Package 'pcds'

March 25, 2020

Type Package

Title Proximity Catch Digraphs and Their Applications

Version 0.1.1

Author Elvan Ceyhan

Maintainer Elvan Ceyhan <elvanceyhan@gmail.com>

Description Contains the functions for generating patterns of segregation, association, CSR (complete spatial randomness)
and Uniform data in one, two and three dimensional cases, for testing these patterns based on two invariants of
various families of the proximity catch digraphs (PCDs) (see (Ceyhan (2005) ISBN:978-3-639-19063-2).
The graph invariants used in testing spatial point data are the domination number (Ceyhan (2011) <doi: 10.1080/03610921003597211>) and arc density (Ceyhan et al. (2006) <doi:10.1016/j.csda.2005.03.002>;
Ceyhan et al. (2007) <doi:10.1002/cjs.5550350106>) of for two-dimensional data for visualization of PCDs for one,
two and three dimensional data. The PCD families considered are Arc-Slice PCDs,
Proportional-Edge PCDs and Central Similarity PCDs.

License GPL-2

Encoding UTF-8

LazyData TRUE

Imports combinat, graphics, interp, plot3D, plotrix, Rdpack (>= 0.7),
stats

RdMacros Rdpack

Suggests scatterplot3d

RoxygenNote 7.0.2

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-25 16:40:02 UTC

R topics documented:

.onAttach . . . 8
.onLoad . . . 8
angle.str2end . . . 9
angle3pnts . . . 11
ArcsASMT . . . 12
ArcsAStri . . . 14
ArcsCS1D . . . 17
ArcsCSend1D . . . 19
ArcsCSMI . . . 21
ArcsCSmid1D . . . 23
ArcsCSMT . . . 25
ArcsCStri . . . 28
ArcsPE1D . . . 31
ArcsPEend1D . . . 33
ArcsPEMI . . . 35
ArcsPEmid1D . . . 37
ArcsPEMT . . . 39
ArcsPEtri . . . 41
area.polygon . . . 44
as.bastri . . . 45
ASarcdens.tri . . . 46
cent.nondeg . . . 48
centersMc . . . 50
centMc . . . 51
circ.cent.bastri . . . 52
circ.cent.tetra . . . 54
circ.cent.tri . . . 55
cl2CC.TbVR . . . 57
cl2CC.VR . . . 59
cl2edgesTe . . . 62
cl2eTbVRcent . . . 64
cl2eVRCC . . . 67
cl2eVRcent . . . 69
cl2eVRcent.alt . . . 72
cl2eVRCM . . . 74
cl2fVRth . . . 76
cl2Mc.int . . . 79
cp2e.bastri . . . 81
cp2e.tri . . . 83
cp2edges.nd . . . 84
CSarcdens.tri . . . 87
dimension . . . 89
Dist . . . 90
dist.pt2set . . . 91
dom.exact . . . 92
dom.greedy . . . 93

dp2l . . . 94
dp2pl . . . 96
fr2eTeER . . . 97
fr2vTbVRCC . . . 99
fr2vVRCC . . . 102
funsAB2CMTe . . . 104
funsAB2MTe . . . 106
funsCartBary . . . 109
funsCSEdgeRegs . . . 111
funsCSGamTe . . . 114
funsCSt1EdgeRegs . . . 118
funsIndDelTri . . . 121
funsMuVarCS1D . . . 123
funsMuVarCS2D . . . 125
funsMuVarCSend1D . . . 127
funsMuVarPE1D . . . 129
funsMuVarPE2D . . . 131
funsMuVarPEend1D . . . 132
funsPG2PE1D . . . 134
funsRankOrderTe . . . 138
funsTbMid2CC . . . 140
Gam1ASbastri . . . 143
Gam1AStri . . . 146
Gam1CS.Te.onesixth . . . 149
Gam1CS1D . . . 152
Gam1CSTe . . . 153
Gam1CSTet1 . . . 156
Gam1PE1D . . . 158
Gam1PEbastri . . . 160
Gam1PEstdTetra . . . 163
Gam1PEtetra . . . 165
Gam1PEtri . . . 168
Gam2ASbastri . . . 171
Gam2AStri . . . 174
Gam2CS.Te.onesixth . . . 176
Gam2PEbastri . . . 178
Gam2PEstdTetra . . . 180
Gam2PEtetra . . . 183
Gam2PEtri . . . 185
Gam3PEstdTetra . . . 188
Gam3PEtetra . . . 190
in.circle . . . 193
in.tetrahedron . . . 194
in.triangle . . . 196
IncMatASMT . . . 197
IncMatAStri . . . 199
IncMatCS1D . . . 200
IncMatCSMT . . . 202

IncMatCSTe . . . 204
IncMatCStri . . . 206
IncMatPE1D . . . 207
IncMatPEMT . . . 209
IncMatPETe . . . 211
IncMatPEtetra . . . 213
IncMatPEtri . . . 214
ind.int.set . . . 216
IndASdomUBtri . . . 217
IndCS.Te.onesixth . . . 219
IndCSdomUBTe . . . 220
IndCSdomUBtri . . . 222
IndCSTe . . . 223
IndCSTe.domset . . . 225
IndCSTeSet . . . 227
IndCSTet1 . . . 229
IndNASbastri . . . 230
IndNAStri . . . 232
IndNAStri.domset . . . 234
IndNAStriSet . . . 236
IndNCSend1D . . . 238
IndNCSint . . . 239
IndNCSmid1D . . . 241
IndNCStri . . . 242
IndNCStri.alt . . . 244
IndNCStri.domset . . . 246
IndNCStriSet . . . 248
IndNPEbastri . . . 250
IndNPEend1D . . . 252
IndNPEint . . . 254
IndNPEmid1D . . . 255
IndNPEstdtetra . . . 257
IndNPETe . . . 258
IndNPETe.domset . . . 260
IndNPETeSet . . . 262
IndNPEtetra . . . 264
IndNPEtri . . . 266
IndNPEtri.domset . . . 268
IndNPEtriSet . . . 270
IndUBdom . . . 272
int.2lines . . . 273
int.circ.line . . . 274
int.line.plane . . . 276
inTriAll . . . 277
is.in.data . . . 279
is.point . . . 280
isStdEqTri . . . 281
Kfr2vTbVRCC . . . 282

Kfr2vVRCC . . . 285
Line . . . 287
Line3D . . . 289
NASbastri . . . 291
NAStri . . . 294
NCSint . . . 297
NCStri . . . 299
NPEbastri . . . 300
NPEint . . . 302
NPEstdtetra . . . 304
NPEtetra . . . 305
NPEtri . . . 307
NumArcsASMT . . . 309
NumArcsAStri . . . 311
NumArcsCSend1D . . . 312
NumArcsCSint . . . 313
NumArcsCSmid1D . . . 315
NumArcsCSMT . . . 316
NumArcsCSTe . . . 318
NumArcsCStri . . . 320
NumArcsPEend1D . . . 322
NumArcsPEint . . . 323
NumArcsPEmid1D . . . 324
NumArcsPEMT . . . 326
NumArcsPETe . . . 328
NumArcsPEtri . . . 330
NumDelTri . . . 331
paraline . . . 332
paraline3D . . . 334
paraplane . . . 336
pcds . . . 338
PEarcdens.tri . . . 339
PEdom.tetra . . . 341
PEdom1D . . . 343
PEdomMT . . . 344
PEdomMTnd . . . 346
PEdomtri . . . 348
perp.ln2pl . . . 350
perpline . . . 352
PG2PE1D.asy . . . 354
PG2PEtri . . . 355
Plane . . . 356
plot.Extrema . . . 358
plot.Lines . . . 359
plot.Lines3D . . . 360
plot.Patterns . . . 361
plot.PCDs . . . 362
plot.Planes . . . 363

plot.TriLines . . . 364
plot.Uniform . . . 365
plotASarcsMT . . . 366
plotASarcsTri . . . 368
plotASregsMT . . . 370
plotASregsTri . . . 372
plotCSarcs1D . . . 374
plotCSarcsMT . . . 377
plotCSarcsTri . . . 379
plotCSregsInt . . . 381
plotCSregsMI . . . 383
plotCSregsMT . . . 385
plotCSregsTri . . . 387
plotDeltri . . . 389
plotIntervals . . . 391
plotPEarcs1D . . . 392
plotPEarcsMT . . . 394
plotPEarcsTri . . . 396
plotPEregsInt . . . 399
plotPEregsMI . . . 401
plotPEregsMT . . . 403
plotPEregsStdTH . . . 405
plotPEregsTH . . . 407
plotPEregsTri . . . 409
print.Extrema . . . 411
print.Lines . . . 412
print.Lines3D . . . 413
print.Patterns . . . 414
print.PCDs . . . 414
print.Planes . . . 415
print.summary.Extrema . . . 416
print.summary.Lines . . . 417
print.summary.Lines3D . . . 417
print.summary.Patterns . . . 418
print.summary.PCDs . . . 418
print.summary.Planes . . . 419
print.summary.TriLines . . . 419
print.summary.Uniform . . . 420
print.TriLines . . . 421
print.Uniform . . . 422
radii . . . 423
radius . . . 425
rasc.disc . . . 426
rasc.matern . . . 428
rasc.tri . . . 431
rascIITe . . . 433
rascMT . . . 434
rascTe . . . 437

re.bastri.cent . . . 439
re.bastriCM . . . 442
re.tri.cent . . . 444
re.triCM . . . 447
redge.triCM . . . 449
redges.tri.cent . . . 451
redges.triCM . . . 453
rel.six.Te . . . 456
reTeCM . . . 458
rseg.disc . . . 460
rseg.tri . . . 463
rsegIITe . . . 465
rsegMT . . . 466
rsegTe . . . 469
runif.bastri . . . 471
runif.stdtetra . . . 473
runif.tetra . . . 475
runif.tri . . . 477
runifMT . . . 478
runifTe . . . 480
runifTe.onesixth . . . 481
rv.bastri.cent . . . 483
rv.bastriCC . . . 485
rv.bastriCM . . . 488
rv.end.int . . . 491
rv.mid.int . . . 492
rv.tetraCC . . . 494
rv.tetraCM . . . 497
rv.tri.cent . . . 499
rv.triCC . . . 501
rv.triCM . . . 504
rverts.tri.cent . . . 506
rverts.tri.cent.alt . . . 509
rverts.tri.nd . . . 511
rverts.triCC . . . 513
rverts.triCM . . . 516
rvTe.cent . . . 518
rvTeCM . . . 520
seg.tri.supp . . . 522
six.ext . . . 524
slope . . . 526
summary.Extrema . . . 527
summary.Lines . . . 528
summary.Lines3D . . . 529
summary.Patterns . . . 530
summary.PCDs . . . 531
summary.Planes . . . 532
summary.TriLines . . . 533

summary.Uniform 534
swamptrees 535
TSArcDensCS1D 536
TSArcDensCSMT 538
TSArcDensPE1D 541
TSArcDensPEMT 543
TSDomPEBin 545
TSDomPEBin1D 548
TSDomPENor 551
XinCHY 553

Index **556**

`.onAttach` *.onAttach start message*

Description

.onAttach start message

Usage

```
.onAttach(libname, pkgname)
```

Arguments

`libname`	defunct
`pkgname`	defunct

Value

invisible()

`.onLoad` *.onLoad getOption package settings*

Description

.onLoad getOption package settings

Usage

```
.onLoad(libname, pkgname)
```

Arguments

libname	defunct
pkgname	defunct

Value

invisible()

Examples

```
getOption("pcds.name")
```

---

angle.str2end *The angles to draw arcs between two line segments*

Description

Gives the two pairs of angles in radians or degrees to draw arcs between two vectors or line segments for the draw.arc function in the plotrix package. The angles are provided with respect to the x-axis in the coordinate system. The line segments are $[ba]$ and $[bc]$ when the argument is given as a,b,c in the function.

radian is a logical argument (default=TRUE) which yields the angle in radians if TRUE, and in degrees if FALSE. The first pair of angles is for drawing arcs in the smaller angle between $[ba]$ and $[bc]$ and the second pair of angles is for drawing arcs in the counter-clockwise order from $[ba]$ to $[bc]$.

Usage

```
angle.str2end(a, b, c, radian = TRUE)
```

Arguments

a, b, c	Three 2D points which represent the intersecting line segments $[ba]$ and $[bc]$.
radian	A logical argument (default=TRUE). If TRUE, the smaller angle or counter-clockwise angle between the line segments $[ba]$ and $[bc]$ is provided in radians, else it is provided in degrees.

Value

A list with two elements

small.arc.angles	Angles of $[ba]$ and $[bc]$ with the x-axis so that difference between them is the smaller angle between $[ba]$ and $[bc]$
ccw.arc.angles	Angles of $[ba]$ and $[bc]$ with the x-axis so that difference between them is the counter-clockwise angle between $[ba]$ and $[bc]$

See Also

angle3pnts

Examples

```
A<-c(.3,.2); B<-c(.6,.3); C<-c(1,1)

pts<-rbind(A,B,C)

Xp<-c(B[1]+max(abs(C[1]-B[1]),abs(A[1]-B[1])),0)

angle.str2end(A,B,C)
angle.str2end(A,B,A)

angle.str2end(A,B,C,radian=FALSE)

#plot of the line segments
ang.rad<-angle.str2end(A,B,C,radian=TRUE); ang.rad
ang.deg<-angle.str2end(A,B,C,radian=FALSE); ang.deg
ang.deg1<-ang.deg$s; ang.deg1
ang.deg2<-ang.deg$c; ang.deg2

rad<-min(Dist(A,B),Dist(B,C))

Xlim<-range(pts[,1],Xp[1],B+Xp,B[1]+c(+rad,-rad))
Ylim<-range(pts[,2],B[2]+c(+rad,-rad))
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

#plot for the smaller arc
plot(pts,pch=1,asp=1,xlab="x",ylab="y",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
L<-rbind(B,B,B); R<-rbind(A,C,B+Xp)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
plotrix::draw.arc(B[1],B[2],radius=.3*rad,angle1=ang.rad$s[1],angle2=ang.rad$s[2])
plotrix::draw.arc(B[1],B[2],radius=.6*rad,angle1=0, angle2=ang.rad$s[1],lty=2,col=2)
plotrix::draw.arc(B[1],B[2],radius=.9*rad,angle1=0,angle2=ang.rad$s[2],col=3)
txt<-rbind(A,B,C)
text(txt+cbind(rep(xd*.02,nrow(txt)),rep(-xd*.02,nrow(txt))),c("A","B","C"))

text(rbind(B)+.5*rad*c(cos(mean(ang.rad$s)),sin(mean(ang.rad$s))),
     paste(abs(round(ang.deg1[2]-ang.deg1[1],2)),"degrees",sep=""))
text(rbind(B)+.6*rad*c(cos(ang.rad$s[1]/2),sin(ang.rad$s[1]/2)),paste(round(ang.deg1[1],2)),col=2)
text(rbind(B)+.9*rad*c(cos(ang.rad$s[2]/2),sin(ang.rad$s[2]/2)),paste(round(ang.deg1[2],2)),col=3)

#plot for the counter-clockwise arc
plot(pts,pch=1,asp=1,xlab="x",ylab="y",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
L<-rbind(B,B,B); R<-rbind(A,C,B+Xp)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
plotrix::draw.arc(B[1],B[2],radius=.3*rad,angle1=ang.rad$c[1],angle2=ang.rad$c[2])
plotrix::draw.arc(B[1],B[2],radius=.6*rad,angle1=0, angle2=ang.rad$s[1],lty=2,col=2)
plotrix::draw.arc(B[1],B[2],radius=.9*rad,angle1=0,angle2=ang.rad$s[2],col=3)
txt<-pts
```

```
text(txt+cbind(rep(xd*.02,nrow(txt)),rep(-xd*.02,nrow(txt))),c("A","B","C"))

text(rbind(B)+.5*rad*c(cos(mean(ang.rad$c)),sin(mean(ang.rad$c))),
     paste(abs(round(ang.deg2[2]-ang.deg2[1],2)),"° degrees",sep=""))
text(rbind(B)+.6*rad*c(cos(ang.rad$s[1]/2),sin(ang.rad$s[1]/2)),paste(round(ang.deg1[1],2)),col=2)
text(rbind(B)+.9*rad*c(cos(ang.rad$s[2]/2),sin(ang.rad$s[2]/2)),paste(round(ang.deg1[2],2)),col=3)
```

`angle3pnts` *The angle between two line segments*

Description

Returns the angle in radians or degrees between two vectors or line segments at the point of intersection. The line segments are $[ba]$ and $[bc]$ when the arguments of the function are given as `a,b,c`. `radian` is a logical argument (default=`TRUE`) which yields the angle in radians if `TRUE`, and in degrees if `FALSE`. The smaller of the angle between the line segments is provided by the function.

Usage

```
angle3pnts(a, b, c, radian = TRUE)
```

Arguments

`a, b, c`	Three 2D points which represent the intersecting line segments $[ba]$ and $[bc]$. The smaller angle between these line segments is to be computed.
`radian`	A logical argument (default=`TRUE`). If `TRUE`, the (smaller) angle between the line segments $[ba]$ and $[bc]$ is provided in radians, else it is provided in degrees.

Value

angle in radians or degrees between the line segments $[ba]$ and $[bc]$

See Also

`angle.str2end`

Examples

```
A<-c(.3,.2); B<-c(.6,.3); C<-c(1,1)
pts<-rbind(A,B,C)

angle3pnts(A,B,C)

angle3pnts(A,B,A)
round(angle3pnts(A,B,A),7)

angle3pnts(A,B,C,radian=FALSE)
```

```
#plot of the line segments
Xlim<-range(pts[,1])
Ylim<-range(pts[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

ang1<-angle3pnts(A,B,C,radian=FALSE)
ang2<-angle3pnts(B+c(1,0),B,C,radian=FALSE)

sa<-angle.str2end(A,B,C,radian=FALSE)$s #small arc angles
ang1<-sa[1]
ang2<-sa[2]

plot(pts,asp=1,pch=1,xlab="x",ylab="y",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
L<-rbind(B,B); R<-rbind(A,C)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
plotrix::draw.arc(B[1],B[2],radius=xd*.1,deg1=ang1,deg2=ang2)
txt<-rbind(A,B,C)
text(txt+cbind(rep(xd*.05,nrow(txt)),rep(-xd*.02,nrow(txt))),c("A","B","C"))

text(rbind(B)+.15*xd*c(cos(pi*(ang2+ang1)/360),sin(pi*(ang2+ang1)/360)),
paste(round(abs(ang1-ang2),2)," degrees"))
```

`ArcsASMT` *The arcs of Arc Slice Proximity Catch Digraph (AS-PCD) for a 2D data set - multiple triangle case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) of AS-PCD whose vertices are the data set `Xp`.

AS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points, i.e., AS proximity regions are defined only for `Xp` points inside the convex hull of `Yp` points. That is, arcs may exist for points only inside the convex hull of `Yp` points. It also provides various descriptions and quantities about the arcs of the AS-PCD such as number of arcs, arc density, etc.

Vertex regions are based on the center `M="CC"` for circumcenter of each Delaunay triangle or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle; default is `M="CC"` i.e. circumcenter of each triangle.

See (Ceyhan (2005, 2010)) for more on AS PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
ArcsASMT(Xp, Yp, M = "CC")
```

Arguments

Xp	A set of 2D points which constitute the vertices of the AS-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangulation. The Delaunay triangles partition the convex hull of Yp points.
M	The center of the triangle. "CC" represents the circumcenter of each Delaunay triangle tri or 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle; default is M="CC" i.e. the circumcenter of each triangle.

Value

A list with the elements

type	A description of the type of the digraph
parameters	Parameters of the digraph, here, it is the center used to construct the vertex regions, i.e. circumcenter.
tess.points	Points on which the tessellation of the study region is performed, here, tessellation is the Delaunay triangulation based on Yp points.
tess.name	Name of data set used in tessellation, i.e., Yp
vertices	Vertices of the digraph, Xp.
vert.name	Name of the data set which constitute the vertices of the digraph
S	Tails (or sources) of the arcs of AS-PCD for 2D data set Xp in the multiple triangle case as the vertices of the digraph
E	Heads (or arrow ends) of the arcs of AS-PCD for 2D data set Xp in the multiple triangle case as the vertices of the digraph
mtitle	Text for "main" title in the plot of the digraph
quant	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

ArcsAStri, ArcsPEtri, ArcsCStri, ArcsPEMT, and ArcsCSMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

ArcsASMT(Xp,Yp,M)

Arcs<-ArcsASMT(Xp,Yp,M)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E
DT<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")

Xlim<-range(Xp[,1],Yp[,1])
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Xp,main=" ", xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),
ylim=Ylim+yd*c(-.05,.05),pch=".",cex=3)
interp::plot.triSht(DT, add=TRUE, do.points = TRUE)
arrows(S[,1], S[,2], E[,1], E[,2], length = 0.1, col= 4)

ArcsASMT(Xp,Yp[1:3,],M)

ArcsASMT(Xp,rbind(Yp,Yp),M)

dat.fr<-data.frame(a=Xp)
ArcsASMT(dat.fr,Yp,M)
```

ArcsAStri *The arcs of Arc Slice Proximity Catch Digraph (AS-PCD) for 2D data - one triangle case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for data set `Xp` as the vertices of AS-PCD.

AS proximity regions are constructed with respect to the triangle `tri`, i.e., arcs may exist for points only inside `tri`. It also provides various descriptions and quantities about the arcs of the AS-PCD such as number of arcs, arc density, etc.

Vertex regions are based on the center `M`="CC" for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M`="CC" the circumcenter of `tri`.

See also (Ceyhan (2005, 2010)).

Usage

```
ArcsAStri(Xp, tri, M = "CC")
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. "CC" stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M`="CC" i.e. the circumcenter of `tri`.

Value

A list with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, it is the center used to construct the vertex regions i.e., circumcenter
`tess.points`	Points on which the tessellation of the study region is performed, here, tessellation is the support triangle.
`tess.name`	Name of data set used in tessellation (i.e., vertices of the triangle).
`vertices`	Vertices of the digraph, `Xp`.
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of AS-PCD for 2D data set `Xp` as vertices of the digraph
`E`	Heads (or arrow ends) of the arcs of AS-PCD for 2D data set `Xp` as vertices of the digraph
`mtitle`	Text for "main" title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

ArcsASMT, ArcsPEtri, ArcsCStri, ArcsPEMT, and ArcsCSMT

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also  #M<-c(1.6,1.2)

ArcsAStri(dat,Tr,M)

Arcs<-ArcsAStri(dat,Tr,M)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

CC<-circ.cent.tri(Tr)
if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
```

```
cent.name<-"CC"
plot(Tr,pch=".",asp=1,xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
} else
{cent<-M
cent.name<-"M"
Ds<-cp2e.tri(Tr,M)
D1<-Ds[1,]; D2<-Ds[2,]; D3<-Ds[3,]
plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
}
polygon(Tr)
points(dat,pch=1,col=1)
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
arrows(S[,1], S[,2], E[,1], E[,2], length = 0.1, col= 4)

txt<-rbind(Tr,cent,Ds)
xc<-txt[,1]+c(-.02,.03,.02,.03,.04,-.03,-.01)
yc<-txt[,2]+c(.02,.02,.03,.06,.04,.05,-.07)
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)

dat.fr<-data.frame(a=dat)
ArcsAStri(dat.fr,Tr,M)
```

`ArcsCS1D` *The arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - multiple interval case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for 1D data set `Xp` as the vertices of CS-PCD. `Yp` determines the end points of the intervals.

For this function, CS proximity regions are constructed data points inside or outside the intervals based on `Yp` points with expansion parameter $t > 0$ and centrality parameter `c` in $(0, 1)$. That is, for this function, arcs may exist for points in the middle or end intervals. It also provides various descriptions and quantities about the arcs of the CS-PCD such as number of arcs, arc density, etc.

Equivalent to function `ArcsCSMI`.

See also (Ceyhan (2016)).

Usage

```
ArcsCS1D(Xp, Yp, t, c)
```

Arguments

Xp	A set or vector of 1D points which constitute the vertices of the CS-PCD.
Yp	A set or vector of 1D points which constitute the end points of the intervals.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A list with the elements

type	A description of the type of the digraph
parameters	Parameters of the digraph, here, they are expansion and centrality parameters.
tess.points	Points on which the tessellation of the study region is performed, here, tessellation is the intervalization of the real line based on Yp points.
tess.name	Name of data set used in tessellation, it is Yp for this function
vertices	Vertices of the digraph, Xp points
vert.name	Name of the data set which constitute the vertices of the digraph
S	Tails (or sources) of the arcs of CS-PCD for 1D data
E	Heads (or arrow ends) of the arcs of CS-PCD for 1D data
mtitle	Text for "main" title in the plot of the digraph
quant	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

ArcsCSmid1D, ArcsCSend1D, ArcsCSMI, ArcsPEmid1D, ArcsPEend1D and ArcsPE1D

Examples

```
t<-1.5
c<-.4
a<-0; b<-10; int<-c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1
```

```
Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

ArcsCS1D(Xp,Yp,t,c)

Arcs<-ArcsCS1D(Xp,Yp,t,c)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),pch=".",
main="arcs of CS-PCD for points in mid and end intervals ", xlab=" ", ylab=" ",
     xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit))
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)
```

ArcsCSend1D *The arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - end interval case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for 1D data set Xp as the vertices of CS-PCD. Yp determines the end points of the end intervals.

For this function, CS proximity regions are constructed data points outside the intervals based on Yp points with expansion parameter $t > 0$. That is, for this function, arcs may exist for points only inside end intervals. It also provides various descriptions and quantities about the arcs of the CS-PCD such as number of arcs, arc density, etc.

See also (Ceyhan (2016)).

Usage

```
ArcsCSend1D(Xp, Yp, t)
```

Arguments

Xp	A set or vector of 1D points which constitute the vertices of the CS-PCD.
Yp	A set or vector of 1D points which constitute the end points of the intervals.
t	A positive real number which serves as the expansion parameter in CS proximity region.

Value

A list with the elements

type	A description of the type of the digraph
parameters	Parameters of the digraph, here, it is the expansion parameter.
tess.points	Points on which the tessellation of the study region is performed, here, tessellation is the intervalization based on Yp.
tess.name	Name of data set used in tessellation, it is Yp for this function
vertices	Vertices of the digraph, Xp points
vert.name	Name of the data set which constitutes the vertices of the digraph
S	Tails (or sources) of the arcs of CS-PCD for 1D data in the end intervals
E	Heads (or arrow ends) of the arcs of CS-PCD for 1D data in the end intervals
mtitle	Text for "main" title in the plot of the digraph
quant	Various quantities for the digraph: number of vertices, number of partition points, number of intervals (which is 2 for end intervals), number of arcs, and arc density.

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

ArcsCSmid1D, ArcsCS1D , ArcsPEmid1D, ArcsPEend1D and ArcsPE1D

Examples

```
t<-1.5
a<-0; b<-10; int<-c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.5

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)
```

```
ArcsCSend1D(Xp,Yp,t)

Arcs<-ArcsCSend1D(Xp,Yp,t)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),pch=".",
main="arcs of CS-PCD with vertices (jittered along y-axis)\n in end intervals ",
     xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit))
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)

ArcsCSend1D(Xp,Yp,t)
```

ArcsCSMI *The arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - multiple interval case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for 1D data set Xp as the vertices of CS-PCD. Yp determines the end points of the intervals.

For this function, CS proximity regions are constructed data points inside or outside the intervals based on Yp points with expansion parameter $t > 0$ and centrality parameter c in $(0, 1)$. That is, for this function, arcs may exist for points in the middle or end intervals. It also provides various descriptions and quantities about the arcs of the CS-PCD such as number of arcs, arc density, etc.

Equivalent to function ArcsCS1D.

See also (Ceyhan (2016)).

Usage

```
ArcsCSMI(Xp, Yp, t, c)
```

Arguments

Xp	A set or vector of 1D points which constitute the vertices of the CS-PCD.
Yp	A set or vector of 1D points which constitute the end points of the intervals.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A list with the elements

type	A description of the type of the digraph
parameters	Parameters of the digraph, here, they are expansion and centrality parameters.
tess.points	Points on which the tessellation of the study region is performed, here, tessellation is the intervalization of the real line based on Yp points.
tess.name	Name of data set used in tessellation, it is Yp for this function
vertices	Vertices of the digraph, Xp points
vert.name	Name of the data set which constitute the vertices of the digraph
S	Tails (or sources) of the arcs of CS-PCD for 1D data
E	Heads (or arrow ends) of the arcs of CS-PCD for 1D data
mtitle	Text for "main" title in the plot of the digraph
quant	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

ArcsCSend1D, ArcsCSmid1D, ArcsCS1D, and ArcsPEMI

Examples

```
t<-2
c<-.4
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1
```

```
Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

Arcs<-ArcsCSMI(Xp,Yp,t,c)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

ArcsCSMI(Xp,Yp,t,c)

ArcsCSMI(Xp,Yp+10,t,c)

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),
main="arcs of CS-PCD for points (jittered along y-axis)\n in middle intervals ",
xlab=" ", ylab=" ", xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit),pch=".")
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)

t<-2
c<-.4
a<-0; b<-10;
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

ArcsCSMI(Xp,Yp,t,c)
```

ArcsCSmid1D *The arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - middle intervals case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for 1D data set Xp as the vertices of CS-PCD.

For this function, CS proximity regions are constructed with respect to the intervals based on Yp points with expansion parameter $t > 0$ and centrality parameter c in $(0, 1)$. That is, for this function, arcs may exist for points only inside the intervals. It also provides various descriptions and quantities about the arcs of the CS-PCD such as number of arcs, arc density, etc.

Vertex regions are based on center M_c of each middle interval.

See also (Ceyhan (2016)).

Usage

```
ArcsCSmid1D(Xp, Yp, t, c)
```

Arguments

Xp	A set or vector of 1D points which constitute the vertices of the CS-PCD.
Yp	A set or vector of 1D points which constitute the end points of the intervals.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A list with the elements

type	A description of the type of the digraph
parameters	Parameters of the digraph, here, they are expansion and centrality parameters.
tess.points	Points on which the tessellation of the study region is performed, here, tessellation is the intervalization based on Yp points.
tess.name	Name of data set used in tessellation, it is Yp for this function
vertices	Vertices of the digraph, i.e. Xp points
vert.name	Name of the data set which constitute the vertices of the digraph
S	Tails (or sources) of the arcs of CS-PCD for 1D data in the middle intervals
E	Heads (or arrow ends) of the arcs of CS-PCD for 1D data in the middle intervals
mtitle	Text for "main" title in the plot of the digraph
quant	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

ArcsPEend1D, ArcsPE1D, ArcsCSmid1D, ArcsCSend1D and ArcsCS1D

Examples

```
t<-1.5
c<-.4
a<-0; b<-10

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

ArcsCSmid1D(Xp,Yp,t,c)
ArcsCSmid1D(Xp,Yp+10,t,c)

Arcs<-ArcsCSmid1D(Xp,Yp,t,c)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),
main="arcs of CS-PCD whose vertices (jittered along y-axis)\n in middle intervals ",
xlab=" ", ylab=" ", xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit),pch=".")
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)

t<-.5
c<-.4
a<-0; b<-10;
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

ArcsCSmid1D(Xp,Yp,t,c)
```

ArcsCSMT *The arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for 2D data - multiple triangle case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) of Central Similarity Proximity Catch Digraph (CS-PCD) whose vertices are the data points in `Xp` in the multiple triangle case.

CS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $\tau > 0$ and edge regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (unscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the number of arcs, loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) for more on CS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
ArcsCSMT(Xp, Yp, tau, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`tau`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle, default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

A list with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, it is the center used to construct the vertex regions.
`tess.points`	Points on which the tessellation of the study region is performed, here, tessellation is Delaunay triangulation based on `Yp` points.
`tess.name`	Name of data set used in tessellation, it is `Yp` for this function
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of CS-PCD for 2D data set `Xp` as vertices of the digraph

E	Heads (or arrow ends) of the arcs of CS-PCD for 2D data set Xp as vertices of the digraph
mtitle	Text for "main" title in the plot of the digraph
quant	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

ArcsCStri, ArcsASMT and ArcsPEMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

tau<-1.5  #try also tau<-2

ArcsCSMT(Xp,Yp,tau,M)

Arcs<-ArcsCSMT(Xp,Yp,tau,M)
Arcs
summary(Arcs)
plot(Arcs)
```

```
S<-Arcs$S
E<-Arcs$E
DT<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")

Xlim<-range(Xp[,1],Yp[,1])
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Xp,main=" ", xlab=" ", ylab=" ",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),pch=".",cex=3)
interp::plot.triSht(DT, add=TRUE, do.points = TRUE)
arrows(S[,1], S[,2], E[,1], E[,2], length = 0.1, col= 4)

ArcsCSMT(Xp,Yp,tau,M)
ArcsCSMT(Xp,Yp[1:3,],tau,M)

ArcsCSMT(Xp,rbind(Yp,Yp),tau,M)

dat.fr<-data.frame(a=Xp)
ArcsCSMT(dat.fr,Yp,tau,M)

dat.fr<-data.frame(a=Yp)
ArcsCSMT(Xp,dat.fr,tau,M)
```

`ArcsCStri` *The arcs of Central Similarity Proximity Catch Digraphs (CS-PCD) for 2D data - one triangle case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for data set `Xp` as the vertices of CS-PCD.

CS proximity regions are constructed with respect to the triangle `tri` with expansion parameter $t > 0$, i.e., arcs may exist for points only inside `tri`. It also provides various descriptions and quantities about the arcs of the CS-PCD such as number of arcs, arc density, etc. Edge regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
ArcsCStri(Xp, tri, t, M = c(1, 1, 1))
```

Arguments

Xp	A set of 2D points which constitute the vertices of the CS-PCD.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
t	A positive real number which serves as the expansion parameter in CS proximity region.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.

Value

A list with the elements

type	A description of the type of the digraph
parameters	Parameters of the digraph, here, it is the center used to construct the vertex regions.
tess.points	Points on which the tessellation of the study region is performed, here, tessellation is the support triangle.
tess.name	Name of data set used in tessellation (i.e., vertices of the triangle)
vertices	Vertices of the digraph, Xp points
vert.name	Name of the data set which constitute the vertices of the digraph
S	Tails (or sources) of the arcs of CS-PCD for 2D data set Xp as vertices of the digraph
E	Heads (or arrow ends) of the arcs of CS-PCD for 2D data set Xp as vertices of the digraph
mtitle	Text for "main" title in the plot of the digraph
quant	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

ArcsCSMT, ArcsAStri and ArcsPEtri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

t<-1.5  #try also t<-2

ArcsCStri(dat,Tr,t,M)

Arcs<-ArcsCStri(dat,Tr,t,M)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

Ds<-cp2e.tri(Tr,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=1,col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
arrows(S[,1], S[,2], E[,1], E[,2], length = 0.1, col= 4)

txt<-rbind(Tr,M,Ds)
xc<-txt[,1]+c(-.02,.03,.02,.03,.04,-.03,-.01)
yc<-txt[,2]+c(.02,.02,.03,.06,.03,.05,-.07)
txt.str<-c("A","B","C","M","D1","D2","D3")
text(xc,yc,txt.str)

t<-2
ArcsCStri(dat,Tr,t,M)

dat.fr<-data.frame(a=dat)
ArcsCStri(dat.fr,Tr,t,M)
```

```
dat.fr<-data.frame(a=Tr)
ArcsCStri(dat,dat.fr,t,M)
```

ArcsPE1D *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - multiple interval case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for 1D data set Xp as the vertices of PE-PCD. Yp determines the end points of the intervals.

For this function, PE proximity regions are constructed data points inside or outside the intervals based on Yp points with expansion parameter $r \geq 1$ and centrality parameter c in $(0, 1)$. That is, for this function, arcs may exist for points in the middle or end intervals. It also provides various descriptions and quantities about the arcs of the PE-PCD such as number of arcs, arc density, etc.

Equivalent to function ArcsPEMI.

See also (Ceyhan (2012)).

Usage

```
ArcsPE1D(Xp, Yp, r, c)
```

Arguments

Xp	A set or vector of 1D points which constitute the vertices of the PE-PCD.
Yp	A set or vector of 1D points which constitute the end points of the intervals.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A list with the elements

type	A description of the type of the digraph
parameters	Parameters of the digraph, here, they are expansion and centrality parameters.
tess.points	Points on which the tessellation of the study region is performed, here, tessellation is the intervalization of the real line based on Yp points.
tess.name	Name of data set used in tessellation, it is Yp for this function
vertices	Vertices of the digraph, Xp points
vert.name	Name of the data set which constitute the vertices of the digraph
S	Tails (or sources) of the arcs of PE-PCD for 1D data

E	Heads (or arrow ends) of the arcs of PE-PCD for 1D data
mtitle	Text for "main" title in the plot of the digraph
quant	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

ArcsPEmid1D, ArcsPEend1D, ArcsPEMI, ArcsCSmid1D, ArcsCSend1D and ArcsCS1D

Examples

```
r<-2
c<-.4
a<-0; b<-10; int<-c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

Arcs<-ArcsPE1D(Xp,Yp,r,c)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

ArcsPE1D(Xp,Yp,r,c)

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),pch=".",main="arcs for points in mid and end intervals ", xlab=" ", ylab=" ",
     xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit))
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)
```

`ArcsPEend1D` *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - end interval case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for 1D data set `Xp` as the vertices of PE-PCD. `Yp` determines the end points of the end intervals.

For this function, PE proximity regions are constructed data points outside the intervals based on `Yp` points with expansion parameter $r \geq 1$. That is, for this function, arcs may exist for points only inside end intervals. It also provides various descriptions and quantities about the arcs of the PE-PCD such as number of arcs, arc density, etc.

See also (Ceyhan (2012)).

Usage

```
ArcsPEend1D(Xp, Yp, r)
```

Arguments

`Xp`	A set or vector of 1D points which constitute the vertices of the PE-PCD.
`Yp`	A set or vector of 1D points which constitute the end points of the intervals.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

Value

A list with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, it is the expansion parameter.
`tess.points`	Points on which the tessellation of the study region is performed, here, tessellation is the intervalization based on `Yp`.
`tess.name`	Name of data set used in tessellation, it is `Yp` for this function
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitutes the vertices of the digraph
`S`	Tails (or sources) of the arcs of PE-PCD for 1D data in the end intervals
`E`	Heads (or arrow ends) of the arcs of PE-PCD for 1D data in the end intervals
`mtitle`	Text for "main" title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals (which is 2 for end intervals), number of arcs, and arc density.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

ArcsPEmid1D, ArcsPE1D , ArcsCSmid1D, ArcsCSend1D and ArcsCS1D

Examples

```
r<-2
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.5

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)  #try also Yp<-runif(ny,a,b)+c(-10,10)

Arcs<-ArcsPEend1D(Xp,Yp,r)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),pch=".",
main="arcs of PE-PCDs for points (jittered along y-axis)\n in end intervals ",
xlab=" ", ylab=" ", xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit))
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)

ArcsPEend1D(Xp,Yp,r)
```

ArcsPEMI *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - multiple interval case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for 1D data set Xp as the vertices of PE-PCD. Yp determines the end points of the intervals.

For this function, PE proximity regions are constructed data points inside or outside the intervals based on Yp points with expansion parameter $r \geq 1$ and centrality parameter c in $(0, 1)$. That is, for this function, arcs may exist for points in the middle or end intervals. It also provides various descriptions and quantities about the arcs of the PE-PCD such as number of arcs, arc density, etc.

Equivalent to function ArcsPE1D.

See also (Ceyhan (2012)).

Usage

```
ArcsPEMI(Xp, Yp, r, c)
```

Arguments

Xp	A set or vector of 1D points which constitute the vertices of the PE-PCD.
Yp	A set or vector of 1D points which constitute the end points of the intervals.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A list with the elements

type	A description of the type of the digraph
parameters	Parameters of the digraph, here, they are expansion and centrality parameters.
tess.points	Points on which the tessellation of the study region is performed, here, tessellation is the intervalization of the real line based on Yp points.
tess.name	Name of data set used in tessellation, it is Yp for this function
vertices	Vertices of the digraph, Xp points
vert.name	Name of the data set which constitute the vertices of the digraph
S	Tails (or sources) of the arcs of PE-PCD for 1D data
E	Heads (or arrow ends) of the arcs of PE-PCD for 1D data
mtitle	Text for "main" title in the plot of the digraph
quant	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

ArcsPE1D, ArcsPEmid1D, ArcsPEend1D, and ArcsCSMI

Examples

```
r<-2
c<-.4
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

Arcs<-ArcsPEMI(Xp,Yp,r,c)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

ArcsPEMI(Xp,Yp,r,c)
ArcsPEMI(Xp,Yp+10,r,c)

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),
main="arcs of PE-PCD for points (jittered along y-axis)\n in middle intervals ",
xlab=" ", ylab=" ", xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit),pch=".")
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
arrows(S, yjit, E, yjit, length = .05, col= 4)

r<-2
c<-.4
a<-0; b<-10;
```

```
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)
ArcsPEMI(Xp,Yp,r,c)
```

`ArcsPEmid1D` *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - middle intervals case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for 1D data set `Xp` as the vertices of PE-PCD.

For this function, PE proximity regions are constructed with respect to the intervals based on `Yp` points with expansion parameter $r \geq 1$ and centrality parameter `c` in $(0, 1)$. That is, for this function, arcs may exist for points only inside the intervals. It also provides various descriptions and quantities about the arcs of the PE-PCD such as number of arcs, arc density, etc.

Vertex regions are based on center M_c of each middle interval.

See also (Ceyhan (2012)).

Usage

```
ArcsPEmid1D(Xp, Yp, r, c)
```

Arguments

`Xp`	A set or vector of 1D points which constitute the vertices of the PE-PCD.
`Yp`	A set or vector of 1D points which constitute the end points of the intervals.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A list with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, they are expansion and centrality parameters.
`tess.points`	Points on which the tessellation of the study region is performed, here, tessellation is the intervalization based on `Yp` points.
`tess.name`	Name of data set used in tessellation, it is `Yp` for this function
`vertices`	Vertices of the digraph, i.e. `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph

S	Tails (or sources) of the arcs of PE-PCD for 1D data in the middle intervals
E	Heads (or arrow ends) of the arcs of PE-PCD for 1D data in the middle intervals
mtitle	Text for "main" title in the plot of the digraph
quant	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

ArcsPEend1D, ArcsPE1D, ArcsCSmid1D, ArcsCSend1D and ArcsCS1D

Examples

```
r<-2
c<-.4
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

Arcs<-ArcsPEmid1D(Xp,Yp,r,c)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

ArcsPEmid1D(Xp,Yp,r,c)
ArcsPEmid1D(Xp,Yp+10,r,c)

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),
main="arcs of PE-PCD for points (jittered along y-axis)\n in middle intervals ",
xlab=" ", ylab=" ", xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit),pch=".")
abline(h=0,lty=1)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
```

```
arrows(S, yjit, E, yjit, length = .05, col= 4)

r<-2
c<-.4
a<-0; b<-10;
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)
ArcsPEmid1D(Xp,Yp,r,c)
```

ArcsPEMT *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D data - multiple triangle case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) of Proportional Edge Proximity Catch Digraph (PE-PCD) whose vertices are the data points in Xp in the multiple triangle case.

PE proximity regions are defined with respect to the Delaunay triangles based on Yp points with expansion parameter $r \geq 1$ and vertex regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (unscaled) basic triangle so that M will be the same type of center for each Delaunay triangle (this conversion is not necessary when M is CM).

Convex hull of Yp is partitioned by the Delaunay triangles based on Yp points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of Yp points). For the number of arcs, loops are not allowed so arcs are only possible for points inside the convex hull of Yp points.

See (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)) for more on the PE-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
ArcsPEMT(Xp, Yp, r, M = c(1, 1, 1))
```

Arguments

Xp	A set of 2D points which constitute the vertices of the PE-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this argument should be set as M="CC"), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

A list with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, it is the center used to construct the vertex regions.
`tess.points`	Points on which the tessellation of the study region is performed, here, tessellation is Delaunay triangulation based on `Yp` points.
`tess.name`	Name of data set used in tessellation, it is `Yp` for this function
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of PE-PCD for 2D data set `Xp` as vertices of the digraph
`E`	Heads (or arrow ends) of the arcs of PE-PCD for 2D data set `Xp` as vertices of the digraph
`mtitle`	Text for "main" title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`ArcsPEtri`, `ArcsASMT` and `ArcsCSMT`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

r<-1.5  #try also r<-2

ArcsPEMT(Xp,Yp,r,M)

Arcs<-ArcsPEMT(Xp,Yp,r,M)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E
DT<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")

Xlim<-range(Xp[,1],Yp[,1])
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Xp,main=" ", xlab=" ", ylab=" ",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),pch=".",cex=3)
interp::plot.triSht(DT, add=TRUE, do.points = TRUE)
arrows(S[,1], S[,2], E[,1], E[,2], length = 0.1, col= 4)

ArcsPEMT(Xp,Yp,r)
ArcsPEMT(Xp,Yp[1:3,],r)

ArcsPEMT(Xp,rbind(Yp,Yp),r)

dat.fr<-data.frame(a=Xp)
ArcsPEMT(dat.fr,Yp,r)

dat.fr<-data.frame(a=Yp)
ArcsPEMT(Xp,dat.fr,r)
```

`ArcsPEtri` *The arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D data - one triangle case*

Description

An object of class "PCDs". Returns arcs as tails (or sources) and heads (or arrow ends) for data set `Xp` as the vertices of PE-PCD.

PE proximity regions are constructed with respect to the triangle `tri` with expansion parameter $r \geq 1$, i.e., arcs may exist for points only inside `tri`. It also provides various descriptions and quantities about the arcs of the PE-PCD such as number of arcs, arc density, etc.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

See also (Ceyhan (2005); Ceyhan et al. (2006)).

Usage

```
ArcsPEtri(Xp, tri, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

Value

A list with the elements

`type`	A description of the type of the digraph
`parameters`	Parameters of the digraph, here, it is the center used to construct the vertex regions.
`tess.points`	Points on which the tessellation of the study region is performed, here, tessellation is the support triangle.
`tess.name`	Name of data set used in tessellation (i.e., vertices of the triangle)
`vertices`	Vertices of the digraph, `Xp` points
`vert.name`	Name of the data set which constitute the vertices of the digraph
`S`	Tails (or sources) of the arcs of PE-PCD for 2D data set `Xp` as vertices of the digraph
`E`	Heads (or arrow ends) of the arcs of PE-PCD for 2D data set `Xp` as vertices of the digraph
`mtitle`	Text for "main" title in the plot of the digraph
`quant`	Various quantities for the digraph: number of vertices, number of partition points, number of intervals, number of arcs, and arc density.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

ArcsPEMT, ArcsAStri and ArcsCStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5  #try also r<-2

ArcsPEtri(dat,Tr,r,M)

Arcs<-ArcsPEtri(dat,Tr,r,M)
Arcs
summary(Arcs)
plot(Arcs)

S<-Arcs$S
E<-Arcs$E

Xlim<-range(Tr[,1],dat[,1],M[1])
Ylim<-range(Tr[,2],dat[,2],M[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

Ds<-cp2e.tri(Tr,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=1,col=1)
L<-rbind(M,M,M); R<-Ds
```

```
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
arrows(S[,1], S[,2], E[,1], E[,2], length = 0.1, col= 4)

txt<-rbind(Tr,M,Ds)
xc<-txt[,1]+c(-.02,.03,.02,.03,.04,-.03,-.01)
yc<-txt[,2]+c(.02,.02,.03,.06,.04,.05,-.07)
txt.str<-c("A","B","C","M","D1","D2","D3")
text(xc,yc,txt.str)

r<-2
ArcsPEtri(dat,Tr,r,M)

dat.fr<-data.frame(a=dat)
ArcsPEtri(dat.fr,Tr,r,M)

dat.fr<-data.frame(a=Tr)
ArcsPEtri(dat,dat.fr,r,M)
```

`area.polygon` *The area of a polygon in R^2*

Description

Returns the area of the polygon, h, in the real plane R^2; the vertices of the polygon h must be provided in clockwise or counter-clockwise order, otherwise the function does not yield the area of the polygon. Also, the polygon could be convex or non-convex. See (Weisstein (2019)).

Usage

```
area.polygon(h)
```

Arguments

h	A vector of n 2D points, stacked row-wise, each row representing a vertex of the polygon, where n is the number of vertices of the polygon.

Value

area of the polygon h

References

Weisstein EW (2019). "Polygon Area." http://mathworld.wolfram.com/PolygonArea.html. From MathWorld–A Wolfram Web Resource.

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(0.5,.8);
Tr<-rbind(A,B,C);
area.polygon(Tr)

A<-c(0,0); B<-c(1,0); C<-c(.7,.6); D<-c(0.3,.8);
h1<-rbind(A,B,C,D);  #try also h1<-rbind(A,B,D,C) or h1<-rbind(A,C,B,D) or h1<-rbind(A,D,C,B);
area.polygon(h1)

Xlim<-range(h1[,1])
Ylim<-range(h1[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(h1,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(h1)
xc<-rbind(A,B,C,D)[,1]+c(-.03,.03,.02,-.01)
yc<-rbind(A,B,C,D)[,2]+c(.02,.02,.02,.04)
txt.str<-c("A","B","C","D")
text(xc,yc,txt.str)

dat.fr<-data.frame(a=Tr)
area.polygon(dat.fr)

#when the triangle is degenerate, it gives zero area
B<-A+2*(C-A);
T2<-rbind(A,B,C)
area.polygon(T2)
```

---

as.bastri *The labels of the vertices of a triangle in the (unscaled) basic triangle form*

Description

Labels the vertices of triangle, `tri`, as ABC so that AB is the longest edge, BC is the second longest and AC is the shortest edge (the order is as in the basic triangle). The new triangle $T(A, B, C)$ is unscaled, i.e., the longest edge AB may not be of unit length.

The basic triangle is $T_b = T((0,0),(1,0),(c_1,c_2))$ where `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

Usage

```
as.bastri(tri)
```

Arguments

tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

A list with two elements

tri	The vertices of the triangle stacked row-wise and labeled row-wise as A, B, C.
desc	Description of the edges based on the vertices, i.e., "Edges (in decreasing length are) AB, BC, and AC".
orig.order	Row order of the input triangle, tri, when converted to the scaled version of the basic triangle

Examples

```
## Not run:
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);

as.bastri(rbind(A,B,C))

as.bastri(rbind(B,C,A))

as.bastri(rbind(B,A,C))
as.bastri(rbind(A,C,B))

A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
as.bastri(rbind(A,B,C))
as.bastri(rbind(A,C,B))
as.bastri(rbind(B,A,C))

A<-runif(2); B<-runif(2); C<-runif(2)
as.bastri(rbind(A,B,C))

dat.fr<-data.frame(a=rbind(A,B,C))
as.bastri(dat.fr)

## End(Not run)
```

ASarcdens.tri *Arc density of Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns the arc density of AS-PCD whose vertex set is the given 2D numerical data set, `Xp`, (some of its members are) in the triangle `tri`.

AS proximity regions is defined with respect to `tri` and vertex regions are defined with the center `M`="CC" for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M`="CC" i.e. circumcenter of `tri`. For the number of arcs, loops are not allowed so arcs are only possible for points inside `tri` for this function.

`tri.cor` is a logical argument for triangle correction (default is `TRUE`), if `TRUE`, only the points inside the triangle are considered (i.e., digraph induced by these vertices are considered) in computing the arc density, otherwise all points are considered (for the number of vertices in the denominator of arc density).

See also (Ceyhan (2005, 2010)).

Usage

```
ASarcdens.tri(Xp, tri, M = "CC", tri.cor = TRUE)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. "CC" stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of `tri`; default is `M`="CC" i.e. the circumcenter of `tri`.
`tri.cor`	A logical argument for computing the arc density for only the points inside the triangle, `tri` (default=`TRUE`), i.e., if `TRUE` only the induced digraph with the vertices inside `tri` are considered in the computation of arc density.

Value

Arc density of AS-PCD whose vertices are the 2D numerical data set, `Xp`; AS proximity regions are defined with respect to the triangle `tri` and CC-vertex regions.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

ASarcdens.tri, CSarcdens.tri, and NumArcsAStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

set.seed(1)
n<-10  #try also n<-20

dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

NumArcsAStri(dat,Tr,M)
ASarcdens.tri(dat,Tr,M)
ASarcdens.tri(dat,Tr,M,tri.cor = FALSE)

ASarcdens.tri(dat,Tr,M)

dat.fr<-data.frame(a=dat)
ASarcdens.tri(dat.fr,Tr,M)

dat.fr<-data.frame(a=Tr)
ASarcdens.tri(dat,dat.fr,M)
```

---

cent.nondeg *Centers for non-degenerate asymptotic distribution of domination number of Proportional Edge Proximity Catch Digraphs (PE-PCDs)*

Description

Returns the centers which yield nondegenerate asymptotic distribution for the domination number of PE-PCD for uniform data in a triangle, $tri = T(v_1, v_2, v_3)$.

PE proximity region is defined with respect to the triangle `tri` with expansion parameter `r` in $(1, 1.5]$.

Vertex regions are defined with the centers that are output of this function. Centers are stacked row-wise with row number is corresponding to the vertex row number in `tri` (see the examples for an illustration). The center labels 1,2,3 correspond to the vertices M_1, M_2, and M_3 (which are the three centers for `r` in $(1, 1.5)$ which becomes center of mass for $r = 1.5$.).

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
cent.nondeg(tri, r)
```

Arguments

`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$ for this function.

Value

The centers (stacked row-wise) which give nondegenerate asymptotic distribution for the domination number of PE-PCD for uniform data in a triangle, `tri`.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
r<-1.35

Ms<-cent.nondeg(Tr,r)
Ms

Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Ms,pch=".",col=1)
polygon(Ms,lty=2)

xc<-Tr[,1]+c(-.02,.02,.02)
yc<-Tr[,2]+c(.02,.02,.03)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)
```

```
xc<-Ms[,1]+c(-.04,.04,.03)
yc<-Ms[,2]+c(.02,.02,.05)
txt.str<-c("M1","M2","M3")
text(xc,yc,txt.str)

cent.nondeg(Tr,r)

dat.fr<-data.frame(a=Tr)
cent.nondeg(dat.fr,r)
```

`centersMc` *Parameterized centers of intervals*

Description

Returns the centers of the intervals based on 1D points in x parameterized by c in $(0, 1)$ so that $100c$ % of the length of interval is to the left of M_c and $100(1 - c)$ % of the length of the interval is to the right of M_c. That is, for an interval (a, b), the parameterized center is $M_c = a + c(b - a)$ x is a vector of 1D points, not necessarily sorted.

See also (Ceyhan (2012, 2016)).

Usage

```
centersMc(x, c)
```

Arguments

x	A vector real numbers that constitute the end points of intervals.
c	A positive real number in $(0, 1)$ parameterizing the centers inside the intervals. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

(parameterized) centers of the intervals based on x points as a vector

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

centMc

Examples

```
n<-10
c<-.4  #try also c<-runif(1)
x<-runif(n)
centersMc(x,c)

n<-10
c<-.3  #try also c<-runif(1)
x<-runif(n,0,10)
centersMc(x,c)
```

---

centMc *Parameterized center of an interval*

Description

Returns the (parameterized) center, M_c, of the interval, $int = (a, b)$, parameterized by c in $(0, 1)$ so that $100c$ % of the length of interval is to the left of M_c and $100(1 - c)$ % of the length of the interval is to the right of M_c. That is, for the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

See also (Ceyhan (2012, 2016)).

Usage

```
centMc(int, c)
```

Arguments

int	A vector with two entries representing an interval.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

(parameterized) center inside int

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

centersMc

Examples

```
c<-.4
a<-0; b<-10; int<-c(a,b)
int
centMc(int,c)

c<-.3
a<-2; b<-4; int<-c(a,b)
centMc(int,c)
```

`circ.cent.bastri` *Circumcenter of a basic triangle*

Description

Returns the circumcenter of a basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ given `c1`, `c2` where `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See (Weisstein (2019); Ceyhan (2010)) for triangle centers and (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)) for the basic triangle.

Usage

```
circ.cent.bastri(c1, c2)
```

Arguments

`c1, c2`	Positive real numbers representing the top vertex in basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, `c1` must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Value

circumcenter of the basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ given `c1`, `c2` as the arguments of the function

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial

segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

Weisstein EW (2019). "Triangle Centers." http://mathworld.wolfram.com/TriangleCenter.html. From MathWorld–A Wolfram Web Resource.

See Also

circ.cent.tri

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);  #the vertices of the basic triangle Tb
Tb<-rbind(A,B,C)
CC<-circ.cent.bastri(c1,c2)  #the circumcenter
CC

D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2; #midpoints of the edges
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1])
Ylim<-range(Tb[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

oldpar <- par(mfrow = c(1,2))
par(pty="s")
plot(A,pch=".",asp=1,xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
points(rbind(CC))
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tb,CC,D1,D2,D3)
xc<-txt[,1]+c(-.03,.04,.03,.06,.06,-.03,0)
yc<-txt[,2]+c(.02,.02,.03,-.03,.02,.04,-.03)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)
par(oldpar)

#for an obtuse triangle
c1<-.4; c2<-.3;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);  #the vertices of the basic triangle Tb
Tb<-rbind(A,B,C)
CC<-circ.cent.bastri(c1,c2)  #the circumcenter
CC

D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2; #midpoints of the edges
```

```
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1],CC[1])
Ylim<-range(Tb[,2],CC[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

oldpar <- par(mfrow = c(1,2))
par(pty="s")
plot(A,pch=".",asp=1,xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
points(rbind(CC))
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tb,CC,D1,D2,D3)
xc<-txt[,1]+c(-.03,.03,.03,.07,.07,-.05,0)
yc<-txt[,2]+c(.02,.02,.04,-.03,.03,.04,.06)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)
par(oldpar)
```

`circ.cent.tetra` *Circumcenter of a general tetrahedron*

Description

Returns the circumcenter a given tetrahedron `th` with vertices stacked row-wise.

Usage

```
circ.cent.tetra(th)
```

Arguments

`th`	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.

Value

circumcenter of the tetrahedron `th`

See Also

`circ.cent.tri`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

CC<-circ.cent.tetra(tetra)
CC

Xlim<-range(tetra[,1],CC[1])
Ylim<-range(tetra[,2],CC[2])
Zlim<-range(tetra[,3],CC[3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(tetra[,1],tetra[,2],tetra[,3], phi =0,theta=40, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
          pch = 20, cex = 1, ticktype = "detailed")
#add the vertices of the tetrahedron
plot3D::points3D(CC[1],CC[2],CC[3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3], labels=c("A","B","C","D"), add=TRUE)

D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty=2)

plot3D::text3D(CC[1],CC[2],CC[3], labels="CC", add=TRUE)

dat.fr<-data.frame(a=tetra)
circ.cent.tetra(dat.fr)
```

`circ.cent.tri` *Circumcenter of a general triangle*

Description

Returns the circumcenter a given triangle, `tri`, with vertices stacked row-wise. See (Weisstein (2019); Ceyhan (2010)) for triangle centers.

Usage

```
circ.cent.tri(tri)
```

Arguments

`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

circumcenter of the triangle `tri`

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Weisstein EW (2019). "Triangle Centers." http://mathworld.wolfram.com/TriangleCenter.html. From MathWorld–A Wolfram Web Resource.

See Also

`circ.cent.bastri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);  #the vertices of the triangle Tr

CC<-circ.cent.tri(Tr)  #the circumcenter
CC

D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2; #midpoints of the edges
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],CC[1])
Ylim<-range(Tr[,2],CC[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

oldpar <- par(mfrow = c(1,2))
par(pty="s")
plot(A,asp=1,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(rbind(CC))
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.08,.08,.08,.12,-.09,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.03,-.06,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)
par(oldpar)
```

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);  #the vertices of the equilateral triangle Te
circ.cent.tri(Te)  #the circumcenter

A<-c(0,0); B<-c(0,1); C<-c(2,0);
Tr<-rbind(A,B,C);  #the vertices of the triangle T
circ.cent.tri(Tr)  #the circumcenter

dat.fr<-data.frame(a=Tr)
circ.cent.tri(dat.fr)
```

`cl2CC.TbVR` *The closest points to circumcenter in each CC-vertex region in a basic triangle*

Description

An object of class "Extrema". Returns the closest data points among the data set, `Dt`, to circumcenter, CC, in each CC-vertex region in the basic triangle $T_b = T(A = (0,0), B = (1,0), C = (c_1, c_2))$=(vertex 1,vertex 2,vertex 3). `ch.all.intri` is for checking whether all data points are inside T_b (default is `FALSE`).

See also (Ceyhan (2005, 2012)).

Usage

```
cl2CC.TbVR(Dt, c1, c2, ch.all.intri = FALSE)
```

Arguments

`Dt`	A set of 2D points representing the set of data points.
`c1, c2`	Positive real numbers which constitute the vertex of the basic triangle. adjacent to the shorter edges; `c1` must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$
`ch.all.intri`	A logical argument for checking whether all data points are inside T_b (default is `FALSE`).

Value

A list with the elements

`txt1`	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
`txt2`	A short description of the distances as "Distances to CC".
`type`	Type of the extrema points
`mtitle`	The "main" title for the plot of the exrema
`ext`	The extrema points, here, closest points to CC in each vertex region.

X	The input data, Dt, can be a matrix or data frame
num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, here, it is T_b.
cent	The center point used for construction of vertex regions
ncent	Name of the center, cent, it is "CC" for this function.
regions	Vertex regions inside the triangle, T_b, provided as a list.
region.names	Names of the vertex regions as "vr=1","vr=2","vr=3"
region.centers	Centers of mass of the vertes regions inside T_b.
dist2ref	Distances from closest points in each vertex region to CC.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

cl2CC.VR, cl2eTbVRcent, cl2eVRcent, cl2eVRCM and fr2eTeER

Examples

```
## Not run:
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-20

set.seed(1)
dat<-runif.bastri(n,c1,c2)$g

Ext<-cl2CC.TbVR(dat,c1,c2)
Ext
summary(Ext)
plot(Ext)

cl2CC.TbVR(dat[1,],c1,c2)

c2CC<-cl2CC.TbVR(dat,c1,c2)

CC<-circ.cent.bastri(c1,c2)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1],dat[,1])
```

```
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(dat)
points(c2CC$ext,pch=4,col=2)

txt<-rbind(Tb,CC,Ds)
xc<-txt[,1]+c(-.03,.03,.02,.07,.06,-.05,.01)
yc<-txt[,2]+c(.02,.02,.03,-.01,.03,.03,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

cl2CC.TbVR(dat,c1,c2)

cl2CC.TbVR(c(.4,.2),c1,c2)

dat.fr<-data.frame(a=dat)
cl2CC.TbVR(dat.fr,c1,c2)

dat2<-rbind(dat,c(.2,.4))
cl2CC.TbVR(dat2,c1,c2)

cl2CC.TbVR(dat2,c1,c2,ch.all.intri = TRUE)
#gives an error message since not all points are in the basic triangle

## End(Not run)
```

`cl2CC.VR` *The closest points to circumcenter in each CC-vertex region in a triangle*

Description

An object of class "Extrema". Returns the closest data points among the data set, `Dt`, to circumcenter, CC, in each CC-vertex region in the triangle $tri = T(A, B, C)$=(vertex 1,vertex 2,vertex 3).

`ch.all.intri` is for checking whether all data points are inside `tri` (default is `FALSE`). If some of the data points are not inside `tri` and `ch.all.intri=TRUE`, then the function yields an error message. If some of the data points are not inside `tri` and `ch.all.intri=FALSE`, then the function yields the closest points to CC among the data points in each CC-vertex region of `tri` (yields `NA` if there are no data points inside `tri`).

See also (Ceyhan (2005, 2012)).

Usage

```
cl2CC.VR(Dt, tri, ch.all.intri = FALSE)
```

Arguments

Dt	A set of 2D points representing the set of data points.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
ch.all.intri	A logical argument (default=FALSE) to check whether all data points are inside the triangle tri. So if it is TRUE, the function checks if all data points are inside the closure of the triangle (i.e. interior and boundary combined) else it does not.

Value

A list with the elements

txt1	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
txt2	A short description of the distances as "Distances to CC ..."
type	Type of the extrema points
mtitle	The "main" title for the plot of the exrema
ext	The extrema points, here, closest points to CC in each CC-vertex region
X	The input data, Dt, can be a matrix or data frame
num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, here, it is tri
cent	The center point used for construction of vertex regions
ncent	Name of the center, cent, it is "CC" for this function
regions	Vertex regions inside the triangle, tri, provided as a list
region.names	Names of the vertex regions as "vr=1","vr=2","vr=3"
region.centers	Centers of mass of the vertex regions inside tri
dist2ref	Distances from closest points in each CC-vertex region to CC.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

cl2CC.TbVR, cl2eTbVRcent, cl2eVRcent, cl2eVRCM and fr2eTeER

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
dat<-runif.tri(n,Tr)$g

Ext<-cl2CC.VR(dat,Tr)
Ext
summary(Ext)
plot(Ext)

cl2CC.VR(dat[1,],Tr)

c2CC<-cl2CC.VR(dat,Tr)

CC<-circ.cent.tri(Tr)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(c2CC$ext,pch=4,col=2)

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.07,.08,.06,.12,-.1,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.03,.0,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

cl2CC.VR(dat,Tr)

cl2CC.VR(c(1.4,1.2),Tr)

dat.fr<-data.frame(a=dat)
cl2CC.VR(dat.fr,Tr)

dat.fr<-data.frame(a=Tr)
cl2CC.VR(dat,dat.fr)

dat2<-rbind(dat,c(.2,.4))
cl2CC.VR(dat2,Tr)
## Not run:
```

```
cl2CC.VR(dat2,Tr,ch.all.intri = TRUE)
#gives an error message since not all points are in the triangle

## End(Not run)
```

`cl2edgesTe` *The closest points in a data set to edges in the standard equilateral triangle*

Description

An object of class "Extrema". Returns the closest points from the 2D data set, `Dt`, to the edges in the standard equilateral triangle $T_e = T(A = (0,0), B = (1,0), C = (1/2, \sqrt{3}/2))$.

`ch.all.intri` is for checking whether all data points are inside T_e (default is `FALSE`).

If some of the data points are not inside T_e and `ch.all.intri=TRUE`, then the function yields an error message. If some of the data points are not inside T_e and `ch.all.intri=FALSE`, then the function yields the closest points to edges among the data points inside T_e (yields `NA` if there are no data points inside T_e).

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan and Priebe (2007)).

Usage

```
cl2edgesTe(Dt, ch.all.intri = FALSE)
```

Arguments

`Dt`	A set of 2D points representing the set of data points.
`ch.all.intri`	A logical argument (default=`FALSE`) to check whether all data points are inside the standard equilateral triangle T_e. So if it is `TRUE`, the function checks if all data points are inside the closure of the triangle (i.e. interior and boundary combined) else it does not.

Value

A list with the elements

`txt1`	Edge labels as AB=3, BC=1, and AC=2 for T_e (corresponds to row number in Extrema Points).
`txt2`	A short description of the distances as "Distances to Edges ...".
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The "main" title for the plot of the exrema
`ext`	The extrema points, i.e., closest points to edges
`X`	The input data, `Dt`, which can be a matrix or data frame

num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, i.e., the standard equilateral triangle T_e
cent	The center point used for construction of edge regions, not required for this extrema, hence it is NULL for this function
ncent	Name of the center, cent, not required for this extrema, hence it is NULL for this function
regions	Edge regions inside the triangle, T_e, not required for this extrema, hence it is NULL for this function
region.names	Names of the edge regions, not required for this extrema, hence it is NULL for this function
region.centers	Centers of mass of the edge regions inside T_e, not required for this extrema, hence it is NULL for this function
dist2ref	Distances from closest points in each edge region to the corresponding edge

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

cl2eTbVRcent, cl2eVRcent, cl2eVRCM and fr2eTeER

Examples

```
n<-10  #try also n<-20
dat<-runifTe(n)$gen.points

Ext<-cl2edgesTe(dat)
Ext
summary(Ext)
plot(Ext,asp=1)

cl2edgesTe(dat[1,])
cl2edgesTe(c(10,10))

ed.clo<-cl2edgesTe(dat)

dat2<-rbind(dat,c(.8,.8))
cl2edgesTe(dat2)
```

```
A<-c(0,0); B<-c(1,0); C<-c(0.5,sqrt(3)/2);
Te<-rbind(A,B,C)
CM<-(A+B+C)/3
p1<-(A+B)/2
p2<-(B+C)/2
p3<-(A+C)/2

Xlim<-range(Te[,1],dat[,1])
Ylim<-range(Te[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(dat,xlab="",ylab="")
points(ed.clo$ext,pty=2,pch=4,col="red")

txt<-rbind(Te,p1,p2,p3)
xc<-txt[,1]+c(-.03,.03,.03,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,0,0,0)
txt.str<-c("A","B","C","re=1","re=2","re=3")
text(xc,yc,txt.str)

dat.fr<-data.frame(a=dat)
cl2edgesTe(dat.fr)
```

`cl2eTbVRcent` *The closest points among a data set in the vertex regions to the corresponding edges in a basic triangle*

Description

An object of class "Extrema". Returns the closest data points among the data set, `Dt`, to edge i in `M`-vertex region i for $i = 1, 2, 3$ in the basic triangle $T_b = T(A = (0, 0), B = (1, 0), C = (c_1, c_2))$ where `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$. Vertex labels are A=1, B=2, and C=3, and corresponding edge labels are BC=1, AC=2, and AB=3.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the basic triangle T_b or based on the circumcenter of T_b.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
cl2eTbVRcent(Dt, c1, c2, M)
```

Arguments

Dt	A set of 2D points representing the set of data points.
c1, c2	Positive real numbers which constitute the vertex of the basic triangle adjacent to the shorter edges; c1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the basic triangle T_b or the circumcenter of T_b.

Value

A list with the elements

txt1	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
txt2	A short description of the distances as "Distances to Edges in the Respective M-Vertex Regions".
type	Type of the extrema points
desc	A short description of the extrema points
mtitle	The "main" title for the plot of the exrema
ext	The extrema points, here, closest points to edges in the corresponding vertex region.
X	The input data, Dt, can be a matrix or data frame
num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, here, it is T_b.
cent	The center point used for construction of vertex regions
ncent	Name of the center, cent, it is "M" or "CC" for this function
regions	Vertex regions inside the triangle, T_b.
region.names	Names of the vertex regions as "vr=1","vr=2","vr=3"
region.centers	Centers of mass of the vertex regions inside T_b.
dist2ref	Distances of closest points in the vertex regions to corresponding edges.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

cl2eVRCM, cl2eVRcent, and cl2edgesTe

Examples

```
## Not run:
c1<-.4;  c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);

set.seed(1)
n<-20
dat<-runif.bastri(n,c1,c2)$g

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.3)

Ext<-cl2eTbVRcent(dat,c1,c2,M)
Ext
summary(Ext)
plot(Ext)

cl2eTbVRcent(dat[1,],c1,c2,M)
cl2eTbVRcent(c(1,2),c1,c2,M)

cl2e<-cl2eTbVRcent(dat,c1,c2,M)
cl2e

Ds<-cp2e.bastri(c1,c2,M)

Xlim<-range(Tb[,1],dat[,1])
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
points(dat,pch=1,col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(cl2e$ext,pch=3,col=2)

xc<-Tb[,1]+c(-.02,.02,0.02)
yc<-Tb[,2]+c(.02,.02,.02)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.04,-.03,0)
yc<-txt[,2]+c(-.02,.02,.02,-.03)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
```

```
cl2eTbVRcent(dat,c1,c2,M)

dat.fr<-data.frame(a=dat)
cl2eTbVRcent(dat.fr,c1,c2,M)

## End(Not run)
```

`cl2eVRCC` *The closest points in a data set to edges in each CC-vertex region in a triangle*

Description

An object of class "Extrema". Returns the closest data points among the data set, `dat`, to edge j in CC-vertex region j for $j = 1, 2, 3$ in the triangle, $tri = T(A, B, C)$, where CC stands for circumcenter. Vertex labels are A=1, B=2, and C=3, and corresponding edge labels are BC=1, AC=2, and AB=3. Function yields `NA` if there are no data points in a CC-vertex region.

See also (Ceyhan (2005, 2010)).

Usage

```
cl2eVRCC(dat, tri)
```

Arguments

`dat`	A set of 2D points representing the set of data points.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

A list with the elements

`txt1`	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
`txt2`	A short description of the distances as "Distances to Edges in the Respective CC-Vertex Regions".
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The "main" title for the plot of the exrema
`ext`	The extrema points, here, closest points to edges in the respective vertex region.
`X`	The input data, `dat`, can be a matrix or data frame
`num.points`	The number of data points, i.e., size of `dat`
`supp`	Support of the data points, here, it is `tri`

cent	The center point used for construction of vertex regions
ncent	Name of the center, cent, it is "CC" for this function
regions	Vertex regions inside the triangle, tri, provided as a list
region.names	Names of the vertex regions as "vr=1","vr=2","vr=3"
region.centers	Centers of mass of the vertex regions inside tri
dist2ref	Distances of closest points in the vertex regions to corresponding edges

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

cl2eTbVRcent, cl2eVRCM, cl2eVRcent, and cl2edgesTe

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

Ext<-cl2eVRCC(dat,Tr)
Ext
summary(Ext)
plot(Ext)

cl2eVRCC(dat[1,],Tr)

cl2e<-cl2eVRCC(dat,Tr)
cl2e

CC<-circ.cent.tri(Tr);
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],dat[,1],CC[1])
Ylim<-range(Tr[,2],dat[,2],CC[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,asp=1,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
```

```
xc<-Tr[,1]+c(-.02,.02,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

points(dat,pch=1,col=1)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(cl2e$ext,pch=3,col=2)

txt<-rbind(CC,Ds)
xc<-txt[,1]+c(-.04,.04,-.03,0)
yc<-txt[,2]+c(-.05,.04,.06,-.08)
txt.str<-c("CC","D1","D2","D3")
text(xc,yc,txt.str)

P<-c(1.4,1.0)
cl2eVRCC(P,Tr)
cl2eVRCC(dat,Tr)

cl2eVRCC(rbind(dat,dat),Tr)

dat.fr<-data.frame(a=dat)
cl2eVRCC(dat.fr,Tr)

dat.fr<-data.frame(a=Tr)
cl2eVRCC(dat,dat.fr)
```

`cl2eVRcent` *The closest points among a data set in the vertex regions to the respective edges in a triangle*

Description

An object of class "Extrema". Returns the closest data points among the data set, `Dt`, to edge i in `M`-vertex region i for $i = 1, 2, 3$ in the triangle $tri = T(A, B, C)$. Vertex labels are A=1, B=2, and C=3, and corresponding edge labels are BC=1, AC=2, and AB=3.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`.

See also (Ceyhan (2005, 2010)).

Usage

```
cl2eVRcent(Dt, tri, M)
```

Arguments

Dt	A set of 2D points representing the set of data points.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri.

Value

A list with the elements

txt1	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
txt2	A short description of the distances as "Distances to Edges in the Respective M-Vertex Regions".
type	Type of the extrema points
desc	A short description of the extrema points
mtitle	The "main" title for the plot of the exrema
ext	The extrema points, here, closest points to edges in the respective vertex region.
X	The input data, Dt, can be a matrix or data frame
num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, here, it is tri
cent	The center point used for construction of vertex regions
ncent	Name of the center, cent, it is "M" or "CC" for this function
regions	Vertex regions inside the triangle, tri, provided as a list
region.names	Names of the vertex regions as "vr=1","vr=2","vr=3"
region.centers	Centers of mass of the vertex regions inside tri
dist2ref	Distances of closest points in the M-vertex regions to corresponding edges.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

cl2eTbVRcent, cl2eVRCM, and cl2edgesTe

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

Ext<-cl2eVRcent(dat,Tr,M)
Ext
summary(Ext)
plot(Ext)

cl2eVRcent(dat[1,],Tr,M)
cl2e<-cl2eVRcent(dat,Tr,M)
cl2e

Ds<-cp2e.tri(Tr,M)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=1,col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(cl2e$ext,pch=3,col=2)

xc<-Tr[,1]+c(-.02,.03,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.05,-.02,-.01)
yc<-txt[,2]+c(-.03,.02,.08,-.07)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)

cl2eVRcent(dat,Tr,M)

dat.fr<-data.frame(a=dat)
```

```
cl2eVRcent(dat.fr,Tr,M)

dat.fr<-data.frame(a=Tr)
cl2eVRcent(dat,dat.fr,M)
```

`cl2eVRcent.alt` *An alternative function to the function* `cl2eVRcent` *which finds the closest points among a data set in the vertex regions to the respective edges in a triangle*

Description

An object of class "Extrema". An alternative function to the function `cl2eVRcent`

Usage

```
cl2eVRcent.alt(dat, tri, M)
```

Arguments

`dat`	A set of 2D points representing the set of data points.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`.

Value

A list with the elements

`txt1`	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
`txt2`	A short description of the distances as "Distances to Edges in the Respective M-Vertex Regions".
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The "main" title for the plot of the exrema
`ext`	The extrema points, here, closest points to edges in the respective vertex region.
`X`	The input data, `Dt`, can be a matrix or data frame
`num.points`	The number of data points, i.e., size of `Dt`
`supp`	Support of the data points, here, it is `tri`
`cent`	The center point used for construction of vertex regions

ncent	Name of the center, cent, it is "M" or "CC" for this function
regions	Vertex regions inside the triangle, tri, provided as a list
region.names	Names of the vertex regions as "vr=1","vr=2","vr=3"
region.centers	Centers of mass of the vertex regions inside tri
dist2ref	Distances of closest points in the M-vertex regions to corresponding edges.

See Also

cl2eVRcent

Examples

```
## Not run:
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
M<-c(1.6,1.0)  #try also M<-c(1.3,1.3)

n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

Ext<-cl2eVRcent.alt(dat,Tr,M)
Ext
summary(Ext)
plot(Ext)

cl2eVRcent.alt(dat[1,],Tr,M)

cl2e<-cl2eVRcent.alt(dat,Tr,M)
cl2e

Ds<-cp2e.tri(Tr,M)

Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=1,col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(cl2e$ext,pch=3,col=2)

xc<-Tr[,1]+c(-.02,.03,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)
```

```
txt<-rbind(M,Ds)
xc<-txt[,1]+c(.02,.04,-.03,0)
yc<-txt[,2]+c(.07,.04,.06,-.08)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)

dat.fr<-data.frame(a=dat)
cl2eVRcent.alt(dat.fr,Tr,M)

dat.fr<-data.frame(a=Tr)
cl2eVRcent.alt(dat,dat.fr,M)

## End(Not run)
```

`cl2eVRCM` *The closest points in a data set to edges in each CM-vertex region in a triangle*

Description

An object of class "Extrema". Returns the closest data points among the data set, `dat`, to edge j in CM-vertex region j for $j = 1, 2, 3$ in the triangle, $tri = T(A, B, C)$, where CM stands for center of mass. Vertex labels are A=1, B=2, and C=3, and corresponding edge labels are BC=1, AC=2, and AB=3. Function yields `NA` if there are no data points in a CM-vertex region.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2010, 2011)).

Usage

```
cl2eVRCM(dat, tri)
```

Arguments

`dat`	A set of 2D points representing the set of data points.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

A list with the elements

`txt1`	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
`txt2`	A short description of the distances as "Distances to Edges in the Respective CM-Vertex Regions".
`type`	Type of the extrema points
`desc`	A short description of the extrema points

mtitle	The "main" title for the plot of the exrema
ext	The extrema points, here, closest points to edges in the respective vertex region.
X	The input data, dat, can be a matrix or data frame
num.points	The number of data points, i.e., size of dat
supp	Support of the data points, here, it is tri
cent	The center point used for construction of vertex regions
ncent	Name of the center, cent, it is "CM" for this function
regions	Vertex regions inside the triangle, tri, provided as a list
region.names	Names of the vertex regions as "vr=1","vr=2","vr=3"
region.centers	Centers of mass of the vertex regions inside tri
dist2ref	Distances of closest points in the vertex regions to corresponding edges

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

cl2eTbVRcent, cl2eVRCC, cl2eVRcent, and cl2edgesTe

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

Ext<-cl2eVRCM(dat,Tr)
Ext
summary(Ext)
plot(Ext)

cl2eVRCM(dat[1,],Tr)
```

```
cl2e<-cl2eVRCM(dat,Tr)
cl2e

CM<-(A+B+C)/3;
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)

xc<-Tr[,1]+c(-.02,.02,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

points(dat,pch=1,col=1)
L<-matrix(rep(CM,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(cl2e$ext,pch=3,col=2)

txt<-rbind(CM,Ds)
xc<-txt[,1]+c(-.04,.04,-.03,0)
yc<-txt[,2]+c(-.05,.04,.06,-.08)
txt.str<-c("CM","D1","D2","D3")
text(xc,yc,txt.str)

P<-c(1.4,1.0)
cl2eVRCM(P,Tr)
cl2eVRCM(dat,Tr)

cl2eVRCM(rbind(dat,dat),Tr)

dat.fr<-data.frame(a=dat)
cl2eVRCM(dat.fr,Tr)

dat.fr<-data.frame(a=Tr)
cl2eVRCM(dat,dat.fr)
```

`cl2fVRth` *The closest points among a data set in the vertex regions to the respective faces in a tetrahedron*

Description

An object of class "Extrema". Returns the closest data points among the data set, `Dt`, to face i in `M`-vertex region i for $i = 1, 2, 3, 4$ in the tetrahedron $th = T(A, B, C, D)$. Vertex labels are A=1, B=2, C=3, and D=4 and corresponding face labels are BCD=1, ACD=2, ABD=3, and ABC=4.

Vertex regions are based on center `M` which can be the center of mass ("CM") or circumcenter ("CC") of `th`.

Usage

```
cl2fVRth(Dt, th, M = "CM")
```

Arguments

`Dt`	A set of 3D points representing the set of data points.
`th`	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes "CC" for circumcenter and "CM" for center of mass; default="CM".

Value

A list with the elements

`txt1`	Vertex labels are A=1, B=2, C=3, and D=4 (corresponds to row number in Extrema Points).
`txt2`	A short description of the distances as "Distances from Closest Points to Faces ...".
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The "main" title for the plot of the exrema
`ext`	The extrema points, here, closest points to faces in the respective vertex region.
`X`	The input data, `Dt`, can be a matrix or data frame
`num.points`	The number of data points, i.e., size of `Dt`
`supp`	Support of the data points, here, it is `th`
`cent`	The center point used for construction of vertex regions, it is circumcenter of center of mass for this function
`ncent`	Name of the center, it is circumcenter "CC" or center of mass "CM" for this function.
`regions`	Vertex regions inside the tetrahedron `th` provided as a list.
`region.names`	Names of the vertex regions as "vr=1","vr=2","vr=3","vr=4"
`region.centers`	Centers of mass of the vertex regions inside `th`.
`dist2ref`	Distances from closest points in each vertex region to the corresponding face.

See Also

fr2vVRCC, fr2eTeER, Kfr2vTbVRCC and Kfr2vVRCC

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
Cent<-"CC"  #try also "CM"

n<-10  #try also n<-20
Dt<-runif.tetra(n,tetra)$g  #try also Dt<-cbind(runif(n),runif(n),runif(n))

Ext<-cl2fVRth(Dt,tetra,Cent)
Ext
summary(Ext)
plot(Ext)

cl2fVRth(c(.5,.5,.5),tetra,Cent)

cl2fVRth(Dt,tetra,Cent)
clf<-cl2fVRth(Dt,tetra,Cent)$ext
clf

if (Cent=="CC") {M<-circ.cent.tetra(tetra)}
if (Cent=="CM") {M<-apply(tetra,2,mean)}

Xlim<-range(tetra[,1],Dt[,1],M[1])
Ylim<-range(tetra[,2],Dt[,2],M[2])
Zlim<-range(tetra[,3],Dt[,3],M[3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(Dt[,1],Dt[,2],Dt[,3], phi =0,theta=40, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
          pch = 20, cex = 1, ticktype = "detailed")
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)
plot3D::points3D(clf[,1],clf[,2],clf[,3], pch=4,col="red", add=TRUE)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3], labels=c("A","B","C","D"), add=TRUE)

#for center of mass use #Cent<-apply(tetra,2,mean)
D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-rbind(M,M,M,M,M,M)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty=2)

cl2fVRth(Dt,tetra,Cent)
```

```
P<-c(.1,.1,.1)
cl2fVRth(P,tetra,Cent)

dat.fr<-data.frame(a=Dt)
cl2fVRth(dat.fr,tetra,Cent)

dat.fr<-data.frame(a=tetra)
cl2fVRth(Dt,dat.fr,Cent)
```

`cl2Mc.int` *The closest points to center in each vertex region in an interval*

Description

An object of class "Extrema". Returns the closest data points among the data set, `Dt`, in each M_c-vertex region i.e., finds the closest points from right and left to M_c among points of the 1D data set `Dt` which reside in in the interval $int = (a, b)$.

M_c is based on the centrality parameter `c` in $(0, 1)$, so that $100c$ % of the length of interval is to the left of M_c and $100(1 - c)$ % of the length of the interval is to the right of M_c. That is, for the interval (a, b), $M_c = a + c(b - a)$. If there are no points from Dt to the left of M_c in the interval, then it yields NA, and likewise for the right of M_c in the interval.

See also (Ceyhan (2012)).

Usage

```
cl2Mc.int(Dt, int, c)
```

Arguments

`Dt`	A set or vector of 1D points from which closest points to M_c are found in the interval `int`.
`int`	A vector of two real numbers representing an interval.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

A list with the elements

`txt1`	Vertex Labels are a=1 and b=2 for the interval (a,b).
`txt2`	A short description of the distances as "Distances from ..."
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The "main" title for the plot of the exrema

ext	The extrema points, here, closest points to M_c in each vertex region
X	The input data vector, Dt.
num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, here, it is int.
cent	The (parameterized) center point used for construction of vertex regions.
ncent	Name of the (parameterized) center, cent, it is "Mc" for this function.
regions	Vertex regions inside the interval, int, provided as a list.
region.names	Names of the vertex regions as "vr=1","vr=2"
region.centers	Centers of mass of the vertex regions inside int.
dist2ref	Distances from closest points in each vertex region to M_c.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

cl2CC.TbVR and cl2CC.VR

Examples

```
c<-.4
a<-0; b<-10; int<-c(a,b)

Mc<-centMc(int,c)

nx<-10
xr<-range(a,b,Mc)
xf<-(xr[2]-xr[1])*.5

dat<-runif(nx,a,b)

Ext<-cl2Mc.int(dat,int,c)
Ext
summary(Ext)
plot(Ext)

cl2Mc.int(dat[1],int,c)
cMc<-cl2Mc.int(dat,int,c)

Xlim<-range(a,b,dat)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),xlab="",pch=".",xlim=Xlim+xd*c(-.05,.05))
abline(h=0)
abline(v=c(a,b,Mc),col=c(1,1,2),lty=2)
points(cbind(dat,0))
points(cbind(c(cMc$ext),0),pch=4,col=2)
```

```
text(cbind(c(a,b,Mc),-0.1),c("a","b","Mc"))

n<-10  #try also n<-20
dat<-runif(n,a-5,b+5)
cl2Mc.int(dat,c(a,b),c)

dat<-runif(n,a+b,b+10)
cl2Mc.int(dat,int,c)

c<-.4
a<-0; b<-10; int<-c(a,b)
n<-10
dat<-runif(n,a,b)
cl2Mc.int(dat,int,c)
```

`cp2e.bastri` *Projections of a point inside the basic triangle to its edges*

Description

Returns the projections from a general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ to the edges on the extension of the lines joining `M` to the vertices (see the examples for an illustration). In the basic triangle T_b, `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
cp2e.bastri(c1, c2, M)
```

Arguments

`c1, c2`	Positive real numbers which constitute the vertex of the basic triangle adjacent to the shorter edges; `c1` must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the basic triangle.

Value

Three projection points (stacked row-wise) from a general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of a basic triangle to the edges on the extension of the lines joining `M` to the vertices; row i is the projection point into edge i, for $i = 1, 2, 3$.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

cp2e.tri and cp2edges.nd

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.2)

cp2e.bastri(c1,c2,M)

Ds<-cp2e.bastri(c1,c2,M)

Xlim<-range(Tb[,1])
Ylim<-range(Tb[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tb)}
#need to run this when M is given in barycentric coordinates

plot(Tb,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.1,.1),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
L<-rbind(M,M,M); R<-Tb
segments(L[,1], L[,2], R[,1], R[,2], lty=3,col=2)

xc<-Tb[,1]+c(-.04,.05,.04)
yc<-Tb[,2]+c(.02,.02,.03)
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.03,-.03,0)
yc<-txt[,2]+c(-.02,.02,.02,-.03)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)
```

```
cp2e.bastri(c1,c2,M)
```

`cp2e.tri` *Projections of a point inside a triangle to its edges*

Description

Returns the projections from a general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of a triangle to the edges on the extension of the lines joining `M` to the vertices (see the examples for an illustration).

See also (Ceyhan (2005, 2010)).

Usage

```
cp2e.tri(tri, M)
```

Arguments

`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.

Value

Three projection points (stacked row-wise) from a general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of a triangle to the edges on the extension of the lines joining `M` to the vertices; row i is the projection point into edge i, for $i = 1, 2, 3$.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`cp2e.bastri` and `cp2edges.nd`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

cp2e.tri(Tr,M)  #try also cp2e.tri(Tr,M=c(1,1))

Ds<-cp2e.tri(Tr,M)

Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

xc<-Tr[,1]
yc<-Tr[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.04,-.04,-.02)
yc<-txt[,2]+c(-.02,.04,.04,-.06)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)

cp2e.tri(Tr,M)

dat.fr<-data.frame(a=Tr)
cp2e.tri(dat.fr,M)
```

`cp2edges.nd` *Projections of Centers for non-degenerate asymptotic distribution of domination number of Proportional Edge Proximity Catch Digraphs (PE-PCDs) to its edges*

Description

Returns the projections from center `cent` to the edges on the extension of the lines joining `cent` to the vertices in the triangle, `tri`. Here M is one of the three centers which gives nondegenerate

asymptotic distribution of the domination number of PE-PCD for uniform data in `tri` for a given expansion parameter `r` in $(1, 1.5]$. The center label `cent` values 1,2,3 correspond to the vertices M_1, M_2, and M_3 (i.e., row numbers in the output of cent.nondeg(tri,r)); default for `cent` is 1. `cent` becomes center of mass CM for $r = 1.5$.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011)).

Usage

```
cp2edges.nd(tri, r, cent = 1)
```

Arguments

`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$ for this function.
`cent`	Index of the center (as 1,2,3 corresponding to M1,M2,M3) which gives nondegenerate asymptotic distribution of the domination number of PE-PCD for uniform data in `tri` for expansion parameter `r` in $(1, 1.5]$; default `cent`=1.

Value

Three projection points (stacked row-wise) from one of the centers (as 1,2,3 corresponding to M1,M2,M3) which gives nondegenerate asymptotic distribution of the domination number of PE-PCD for uniform data in `tri` for expansion parameter `r` in $(1, 1.5]$.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

`cp2e.bastri` and `cp2e.tri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
r<-1.35

cp2edges.nd(Tr,r,cent=2)
```

```
Ms<-cent.nondeg(Tr,r)

Ds<-cp2edges.nd(Tr,r,cent=1)
D1<-Ds[1,]; D2<-Ds[2,]; D3<-Ds[3,]

Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(Ms,pch=".",col=1)
polygon(Ms,lty=2)

xc<-Tr[,1]
yc<-Tr[,2]
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

txt<-Ms
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("M1","M2","M3")
text(xc,yc,txt.str)

points(Ds,pch=4,col=2)
L<-Tr; R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2,col=2)
txt<-Ds
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("D1","D2","D3")
text(xc,yc,txt.str)

cp2edges.nd(Tr,r,1)

dat.fr<-data.frame(a=Tr)
cp2edges.nd(dat.fr,r,1)

## Not run:
cp2edges.nd(Tr,r,cent=5)
#gives an error message since center index, cent, must be 1, 2 or 3
cp2edges.nd(Tr,r=2,cent=2)
#gives an error message since r must be a scalar in (1,1.5]

T2<-rbind(A,A+2*(C-A),C)
cp2edges.nd(T2,r,cent=2)
#gives an error message since the triangle is degenerate

## End(Not run)
```

`CSarcdens.tri` *Arc density of Central Similarity Proximity Catch Digraphs (CS-PCDs) - one triangle case*

Description

Returns the arc density of CS-PCD whose vertex set is the given 2D numerical data set, `Xp`, (some of its members are) in the triangle `tri`.

CS proximity regions is defined with respect to `tri` with expansion parameter $t > 0$ and edge regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`. The function also provides arc density standardized by the mean and asymptotic variance of the arc density of CS-PCD for uniform data in the triangle `tri`. For the number of arcs, loops are not allowed.

`tri.cor` is a logical argument for triangle correction (default is `TRUE`), if `TRUE`, only the points inside the triangle are considered (i.e., digraph induced by these vertices are considered) in computing the arc density, otherwise all points are considered (for the number of vertices in the denominator of arc density).

Caveat: The standardized arc density is only correct when `M` is the center of mass in the current version.

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) for more on CS-PCDs.

Usage

```
CSarcdens.tri(Xp, tri, t, M = c(1, 1, 1), tri.cor = TRUE)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.
`tri.cor`	A logical argument for computing the arc density for only the points inside the triangle, `tri` (default=`TRUE`), i.e., if `TRUE` only the induced digraph with the vertices inside `tri` are considered in the computation of arc density.

Value

A list with the elements

arc.dens	Arc density of CS-PCD whose vertices are the 2D numerical data set, Xp; CS proximity regions are defined with respect to the triangle tri and M-edge regions
std.arc.dens	Arc density standardized by the mean and asymptotic variance of the arc density of CS-PCD for uniform data in the triangle tri.
caveat	The warning as "The standardized arc density is only correct when M is the center of mass in the current version".

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

ASarcdens.tri, PEarcdens.tri, and NumArcsCStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

NumArcsCStri(dat,Tr,t=.5,M)
CSarcdens.tri(dat,Tr,t=.5,M)
CSarcdens.tri(dat,Tr,t=.5,M,tri.cor = FALSE)

NumArcsCStri(dat,Tr,t=1,M)
CSarcdens.tri(dat,Tr,t=1,M)

NumArcsCStri(dat,Tr,t=1.5,M)
CSarcdens.tri(dat,Tr,t=1.5,M)

t<-2
CSarcdens.tri(dat,Tr,t,M)

dat.fr<-data.frame(a=dat)
CSarcdens.tri(dat.fr,Tr,t,M)

dat.fr<-data.frame(a=Tr)
```

```
CSarcdens.tri(dat,dat.fr,t,M)
```

`dimension` *The dimension of a vector or matrix or a data frame*

Description

Returns the dimension (i.e., # of columns) of `x`, which is a matrix or a vector or a data frame. This is different than the `dim` function in the base distribution of R, in the sense that, `dimension` gives only the number of columns of the argument `x`, while `dim` gives the number of rows and columns of `x`. `dimension` also works for a scalar or a vector, while `dim` yields `NULL` for such arguments.

Usage

```
dimension(x)
```

Arguments

`x`	A vector or a matrix or a data frame whose dimension is to be determined.

Value

Dimension (i.e., # of columns) of `x`

See Also

`is.point` and `dim` from the base distribution of R

Examples

```
dimension(3)
dim(3)

A<-c(1,2)
dimension(A)
dim(A)

B<-c(2,3)
dimension(rbind(A,B,A))
dimension(cbind(A,B,A))

M<-matrix(runif(20),ncol=5)
dimension(M)
dim(M)

dat.fr<-data.frame(a=A,b=B)
dimension(dat.fr)
dim(dat.fr)
```

```
dimension(c("a","b"))
```

`Dist` *The distance between two vectors, matrices, or data frames*

Description

Returns the Euclidean distance between `x` and `y` which can be vectors or matrices or data frames of any dimension (`x` and `y` should be of same dimension).

This function is different from the `dist` function in the `stats` package of the standard R distribution. `dist` requires its argument to be a data matrix and `dist` computes and returns the distance matrix computed by using the specified distance measure to compute the distances between the rows of a data matrix (Becker et al. (1988)), while `Dist` needs two arguments to find the distances between. For two data matrices A and B, `dist(rbind(as.vector(A),as.vector(B)))` and `Dist(A,B)` yield the same result.

Usage

```
Dist(x, y)
```

Arguments

`x, y`	Vectors, matrices or data frames (both should be of the same type).

Value

Euclidean distance between `x` and `y`

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Wadsworth \& Brooks/Cole.

See Also

`dist` from the base package `stats`

Examples

```
B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Dist(B,C);
dist(rbind(B,C))
dist(rbind(as.vector(B),as.vector(C)))
Dist(B,B);

x<-runif(10)
y<-runif(10)
Dist(x,y)
```

```
xm<-matrix(x,ncol=2)
ym<-matrix(y,ncol=2)
Dist(xm,ym)
dist(rbind(as.vector(xm),as.vector(ym)))

Dist(xm,xm)

dat.fr<-data.frame(b=B,c=C)
Dist(dat.fr,dat.fr)
Dist(dat.fr,cbind(B,C))
```

`dist.pt2set` *Distance from a point to a set of finite cardinality*

Description

Returns the Euclidean distance between a point `x` and set of points `Y` and the closest point in set `Y` to `x`. Distance between a point and a set is by definition the distance from the point to the closest point in the set. `x` should be of finite dimension and `Y` should be of finite cardinality and `x` and elements of `Y` must have the same dimension.

Usage

```
dist.pt2set(x, Y)
```

Arguments

`x`	A vector (i.e., a point in R^d).
`Y`	A set of d-dimensional points.

Value

A list with the elements

`distance`	Distance from point `x` to set `Y`
`ind.cl.point`	Index of the closest point in set `Y` to the point `x`
`closest.point`	The closest point in set `Y` to the point `x`

See Also

`dp2l` and `dp2pl`

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
dist.pt2set(c(1,2),Te)

X2<-cbind(runif(10),runif(10))
dist.pt2set(c(1,2),X2)

dist.pt2set(C,C)
dist.pt2set(B,C)

## Not run:
x<-runif(1)
y<-runif(10)
dist.pt2set(x,y)
#this does not work because both entries are treated as vectors

## End(Not run)

x<-runif(1)
y<-as.matrix(runif(10))
dist.pt2set(x,y) #this works, because x is a 1D point, and y is treated as a set of 10 1D points

dat.fr<-data.frame(b=B,c=C)
dist.pt2set(A,dat.fr)
```

`dom.exact` *Exact domination number (i.e. domination number by the exact algorithm)*

Description

Returns the (exact) domination number based on the incidence matrix `Inc.Mat` of a graph or a digraph and the indices (i.e. row numbers of `Inc.Mat`) for the corresponding (exact) minimum dominating set. Here the row number in the incidence matrix corresponds to the index of the vertex (i.e. index of the data point). The function works whether loops are allowed or not (i.e., whether the first diagonal is all 1 or all 0). It takes a rather long time for large number of vertices (i.e., large number of row numbers).

Usage

```
dom.exact(Inc.Mat)
```

Arguments

`Inc.Mat`	A square matrix consisting of 0's and 1's which represents the incidence matrix of a graph or digraph.

Value

A list with two elements

dom.num	The cardinality of the (exact) minimum dominating set i.e. (exact) domination number of the graph or digraph whose incidence matrix Inc.Mat is given as input.
ind.mds	Indices of the rows in the incidence matrix Inc.Mat for the (exact) minimum dominating set. The row numbers in the incidence matrix correspond to the indices of the vertices (i.e. indices of the data points).

See Also

dom.greedy, PEdom1D, PEdomtri, PEdomMTnd, and IndCSdomUBtri

Examples

```
n<-10
M<-matrix(sample(c(0,1),n^2,replace=TRUE),nrow=n)
diag(M)<-1

dom.greedy(M)
IndUBdom(M,2)
dom.exact(M)
```

---

dom.greedy *Approximate domination number and approximate dominating set by the greedy algorithm*

Description

Returns the (approximate) domination number and the indices (i.e. row numbers) for the corresponding (approximate) minimum dominating set based on the incidence matrix Inc.Mat of a graph or a digraph by using the greedy algorithm (Chvatal (1979)). Here the row number in the incidence matrix corresponds to the index of the vertex (i.e. index of the data point). The function works whether loops are allowed or not (i.e., whether the first diagonal is all 1 or all 0). This function may yield the actual domination number or overestimates it.

Usage

```
dom.greedy(Inc.Mat)
```

Arguments

Inc.Mat	A square matrix consisting of 0's and 1's which represents the incidence matrix of a graph or digraph.

Value

A list with two elements

dom.num	The cardinality of the (approximate) minimum dominating set found by the greedy algorithm. i.e. (approximate) domination number of the graph or digraph whose incidence matrix Inc.Mat is given as input.
ind.domset	Indices of the rows in the incidence matrix Inc.Mat for the ((approximate) minimum dominating set). The row numbers in the incidence matrix correspond to the indices of the vertices (i.e. indices of the data points).

References

Chvatal V (1979). "A greedy heuristic for the set-covering problem." *Mathematics of Operations Research*, **4**(3), 233–235.

See Also

dom.exact, PEdom1D, PEdomtri, PEdomMTnd, and IndCSdomUBtri

Examples

```
n<-10
M<-matrix(sample(c(0,1),n^2,replace=TRUE),nrow=n)
diag(M)<-1

dom.greedy(M)
```

---

dp2l *The distance from a point to a line*

Description

Returns the distance from a point p to the line joining points a and b in 2D space.

Usage

```
dp2l(p, a, b)
```

Arguments

p	A 2D point, distance from p to the line passing through points a and b are to be computed.
a, b	2D points that determine the straight line (i.e., through which the straight line passes).

Value

A list with two elements

dis	Distance from point p to the line passing through a and b
cl2p	The closest point on the line passing through a and b to the point p

See Also

dp2pl, dist.pt2set and Dist

Examples

```
A<-c(1,2); B<-c(2,3); P<-c(3,1.5)

dpl<-dp2l(P,A,B);
dpl
C<-dpl$cl2p
pts<-rbind(A,B,C,P)

dp2l(A,A,B);
dp2l((A+B)/2,A,B);

xr<-range(pts[,1])
xf<-(xr[2]-xr[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=20)  #try also l=100
lnAB<-Line(A,B,x)
y<-lnAB$y
int<-lnAB$intercept #intercept
sl<-lnAB$slope #slope

xsq<-seq(min(A[1],B[1],P[1])-xf,max(A[1],B[1],P[1])+xf,l=20)  #try also l=100
pline<-(-1/sl)*(xsq-P[1])+P[2] #line passing thru P and perpendicular to AB

Xlim<-range(pts[,1],x)
Ylim<-range(pts[,2],y)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(rbind(P),asp=1,pch=1,xlab="x",ylab="y",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(rbind(A,B),pch=1)
lines(x,y,lty=1,xlim=Xlim,ylim=Ylim)
int<-round(int,2);sl<-round(sl,2)
text(rbind((A+B)/2+xd*c(-.01,-.01)),ifelse(sl==0,paste("y=",int),
ifelse(sl==1,paste("y=x+",int),
ifelse(int==0,paste("y=",sl,"x"),paste("y=",sl,"x+",int)))))
text(rbind(A+xd*c(0,-.01),B+xd*c(.0,-.01),P+xd*c(.01,-.01)),c("A","B","P"))
lines(xsq,pline,lty=2)
segments(P[1],P[2], C[1], C[2], lty=1,col=2,lwd=2)
text(rbind(C+xd*c(-.01,-.01)),"C")

text(rbind((P+C)/2),col=2,paste("d=",round(dpl$dis,2)))
```

```
A<-c(1,2); B<-c(1,3)
P<-c(3,1.5)
dp2l(P,A,B);

A<-c(1,2); B<-c(2,2)
P<-c(3,1.5)
dp2l(P,A,B);
```

---

dp2pl *The distance from a point to a plane*

Description

Returns the distance from a point p to the plane passing through points a, b, and c in 3D space.

Usage

```
dp2pl(p, a, b, c)
```

Arguments

p	A 3D point, distance from p to the plane passing through points a, b, and c are to be computed.
a, b, c	3D points that determine the plane (i.e., through which the plane is passing).

Value

A list with two elements

dis	Distance from point p to the plane spanned by 3D points a, b, and c
cl2p	The closest point on the plane spanned by 3D points a, b, and c to the point p

See Also

dp2l, dist.pt2set and Dist

Examples

```
A<-c(1,2,3); B<-c(3,9,12); C<-c(1,1,3); P<-c(5,2,40)

dis<-dp2pl(P,A,B,C);
dis
Pr<-dis$Pr2pl #projection on the plane

P2<-c(0,0,0)
dp2pl(P2,A,B,C);
```

```
dp2pl(A,A,B,C);

xseq<-seq(0,10,l=20)  #try also l=100
yseq<-seq(0,10,l=20)  #try also l=100

pl.grid<-Plane(A,B,C,xseq,yseq)$z

plot3D::persp3D(z = pl.grid, x = xseq, y = yseq, theta =225, phi = 30, ticktype = "detailed")
#plane spanned by points A, B, C
#add the vertices of the tetrahedron
plot3D::points3D(P[1],P[2],P[3], add=TRUE)
plot3D::points3D(Pr[1],Pr[2],Pr[3], add=TRUE)
plot3D::segments3D(P[1], P[2], P[3], Pr[1], Pr[2],Pr[3], add=TRUE,lwd=2)

plot3D::text3D(P[1]-.5,P[2],P[3]+1, c("P"),add=TRUE)
plot3D::text3D(Pr[1]-.5,Pr[2],Pr[3]+2, c("Pr"),add=TRUE)

persp(xseq,yseq,pl.grid, xlab="x",ylab="y",zlab="z",theta = -30,
phi = 30, expand = 0.5, col = "lightblue",
      ltheta = 120, shade = 0.05, ticktype = "detailed")

dp2pl(P,A,B,C)
```

`fr2eTeER` *The furthest points in a data set from edges in each CM-edge region in the standard equilateral triangle*

Description

An object of class "Extrema". Returns the furthest data points among the data set, `Dt`, in each CM-edge region from the edge in the standard equilateral triangle $T_e = T(A = (0,0), B = (1,0), C = (1/2, \sqrt{3}/2))$.

`ch.all.intri` is for checking whether all data points are inside T_e (default is `FALSE`).

See also (Ceyhan (2005)).

Usage

```
fr2eTeER(Dt, ch.all.intri = FALSE)
```

Arguments

`Dt`	A set of 2D points, some could be inside and some could be outside standard equilateral triangle T_e.
`ch.all.intri`	A logical argument used for checking whether all data points are inside T_e (default is `FALSE`).

Value

A list with the elements

txt1	Edge labels as AB=3, BC=1, and AC=2 for T_e (corresponds to row number in Extrema Points).
txt2	A short description of the distances as "Distances to Edges".
type	Type of the extrema points
desc	A short description of the extrema points
mtitle	The "main" title for the plot of the exrema
ext	The extrema points, here, furthest points from edges in each edge region.
X	The input data, Dt, can be a matrix or data frame
num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, here, it is T_e.
cent	The center point used for construction of edge regions.
ncent	Name of the center, cent, it is center of mass "CM" for this function.
regions	Edge regions inside the triangle, T_e, provided as a list.
region.names	Names of the edge regions as "er=1","er=2","er=3".
region.centers	Centers of mass of the edge regions inside T_e.
dist2ref	Distances from furthest points in each edge region to the corresponding edge.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

See Also

fr2vTbVRCC, fr2vVRCC, Kfr2vTbVRCC, Kfr2vVRCC, and cl2edgesTe

Examples

```
## Not run:
n<-20
dat<-runifTe(n)$gen.points

Ext<-fr2eTeER(dat)
Ext
summary(Ext)
plot(Ext,asp=1)

fr2eTeER(dat[1,])
ed.far<-fr2eTeER(dat)

dat2<-rbind(dat,c(.8,.8))
fr2eTeER(dat2)
```

```
fr2eTeER(dat2,ch.all.intri = TRUE)

A<-c(0,0); B<-c(1,0); C<-c(0.5,sqrt(3)/2);
Te<-rbind(A,B,C)
CM<-(A+B+C)/3
p1<-(A+B)/2
p2<-(B+C)/2
p3<-(A+C)/2

Xlim<-range(Te[,1],dat[,1])
Ylim<-range(Te[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-Te; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(dat,xlab="",ylab="")
points(ed.far$ext,pty=2,pch=4,col="red")

txt<-rbind(Te,CM,p1,p2,p3)
xc<-txt[,1]+c(-.03,.03,.03,-.06,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,.02,0,0,0)
txt.str<-c("A","B","C","CM","re=2","re=3","re=1")
text(xc,yc,txt.str)

## End(Not run)
```

`fr2vTbVRCC` *The furthest points from vertices in each CC-vertex region in a basic triangle*

Description

An object of class "Extrema". Returns the furthest data points among the data set, `Dt`, in each CC-vertex region from the corresponding vertex in the basic triangle $T_b = T(A = (0,0), B = (1,0), C = (c_1, c_2))$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

`ch.all.intri` is for checking whether all data points are inside T_b (default is `FALSE`).

See also (Ceyhan (2005, 2012)).

Usage

```
fr2vTbVRCC(Dt, c1, c2, ch.all.intri = FALSE)
```

Arguments

Dt	A set of 2D points.
c1, c2	Positive real numbers which constitute the vertex of the basic triangle. adjacent to the shorter edges; c1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$
ch.all.intri	A logical argument for checking whether all data points are inside T_b (default is FALSE).

Value

A list with the elements

txt1	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
txt2	A short description of the distances as "Distances to Vertices".
type	Type of the extrema points
desc	A short description of the extrema points
mtitle	The "main" title for the plot of the exrema
ext	The extrema points, here, furthest points from vertices in each vertex region.
X	The input data, Dt, can be a matrix or data frame
num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, here, it is T_b.
cent	The center point used for construction of edge regions.
ncent	Name of the center, cent, it is circumcenter "CC" for this function.
regions	Vertex regions inside the triangle, T_b, provided as a list.
region.names	Names of the vertex regions as "vr=1","vr=2","vr=3"
region.centers	Centers of mass of the vertex regions inside T_b.
dist2ref	Distances from furthest points in each vertex region to the corresponding vertex.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

fr2vVRCC, fr2eTeER, Kfr2vTbVRCC and Kfr2vVRCC

Examples

```
## Not run:
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-20

set.seed(1)
dat<-runif.bastri(n,c1,c2)$g

Ext<-fr2vTbVRCC(dat,c1,c2)
Ext
summary(Ext)
plot(Ext)

fr2vTbVRCC(dat[1,],c1,c2)

dat2<-rbind(dat,c(.2,.4))
fr2vTbVRCC(dat2,c1,c2)

f2v<-fr2vTbVRCC(dat,c1,c2)

CC<-circ.cent.bastri(c1,c2)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1],dat[,1])
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(dat)
points(rbind(f2v$ext),pch=4,col=2)

txt<-rbind(Tb,CC,D1,D2,D3)
xc<-txt[,1]+c(-.03,.03,0.02,.07,.06,-.05,.01)
yc<-txt[,2]+c(.02,.02,.03,.01,.02,.02,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

set.seed(1)
dat<-runif.bastri(n,c1,c2)$g
fr2vTbVRCC(dat,c1,c2)

fr2vTbVRCC(c(.4,.2),c1,c2)

dat.fr<-data.frame(a=dat)
fr2vTbVRCC(dat.fr,c1,c2)
```

```
## End(Not run)
```

`fr2vVRCC` *The furthest points in a data set from vertices in each CC-vertex region in a triangle*

Description

An object of class "Extrema". Returns the furthest data points among the data set, `Dt`, in each CC-vertex region from the vertex in the triangle, $tri = T(A, B, C)$. Vertex region labels/numbers correspond to the row number of the vertex in `tri`. `ch.all.intri` is for checking whether all data points are inside `tri` (default is `FALSE`).

If some of the data points are not inside `tri` and `ch.all.intri=TRUE`, then the function yields an error message. If some of the data points are not inside `tri` and `ch.all.intri=FALSE`, then the function yields the closest points to edges among the data points inside `tri` (yields `NA` if there are no data points inside `tri`).

See also (Ceyhan (2005, 2012)).

Usage

```
fr2vVRCC(Dt, tri, ch.all.intri = FALSE)
```

Arguments

`Dt`	A set of 2D points representing the set of data points.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`ch.all.intri`	A logical argument (default=`FALSE`) to check whether all data points are inside the triangle `tri`. So if it is `TRUE`, the function checks if all data points are inside the closure of the triangle (i.e. interior and boundary combined) else it does not.

Value

A list with the elements

`txt1`	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
`txt2`	A short description of the distances as "Distances to Vertices...".
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The "main" title for the plot of the exrema
`ext`	The extrema points, here, furthest points from vertices in each CC-vertex region in the triangle `tri`.

X	The input data, Dt, can be a matrix or data frame
num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, here, it is the triangle tri for this function.
cent	The center point used for construction of edge regions.
ncent	Name of the center, cent, it is circumcenter "CC" for this function
regions	CC-Vertex regions inside the triangle, tri, provided as a list
region.names	Names of the vertex regions as "vr=1","vr=2","vr=3"
region.centers	Centers of mass of the vertex regions inside tri
dist2ref	Distances from furthest points in each vertex region to the corresponding vertex

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

fr2vTbVRCC, fr2eTeER, Kfr2vTbVRCC and Kfr2vVRCC

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
dat<-runif.tri(n,Tr)$g

Ext<-fr2vVRCC(dat,Tr)
Ext
summary(Ext)
plot(Ext)

fr2vVRCC(dat[1,],Tr)
f2v<-fr2vVRCC(dat,Tr)

CC<-circ.cent.tri(Tr)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
```

```
plot(Tr,xlab="",asp=1,ylab="",pch=".",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(dat)
points(rbind(f2v$ext),pch=4,col=2)

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.06,.08,.05,.12,-.1,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.05,.0,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

fr2vVRCC(dat,Tr)

fr2vVRCC(c(1.4,1.2),Tr)

dat.fr<-data.frame(a=dat)
fr2vVRCC(dat.fr,Tr)

dat.fr<-data.frame(a=Tr)
fr2vVRCC(dat,dat.fr)

dat2<-rbind(dat,c(.2,.4))
fr2vVRCC(dat2,Tr)
## Not run:
fr2vVRCC(dat2,Tr,ch.all.intri = TRUE)
#gives an error message since not all points in the data set are in the triangle

## End(Not run)
```

`funsAB2CMTe` *The lines joining two vertices to the center of mass in standard equilateral triangle*

Description

Two functions, `lA_CM.Te` and `lB_CM.Te` of class "TriLines". Returns the `equation`, `slope`, `intercept`, and y-coordinates of the lines joining A and CM and also B and CM.

`lA_CM.Te` is the line joining A to the center of mass, CM, and

`lB_CM.Te` is the line joining B to the center of mass, CM, in the standard equilateral triangle $T_e = (A, B, C)$ with $A = (0, 0)$; $B = (1, 0)$; $C = (1/2, \sqrt{3}/2)$; x-coordinates are provided in vector `x`.

Usage

```
lA_CM.Te(x)

lB_CM.Te(x)
```

Arguments

x	A single scalar or a vector of scalars which is the argument of the functions lA_CM.Te and lB_CM.Te.

Value

A list with the elements

txt1	Longer description of the line.
txt2	Shorter description of the line (to be inserted over the line in the plot).
mtitle	The "main" title for the plot of the line.
cent	The center chosen inside the standard equilateral triangle.
cent.name	The name of the center inside the standard equilateral triangle. It is "CM" for these two functions.
tri	The triangle (it is the standard equilateral triangle for this function).
x	The input vector, can be a scalar or a vector of scalars, which constitute the x-coordinates of the point(s) of interest on the line.
y	The output vector, will be a scalar if x is a scalar or a vector of scalars if x is a vector of scalar, constitutes the y-coordinates of the point(s) of interest on the line.
slope	Slope of the line.
intercept	Intercept of the line.
equation	Equation of the line.

See Also

lA_M.Te, lB_M.Te, and lC_M.Te

Examples

```
#Examples for lA_CM.Te
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.1)  #try also by=.01

lnACM<-lA_CM.Te(x)
lnACM
summary(lnACM)
plot(lnACM)

CM<-(A+B+C)/3;
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Te[,1])
```

```
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-Te; R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Te,CM,D1,D2,D3,c(.25,lA_CM.Te(.25)$y),c(.75,lB_CM.Te(.75)$y))
xc<-txt[,1]+c(-.02,.02,.02,.05,.05,-.03,.0,0,0)
yc<-txt[,2]+c(.02,.02,.02,.02,0,.02,-.04,0,0)
txt.str<-c("A","B","C","CM","D1","D2","D3","lA_CM.Te(x)","lB_CM.Te(x)")
text(xc,yc,txt.str)

lA_CM.Te(.25)$y

## End(Not run)

#Examples for lB_CM.Te
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.1)  #try also by=.01

lnBCM<-lB_CM.Te(x)
lnBCM
summary(lnBCM)
plot(lnBCM,xlab="x",ylab="y")

lB_CM.Te(.25)$y

## End(Not run)
```

---

funsAB2MTe *The lines joining the three vertices of the standard equilateral triangle to a center,* M*, of it*

Description

Three functions, `lA_M.Te`, `lB_M.Te` and `lC_M.Te` of class "TriLines". Returns the `equation`, `slope`, `intercept`, and y-coordinates of the lines joining A and `M`, B and `M`, and also C and `M`.

`lA_M.Te` is the line joining A to the center, `M`, `lB_M.Te` is the line joining B to `M`, and `lC_M.Te` is the line joining C to `M`, in the standard equilateral triangle $T_e = (A, B, C)$ with $A = (0, 0)$; $B = (1, 0)$; $C = (1/2, sqrt3/2)$; x-coordinates are provided in vector `x`

Usage

```
lA_M.Te(x, M)

lB_M.Te(x, M)

lC_M.Te(x, M)
```

Arguments

x	A single scalar or a vector of scalars.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle.

Value

A list with the elements

txt1	Longer description of the line.
txt2	Shorter description of the line (to be inserted over the line in the plot).
mtitle	The "main" title for the plot of the line.
cent	The center chosen inside the standard equilateral triangle.
cent.name	The name of the center inside the standard equilateral triangle.
tri	The triangle (it is the standard equilateral triangle for this function).
x	The input vector, can be a scalar or a vector of scalars, which constitute the x-coordinates of the point(s) of interest on the line.
y	The output vector, will be a scalar if x is a scalar or a vector of scalars if x is a vector of scalar, constitutes the y-coordinates of the point(s) of interest on the line.
slope	Slope of the line.
intercept	Intercept of the line.
equation	Equation of the line.

See Also

lA_CM.Te and lB_CM.Te

Examples

```
#Examples for lA_M.Te
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)

M<-c(.65,.2)  #try also M<-c(1,1,1)

xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
```

```
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.1)  #try also by=.01

lnAM<-lA_M.Te(x,M)
lnAM
summary(lnAM)
plot(lnAM)

Ds<-cp2e.tri(Te,M)
#finds the projections from a point M=(m1,m2) to the edges on the
#extension of the lines joining M to the vertices in the triangle Te

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,pch=".",xlab="",ylab="",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-Te; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
L<-Ds; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty=3,col=2)

txt<-rbind(Te,M,Ds,c(.25,lA_M.Te(.25,M)$y),c(.4,lB_M.Te(.4,M)$y),
c(.60,lC_M.Te(.60,M)$y))
xc<-txt[,1]+c(-.02,.02,.02,.02,.04,-.03,.0,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,.05,.02,.03,-.03,0,0,0)
txt.str<-c("A","B","C","M","D1","D2","D3","lA_M.Te(x)","lB_M.Te(x)","lC_M.Te(x)")
text(xc,yc,txt.str)

lA_M.Te(.25,M)

## End(Not run)

#Examples for lB_M.Te
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)

M<-c(.65,.2)  #try also M<-c(1,1,1)

xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.1)  #try also by=.01

lnBM<-lB_M.Te(x,M)
lnBM
summary(lnBM)
plot(lnBM)

lB_M.Te(.25,M)

## End(Not run)
```

```
#Examples for lC_M.Te
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)

M<-c(.65,.2)  #try also M<-c(1,1,1)

xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.1)  #try also by=.01

lnCM<-lC_M.Te(x,M)
lnCM
summary(lnCM)
plot(lnCM)

lC_M.Te(.25,M)

## End(Not run)
```

`funsCartBary` *Converts of a point in Cartesian coordinates to Barycentric coordinates and vice versa*

Description

Two functions: `cart2bary` and `bary2cart`.

`cart2bary` converts Cartesian coordinates of a given point `P`=(x,y) to barycentric coordinates (in the normalized form) with respect to the triangle `tri`=(v_1, v_2, v_3) with vertex labeling done row-wise in `tri` (i.e., row i corresponds to vertex v_i for $i = 1, 2, 3$).

`bary2cart` converts barycentric coordinates of the point `P`=(t_1, t_2, t_3) (not necessarily normalized) to Cartesian coordinates according to the coordinates of the triangle, `tri`. For information on barycentric coordinates, see (Weisstein (2019)).

Usage

```
cart2bary(P, tri)

bary2cart(P, tri)
```

Arguments

`P`	A 2D point for `cart2bary`, and a vector of three numeric entries for `bary2cart`.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

cart2bary returns the barycentric coordinates of a given point P=(x, y) and bary2cart returns the Cartesian coordinates of the point P=(t_1, t_2, t_3) (not necessarily normalized)

References

Weisstein EW (2019). "Barycentric Coordinates." http://mathworld.wolfram.com/BarycentricCoordinates.html. From MathWorld–A Wolfram Web Resource.

Examples

```
#Examples for cart2bary
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tr<-rbind(A,B,C)

cart2bary(A,Tr)
cart2bary(c(.3,.2),Tr)
cart2bary(c(.4,.2),Tr)
cart2bary(c(.5,.2),Tr)
cart2bary(c(.6,.2),Tr)

P<-c(.8,.2)
round(cart2bary(P,Tr),2)

P<-c(.5,.61)
cart2bary(P,Tr)

CM<-(A+B+C)/3
cart2bary(CM,Tr)

A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
cart2bary(c(1.4,1.2),Tr)

cart2bary(c(.8,.2),Tr)

cart2bary(c(1.5,1.61),Tr)

dat.fr<-data.frame(a=Tr)
cart2bary(c(.8,.2),dat.fr)

#Examples for bary2cart
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tr<-rbind(A,B,C)

bary2cart(c(.3,.2,.5),Tr)
bary2cart(c(.4,.2,.4),Tr)
bary2cart(c(.5,.2,.3),Tr)
bary2cart(c(6,2,4),Tr)
```

```
P<-c(.8,.2,.3)
bary2cart(P,Tr)

P<-c(-.5,.4,.2)
bary2cart(P,Tr)

CM<-(A+B+C)/3; CM
bary2cart(c(1,1,1),Tr)

A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
bary2cart(c(1.4,1.2,1),Tr)

bary2cart(c(.8,.2,.6),Tr)

bary2cart(c(1,2,3),Tr)

dat.fr<-data.frame(a=Tr)
bary2cart(c(.8,.2,.3),dat.fr)
```

`funsCSEdgeRegs` *Each function is for the presence of an arc from a point in one of the edge regions to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Three indicator functions: `IndCSTeRAB`, `IndCSTeRBC` and `IndCSTeRAC`.

The function `IndCSTeRAB` returns I(`pt2` is in $NCS(pt1, t)$ for `pt1` in RAB (edge region for edge AB, i.e., edge 3) in the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$;

`IndCSTeRBC` returns I(`pt2` is in $NCS(pt1, t)$ for `pt1` in RBC (edge region for edge BC, i.e., edge 1) in T_e; and

`IndCSTeRAC` returns I(`pt2` is in $NCS(pt1, t)$ for `pt1` in RAC (edge region for edge AC, i.e., edge 2) in T_e. That is, each function returns 1 if `pt2` is in $NCS(pt1, t)$, returns 0 otherwise.

CS proximity region is defined with respect to T_e whose vertices are also labeled as $T_e = T(v = 1, v = 2, v = 3)$ with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e

If `pt1` and `pt2` are distint and `pt1` is outside the corresponding edge region and `pt2` is outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their location (i.e., it allows loops).

See also (Ceyhan (2005, 2010)).

Usage

```
IndCSTeRAB(pt1, pt2, t, M)

IndCSTeRBC(pt1, pt2, t, M)
```

```
IndCSTeRAC(pt1, pt2, t, M)
```

Arguments

pt1	A 2D point whose CS proximity region is constructed.
pt2	A 2D point. The function determines whether pt2 is inside the CS proximity region of pt1 or not.
t	A positive real number which serves as the expansion parameter in CS proximity region.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e.

Value

Each function returns I(pt2 is in $NCS(pt1, t)$) for pt1, that is, returns 1 if pt2 is in $NCS(pt1, t)$, returns 0 otherwise

See Also

IndCSTeRABt1, IndCSTeRBCt1 and IndCSTeRACt1

Examples

```
#Examples for IndCSTeRAB
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T3<-rbind(A,B,CM);

set.seed(1)
dat3<-runif.tri(10,T3)$g
dat<-runifTe(10)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

t<-1

IndCSTeRAB(dat3[1,],dat[2,],t,M)
IndCSTeRAB(dat3[1,],dat3[1,],t,M)
IndCSTeRAB(dat[1,],dat[1,],t,M)

IndCSTeRAB(dat3[1,],dat[2,],t=20,M)
IndCSTeRAB(dat3[1,],dat3[3,],t,M)
IndCSTeRAB(c(.2,.5),dat[2,],t,M)

IndCSTeRAB(dat[1,],dat[2,],t,M)

## End(Not run)

#Examples for IndCSTeRBC
```

```
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T1<-rbind(B,C,CM);

set.seed(1)
dat1<-runif.tri(10,T1)$g
dat<-runifTe(10)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

t<-1

IndCSTeRBC(dat1[1,],dat[2,],t,M)
IndCSTeRBC(dat1[1,],dat1[1,],t,M)

IndCSTeRBC(dat[2,],dat[2,],t,M)
IndCSTeRBC(dat1[1,],dat[2,],t=20,M)
IndCSTeRBC(dat1[1,],dat1[3,],t,M)
IndCSTeRBC(c(.2,.5),dat[2,],t,M)

IndCSTeRBC(c(.2,.5),c(.2,.5),t,M)

IndCSTeRBC(dat[1,],dat[2,],t,M)

## End(Not run)

#Examples for IndCSTeRAC
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T2<-rbind(A,C,CM);

set.seed(1)
dat2<-runif.tri(10,T2)$g
dat<-runifTe(10)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

t<-1

IndCSTeRAC(dat2[1,],dat[2,],t,M)
IndCSTeRAC(dat2[1,],dat2[1,],t,M)
IndCSTeRAC(dat[1,],dat[1,],t,M)

IndCSTeRAC(dat2[1,],dat2[3,],t,M)
IndCSTeRAC(dat2[1,],dat2[3,],t=20,M)
IndCSTeRAC(c(.2,.5),dat[2,],t,M)

IndCSTeRAC(c(.2,.5),c(.2,.5),t,M)

IndCSTeRAC(dat[1,],dat[2,],t,M)
```

```
## End(Not run)
```

`funsCSGamTe` *The function* `GamkCSTe` *is for* k *(*$k = 2, 3, 4, 5$*) points constituting a dominating set for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Four indicator functions: `Gam2CSTe`, `Gam3CSTe`, `Gam4CSTe`, `Gam5CSTe` and `Gam6CSTe`.

The function `GamkCSTe` returns I({`pt1`,...,`ptk`} is a dominating set of the CS-PCD) where vertices of CS-PCD are the 2D data set `Dt`, that is, returns 1 if {`pt1`,...,`ptk`} is a dominating set of CS-PCD, returns 0 otherwise for $k = 2, 3, 4, 5, 6$.

CS proximity region is constructed with respect to $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $t > 0$ and edge regions are based on center of mass $CM = (1/2, \sqrt{3}/6)$.

`ch.data.pnts` is for checking whether points `pt1`,...,`ptk` are data points in `Dt` or not (default is `FALSE`), so by default this function checks whether the points `pt1`,...,`ptk` would be a dominating set if they actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam2CSTe(pt1, pt2, Dt, t, ch.data.pnts = FALSE)

Gam3CSTe(pt1, pt2, pt3, Dt, t, ch.data.pnts = FALSE)

Gam4CSTe(pt1, pt2, pt3, pt4, Dt, t, ch.data.pnts = FALSE)

Gam5CSTe(pt1, pt2, pt3, pt4, pt5, Dt, t, ch.data.pnts = FALSE)

Gam6CSTe(pt1, pt2, pt3, pt4, pt5, pt6, Dt, t, ch.data.pnts = FALSE)
```

Arguments

`pt1, pt2, pt3, pt4, pt5, pt6`	The points $\{pt1, ..., ptk\}$ are k 2D points (for $k = 2, 3, 4, 5, 6$) to be tested for constituting a dominating set of the CS-PCD.
`Dt`	A set of 2D points which constitutes the vertices of the CS-PCD.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`ch.data.pnts`	A logical argument for checking whether points $\{pt1, ..., ptk\}$ are data points in `Dt` or not (default is `FALSE`).

Value

The function GamkCSTe returns {pt1,...,ptk} is a dominating set of the CS-PCD) where vertices of the CS-PCD are the 2D data set Dt), that is, returns 1 if {pt1,...,ptk} is a dominating set of CS-PCD, returns 0 otherwise.

See Also

Gam1CSTe, Gam2PEtri and Gam2PEtetra

Examples

```
#Examples for Gam2CSTe
## Not run:
t<-1.5
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

Gam2CSTe(dat[1,],dat[2,],dat,t)
Gam2CSTe(dat[1,],dat[1,],dat,t)

Gam2CSTe(dat[1,],dat[3,],dat,t)

Gam2CSTe(c(.2,.2),dat[2,],dat,t)

Gam2CSTe(c(.2,.2),c(.2,.3),dat,t)

Gam2CSTe(c(.2,.2),c(.2,.3),rbind(c(.2,.2),c(.2,.3)),t)
Gam2CSTe(c(1.2,1.2),c(1.2,1.3),rbind(c(1.2,1.2),c(1.2,1.3)),t)

ind.gam2<-vector()
for (i in 1:(n-1))
 for (j in (i+1):n)
 {if (Gam2CSTe(dat[i,],dat[j,],dat,t)==1)
   ind.gam2<-rbind(ind.gam2,c(i,j))}

ind.gam2

P1<-c(.4,.2); P2<-c(.6,.4)
Gam2CSTe(P1,P2,dat,t)

dat.fr<-data.frame(a=dat)
Gam2CSTe(P1,P2,dat.fr,t)

Gam2CSTe(c(.2,.2),dat[2,],dat,t,ch.data.pnts = TRUE)
#gives an error message since not both points are data points in Dt
Gam2CSTe(c(.2,.2),c(.2,.3),dat,t,ch.data.pnts = TRUE)
#gives an error message since not both points are data points in Dt

## End(Not run)
```

```
#Examples for Gam3CSTe
## Not run:
t<-1.5
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

Gam3CSTe(dat[1,],dat[2,],dat[3,],dat,t)

Gam3CSTe(dat[1,],dat[3,],dat[4,],dat,t)

Gam3CSTe(c(.2,.2),dat[2,],dat[3,],dat,t)

Gam3CSTe(c(.2,.2),c(.2,.3),c(.2,.4),rbind(c(.2,.2),c(.2,.3),c(.2,.4)),t)
Gam3CSTe(c(1.2,1.2),c(1.2,1.3),c(1.2,1.4),rbind(c(1.2,1.2),c(1.2,1.3),c(1.2,1.4)),t)

ind.gam3<-vector()
for (i in 1:(n-2))
 for (j in (i+1):(n-1))
   for (k in (j+1):n)
   {if (Gam3CSTe(dat[i,],dat[j,],dat[k,],dat,t)==1)
     ind.gam3<-rbind(ind.gam3,c(i,j,k))}

ind.gam3

P1<-c(.4,.2); P2<-c(.6,.4); P3<-c(.5,.1)
Gam3CSTe(P1,P2,P3,dat,t)

dat.fr<-data.frame(a=dat)
Gam3CSTe(P1,P2,P3,dat.fr,t)

Gam3CSTe(c(.2,.2),dat[2,],dat[3,],dat,t,ch.data.pnts = TRUE)
#gives an error message since not all points, pt1, pt2, and pt3, are data points in Dt

## End(Not run)

#Examples for Gam4CSTe
## Not run:
t<-1.5
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

Gam4CSTe(dat[1,],dat[2,],dat[3,],dat[4,],dat,t)

Gam4CSTe(dat[1,],dat[3,],dat[4,],dat[5,],dat,t)

Gam4CSTe(c(.2,.2),dat[2,],dat[3,],dat[4,],dat,t)

Gam4CSTe(c(.2,.2),c(.2,.3),c(.2,.4),c(.2,.5),rbind(c(.2,.2),c(.2,.3),c(.2,.4),c(.2,.5)) ,t)
Gam4CSTe(c(1.2,1.2),c(1.2,1.3),c(1.2,1.4),c(1.2,1.5),
```

```
rbind(c(1.2,1.2),c(1.2,1.3),c(1.2,1.4),c(1.2,1.5)),t)

ind.gam4<-vector()
for (i in 1:(n-3))
 for (j in (i+1):(n-2))
   for (k in (j+1):(n-1))
     for (l in (k+1):n)
     {if (Gam4CSTe(dat[i,],dat[j,],dat[k,],dat[l,],dat,t)==1)
       ind.gam4<-rbind(ind.gam4,c(i,j,k,l))}

ind.gam4

P1<-c(.4,.2); P2<-c(.6,.4); P3<-c(.5,.1); P4<-c(.5,.29)
Gam4CSTe(P1,P2,P3,P4,dat,t)

dat.fr<-data.frame(a=dat)
Gam4CSTe(P1,P2,P3,P4,dat.fr,t)

Gam4CSTe(c(.2,.2),dat[2,],dat[3,],dat[4,],dat,t,ch.data.pnts = TRUE)
#gives an error message since not all points are data points in Dt

## End(Not run)

#Examples for Gam5CSTe
## Not run:
t<-1.5
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

Gam5CSTe(dat[1,],dat[2,],dat[3,],dat[4,],dat[5,],dat,t)

Gam5CSTe(dat[1,],dat[3,],dat[4,],dat[5,],dat[6,],dat,t)

Gam5CSTe(c(.2,.2),dat[2,],dat[3,],dat[4,],dat[5,],dat,t)

ind.gam5<-vector()
for (i1 in 1:(n-4))
 for (i2 in (i1+1):(n-3))
   for (i3 in (i2+1):(n-2))
     for (i4 in (i3+1):(n-1))
       for (i5 in (i4+1):n)
       {if (Gam5CSTe(dat[i1,],dat[i2,],dat[i3,],dat[i4,],dat[i5,],dat,t)==1)
         ind.gam5<-rbind(ind.gam5,c(i1,i2,i3,i4,i5))}

ind.gam5

P1<-c(.4,.2); P2<-c(.6,.4); P3<-c(.5,.1); P4<-c(.5,.29); P5<-c(.3,.3)
Gam5CSTe(P1,P2,P3,P4,P5,dat,t)

dat.fr<-data.frame(a=dat)
Gam5CSTe(P1,P2,P3,P4,P5,dat.fr,t)
```

```
Gam5CSTe(c(.2,.2),dat[2,],dat[3,],dat[4,],dat[5,],dat,t,ch.data.pnts = TRUE)
#gives an error message since not all points are data points in Dt

## End(Not run)

#Examples for Gam6CSTe
## Not run:
t<-1.5
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

Gam6CSTe(dat[1,],dat[2,],dat[3,],dat[4,],dat[5,],dat[6,],dat,t)

Gam6CSTe(dat[1,],dat[3,],dat[4,],dat[5,],dat[6,],dat[7,],dat,t)

Gam6CSTe(c(.2,.2),dat[2,],dat[3,],dat[4,],dat[5,],dat[6,],dat,t)

ind.gam6<-vector()
for (i1 in 1:(n-5))
 for (i2 in (i1+1):(n-4))
   for (i3 in (i2+1):(n-3))
     for (i4 in (i3+1):(n-2))
       for (i5 in (i4+1):(n-1))
         for (i6 in (i5+1):n)
         {if (Gam6CSTe(dat[i1,],dat[i2,],dat[i3,],dat[i4,],dat[i5,],dat[i6,],dat,t)==1)
           ind.gam6<-rbind(ind.gam6,c(i1,i2,i3,i4,i5,i6))}

ind.gam6

P1<-c(.4,.2); P2<-c(.6,.4); P3<-c(.5,.1); P4<-c(.5,.29); P5<-c(.3,.3); P6<-c(.4,.4)
Gam6CSTe(P1,P2,P3,P4,P5,P6,dat,t)

dat.fr<-data.frame(a=dat)
Gam6CSTe(P1,P2,P3,P4,P5,P6,dat.fr,t)

Gam6CSTe(c(.2,.2),dat[2,],dat[3,],dat[4,],dat[5,],dat[6,],dat,t,ch.data.pnts = TRUE)
#gives an error message since not all points are data points in Dt

## End(Not run)
```

---

funsCSt1EdgeRegs — *Each function is for the presence of an arc from a point in one of the edge regions to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case with* $t = 1$

Description

Three indicator functions: `IndCSTeRABt1`, `IndCSTeRBCt1` and `IndCSTeRACt1`.

The function `IndCSTeRABt1` returns I(`pt2` is in $NCS(pt1, t = 1)$ for `pt1` in RAB (edge region for edge AB, i.e., edge 3) in the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$;

`IndCSTeRBCt1` returns I(`pt2` is in $NCS(pt1, t = 1)$ for `pt1` in RBC (edge region for edge BC, i.e., edge 1) in T_e; and

`IndCSTeRACt1` returns I(`pt2` is in $NCS(pt1, t = 1)$ for `pt1` in RAC (edge region for edge AC, i.e., edge 2) in T_e.

That is, each function returns 1 if `pt2` is in $NCS(pt1, t = 1)$, returns 0 otherwise, where $NCS(x, t)$ is the CS proximity region for point x with expansion parameter $t = 1$.

Usage

```
IndCSTeRABt1(pt1, pt2)

IndCSTeRBCt1(pt1, pt2)

IndCSTeRACt1(pt1, pt2)
```

Arguments

`pt1`	A 2D point whose CS proximity region is constructed.
`pt2`	A 2D point. The function determines whether `pt2` is inside the CS proximity region of `pt1` or not.

Value

Each function returns I(`pt2` is in $NCS(pt1, t = 1)$) for `pt1`, that is, returns 1 if `pt2` is in $NCS(pt1, t = 1)$, returns 0 otherwise

See Also

`IndCSTeRAB`, `IndCSTeRBC` and `IndCSTeRAC`

Examples

```
#Examples for IndCSTeRABt1
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T3<-rbind(A,B,CM);

set.seed(1)
dat3<-runif.tri(10,T3)$g
dat<-runifTe(10)$gen.points

IndCSTeRABt1(dat3[1,],dat[2,])
IndCSTeRABt1(dat3[1,],dat3[1,])
IndCSTeRABt1(dat[2,],dat[2,])
```

```
IndCSTeRABt1(dat3[1,],dat3[3,])
IndCSTeRABt1(c(.2,.5),dat[2,])
IndCSTeRABt1(c(.2,.5),c(.2,.5))

## End(Not run)

#Examples for IndCSTeRBCt1
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T1<-rbind(B,C,CM);

set.seed(1)
dat1<-runif.tri(10,T1)$g
dat<-runifTe(10)$gen.points

IndCSTeRBCt1(dat1[1,],dat[2,])
IndCSTeRBCt1(dat1[1,],dat1[1,])
IndCSTeRBCt1(dat[2,],dat[2,])

IndCSTeRBCt1(dat1[1,],dat1[3,])
IndCSTeRBCt1(c(.2,.5),dat[2,])
IndCSTeRBCt1(c(.2,.5),c(.2,.5))

IndCSTeRBCt1(dat[2,],dat[2,])

## End(Not run)

#Examples for IndCSTeRACt1
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
T2<-rbind(A,C,CM);

set.seed(1)
dat2<-runif.tri(10,T2)$g
dat<-runifTe(10)$gen.points

IndCSTeRACt1(dat2[1,],dat[2,])
IndCSTeRACt1(dat2[1,],dat2[1,])
IndCSTeRACt1(dat[2,],dat[2,])

IndCSTeRACt1(dat2[1,],dat2[3,])
IndCSTeRACt1(c(1,2),c(1,2))

IndCSTeRACt1(dat[2,],dat[2,])

## End(Not run)
```

funsIndDelTri *Functions provide the indices of the Delaunay triangles where the points reside*

Description

Two functions: ind.Del.tri and indices.Del.tri.

ind.Del.tri finds the index of the Delaunay triangle in which the given point, pt, resides.

indices.Del.tri finds the indices of triangles for all the points in data set, dat, as a vector.

Delaunay triangulation is based on Yp and DTmesh are the Delaunay triangles with default NULL. The function returns NA for a point not inside the convex hull of Yp. Number of Yp points (i.e., size of Yp) should be at least three and the points should be in general position so that Delaunay triangulation is (uniquely) defined.

If the number of Yp points is 3, then there is only one Delaunay triangle and the indices of all the points inside this triangle are all 1.

See (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
ind.Del.tri(pt, Yp, DTmesh = NULL)

indices.Del.tri(dat, Yp, DTmesh = NULL)
```

Arguments

pt	A 2D point; the index of the Delaunay triangle in which pt resides is to be determined. It is an argument for ind.Del.tri.
Yp	A set of 2D points from which Delaunay triangulation is constructed.
DTmesh	Delaunay triangles based on Yp, default is NULL, which is computed via tri.mesh function in interp package. triangles function yields a triangulation data structure from the triangulation object created by tri.mesh.
dat	A set of 2D points representing the set of data points for which the indices of the Delaunay triangles they reside is to be determined. It is an argument for indices.Del.tri.

Value

ind.Del.tri returns the index of the Delaunay triangle in which the given point, pt, resides and indices.Del.tri returns the vector of indices of the Delaunay triangles in which points in the data set, dat, reside

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

Examples

```
#Examples for ind.Del.tri
nx<-100 #number of X points (target)
ny<-10 #number of Y points (nontarget)

Yp<-cbind(runif(ny),runif(ny))

dat<-runifMT(nx,Yp)$g #data under CSR in the convex hull of Ypoints
#try also dat<-cbind(runif(nx),runif(nx))

ind.Del.tri(dat[10,],Yp)

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")  #Delaunay triangulation
TRY<-interp::triangles(DTY)[,1:3];
ind.Del.tri(dat[10,],Yp,DTY)

ind.Del.tri(c(.5,.5),Yp)

ind.Del.tri(c(1.5,.5),Yp)

ind.DT<-vector()
for (i in 1:nx)
  ind.DT<-c(ind.DT,ind.Del.tri(dat[i,],Yp))
ind.DT

Xlim<-range(Yp[,1],dat[,1])
Ylim<-range(Yp[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points

#plot of the data in the convex hull of Y points together with the Delaunay triangulation
plot(dat,main=" ", xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),type="n")
interp::plot.triSht(DTY, add=TRUE, do.points = TRUE,pch=16,col="blue")
points(dat,pch=".",cex=3)
text(dat,labels = factor(ind.DT) )

Yp<-rbind(c(.3,.2),c(.4,.5),c(.14,.15))
ind.Del.tri(c(.25,.25),Yp)
```

```
dat.fr<-data.frame(a=Yp)
ind.Del.tri(c(.25,.25),dat.fr)

#Examples for indices.Del.tri
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Yp<-cbind(runif(ny),runif(ny))
dat<-runifMT(nx,Yp)$g #data under CSR in the convex hull of Ypoints
#try also dat<-cbind(runif(nx),runif(nx))

tr.ind<-indices.Del.tri(dat,Yp)  #indices of the Delaunay triangles
tr.ind

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points
tr.ind<-indices.Del.tri(dat,Yp,DTY)  #indices of the Delaunay triangles
tr.ind

Xlim<-range(Yp[,1],dat[,1])
Ylim<-range(Yp[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

#plot of the data in the convex hull of Y points together with the Delaunay triangulation

oldpar <- par(mfrow = c(1,2))
par(pty="s")
plot(dat,main=" ", xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),pch=".")
interp::plot.triSht(DTY, add=TRUE, do.points = TRUE,pch=16,col="blue")
text(dat,labels = factor(tr.ind) )
par(oldpar)

Yp<-rbind(c(.3,.2),c(.4,.5),c(.14,.15))
indices.Del.tri(c(.25,.25),Yp)

dat<-rbind(c(.25,.25),c(.15,.25))
indices.Del.tri(dat,Yp)

dat.fr<-data.frame(a=dat)
indices.Del.tri(dat.fr,Yp)

dat.fr<-data.frame(a=Yp)
indices.Del.tri(c(.25,.25),dat.fr)
```

`funsMuVarCS1D` *Returning the mean and (asymptotic) variance of arc density of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - middle interval case*

Description

Two functions: `muCS1D` and `asyvarCS1D`.

`muCS1D` returns the mean of the (arc) density of CS-PCD and `asyvarCS1D` returns the (asymptotic) variance of the arc density of CS-PCD for a given centrality parameter `c` in $(0, 1)$ and an expansion parameter $t > 0$ and 1D uniform data in a finite interval (a, b), i.e. data from $U(a, b)$ distribution.

See also (Ceyhan (2016)).

Usage

```
muCS1D(t, c)

asyvarCS1D(t, c)
```

Arguments

`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

`muCS1D` returns the mean and `asyvarCS1D` returns the asymptotic variance of the arc density of CS-PCD for uniform data in an interval

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`muPE1D` and `asyvarPE1D`

Examples

```
#Examples for muCS1D
muCS1D(1.2,.4)
muCS1D(1.2,.6)

tseq<-seq(0.01,5,by=.05)
cseq<-seq(0.01,.99,by=.05)
```

```
ltseq<-length(tseq)
lcseq<-length(cseq)

mu.grid<-matrix(0,nrow=ltseq,ncol=lcseq)
for (i in 1:ltseq)
  for (j in 1:lcseq)
  {
    mu.grid[i,j]<-muCS1D(tseq[i],cseq[j])
  }

persp(tseq,cseq,mu.grid, xlab="t", ylab="c", zlab="mu(t,c)",theta = -30,
phi = 30, expand = 0.5, col = "lightblue", ltheta = 120,
shade = 0.05, ticktype = "detailed")



#Examples for asyvarCS1D
asyvarCS1D(1.2,.8)

tseq<-seq(0.01,5,by=.05)
cseq<-seq(0.01,.99,by=.05)

ltseq<-length(tseq)
lcseq<-length(cseq)

var.grid<-matrix(0,nrow=ltseq,ncol=lcseq)
for (i in 1:ltseq)
  for (j in 1:lcseq)
  {
    var.grid[i,j]<-asyvarCS1D(tseq[i],cseq[j])
  }

persp(tseq,cseq,var.grid, xlab="t", ylab="c", zlab="var(t,c)", theta = -30,
phi = 30, expand = 0.5, col = "lightblue", ltheta = 120,
shade = 0.05, ticktype = "detailed")
```

`funsMuVarCS2D` *Returns the mean and (asymptotic) variance of arc density of Central Similarity Proximity Catch Digraph (CS-PCD) for 2D uniform data in one triangle*

Description

Two functions: `muCS2D` and `asyvarCS2D`.

`muCS2D` returns the mean of the (arc) density of CS-PCD and `asyvarCS2D` returns the asymptotic variance of the arc density of CS-PCD with expansion parameter $t > 0$ for 2D uniform data in a triangle.

CS proximity regions are defined with respect to the triangle and vertex regions are based on center of mass, CM of the triangle.

See also (Ceyhan (2005); Ceyhan et al. (2007)).

Usage

```
muCS2D(t)

asyvarCS2D(t)
```

Arguments

t	A positive real number which serves as the expansion parameter in CS proximity region.

Value

muCS2D returns the mean and asyvarCS2D returns the (asymptotic) variance of the arc density of CS-PCD for uniform data in any triangle

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

muPE2D and asyvarPE2D

Examples

```
#Examples for muCS2D
muCS2D(.5)

tseq<-seq(0.01,5,by=.05)
ltseq<-length(tseq)

mu<-vector()
for (i in 1:ltseq)
{
  mu<-c(mu,muCS2D(tseq[i]))
}

plot(tseq, mu,type="l",xlab="t",ylab=expression(mu(t)),lty=1,xlim=range(tseq))

#Examples for asyvarCS2D
asyvarCS2D(.5)
```

```
tseq<-seq(0.01,10,by=.05)
ltseq<-length(tseq)

asyvar<-vector()
for (i in 1:ltseq)
{
  asyvar<-c(asyvar,asyvarCS2D(tseq[i]))
}

oldpar <- par(mfrow = c(1,2))
par(mar=c(5,5,4,2))
plot(tseq, asyvar,type="l",xlab="t",ylab=expression(paste(sigma^2,"(t)")),lty=1,xlim=range(tseq))
par(oldpar)
```

`funsMuVarCSend1D` *Returns the mean and (asymptotic) variance of arc density of Central Similarity Proximity Catch Digraph (CS-PCD) for 1D data - end interval case*

Description

Two functions: `muCSend1D` and `asyvarCSend1D`.

`muCSend1D` returns the mean of the arc density of CS-PCD and `asyvarCSend1D` returns the asymptotic variance of the arc density of CS-PCD for a given expansion parameter $t > 0$ for 1D uniform data in the left and right end intervals for the interval (a, b).

See also (Ceyhan (2016)).

Usage

```
muCSend1D(t)

asyvarCSend1D(t)
```

Arguments

`t`	A positive real number which serves as the expansion parameter in CS proximity region.

Details

funsMuVarCSend1D

Value

`muCSend1D` returns the mean and `asyvarCSend1D` returns the asymptotic variance of the arc density of CS-PCD for uniform data in end intervals

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

muPEend1D and asyvarPEend1D

Examples

```
#Examples for muCSend1D
muCSend1D(1.2)

tseq<-seq(0.01,5,by=.05)
ltseq<-length(tseq)

mu.end<-vector()
for (i in 1:ltseq)
{
  mu.end<-c(mu.end,muCSend1D(tseq[i]))
}

oldpar <- par(mfrow = c(1,2))
par(mar = c(5,4,4,2) + 0.1)
plot(tseq, mu.end,type="l",
ylab=expression(paste(mu,"(t)")),xlab="t",lty=1,xlim=range(tseq),ylim=c(0,1))
par(oldpar)


#Examples for asyvarCSend1D
asyvarCSend1D(1.2)

tseq<-seq(.01,5,by=.05)
ltseq<-length(tseq)

var.end<-vector()
for (i in 1:ltseq)
{
  var.end<-c(var.end,asyvarCSend1D(tseq[i]))
}

oldpar <- par(mfrow = c(1,2))
par(mar=c(5,5,4,2))
plot(tseq, var.end,type="l",xlab="t",ylab=expression(paste(sigma^2,"(t)")),lty=1,xlim=range(tseq))
par(oldpar)
```

funsMuVarPE1D *Returns the mean and (asymptotic) variance of arc density of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - middle interval case*

Description

The functions muPE1D and asyvarPE1D and their auxiliary functions.

muPE1D returns the mean of the (arc) density of PE-PCD and asyvarPE1D returns the (asymptotic) variance of the arc density of PE-PCD for a given centrality parameter c in $(0, 1)$ and an expansion parameter $r \geq 1$ and for 1D uniform data in a finite interval (a, b), i.e. data from $U(a, b)$ distribution.

muPE1D uses auxiliary (internal) function mu1PE1D which yields mean (i.e., expected value) of the arc density of PE-PCD for a given c in (0,1/2) and $r \geq 1$.

asyvarPE1D uses auxiliary (internal) functions fvar1 which yields asymptotic variance of the arc density of PE-PCD for c in (1/4,1/2) and $r \geq 1$; and fvar2 which yields asymptotic variance of the arc density of PE-PCD for c in (0,1/4) and $r \geq 1$.

See also (Ceyhan (2012)).

Usage

```
mu1PE1D(r, c)

muPE1D(r, c)

fvar1(r, c)

fvar2(r, c)

asyvarPE1D(r, c)
```

Arguments

r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

muPE1D returns the mean and asyvarPE1D returns the asymptotic variance of the arc density of PE-PCD for $U(a, b)$ data

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

muCS1D and asyvarCS1D

Examples

```
## Not run:
mu1PE1D(2.25,.45)

## End(Not run)

#Examples for muPE1D
muPE1D(1.2,.4)
muPE1D(1.2,.6)

rseq<-seq(1.01,5,by=.05)
cseq<-seq(0.01,.99,by=.05)

lrseq<-length(rseq)
lcseq<-length(cseq)

mu.grid<-matrix(0,nrow=lrseq,ncol=lcseq)
for (i in 1:lrseq)
  for (j in 1:lcseq)
  {
    mu.grid[i,j]<-muPE1D(rseq[i],cseq[j])
  }

persp(rseq,cseq,mu.grid, xlab="r", ylab="c", zlab="mu(r,c)", theta = -30, phi = 30,
expand = 0.5, col = "lightblue", ltheta = 120, shade = 0.05, ticktype = "detailed")

## Not run:
fvar1(2.25,.45)

## End(Not run)

## Not run:
fvar2(2.25,.15)

## End(Not run)

#Examples for asyvarPE1D
asyvarPE1D(1.2,.8)

rseq<-seq(1.01,5,by=.05)
cseq<-seq(0.01,.99,by=.05)

lrseq<-length(rseq)
lcseq<-length(cseq)

var.grid<-matrix(0,nrow=lrseq,ncol=lcseq)
for (i in 1:lrseq)
  for (j in 1:lcseq)
```

```
  {
    var.grid[i,j]<-asyvarPE1D(rseq[i],cseq[j])
  }

persp(rseq,cseq,var.grid, xlab="r", ylab="c", zlab="var(r,c)", theta = -30, phi = 30,
expand = 0.5, col = "lightblue", ltheta = 120, shade = 0.05, ticktype = "detailed")
```

`funsMuVarPE2D` *Returns the mean and (asymptotic) variance of arc density of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D uniform data in one triangle*

Description

Two functions: `muPE2D` and `asyvarPE2D`.

`muPE2D` returns the mean of the (arc) density of PE-PCD and `asyvarPE2D` returns the asymptotic variance of the arc density of PE-PCD with expansion parameter $r \geq 1$ for 2D uniform data in a triangle.

PE proximity regions are defined with respect to the triangle and vertex regions are based on center of mass, CM of the triangle.

See also (Ceyhan et al. (2006)).

Usage

```
muPE2D(r)

asyvarPE2D(r)
```

Arguments

`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

Value

`muPE2D` returns the mean and `asyvarPE2D` returns the (asymptotic) variance of the arc density of PE-PCD for uniform data in any triangle

References

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

`muCS2D` and `asyvarCS2D`

Examples

```
#Examples for muPE2D
muPE2D(1.2)

rseq<-seq(1.01,5,by=.05)
lrseq<-length(rseq)

mu<-vector()
for (i in 1:lrseq)
{
  mu<-c(mu,muPE2D(rseq[i]))
}

plot(rseq, mu,type="l",xlab="r",ylab=expression(mu(r)),lty=1,xlim=range(rseq),ylim=c(0,1))

#Examples for asyvarPE2D
asyvarPE2D(1.2)

rseq<-seq(1.01,5,by=.05)
lrseq<-length(rseq)

avar<-vector()
for (i in 1:lrseq)
{
  avar<-c(avar,asyvarPE2D(rseq[i]))
}

oldpar <- par(mfrow = c(1,2))
par(mar=c(5,5,4,2))
plot(rseq, avar,type="l",xlab="r",ylab=expression(paste(sigma^2,"(r)")),lty=1,xlim=range(rseq))
par(oldpar)
```

`funsMuVarPEend1D` *Returns the mean and (asymptotic) variance of arc density of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - end interval case*

Description

Two functions: `muPEend1D` and `asyvarPEend1D`.

`muPEend1D` returns the mean of the arc density of PE-PCD and `asyvarPEend1D` returns the asymptotic variance of the arc density of PE-PCD for a given expansion parameter $r \geq 1$ for 1D uniform data in the left and right end intervals for the interval (a, b).

See also (Ceyhan (2012)).

Usage

```
muPEend1D(r)

asyvarPEend1D(r)
```

Arguments

r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

Value

muPEend1D returns the mean and asyvarPEend1D returns the asymptotic variance of the arc density of PE-PCD for uniform data in end intervals

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

muCSend1D and asyvarCSend1D

Examples

```
#Examples for muPEend1D
muPEend1D(1.2)

rseq<-seq(1.01,5,by=.05)
lrseq<-length(rseq)

mu.end<-vector()
for (i in 1:lrseq)
{
  mu.end<-c(mu.end,muPEend1D(rseq[i]))
}

plot(rseq, mu.end,type="l",
ylab=expression(paste(mu,"(r)")),xlab="r",lty=1,xlim=range(rseq),ylim=c(0,1))

#Examples for asyvarPEend1D
asyvarPEend1D(1.2)

rseq<-seq(1.01,5,by=.05)
lrseq<-length(rseq)

var.end<-vector()
for (i in 1:lrseq)
{
  var.end<-c(var.end,asyvarPEend1D(rseq[i]))
```

```
}

oldpar <- par(mfrow = c(1,2))
par(mar=c(5,5,4,2))
plot(rseq, var.end,type="l",
xlab="r",ylab=expression(paste(sigma^2,"(r)")),lty=1,xlim=range(rseq))
par(oldpar)
```

---

funsPG2PE1D *The functions for probability of domination number=2 for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - middle interval case*

Description

The function PG2PE1D and its auxiliary functions.

Returns $P(\gamma = 2)$ for PE-PCD whose vertices are a uniform data set of size n in a finite interval (a,b)where γ stans for the domination number.

The PE proximity region $NPE(x, r, c)$ is defined with respect to (a,b) with centrality parameter c in $(0, 1)$ and expansion parameter $r \geq 1$.

To compute the probability $P(\gamma = 2)$ for PE-PCD in the 1D case, we partition the domain $(r, c) = (1, \infty) \times (0, 1)$, and compute the probability for each partition set. The sample size (i.e. number of vertices or data points) is a positive integer, n.

Usage

```
PG2AI(r, c, n)

PG2AII(r, c, n)

PG2AIII(r, c, n)

PG2AIV(r, c, n)

PG2A(r, c, n)

PG2Asym(r, c, n)

PG2BIII(r, c, n)

PG2B(r, c, n)

PG2Bsym(r, c, n)

PG2CIV(r, c, n)
```

```
PG2C(r, c, n)

PG2Csym(r, c, n)

PG2PE1D(r, c, n)
```

Arguments

r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
n	A positive integer representing the size of the uniform data set.

Value

P(domination number=2) for PE-PCD whose vertices are a uniform data set of size n in a finite interval (a, b)

Auxiliary Functions for PG2PE1D

The auxiliary functions are PG2AI,PG2AII,PG2AIII,PG2AIV,PG2A,PG2Asym,PG2BIII,PG2B,PG2B,PG2Bsym,PG2CIV,PG2C and PG2Csym, each corresponding to a partition of the domain of r and c. In particular, the domain partition is handled in 3 cases as

CASE A: c in $((3 - \sqrt{5})/2, 1/2)$

CASE B: c in $(1/4, (3 - \sqrt{5})/2)$ and

CASE C: c in $(0, 1/4)$.

Case A - c in $((3 - \sqrt{5})/2, 1/2)$

In Case A, we compute $P(\gamma = 2)$ with

PG2AIV(r,c,n) if $1 < r < (1 - c)/c$;

PG2AIII(r,c,n) if $(1 - c)/c < r < 1/(1 - c)$;

PG2AII(r,c,n) if $1/(1 - c) < r < 1/c$;

and PG2AI(r,c,n) otherwise.

PG2A(r,c,n) combines these functions in Case A: c in $((3 - \sqrt{5})/2, 1/2)$. Due to the symmetry in the PE proximity regions, we use PG2Asym(r,c,n) for c in $(1/2, (\sqrt{5} - 1)/2)$ with the same auxiliary functions

PG2AIV(r,1-c,n) if $1 < r < c/(1 - c)$;

PG2AIII(r,1-c,n) if $(c/(1 - c) < r < 1/c$;

PG2AII(r,1-c,n) if $1/c < r < 1/(1 - c)$;

and PG2AI(r,1-c,n) otherwise.

Case B - c in $(1/4, (3-\sqrt{5})/2)$

In Case B, we compute $P(\gamma = 2)$ with

PG2AIV(r,c,n) if $1 < r < 1/(1-c)$;

PG2BIII(r,c,n) if $1/(1-c) < r < (1-c)/c$;

PG2AII(r,c,n) if $(1-c)/c < r < 1/c$;

and PG2AI(r,c,n) otherwise.

PG2B(r,c,n) combines these functions in Case B: c in $(1/4, (3-\sqrt{5})/2)$. Due to the symmetry in the PE proximity regions, we use PG2Bsym(r,c,n) for c in $((\sqrt{5}-1)/2, 3/4)$ with the same auxiliary functions

PG2AIV(r,1-c,n) if $1 < r < 1/c$;

PG2BIII(r,1-c,n) if $1/c < r < c/(1-c)$;

PG2AII(r,1-c,n) if $c/(1-c) < r < 1/(1-c)$;

and PG2AI(r,1-c,n) otherwise.

Case C - c in $(0, 1/4)$

In Case C, we compute $P(\gamma = 2)$ with

PG2AIV(r,c,n) if $1 < r < 1/(1-c)$;

PG2BIII(r,c,n) if $1/(1-c) < r < (1-\sqrt{1-4c})/(2c)$;

PG2CIV(r,c,n) if $(1-\sqrt{1-4c})/(2c) < r < (1+\sqrt{1-4c})/(2c)$;

PG2BIII(r,c,n) if $(1+\sqrt{1-4c})/(2c) < r < 1/(1-c)$;

PG2AII(r,c,n) if $1/(1-c) < r < 1/c$;

and PG2AI(r,c,n) otherwise.

PG2C(r,c,n) combines these functions in CaseE C: c in $(0, 1/4)$. Due to the symmetry in the PE proximity regions, we use PG2Csym(r,c,n) for c in $(3/4, 1)$ with the same auxiliary functions

PG2AIV(r,1-c,n) if $1 < r < 1/c$;

PG2BIII(r,1-c,n) if $1/c < r < (1-\sqrt{1-4(1-c)})/(2(1-c))$;

PG2CIV(r,1-c,n) if $(1-\sqrt{1-4(1-c)})/(2(1-c)) < r < (1+\sqrt{1-4(1-c)})/(2(1-c))$;

PG2BIII(r,1-c,n) if $(1+\sqrt{1-4(1-c)})/(2(1-c)) < r < c/(1-c)$;

PG2AII(r,1-c,n) if $c/(1-c) < r < 1/(1-c)$;

and PG2AI(r,1-c,n) otherwise.

Combining Cases A, B, and C, we get our main function `PG2PE1D` which computes $P(\gamma = 2)$ for any (`r`,`c`) in its domain.

See Also

`PG2PEtri` and `PG2PE1D.asy`

Examples

```
#Examples for the auxiliary functions
## Not run:
PG2AI(2.25,.45,10)

## End(Not run)

## Not run:
PG2AII(2.2,.4,10)

## End(Not run)

## Not run:
PG2AIII(1.2,.25,10)

## End(Not run)

## Not run:
PG2AIV(1.2,.25,10)

## End(Not run)

## Not run:
PG2A(1.2,.25,10)

## End(Not run)

## Not run:
PG2Asym(1.2,.75,10)

## End(Not run)

## Not run:
PG2BIII(1.5,.45,10)

## End(Not run)

## Not run:
PG2B(1.5,.30,10)

## End(Not run)

## Not run:
PG2Bsym(1.5,.70,10)

## End(Not run)

## Not run:
PG2CIV(1.5,.2,10)

## End(Not run)
```

```
## Not run:
PG2C(1.5,.2,10)

## End(Not run)

## Not run:
PG2Csym(1.5,.8,10)

## End(Not run)

#Examples for the main function PG2PE1D
r<-2
c<-.5
n<-10

PG2PE1D(r,c,n)

PG2PE1D(r=1.5,c=1/1.5,n=100)
```

`funsRankOrderTe` *Returns the ranks and orders of points in decreasing distance to the edges of the triangle*

Description

Two functions: `rank.d2e.Te` and `order.d2e.Te`.

`rank.d2e.Te` finds the ranks of the distances of points in data, `Dt`, to the edges of the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$

`dec` is a logical argument, default is `TRUE`, so the ranks are for decreasing distances, if `FALSE` it will be in increasing distances.

`order.d2e.Te` finds the orders of the distances of points in data, `Dt`, to the edges of T_e. The arguments are as in `rank.d2e.Te`.

Usage

```
rank.d2e.Te(Dt, dec = TRUE)

order.d2e.Te(Dt, dec = TRUE)
```

Arguments

`Dt`	A set of 2D points representing the data set in which ranking in terms of the distance to the edges of T_e is performed.
`dec`	A logical argument indicating the how the ranking will be performed. If `TRUE`, the ranks are for decreasing distances,and if `FALSE` they will be in increasing distances, default is `TRUE`.

Value

A list with two elements

distances	Distances from data points to the edges of T_e
dist.rank	The ranks of the data points in decreasing distances to the edges of T_e

Examples

```
#Examples for rank.d2e.Te
## Not run:
n<-20
set.seed(1)
dat<-runifTe(n)$gen.points

dec.dist<-rank.d2e.Te(dat)
dec.dist
dec.dist.rank<-dec.dist[[2]] #the rank of distances to the edges in decreasing order
dec.dist.rank
dist<-dec.dist[[1]] #distances to the edges of the std eq. triangle
dist

A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.0,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(dat,pch=".")
text(dat,labels = factor(dec.dist.rank) )

inc.dist<-rank.d2e.Te(dat,dec = FALSE)
inc.dist
inc.dist.rank<-inc.dist[[2]] #the rank of distances to the edges in increasing order
inc.dist.rank
dist<-inc.dist[[1]] #distances to the edges of the std eq. triangle
dist

plot(A,pch=".",xlab="",ylab="",xlim=Xlim,ylim=Ylim)
polygon(Te)
points(dat,pch=".",xlab="",ylab="", main="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
text(dat,labels = factor(inc.dist.rank) )

rank.d2e.Te(rbind(dat,dat))

dat.fr<-data.frame(a=dat)
rank.d2e.Te(dat.fr)

## End(Not run)
```

```
#Examples for order.d2e.Te
## Not run:
n<-20
set.seed(1)
dat<-runifTe(n)$gen.points  #try also dat<-cbind(runif(n),runif(n))

dec.dist<-order.d2e.Te(dat)
dec.dist
dec.dist.order<-dec.dist[[2]] #the order of distances to the edges in decreasing order
dec.dist.order
dist<-dec.dist[[1]] #distances to the edges of the std eq. triangle
dist
dist[dec.dist.order] #distances in decreasing order
dat[dec.dist.order,] #data in decreasing order

A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(dat,pch=".")
text(dat[dec.dist.order,],labels = factor(1:n) )

inc.dist<-order.d2e.Te(dat,dec = FALSE)
inc.dist
inc.dist.order<-inc.dist[[2]] #the order of distances to the edges in increasing order
inc.dist.order
dist<-inc.dist[[1]] #distances to the edges of the std eq. triangle
dist
dist[inc.dist.order] #distances in increasing order

plot(A,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(dat,pch=".")
text(dat[inc.dist.order,],labels = factor(1:n) )

order.d2e.Te(rbind(dat,dat))

dat.fr<-data.frame(a=dat)
order.d2e.Te(dat.fr)

## End(Not run)
```

funsTbMid2CC *Two functions* lD1CCinTb *and* lD2CCinTb *which are of class "TriLines" — The lines joining the midpoints of edges to the circumcenter (CC) in the basic triangle.*

Description

Returns the equation,slope,intercept, and y-coordinates of the lines joining D1 and CC and also D2 and CC, in the basic triangle $T_b = T(A = (0,0), B = (1,0), C = (c_1, c_2))$ where c1 is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis. Here the midpoints are $D_1 = (B + C)/2$ and $D_2 = (A + C)/2$. x-coordinates are provided in vector x.

Usage

```
lD1CCinTb(x, c1, c2)

lD2CCinTb(x, c1, c2)
```

Arguments

x	A single scalar or a vector of scalars.
c1, c2	Positive real numbers which constitute the vertex of the basic triangle adjacent to the shorter edges; c1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Value

A list with the elements

txt1	Longer description of the line.
txt2	Shorter description of the line (to be inserted over the line in the plot).
mtitle	The "main" title for the plot of the line.
cent	The center chosen inside the standard equilateral triangle.
cent.name	The name of the center inside the basic triangle. It is "CC" for these two functions.
tri	The triangle (it is the basic triangle for this function).
x	The input vector, can be a scalar or a vector of scalars, which constitute the x-coordinates of the point(s) of interest on the line.
y	The output vector, will be a scalar if x is a scalar or a vector of scalars if x is a vector of scalar, constitutes the y-coordinates of the point(s) of interest on the line.
slope	Slope of the line.
intercept	Intercept of the line.
equation	Equation of the line.

See Also

lA_CM.Te, lB_CM.Te, lA_M.Te, lB_M.Te, and lC_M.Te

Examples

```
#Examples for lD1CCinTb
## Not run:
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);  #the vertices of the basic triangle Tb

Tb<-rbind(A,B,C)

xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.1)  #try also by=.01

lnD1CC<-lD1CCinTb(x,c1,c2)
lnD1CC
summary(lnD1CC)
plot(lnD1CC)

CC<-circ.cent.bastri(c1,c2)  #the circumcenter
CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2; #midpoints of the edges
Ds<-rbind(D1,D2,D3)

x1<-seq(0,1,by=.1)  #try also by=.01
y1<-lD1CCinTb(x1,c1,c2)$y

Xlim<-range(Tb[,1],x1)
Ylim<-range(Tb[,2],y1)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tb,CC,D1,D2,D3)
xc<-txt[,1]+c(-.03,.04,.03,.02,.09,-.08,0)
yc<-txt[,2]+c(.02,.02,.04,.08,.03,.03,-.05)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

lines(x1,y1,type="l",lty=2)
text(.8,.5,"lD1CCinTb")

c1<-.4; c2<-.6;
x1<-seq(0,1,by=.1)  #try also by=.01
lD1CCinTb(x1,c1,c2)

## End(Not run)
```

```
#Examples for lD2CCinTb
## Not run:
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);  #the vertices of the basic triangle Tb

Tb<-rbind(A,B,C)

xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.1)  #try also by=.01

lnD2CC<-lD2CCinTb(x,c1,c2)
lnD2CC
summary(lnD2CC)
plot(lnD2CC)

CC<-circ.cent.bastri(c1,c2)  #the circumcenter
CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2; #midpoints of the edges
Ds<-rbind(D1,D2,D3)

x2<-seq(0,1,by=.1)  #try also by=.01
y2<-lD2CCinTb(x2,c1,c2)$y

Xlim<-range(Tb[,1],x1)
Ylim<-range(Tb[,2],y2)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tb,CC,D1,D2,D3)
xc<-txt[,1]+c(-.03,.04,.03,.02,.09,-.08,0)
yc<-txt[,2]+c(.02,.02,.04,.08,.03,.03,-.05)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

lines(x2,y2,type="l",lty=2)
text(0,.5,"lD2CCinTb")

## End(Not run)
```

Gam1ASbastri *The indicator for a point being a dominating point for Arc Slice Proximity Catch Digraphs (AS-PCDs) - basic triangle case*

Description

Returns I(p is a dominating point of the AS-PCD) where the vertices of the AS-PCD are the 2D data set Dt, that is, returns 1 if p is a dominating point of AS-PCD, returns 0 otherwise. AS proximity regions are defined with respect to the basic triangle, T_b, c1 is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

Vertex regions are based on the center M="CC" for circumcenter of T_b; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_b; default is M="CC". Point, p, is in the vertex region of vertex rv (default is NULL); vertices are labeled as 1,2,3 in the order they are stacked row-wise.

ch.data.pnt is for checking whether point p is a data point in Dt or not (default is FALSE), so by default this function checks whether the point p would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam1ASbastri(p, Dt, c1, c2, M = "CC", rv = NULL, ch.data.pnt = FALSE)
```

Arguments

p	A 2D point that is to be tested for being a dominating point or not of the AS-PCD.
Dt	A set of 2D points which constitutes the vertices of the AS-PCD.
c1, c2	Positive real numbers which constitute the vertex of the basic triangle adjacent to the shorter edges; c1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.
M	The center of the triangle. "CC" stands for circumcenter of the triangle T_b or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is M="CC" i.e. the circumcenter of T_b.
rv	Index of the vertex whose region contains point p, rv takes the vertex labels as 1,2,3 as in the row order of the vertices in T_b.
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of the AS-PCD) where the vertices of the AS-PCD are the 2D data set Dt, that is, returns 1 if p is a dominating point, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

Gam1AStri and Gam1PEbastri

Examples

```
## Not run:
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-20

set.seed(1)
dat<-runif.bastri(n,c1,c2)$g

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.2)

Gam1ASbastri(dat[1,],dat,c1,c2,M)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1ASbastri(dat[i,],dat,c1,c2,M))}

ind.gam1<-which(gam.vec==1)
ind.gam1

#or try
Rv<-rv.bastriCC(dat[1,],c1,c2)$rv
Gam1ASbastri(dat[1,],dat,c1,c2,M,Rv)

Gam1ASbastri(c(.2,.4),dat,c1,c2,M)
Gam1ASbastri(c(.2,.4),c(.2,.4),c1,c2,M)

dat2<-rbind(dat,c(.2,.4))
Gam1ASbastri(dat[1,],dat2,c1,c2,M)

CC<-circ.cent.bastri(c1,c2)  #the circumcenter

if (dimension(M)==3) {M<-bary2cart(M,Tb)}
```

```
#need to run this when M is given in barycentric coordinates

if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
} else
{cent<-M
cent.name<-"M"
Ds<-cp2e.bastri(c1,c2,M)
D1<-Ds[1,]; D2<-Ds[2,]; D3<-Ds[3,]
}

Xlim<-range(Tb[,1],dat[,1])
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(dat)
points(rbind(dat[ind.gam1,]),pch=4,col=2)

txt<-rbind(Tb,cent,D1,D2,D3)
xc<-txt[,1]+c(-.03,.03,.02,.06,.06,-0.05,.01)
yc<-txt[,2]+c(.02,.02,.03,.0,.03,.03,-.03)
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)

Gam1ASbastri(c(.4,.2),dat,c1,c2,M)

dat.fr<-data.frame(a=dat)
Gam1ASbastri(c(.4,.2),dat.fr,c1,c2,M)

Gam1ASbastri(c(.5,.11),dat,c1,c2,M)

Gam1ASbastri(c(.5,.11),dat,c1,c2,M,ch.data.pnt=TRUE)
#gives an error message since the point is not in the basic triangle

## End(Not run)
```

`Gam1AStri` *The indicator for a point being a dominating point for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns I(p is a dominating point of the AS-PCD whose vertices are the 2D data set Dt), that is, returns 1 if p is a dominating point of AS-PCD, returns 0 otherwise. Point, p, is in the region of vertex rv (default is NULL); vertices are labeled as 1,2,3 in the order they are stacked row-wise in tri.

AS proximity regions are defined with respect to the triangle tri and vertex regions are based on the center M="CC" for circumcenter of tri; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle tri; default is M="CC" the circumcenter of tri.

ch.data.pnt is for checking whether point p is a data point in Dt or not (default is FALSE), so by default this function checks whether the point p would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam1AStri(p, Dt, tri, M = "CC", rv = NULL, ch.data.pnt = FALSE)
```

Arguments

p	A 2D point that is to be tested for being a dominating point or not of the AS-PCD.
Dt	A set of 2D points which constitutes the vertices of the AS-PCD.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	The center of the triangle. "CC" stands for circumcenter of the triangle tri or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is M="CC" i.e. the circumcenter of tri.
rv	Index of the vertex whose region contains point p, rv takes the vertex labels as 1,2,3 as in the row order of the vertices in tri.
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of the AS-PCD whose vertices are the 2D data set Dt), that is, returns 1 if p is a dominating point of the AS-PCD, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions."

Computational Geometry: Theory and Applications, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

Gam1ASbastri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

Gam1AStri(dat[1,],dat,Tr,M)
Gam1AStri(dat[1,],dat[1,],Tr,M)
Gam1AStri(c(1.5,1.5),c(1.6,1),Tr,M)
Gam1AStri(c(1.6,1),c(1.5,1.5),Tr,M)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1AStri(dat[i,],dat,Tr,M))}

ind.gam1<-which(gam.vec==1)
ind.gam1

#or try
Rv<-rv.triCC(dat[1,],Tr)$rv
Gam1AStri(dat[1,],dat,Tr,M,Rv)

Gam1AStri(c(.2,.4),dat,Tr,M)
Gam1AStri(c(.2,.4),c(.2,.4),Tr,M)

dat2<-rbind(dat,c(.2,.4))
Gam1AStri(dat[1,],dat2,Tr,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

CC<-circ.cent.tri(Tr)  #the circumcenter

if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
```

```
cent.name<-"CC"
} else
{cent<-M
cent.name<-"M"
Ds<-cp2e.tri(Tr,M)
D1<-Ds[1,]; D2<-Ds[2,]; D3<-Ds[3,]
}

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat)
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(rbind(dat[ind.gam1,]),pch=4,col=2)

txt<-rbind(Tr,cent,Ds)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)

Gam1AStri(c(1.5,1.1),dat,Tr,M)

dat.fr<-data.frame(a=dat)
Gam1AStri(c(1.5,1.1),dat.fr,Tr,M)

dat.fr<-data.frame(a=Tr)
Gam1AStri(c(1.5,1.1),dat,dat.fr,M)

Gam1AStri(c(1.5,1.1),dat,Tr,M)
## Not run:
Gam1AStri(c(1.5,1.1),dat,Tr,M,ch.data.pnt=TRUE)
#gives an error message since point p is not a data point in Dt

## End(Not run)
```

`Gam1CS.Te.onesixth` *The indicator for a point being a dominating point for Central Similarity Proximity Catch Digraphs (CS-PCDs) - first one sixth of the standard equilateral triangle case*

Description

Returns I(p is a dominating point of the 2D data set Dt of CS-PCD) in the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$, that is, returns 1 if p is a dominating

point of CS-PCD, returns 0 otherwise.

Point, p, must lie in the first one-sixth of T_e, which is the triangle with vertices $T(A, D_3, CM) = T((0,0),(1/2,0),CM)$.

CS proximity region is constructed with respect to T_e with expansion parameter $t = 1$.

ch.data.pnt is for checking whether point p is a data point in Dt or not (default is FALSE), so by default this function checks whether the point p would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005)).

Usage

```
Gam1CS.Te.onesixth(p, Dt, ch.data.pnt = FALSE)
```

Arguments

p	A 2D point that is to be tested for being a dominating point or not of the CS-PCD.
Dt	A set of 2D points which constitutes the vertices of the CS-PCD.
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of the CS-PCD) where the vertices of the CS-PCD are the 2D data set Dt, that is, returns 1 if p is a dominating point, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

See Also

Gam1CSTe and Gam1CSTet1

Examples

```
## Not run:
n<-5
set.seed(1)
dat<-rbind(runifTe.onesixth(n)$gen.points,runifTe(n)$gen.points)

Gam1CS.Te.onesixth(dat[1,],dat[2,])
Gam1CS.Te.onesixth(c(.2,.5),dat[2,])
Gam1CS.Te.onesixth(c(.2,.5),c(.2,.5))

Gam1CS.Te.onesixth(dat[2,],dat)
Gam1CS.Te.onesixth(dat[9,],dat)
```

```
Gam1CS.Te.onesixth(c(.49,.49),dat)

n<-nrow(dat)
gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1CS.Te.onesixth(dat[i,],dat))}

ind.gam1<-which(gam.vec==1)

ind.gam1
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
CM<-(A+B+C)/3;
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-matrix(rep(CM,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
segments(L[,1], L[,2], Te[,1], Te[,2], lty=2)
polygon(rbind(A,D3,CM))
points(dat,pch=1,col=1)
points(rbind(dat[ind.gam1,]),pch=4,col=2)

txt<-rbind(A,B,C,CM,D1,D2,D3)
xc<-txt[,1]+c(-.02,.02,.01,.05,.04,-.03,.03)
yc<-txt[,2]+c(.02,.02,.03,-.02,.03,.02,-0.04)
txt.str<-c("A","B","C","CM","D1","D2","D3")
text(xc,yc,txt.str)

Gam1CS.Te.onesixth(c(.4,.1),c(.4,.1))
Gam1CS.Te.onesixth(c(.49,.19),c(.4,.1))

Gam1CS.Te.onesixth(c(.4,.2),dat)

dat.fr<-data.frame(a=dat)
Gam1CS.Te.onesixth(c(.4,.2),dat.fr)

Gam1CS.Te.onesixth(c(.49,.49),dat,ch.data.pnt = TRUE)
#gives an error message since point, p, is not a data point in Dt

## End(Not run)
```

Gam1CS1D *The indicator for a point being a dominating point for Central Similarity Proximity Catch Digraphs (CS-PCDs) for an interval*

Description

Returns I(p is a dominating point of CS-PCD) where the vertices of the CS-PCD are the 1D data set Dt).

CS proximity region is defined with respect to the interval int with an expansion parameter, $t > 0$, and a centrality parameter, c in $(0, 1)$, so arcs may exist for Dt points inside the interval $int = (a, b)$.

Vertex regions are based on the center associated with the centrality parameter c in $(0, 1)$. rv is the index of the vertex region p resides, with default=NULL.

ch.data.pnt is for checking whether point p is a data point in Dt or not (default is FALSE), so by default this function checks whether the point p would be a dominating point if it actually were in the data set.

Usage

```
Gam1CS1D(p, Dt, t, c, int, rv = NULL, ch.data.pnt = FALSE)
```

Arguments

p	A 1D point that is to be tested for being a dominating point or not of the CS-PCD.
Dt	A set of 1D points which constitutes the vertices of the CS-PCD.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$; default c=.5.
int	A vector of two real numbers representing an interval.
rv	Index of the vertex region in which the point resides, either 1, 2 or NULL (default is NULL).
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of CS-PCD) where the vertices of the CS-PCD are the 1D data set Dt), that is, returns 1 if p is a dominating point, returns 0 otherwise

See Also

Gam1PE1D

Examples

```
t<-2
c<-.4
a<-0; b<-10; int<-c(a,b)

Mc<-centMc(int,c)
n<-10

set.seed(1)
dat<-runif(n,a,b)

Gam1CS1D(dat[5],dat,t,c,int)
## Not run:
Gam1CS1D(2,dat,t,c,int,ch.data.pnt = TRUE)  #p is not a data point in Dt

## End(Not run)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1CS1D(dat[i],dat,t,c,int))}

ind.gam1<-which(gam.vec==1)
ind.gam1

domset<-dat[ind.gam1]
if (length(ind.gam1)==0)
{domset<-NA}

#or try
Rv<-rv.mid.int(dat[5],c,int)$rv
Gam1CS1D(dat[5],dat,t,c,int,Rv)

Xlim<-range(a,b,dat)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),xlab="",pch=".",xlim=Xlim+xd*c(-.05,.05))
abline(h=0)
abline(v=c(a,b,Mc),col=c(1,1,2),lty=2)
points(cbind(dat,0))
points(cbind(domset,0),pch=4,col=2)
text(cbind(c(a,b,Mc),-0.1),c("a","b","Mc"))

Gam1CS1D(dat[5],dat,t,c,int)

n<-10
dat2<-runif(n,a+b,b+10)
Gam1CS1D(5,dat2,t,c,int)
```

Gam1CSTe *The indicator for a point being a dominating point for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns I(p is a dominating point of the CS-PCD) where the vertices of the CS-PCD are the 2D data set Dt in the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$, that is, returns 1 if p is a dominating point of CS-PCD, returns 0 otherwise.

CS proximity region is constructed with respect to T_e with expansion parameter $t > 0$ and edge regions are based on center of mass $CM = (1/2, \sqrt{3}/6)$.

ch.data.pnt is for checking whether point p is a data point in Dt or not (default is FALSE), so by default this function checks whether the point p would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam1CSTe(p, Dt, t, ch.data.pnt = FALSE)
```

Arguments

p	A 2D point that is to be tested for being a dominating point or not of the CS-PCD.
Dt	A set of 2D points which constitutes the vertices of the CS-PCD.
t	A positive real number which serves as the expansion parameter in CS proximity region.
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of the CS-PCD) where the vertices of the CS-PCD are the 2D data set Dt, that is, returns 1 if p is a dominating point, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

Gam1CSTet1

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
Te<-rbind(A,B,C);
t<-1.5
n<-10  #try also n<-20

set.seed(1)
dat<-runifTe(n)$gen.points

Gam1CSTe(dat[3,],dat,t)

Gam1CSTe(c(1,2),dat,t)

Gam1CSTe(c(1,2),c(1,2),t)
Gam1CSTe(c(1,2),c(1,2),t,ch.data.pnt = TRUE)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1CSTe(dat[i,],dat,t))}

ind.gam1<-which(gam.vec==1)
ind.gam1

Xlim<-range(Te[,1],dat[,1])
Ylim<-range(Te[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(dat)
L<-Te; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE);
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(rbind(dat[ind.gam1,]),pch=4,col=2)
#rbind is to insert the points correctly if there is only one dominating point

txt<-rbind(Te,CM)
xc<-txt[,1]+c(-.02,.02,.01,.05)
yc<-txt[,2]+c(.02,.02,.03,.02)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)

## Not run:
Gam1CSTe(c(1,2),dat,t,ch.data.pnt = TRUE)
#gives an error message since p is not a data point

## End(Not run)
```

Gam1CSTet1 *The indicator for a point being a dominating point for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case with $t = 1$*

Description

Returns I(p is a dominating point of the CS-PCD) where the vertices of the CS-PCD are the 2D data set Dt in the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$, that is, returns 1 if p is a dominating point of CS-PCD, returns 0 otherwise.

Point, p, is in the edge region of edge re (default is NULL); vertices are labeled as 1,2,3 in the order they are stacked row-wise in T_e, and the opposite edges are labeled with label of the vertices (that is, edge numbering is 1,2,3 for edges AB,BC,AC).

CS proximity region is constructed with respect to T_e with expansion parameter $t = 1$ and edge regions are based on center of mass $CM = (1/2, \sqrt{3}/6)$.

ch.data.pnt is for checking whether point p is a data point in Dt or not (default is FALSE), so by default this function checks whether the point p would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam1CSTet1(p, Dt, re = NULL, ch.data.pnt = FALSE)
```

Arguments

p	A 2D point that is to be tested for being a dominating point or not of the CS-PCD.
Dt	A set of 2D points which constitutes the vertices of the CS-PCD.
re	The index of the edge region in T_e containing the point, either 1, 2, 3 or NULL (default is NULL).
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of the CS-PCD) where the vertices of the CS-PCD are the 2D data set Dt, that is, returns 1 if p is a dominating point, returns 0 otherwise.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

Gam1CSTe

Examples

```
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
CM<-(A+B+C)/3
Te<-rbind(A,B,C);
n<-20

set.seed(1)
dat<-runifTe(n)$gen.points

Gam1CSTet1(dat[3,],dat)

Gam1CSTet1(c(1,2),dat)

Gam1CSTet1(c(1,2),c(1,2))
Gam1CSTet1(c(1,2),c(1,2),ch.data.pnt = TRUE)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1CSTet1(dat[i,],dat))}

ind.gam1<-which(gam.vec==1)
ind.gam1

Xlim<-range(Te[,1],dat[,1])
Ylim<-range(Te[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(dat)
L<-Te; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE);
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
```

```
points(rbind(dat[ind.gam1,]),pch=4,col=2)
#rbind is to insert the points correctly if there is only one dominating point

txt<-rbind(Te,CM)
xc<-txt[,1]+c(-.02,.02,.01,.05)
yc<-txt[,2]+c(.02,.02,.03,.02)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)

P<-c(.4,.2)
Gam1CSTe(P,dat,t)

dat.fr<-data.frame(a=dat)
Gam1CSTe(P,dat.fr,t)

Gam1CSTet1(c(1,2),dat,ch.data.pnt = TRUE)
#gives an error message since p is not a data point

## End(Not run)
```

`Gam1PE1D` *The indicator for a point being a dominating point for Proportional Edge Proximity Catch Digraphs (PE-PCDs) for an interval*

Description

Returns I(p is a dominating point of the PE-PCD) where the vertices of the PE-PCD are the 1D data set `Dt`.

PE proximity region is defined with respect to the interval `int` with an expansion parameter, $r \geq 1$, and a centrality parameter, `c` in $(0, 1)$, so arcs may exist for `Dt` points inside the interval $int = (a, b)$.

Vertex regions are based on the center associated with the centrality parameter `c` in $(0, 1)$. `rv` is the index of the vertex region p resides, with default=`NULL`.

`ch.data.pnt` is for checking whether point `p` is a data point in `Dt` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

Usage

```
Gam1PE1D(p, Dt, r, c, int, rv = NULL, ch.data.pnt = FALSE)
```

Arguments

`p`	A 1D point that is to be tested for being a dominating point or not of the PE-PCD.
`Dt`	A set of 1D points which constitutes the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$; default c=.5.
int	A vector of two real numbers representing an interval.
rv	Index of the vertex region in which the point resides, either 1, 2 or NULL (default is NULL).
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of the PE-PCD) where the vertices of the PE-PCD are the 1D data set Dt, that is, returns 1 if p is a dominating point, returns 0 otherwise

See Also

Gam1PEtri

Examples

```
r<-2
c<-.4
a<-0; b<-10; int<-c(a,b)

Mc<-centMc(int,c)

n<-10

set.seed(1)
dat<-runif(n,a,b)

Gam1PE1D(dat[5],dat,r,c,int)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1PE1D(dat[i],dat,r,c,int))}

ind.gam1<-which(gam.vec==1)
ind.gam1

domset<-dat[ind.gam1]
if (length(ind.gam1)==0)
{domset<-NA}

#or try
Rv<-rv.mid.int(dat[5],c,int)$rv
Gam1PE1D(dat[5],dat,r,c,int,Rv)

Xlim<-range(a,b,dat)
xd<-Xlim[2]-Xlim[1]
```

```
plot(cbind(a,0),xlab="",pch=".",xlim=Xlim+xd*c(-.05,.05))
abline(h=0)
points(cbind(dat,0))
abline(v=c(a,b,Mc),col=c(1,1,2),lty=2)
points(cbind(domset,0),pch=4,col=2)
text(cbind(c(a,b,Mc),-0.1),c("a","b","Mc"))

Gam1PE1D(dat[5],dat,r,c,int)

n<-10
dat2<-runif(n,a+b,b+10)
Gam1PE1D(5,dat2,r,c,int)

## Not run:
Gam1PE1D(2,dat,r,c,int,ch.data.pnt = TRUE)  #point p is not a data point in Dt

## End(Not run)
```

`Gam1PEbastri` *The indicator for a point being a dominating point or not for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - basic triangle case*

Description

Returns I(p is a dominating point of the PE-PCD) where the vertices of the PE-PCD are the 2D data set `Dt` for data in the basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, that is, returns 1 if `p` is a dominating point of PE-PCD, returns 0 otherwise.

PE proximity regions are defined with respect to the basic triangle T_b. In the basic triangle, T_b, `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of a basic triangle to the edges on the extension of the lines joining `M` to the vertices or based on the circumcenter of T_b; default is $M = (1,1,1)$ i.e. the center of mass of T_b. Point, `p`, is in the vertex region of vertex `rv` (default is `NULL`); vertices are labeled as 1,2,3 in the order they are stacked row-wise.

`ch.data.pnt` is for checking whether point `p` is a data point in `Dt` or not (default is `FALSE`), so by default this function checks whether the point `p` would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2011)).

Usage

```
Gam1PEbastri(p, Dt, r, c1, c2, M = c(1, 1, 1), rv = NULL, ch.data.pnt = FALSE)
```

Arguments

p	A 2D point that is to be tested for being a dominating point or not of the PE-PCD.
Dt	A set of 2D points which constitutes the vertices of the PE-PCD.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c1, c2	Positive real numbers which constitute the vertex of the basic triangle adjacent to the shorter edges; c1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the basic triangle T_b or the circumcenter of T_b; default is $M = (1, 1, 1)$ i.e. the center of mass of T_b.
rv	Index of the vertex whose region contains point p, rv takes the vertex labels as 1,2,3 as in the row order of the vertices in T_b.
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of the PE-PCD) where the vertices of the PE-PCD are the 2D data set Dt, that is, returns 1 if p is a dominating point, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

Gam1ASbastri and Gam1AStri

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-10  #try also n<-20

set.seed(1)
dat<-runif.bastri(n,c1,c2)$g

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.3)
r<-2

P<-c(.4,.2)
```

```
Gam1PEbastri(P,dat,r,c1,c2,M)
Gam1PEbastri(P,P,r,c1,c2,M)

Gam1PEbastri(dat[1,],dat,r,c1,c2,M)

Gam1PEbastri(c(1,1),dat,r,c1,c2,M)
## Not run:
Gam1PEbastri(c(1,1),dat,r,c1,c2,M,ch.data.pnt = TRUE)
#gives an error message since point p=c(1,1) is not a data point in Dt

## End(Not run)
Gam1PEbastri(c(1,1),c(1,1),r,c1,c2,M)

#or try
Rv<-rv.bastri.cent(dat[1,],c1,c2,M)$rv
Gam1PEbastri(dat[1,],dat,r,c1,c2,M,Rv)

Gam1PEbastri(c(2,1),dat,r,c1,c2,M)

Gam1PEbastri(c(.2,.1),dat,r,c1,c2,M)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1PEbastri(dat[i,],dat,r,c1,c2,M))}

ind.gam1<-which(gam.vec==1)
ind.gam1

Xlim<-range(Tb[,1],dat[,1])
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tb)}
#need to run this when M is given in barycentric coordinates

if (identical(M,circ.cent.tri(Tb)))
{
  plot(Tb,pch=".",asp=1,xlab="",ylab="",axes=TRUE,
  xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
  polygon(Tb)
  points(dat,pch=1,col=1)
  Ds<-rbind((B+C)/2,(A+C)/2,(A+B)/2)
} else
{plot(Tb,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
  polygon(Tb)
  points(dat,pch=1,col=1)
  Ds<-cp2e.bastri(c1,c2,M)}
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(rbind(dat[ind.gam1,]),pch=4,col=2)
```

```
txt<-rbind(Tb,M,Ds)
xc<-txt[,1]+c(-.02,.02,.02,-.02,.03,-.03,.01)
yc<-txt[,2]+c(.02,.02,.02,-.02,.02,.02,-.03)
txt.str<-c("A","B","C","M","D1","D2","D3")
text(xc,yc,txt.str)

P<-c(.4,.2)
Gam1PEbastri(P,dat,r,c1,c2,M)

Gam1PEbastri(P,rbind(dat,dat),r,c1,c2,M)

dat.fr<-data.frame(a=dat)
Gam1PEbastri(P,dat.fr,r,c1,c2,M)

## Not run:
Gam1PEbastri(c(.2,.1),dat,r,c1,c2,M,ch.data.pnt=TRUE)
#gives an error message since point p is not a data point in Dt

## End(Not run)
```

`Gam1PEstdTetra` *The indicator for a 3D point being a dominating point for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard regular tetrahedron case*

Description

Returns I(p is a dominating point of the PE-PCD) where the vertices of teh PE-PCD are the 3D data set `Dt` in the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/3))$, that is, returns 1 if p is a dominating point of PE-PCD, returns 0 otherwise.

Point, p, is in the vertex region of vertex `rv` (default is `NULL`); vertices are labeled as 1,2,3,4 in the order they are stacked row-wise in T_h.

PE proximity region is constructed with respect to the tetrahedron T_h with expansion parameter $r \geq 1$ and vertex regions are based on center of mass CM (equivalent to circumcenter in this case).

`ch.data.pnt` is for checking whether point p is a data point in `Dt` or not (default is `FALSE`), so by default this function checks whether the point p would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam1PEstdTetra(p, Dt, r, rv = NULL, ch.data.pnt = FALSE)
```

Arguments

p	A 3D point that is to be tested for being a dominating point or not of the PE-PCD.
Dt	A set of 3D points which constitutes the vertices of the PE-PCD.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
rv	Index of the vertex whose region contains point p, rv takes the vertex labels as 1,2,3,4 as in the row order of the vertices in standard regular tetrahedron, default is NULL.
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of the PE-PCD) where the vertices of teh PE-PCD are the 3D data set Dt, that is, returns 1 if p is a dominating point, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

Gam1PEtetra, Gam1PEtri and Gam1PEbastri

Examples

```
## Not run:
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-20
dat<-runif.stdtetra(n)$g  #try also dat<-cbind(runif(n),runif(n),runif(n))
r<-1.5

P<-c(.4,.1,.2)
Gam1PEstdTetra(P,P,r)
Gam1PEstdTetra(dat[1,],dat,r)
Gam1PEstdTetra(P,dat,r)

Gam1PEstdTetra(dat[1,],dat,r)
Gam1PEstdTetra(dat[1,],dat[1,],r)

#or try
```

```
RV<-rv.tetraCC(dat[1,],tetra)$rv
Gam1PEstdTetra(dat[1,],dat,r,rv=RV)

Gam1PEstdTetra(c(-1,-1,-1),dat,r)
Gam1PEstdTetra(c(-1,-1,-1),c(-1,-1,-1),r)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1PEstdTetra(dat[i,],dat,r))}

ind.gam1<-which(gam.vec==1)
ind.gam1
g1.pts<-dat[ind.gam1,]

Xlim<-range(tetra[,1],dat[,1])
Ylim<-range(tetra[,2],dat[,2])
Zlim<-range(tetra[,3],dat[,3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(dat[,1],dat[,2],dat[,3], phi =0,theta=40, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
         pch = 20, cex = 1, ticktype = "detailed")
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)
if (length(g1.pts)!=0)
{plot3D::points3D(g1.pts[,1],g1.pts[,2],g1.pts[,3], pch=4,col="red", add=TRUE)}

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3], labels=c("A","B","C","D"), add=TRUE)

CM<-apply(tetra,2,mean)
D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-matrix(rep(CM,6),ncol=3,byrow=TRUE)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty=2)

P<-c(.4,.1,.2)
Gam1PEstdTetra(P,dat,r)

dat.fr<-data.frame(a=dat)
Gam1PEstdTetra(P,dat.fr,r)

Gam1PEstdTetra(c(-1,-1,-1),dat,r,ch.data.pnt = TRUE)
#gives an error message since p is not a data point

## End(Not run)
```

Gam1PEtetra *The indicator for a 3D point being a dominating point for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one tetrahedron case*

Description

Returns I(p is a dominating point of the PE-PCD) where the vertices of the PE-PCD are the 2D data set Dt in the tetrahedron th, that is, returns 1 if p is a dominating point of PE-PCD, returns 0 otherwise.

Point, p, is in the vertex region of vertex rv (default is NULL); vertices are labeled as 1,2,3,4 in the order they are stacked row-wise in th.

PE proximity region is constructed with respect to the tetrahedron th with expansion parameter $r \geq 1$ and vertex regions are based on center of mass (M="CM") or circumcenter (M="CC") only. and vertex regions are based on center of mass CM (equivalent to circumcenter in this case).

ch.data.pnt is for checking whether point p is a data point in Dt or not (default is FALSE), so by default this function checks whether the point p would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam1PEtetra(p, Dt, r, th, M = "CM", rv = NULL, ch.data.pnt = FALSE)
```

Arguments

p	A 3D point that is to be tested for being a dominating point or not of the PE-PCD.
Dt	A set of 3D points which constitutes the vertices of the PE-PCD.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
th	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.
M	The center to be used in the construction of the vertex regions in the tetrahedron, th. Currently it only takes "CC" for circumcenter and "CM" for center of mass; default="CM".
rv	Index of the vertex whose region contains point p, rv takes the vertex labels as 1,2,3,4 as in the row order of the vertices in standard tetrahedron, default is NULL.
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of the PE-PCD) where the vertices of the PE-PCD are the 2D data set Dt, that is, returns 1 if p is a dominating point, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

Gam1PEstdTetra, Gam1PEtri and Gam1PEbastri

Examples

```
## Not run:
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-20

dat<-runif.tetra(n,tetra)$g  #try also dat<-cbind(runif(n),runif(n),runif(n))

M<-"CM"; cent<-apply(tetra,2,mean)  #center of mass
#try also M<-"CC"; cent<-circ.cent.tetra(tetra)  #circumcenter

r<-2

P<-c(.4,.1,.2)
Gam1PEtetra(P,P,r,tetra,M)
Gam1PEtetra(dat[1,],dat,r,tetra,M)
Gam1PEtetra(P,dat,r,tetra,M)

Gam1PEtetra(dat[1,],dat,r,tetra,M)
Gam1PEtetra(dat[1,],dat[1,],r,tetra,M)

#or try
RV<-rv.tetraCC(dat[1,],tetra)$rv
Gam1PEtetra(dat[1,],dat,r,tetra,M,rv=RV)

Gam1PEtetra(c(-1,-1,-1),dat,r,tetra,M)
Gam1PEtetra(c(-1,-1,-1),c(-1,-1,-1),r,tetra,M)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1PEtetra(dat[i,],dat,r,tetra,M))}

ind.gam1<-which(gam.vec==1)
ind.gam1
g1.pts<-dat[ind.gam1,]

Xlim<-range(tetra[,1],dat[,1],cent[1])
Ylim<-range(tetra[,2],dat[,2],cent[2])
```

```
Zlim<-range(tetra[,3],dat[,3],cent[3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(dat[,1],dat[,2],dat[,3], phi =0,theta=40, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
         pch = 20, cex = 1, ticktype = "detailed")
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)
if (length(g1.pts)!=0)
{plot3D::points3D(g1.pts[,1],g1.pts[,2],g1.pts[,3], pch=4,col="red", add=TRUE)}

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3], labels=c("A","B","C","D"), add=TRUE)

D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-rbind(cent,cent,cent,cent,cent,cent)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty=2)

P<-c(.4,.1,.2)
Gam1PEtetra(P,dat,r,tetra,M)

dat.fr<-data.frame(a=dat)
Gam1PEtetra(P,dat.fr,r,tetra,M)

Gam1PEtetra(c(-1,-1,-1),dat,r,tetra,M,ch.data.pnt = TRUE)
#gives an error message since p is not a data point

## End(Not run)
```

`Gam1PEtri` *The indicator for a point being a dominating point for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns I(p is a dominating point of the PE-PCD) where the vertices of the PE-PCD are the 2D data set `Dt` in the triangle `tri`, that is, returns 1 if `p` is a dominating point of PE-PCD, returns 0 otherwise.

Point, `p`, is in the vertex region of vertex `rv` (default is `NULL`); vertices are labeled as 1,2,3 in the order they are stacked row-wise in `tri`.

PE proximity region is constructed with respect to the triangle `tri` with expansion parameter $r \geq 1$ and vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

ch.data.pnt is for checking whether point p is a data point in Dt or not (default is FALSE), so by default this function checks whether the point p would be a dominating point if it actually were in the data set.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
Gam1PEtri(p, Dt, tri, r, M = c(1, 1, 1), rv = NULL, ch.data.pnt = FALSE)
```

Arguments

p	A 2D point that is to be tested for being a dominating point or not of the PE-PCD.
Dt	A set of 2D points which constitutes the vertices of the PE-PCD.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.
rv	Index of the vertex whose region contains point p, rv takes the vertex labels as 1,2,3 as in the row order of the vertices in tri.
ch.data.pnt	A logical argument for checking whether point p is a data point in Dt or not (default is FALSE).

Value

I(p is a dominating point of the PE-PCD) where the vertices of the PE-PCD are the 2D data set Dt, that is, returns 1 if p is a dominating point, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

Gam1PEbastri and Gam1AStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5  #try also r<-2

Gam1PEtri(c(1.4,1),c(1.4,1),Tr,r,M)
Gam1PEtri(dat[1,],dat,Tr,r,M)
Gam1PEtri(c(1,2),dat,Tr,r,M)

Gam1PEtri(c(1,2),c(1,2),Tr,r,M)
Gam1PEtri(c(1,2),c(1,2),Tr,r,M,ch.data.pnt = TRUE)

gam.vec<-vector()
for (i in 1:n)
{gam.vec<-c(gam.vec,Gam1PEtri(dat[i,],dat,Tr,r,M))}

ind.gam1<-which(gam.vec==1)
ind.gam1

#or try
Rv<-rv.tri.cent(dat[1,],Tr,M)$rv
Gam1PEtri(dat[1,],dat,Tr,r,M,Rv)

Ds<-cp2e.tri(Tr,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

Xlim<-range(Tr[,1],dat[,1],M[1])
Ylim<-range(Tr[,2],dat[,2],M[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=1,col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(rbind(dat[ind.gam1,]),pch=4,col=2)
#rbind is to insert the points correctly if there is only one dominating point
```

```
txt<-rbind(Tr,M,Ds)
xc<-txt[,1]+c(-.02,.03,.02,-.02,.04,-.03,.0)
yc<-txt[,2]+c(.02,.02,.05,-.03,.04,.06,-.07)
txt.str<-c("A","B","C","M","D1","D2","D3")
text(xc,yc,txt.str)

P<-c(1.4,1)
Gam1PEtri(P,P,Tr,r,M)
Gam1PEtri(dat[1,],dat,Tr,r,M)

dat.fr<-data.frame(a=dat)
Gam1PEtri(P,dat.fr,Tr,r,M)

dat.fr<-data.frame(a=Tr)
Gam1PEtri(P,dat,dat.fr,r,M)

## Not run:
Gam1PEtri(c(1,2),dat,Tr,r,M,ch.data.pnt = TRUE)
#gives an error message since p is not a data point

## End(Not run)
```

`Gam2ASbastri` *The indicator for two points being a dominating set for Arc Slice Proximity Catch Digraphs (AS-PCDs) - basic triangle case*

Description

Returns I({pt1,pt2} is a dominating set of AS-PCD) where vertices of AS-PCD are the 2D data set Dt), that is, returns 1 if {pt1,pt2} is a dominating set of AS-PCD, returns 0 otherwise.

AS proximity regions are defined with respect to the basic triangle $T_b = T(c(0,0), c(1,0), c(c1,c2))$, In the basic triangle, T_b, c1 is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

Point, pt1, is in the vertex region of vertex rv1 (default is NULL) and point, pt2, is in the vertex region of vertex rv2 (default is NULL); vertices are labeled as 1,2,3 in the order they are stacked row-wise.

Vertex regions are based on the center M="CC" for circumcenter of T_b; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_b; default is M="CC".

ch.data.pnts is for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE), so by default this function checks whether the points pt1 and pt2 would be a dominating set if they actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam2ASbastri(
  pt1,
  pt2,
  Dt,
  c1,
  c2,
  M = "CC",
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

pt1, pt2	Two 2D points to be tested for constituting a dominating set of the AS-PCD.
Dt	A set of 2D points which constitutes the vertices of the AS-PCD.
c1, c2	Positive real numbers which constitute the vertex of the basic triangle adjacent to the shorter edges; c1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.
M	The center of the triangle. "CC" stands for circumcenter of the triangle T_b or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is M="CC" i.e. the circumcenter of T_b.
rv1, rv2	The indices of the vertices whose regions contains pt1 and pt2, respectively. They take the vertex labels as 1,2,3 as in the row order of the vertices in T_b (default is NULL for both).
ch.data.pnts	A logical argument for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE).

Value

I({pt1,pt2} is a dominating set of the AS-PCD) where the vertices of AS-PCD are the 2D data set Dt), that is, returns 1 if {pt1,pt2} is a dominating set of AS-PCD, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

Gam2AStri

Examples

```
## Not run:
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-20

set.seed(1)
dat<-runif.bastri(n,c1,c2)$g

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.2)

Gam2ASbastri(dat[1,],dat[2,],dat,c1,c2,M)
Gam2ASbastri(dat[1,],dat[1,],dat,c1,c2,M)  #one point can not a dominating set of size two

Gam2ASbastri(c(.2,.4),c(.2,.5),rbind(c(.2,.4),c(.2,.5)),c1,c2,M)

ind.gam2<-vector()
for (i in 1:(n-1))
  for (j in (i+1):n)
  {if (Gam2ASbastri(dat[i,],dat[j,],dat,c1,c2,M)==1)
    ind.gam2<-rbind(ind.gam2,c(i,j))}
ind.gam2

#or try
rv1<-rv.bastriCC(dat[1,],c1,c2)$rv
rv2<-rv.bastriCC(dat[2,],c1,c2)$rv
Gam2ASbastri(dat[1,],dat[2,],dat,c1,c2,M,rv1,rv2)
Gam2ASbastri(c(.2,.4),dat[2,],dat,c1,c2,M,rv1,rv2)

#or try
rv1<-rv.bastriCC(dat[1,],c1,c2)$rv
Gam2ASbastri(dat[1,],dat[2,],dat,c1,c2,M,rv1)

#or try
Rv2<-rv.bastriCC(dat[2,],c1,c2)$rv
Gam2ASbastri(dat[1,],dat[2,],dat,c1,c2,M,rv2=Rv2)

Gam2ASbastri(c(.3,.2),c(.35,.25),dat,c1,c2,M)

dat.fr<-data.frame(a=dat)
Gam2ASbastri(c(.3,.2),c(.35,.25),dat.fr,c1,c2,M)

## End(Not run)
```

Gam2AStri *The indicator for two points constituting a dominating set for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns I({pt1,pt2} is a dominating set of the AS-PCD) where vertices of the AS-PCD are the 2D data set Dt), that is, returns 1 if {pt1,pt2} is a dominating set of AS-PCD, returns 0 otherwise.

AS proximity regions are defined with respect to the triangle tri. Point, pt1, is in the region of vertex rv1 (default is NULL) and point, pt2, is in the region of vertex rv2 (default is NULL); vertices (and hence rv1 and rv2) are labeled as 1,2,3 in the order they are stacked row-wise in tri.

Vertex regions are based on the center M="CC" for circumcenter of tri; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle tri; default is M="CC" the circumcenter of tri.

ch.data.pnts is for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE), so by default this function checks whether the points pt1 and pt2 would constitute dominating set if they actually were in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam2AStri(
  pt1,
  pt2,
  Dt,
  tri,
  M = "CC",
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

pt1, pt2	Two 2D points to be tested for constituting a dominating set of the AS-PCD.
Dt	A set of 2D points which constitutes the vertices of the AS-PCD.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	The center of the triangle. "CC" stands for circumcenter of the triangle tri or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is M="CC" i.e. the circumcenter of tri.
rv1, rv2	The indices of the vertices whose regions contains pt1 and pt2, respectively. They take the vertex labels as 1,2,3 as in the row order of the vertices in tri (default is NULL for both).

ch.data.pnts A logical argument for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE).

Value

I({pt1,pt2} is a dominating set of the AS-PCD) where vertices of the AS-PCD are the 2D data set Dt), that is, returns 1 if {pt1,pt2} is a dominating set of AS-PCD, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

Gam2ASbastri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

Gam2AStri(dat[1,],dat[2,],dat,Tr,M)
Gam2AStri(dat[1,],dat[1,],dat,Tr,M)  #same two points cannot be a dominating set of size 2

Gam2AStri(c(.2,.4),dat[2,],dat,Tr,M)
Gam2AStri(c(.2,.4),c(.2,.5),dat,Tr,M)
Gam2AStri(c(.2,.4),c(.2,.5),rbind(c(.2,.4),c(.2,.5)),Tr,M)

#or try
rv1<-rv.triCC(c(.2,.4),Tr)$rv
rv2<-rv.triCC(c(.2,.5),Tr)$rv
Gam2AStri(c(.2,.4),c(.2,.5),rbind(c(.2,.4),c(.2,.5)),Tr,M,rv1,rv2)

ind.gam2<-vector()
for (i in 1:(n-1))
  for (j in (i+1):n)
```

```
  {if (Gam2AStri(dat[i,],dat[j,],dat,Tr,M)==1)
    ind.gam2<-rbind(ind.gam2,c(i,j))}
ind.gam2

#or try
rv1<-rv.triCC(dat[1,],Tr)$rv
rv2<-rv.triCC(dat[2,],Tr)$rv
Gam2AStri(dat[1,],dat[2,],dat,Tr,M,rv1,rv2)

#or try
rv1<-rv.triCC(dat[1,],Tr)$rv
Gam2AStri(dat[1,],dat[2,],dat,Tr,M,rv1)

#or try
Rv2<-rv.triCC(dat[2,],Tr)$rv
Gam2AStri(dat[1,],dat[2,],dat,Tr,M,rv2=Rv2)

Gam2AStri(c(1.3,1.2),c(1.35,1.25),dat,Tr,M)

dat.fr<-data.frame(a=dat)
Gam2AStri(c(.3,.2),c(.35,.25),dat.fr,Tr,M)

dat.fr<-data.frame(a=Tr)
Gam2AStri(c(.3,.2),c(.35,.25),dat,dat.fr,M)
```

`Gam2CS.Te.onesixth` *The indicator for two points constituting a dominating set for Central Similarity Proximity Catch Digraphs (CS-PCDs) - first one sixth of the standard equilateral triangle case*

Description

Returns I({`pt1`,`pt2`} is a dominating set of the CS-PCD) where the vertices of the CS-PCD are the 2D data set `Dt`), that is, returns 1 if `p` is a dominating point of CS-PCD, returns 0 otherwise.

CS proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ and with expansion parameter $t = 1$. Point, `pt1`, must lie in the first one-sixth of T_e, which is the triangle with vertices $T(A, D_3, CM) = T((0, 0), (1/2, 0), CM)$.

`ch.data.pnts` is for checking whether points `pt1` and `pt2` are data points in `Dt` or not (default is `FALSE`), so by default this function checks whether the points `pt1` and `pt2` would be a dominating set if they actually were in the data set.

See also (Ceyhan (2005)).

Usage

```
Gam2CS.Te.onesixth(pt1, pt2, Dt, ch.data.pnts = FALSE)
```

Arguments

pt1, pt2	Two 2D points to be tested for constituting a dominating set of the CS-PCD.
Dt	A set of 2D points which constitutes the vertices of the CS-PCD.
ch.data.pnts	A logical argument for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE).

Value

I({pt1,pt2} is a dominating set of the CS-PCD) where the vertices of the CS-PCD are the 2D data set Dt), that is, returns 1 if {pt1,pt2} is a dominating set of CS-PCD, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

See Also

Gam2CSTe

Examples

```
## Not run:
n<-5

set.seed(1)
dat<-rbind(runifTe.onesixth(n)$gen.points,runifTe(n)$gen.points)

Gam2CS.Te.onesixth(dat[1,],dat[2,],dat)
Gam2CS.Te.onesixth(dat[1,],dat[3,],dat)
Gam2CS.Te.onesixth(c(.2,.2),dat[2,],dat)

Gam2CS.Te.onesixth(c(.2,.1),c(.3,.1),rbind(c(.2,.1),c(.3,.1)))
Gam2CS.Te.onesixth(c(1.2,1.1),c(1.3,1.1),rbind(c(1.2,1.1),c(1.3,1.1)))

n<-nrow(dat)
ind.gam2<-vector()
for (i in 1:(n-1))
 for (j in (i+1):n)
 {if (Gam2CS.Te.onesixth(dat[i,],dat[j,],dat)==1)
   ind.gam2<-rbind(ind.gam2,c(i,j))}

ind.gam2

P1<-c(.2,.1)
P2<-c(.3,.1)
Gam2CS.Te.onesixth(P1,c(.4,.2),dat)

dat.fr<-data.frame(a=dat)
Gam2CS.Te.onesixth(c(.4,.2),P2,dat.fr)
```

```
Gam2CS.Te.onesixth(c(.2,.2),dat[2,],dat,ch.data.pnts = TRUE)
#gives an error message since not both points are data points in Dt

## End(Not run)
```

`Gam2PEbastri` *The indicator for two points being a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - basic triangle case*

Description

Returns I({`pt1`,`pt2`} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 2D data set `Dt` in the basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, that is, returns 1 if {`pt1`,`pt2`} is a dominating set of PE-PCD, returns 0 otherwise.

PE proximity regions are defined with respect to T_b. In the basic triangle, T_b, `c1` is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of a basic triangle T_b; default is $M = (1,1,1)$ i.e. the center of mass of T_b. Point, `pt1`, is in the vertex region of vertex `rv1` (default is `NULL`); and point, `pt2`, is in the vertex region of vertex `rv2` (default is `NULL`); vertices are labeled as 1,2,3 in the order they are stacked row-wise.

`ch.data.pnts` is for checking whether points `pt1` and `pt2` are both data points in `Dt` or not (default is `FALSE`), so by default this function checks whether the points `pt1` and `pt2` would constitute a dominating set if they both were actually in the data set.

See also (Ceyhan (2005, 2011)).

Usage

```
Gam2PEbastri(
  pt1,
  pt2,
  Dt,
  r,
  c1,
  c2,
  M = c(1, 1, 1),
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

pt1, pt2	Two 2D points to be tested for constituting a dominating set of the PE-PCD.
Dt	A set of 2D points which constitutes the vertices of the PE-PCD.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c1, c2	Positive real numbers which constitute the vertex of the basic triangle. adjacent to the shorter edges; c1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the basic triangle T_b or the circumcenter of T_b; default is $M = (1, 1, 1)$ i.e. the center of mass of T_b.
rv1, rv2	The indices of the vertices whose regions contains pt1 and pt2, respectively. They take the vertex labels as 1,2,3 as in the row order of the vertices in T_b (default is NULL for both).
ch.data.pnts	A logical argument for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE).

Value

I({pt1,pt2} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 2D data set Dt), that is, returns 1 if {pt1,pt2} is a dominating set of PE-PCD, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

Gam2PEtri, Gam2ASbastri, and Gam2AStri

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-10  #try also n<-20

set.seed(1)
dat<-runif.bastri(n,c1,c2)$g

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.3)

r<-2
```

```
Gam2PEbastri(dat[1,],dat[2,],dat,r,c1,c2,M)
Gam2PEbastri(c(1,1),dat[2,],dat,r,c1,c2,M)

Gam2PEbastri(c(1,2),dat[2,],dat,r,c1,c2,M)

Gam2PEbastri(c(1,2),c(1,3),rbind(c(1,2),c(1,3)),r,c1,c2,M)
Gam2PEbastri(c(1,2),c(1,3),rbind(c(1,2),c(1,3)),r,c1,c2,M,ch.data.pnts = TRUE)

ind.gam2<-vector()
for (i in 1:(n-1))
  for (j in (i+1):n)
  {if (Gam2PEbastri(dat[i,],dat[j,],dat,r,c1,c2,M)==1)
    ind.gam2<-rbind(ind.gam2,c(i,j))}
ind.gam2

#or try
rv1<-rv.bastri.cent(dat[1,],c1,c2,M)$rv;
rv2<-rv.bastri.cent(dat[2,],c1,c2,M)$rv;
Gam2PEbastri(dat[1,],dat[2,],dat,r,c1,c2,M,rv1,rv2)

#or try
rv1<-rv.bastri.cent(dat[1,],c1,c2,M)$rv;
Gam2PEbastri(dat[1,],dat[2,],dat,r,c1,c2,M,rv1)

#or try
rv2<-rv.bastri.cent(dat[2,],c1,c2,M)$rv;
Gam2PEbastri(dat[1,],dat[2,],dat,r,c1,c2,M,rv2=rv2)

P1<-c(.4,.2)
P2<-c(.6,.2)
Gam2PEbastri(P1,P2,dat,r,c1,c2,M)

Gam2PEbastri(P1,P2,rbind(dat,dat),r,c1,c2,M)

dat.fr<-data.frame(a=dat)
Gam2PEbastri(P1,P2,dat.fr,r,c1,c2,M)

## Not run:
Gam2PEbastri(c(1,2),dat[2,],dat,r,c1,c2,M,ch.data.pnts = TRUE)
#gives an error message since not both points are data points in Dt

## End(Not run)
```

Gam2PEstdTetra

The indicator for two 3D points constituting a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard regular tetrahedron case

---

Description

Returns I({pt1,pt2} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 3D data set Dt in the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/3))$, that is, returns 1 if {pt1,pt2} is a dominating set of PE-PCD, returns 0 otherwise.

Point, pt1, is in the region of vertex rv1 (default is NULL) and point, pt2, is in the region of vertex rv2 (default is NULL); vertices (and hence rv1 and rv2) are labeled as 1,2,3,4 in the order they are stacked row-wise in T_h.

PE proximity region is constructed with respect to the tetrahedron T_h with expansion parameter $r \geq 1$ and vertex regions are based on center of mass CM (equivalent to circumcenter in this case).

ch.data.pnts is for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE), so by default this function checks whether the points pt1 and pt2 would constitute a dominating set if they actually were both in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam2PEstdTetra(pt1, pt2, Dt, r, rv1 = NULL, rv2 = NULL, ch.data.pnts = FALSE)
```

Arguments

pt1, pt2	Two 3D points to be tested for constituting a dominating set of the PE-PCD.
Dt	A set of 3D points which constitutes the vertices of the PE-PCD.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
rv1, rv2	The indices of the vertices whose regions contains pt1 and pt2, respectively. They take the vertex labels as 1,2,3,4 as in the row order of the vertices in T_h (default is NULL for both).
ch.data.pnts	A logical argument for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE).

Value

I({pt1,pt2} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 3D data set Dt), that is, returns 1 if {pt1,pt2} is a dominating set of PE-PCD, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

Gam2PEtetra, Gam2PEtri and Gam2PEbastri

Examples

```
## Not run:
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-20
dat<-runif.stdtetra(n)$g  #try also dat<-cbind(runif(n),runif(n),runif(n))
r<-1.5

Gam2PEstdTetra(dat[1,],dat[2,],dat,r)
Gam2PEstdTetra(dat[1,],dat[1,],dat,r)

Gam2PEstdTetra(c(-1,-1,-1),dat[2,],dat,r)

Gam2PEstdTetra(c(-1,-1,-1),c(-1,-1,-2),rbind(c(-1,-1,-1),c(-1,-1,-2)),r)

ind.gam2<-vector()
for (i in 1:(n-1))
 for (j in (i+1):n)
 {if (Gam2PEstdTetra(dat[i,],dat[j,],dat,r)==1)
   ind.gam2<-rbind(ind.gam2,c(i,j))}

ind.gam2

#or try
rv1<-rv.tetraCC(dat[1,],tetra)$rv;rv2<-rv.tetraCC(dat[2,],tetra)$rv
Gam2PEstdTetra(dat[1,],dat[2,],dat,r,rv1,rv2)

#or try
rv1<-rv.tetraCC(dat[1,],tetra)$rv;
Gam2PEstdTetra(dat[1,],dat[2,],dat,r,rv1)

#or try
rv2<-rv.tetraCC(dat[2,],tetra)$rv
Gam2PEstdTetra(dat[1,],dat[2,],dat,r,rv2=rv2)

P1<-c(.1,.1,.1)
P2<-c(.4,.1,.2)
Gam2PEstdTetra(P1,P2,dat,r)

Gam2PEstdTetra(dat[1,],dat[2,],dat,r)

dat.fr<-data.frame(a=dat)
Gam2PEstdTetra(P1,P2,dat.fr,r)

Gam2PEstdTetra(c(-1,-1,-1),dat[2,],dat,r,ch.data.pnts = TRUE)
#gives an error message since not both points, pt1 and pt2, are data points in Dt

## End(Not run)
```

`Gam2PEtetra` *The indicator for two 3D points constituting a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one tetrahedron case*

Description

Returns I({pt1,pt2} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 3D data set Dt in the tetrahedron th, that is, returns 1 if {pt1,pt2} is a dominating set of PE-PCD, returns 0 otherwise.

Point, pt1, is in the region of vertex rv1 (default is NULL) and point, pt2, is in the region of vertex rv2 (default is NULL); vertices (and hence rv1 and rv2) are labeled as 1,2,3,4 in the order they are stacked row-wise in th.

PE proximity region is constructed with respect to the tetrahedron th with expansion parameter $r \geq 1$ and vertex regions are based on center of mass (M="CM") or circumcenter (M="CC") only.

ch.data.pnts is for checking whether points pt1 and pt2 are both data points in Dt or not (default is FALSE), so by default this function checks whether the points pt1 and pt2 would constitute a dominating set if they actually were both in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam2PEtetra(
  pt1,
  pt2,
  Dt,
  r,
  th,
  M = "CM",
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

pt1, pt2	Two 3D points to be tested for constituting a dominating set of the PE-PCD.
Dt	A set of 3D points which constitutes the vertices of the PE-PCD.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
th	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.
M	The center to be used in the construction of the vertex regions in the tetrahedron, th. Currently it only takes "CC" for circumcenter and "CM" for center of mass; default="CM".

rv1, rv2	The indices of the vertices whose regions contains pt1 and pt2, respectively. They take the vertex labels as 1,2,3,4 as in the row order of the vertices in th (default is NULL for both).
ch.data.pnts	A logical argument for checking whether both points pt1 and pt2 are data points in Dt or not (default is FALSE).

Value

I({pt1,pt2} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 3D data set Dt), that is, returns 1 if {pt1,pt2} is a dominating set of PE-PCD, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

Gam2PEstdTetra, Gam2PEtri and Gam2PEbastri

Examples

```
## Not run:
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-20

dat<-runif.tetra(n,tetra)$g  #try also dat<-cbind(runif(n),runif(n),runif(n))

M<-"CM";  #try also M<-"CC";
r<-1.5

Gam2PEtetra(dat[1,],dat[2,],dat,r,tetra,M)
Gam2PEtetra(dat[1,],dat[1,],dat,r,tetra,M)

Gam2PEtetra(c(-1,-1,-1),dat[2,],dat,r,tetra,M)

Gam2PEtetra(c(-1,-1,-1),c(-1,-1,-2),rbind(c(-1,-1,-1),c(-1,-1,-2)),r,tetra,M)

ind.gam2<-ind.gamn2<-vector()
for (i in 1:(n-1))
 for (j in (i+1):n)
 {if (Gam2PEtetra(dat[i,],dat[j,],dat,r,tetra,M)==1)
 {ind.gam2<-rbind(ind.gam2,c(i,j))
 }
 }
ind.gam2
```

```
#or try
rv1<-rv.tetraCC(dat[1,],tetra)$rv;rv2<-rv.tetraCC(dat[2,],tetra)$rv
Gam2PEtetra(dat[1,],dat[2,],dat,r,tetra,M,rv1,rv2)

#or try
rv1<-rv.tetraCC(dat[1,],tetra)$rv;
Gam2PEtetra(dat[1,],dat[2,],dat,r,tetra,M,rv1)

#or try
rv2<-rv.tetraCC(dat[2,],tetra)$rv
Gam2PEtetra(dat[1,],dat[2,],dat,r,tetra,M,rv2=rv2)

P1<-c(.1,.1,.1)
P2<-c(.4,.1,.2)
Gam2PEtetra(P1,P2,dat,r,tetra,M)

dat.fr<-data.frame(a=dat)
Gam2PEtetra(P1,P2,dat.fr,r,tetra,M)

Gam2PEtetra(c(-1,-1,-1),dat[2,],dat,r,tetra,M,ch.data.pnts = TRUE)
#gives an error message since not both points, pt1 and pt2, are data points in Dt

## End(Not run)
```

`Gam2PEtri` *The indicator for two points constituting a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns I({`pt1`,`pt2`} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 2D data set `Dt`), that is, returns 1 if {`pt1`,`pt2`} is a dominating set of PE-PCD, returns 0 otherwise.

Point, `pt1`, is in the region of vertex `rv1` (default is `NULL`) and point, `pt2`, is in the region of vertex `rv2` (default is `NULL`); vertices (and hence `rv1` and `rv2`) are labeled as 1,2,3 in the order they are stacked row-wise in `tri`.

PE proximity regions are defined with respect to the triangle `tri` and vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

`ch.data.pnts` is for checking whether points `pt1` and `pt2` are data points in `Dt` or not (default is `FALSE`), so by default this function checks whether the points `pt1` and `pt2` would be a dominating set if they actually were in the data set.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
Gam2PEtri(
  pt1,
  pt2,
  Dt,
  tri,
  r,
  M = c(1, 1, 1),
  rv1 = NULL,
  rv2 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

pt1, pt2	Two 2D points to be tested for constituting a dominating set of the PE-PCD.
Dt	A set of 2D points which constitutes the vertices of the PE-PCD.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.
rv1, rv2	The indices of the vertices whose regions contains pt1 and pt2, respectively. They take the vertex labels as 1,2,3 as in the row order of the vertices in tri (default is NULL for both).
ch.data.pnts	A logical argument for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE).

Value

I({pt1,pt2} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 2D data set Dt), that is, returns 1 if {pt1,pt2} is a dominating set of PE-PCD, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

Gam2PEbastri, Gam2AStri, and Gam2PEtetra

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5  #try also r<-2

Gam2PEtri(dat[1,],dat[2,],dat,Tr,r,M)
Gam2PEtri(dat[1,],dat[4,],dat,Tr,r,M)
Gam2PEtri(dat[4,],dat[4,],dat,Tr,r,M)

Gam2PEtri(dat[1,],c(1,2),dat,Tr,r,M)

Gam2PEtri(c(1,2),c(1,3),rbind(c(1,2),c(1,3)),Tr,r,M)

ind.gam2<-vector()
for (i in 1:(n-1))
  for (j in (i+1):n)
  {if (Gam2PEtri(dat[i,],dat[j,],dat,Tr,r,M)==1)
    ind.gam2<-rbind(ind.gam2,c(i,j))}
ind.gam2

#or try
rv1<-rv.tri.cent(dat[1,],Tr,M)$rv;
rv2<-rv.tri.cent(dat[2,],Tr,M)$rv
Gam2PEtri(dat[1,],dat[2,],dat,Tr,r,M,rv1,rv2)

#or try
rv1<-rv.tri.cent(dat[1,],Tr,M)$rv;
Gam2PEtri(dat[1,],dat[2,],dat,Tr,r,M,rv1)

#or try
rv2<-rv.tri.cent(dat[2,],Tr,M)$rv
Gam2PEtri(dat[1,],dat[2,],dat,Tr,r,M,rv2=rv2)

P1<-c(1.4,1)
P2<-c(1.6,1)
Gam2PEtri(P1,P2,dat,Tr,r,M)
```

```
Gam2PEtri(dat[1,],dat[2,],dat,Tr,r,M)

dat.fr<-data.frame(a=dat)
Gam2PEtri(P1,P2,dat.fr,Tr,r,M)

dat.fr<-data.frame(a=Tr)
Gam2PEtri(P1,P2,dat,dat.fr,r,M)

## Not run:
Gam2PEtri(dat[1,],c(1,2),dat,Tr,r,M,ch.data.pnts = TRUE)
#gives an error message since not both points, pt1 and pt2, are data points in Dt

## End(Not run)
```

`Gam3PEstdTetra` *The indicator for three 3D points constituting a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard regular tetrahedron case*

Description

Returns I({`pt1`,`pt2`,`pt3`} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 3D data set `Dt` in the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/3))$, that is, returns 1 if {`pt1`,`pt2`,`pt3`} is a dominating set of PE-PCD, returns 0 otherwise.

Point, `pt1`, is in the region of vertex `rv1` (default is `NULL`), point, `pt2`, is in the region of vertex `rv2` (default is `NULL`); point, `pt3`), is in the region of vertex `rv3`) (default is `NULL`); vertices (and hence `rv1`,`rv2` and `rv3`) are labeled as 1,2,3,4 in the order they are stacked row-wise in T_h.

PE proximity region is constructed with respect to the tetrahedron T_h with expansion parameter $r \geq 1$ and vertex regions are based on center of mass CM (equivalent to circumcenter in this case).

`ch.data.pnts` is for checking whether points `pt1`, `pt2` and `pt3` are all data points in `Dt` or not (default is `FALSE`), so by default this function checks whether the points `pt1`, `pt2` and `pt3` would constitute a dominating set if they actually were all in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam3PEstdTetra(
  pt1,
  pt2,
  pt3,
  Dt,
  r,
  rv1 = NULL,
  rv2 = NULL,
  rv3 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

pt1, pt2, pt3	Three 3D points to be tested for constituting a dominating set of the PE-PCD.
Dt	A set of 3D points which constitutes the vertices of the PE-PCD.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
rv1, rv2, rv3	The indices of the vertices whose regions contains pt1, pt2 and pt3, respectively. They take the vertex labels as 1,2,3,4 as in the row order of the vertices in T_h (default is NULL for all).
ch.data.pnts	A logical argument for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE).

Value

I({pt1,pt2,pt3} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 3D data set Dt), that is, returns 1 if {pt1,pt2,pt3} is a dominating set of PE-PCD, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

Gam3PEtetra

Examples

```
## Not run:
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-20
dat<-runif.stdtetra(n)$g  #try also dat<-cbind(runif(n),runif(n),runif(n))
r<-1.25

Gam3PEstdTetra(dat[1,],dat[2,],dat[3,],dat,r)

Gam3PEstdTetra(dat[1,],dat[2,],dat[2,],dat,r)

Gam3PEstdTetra(dat[1,],c(1,1,1),dat[3,],dat,r)

Gam3PEstdTetra(c(-1,1,1),c(1,1,1),c(1,1,-1),rbind(c(-1,1,1),c(1,1,1),c(1,1,-1)),r)

ind.gam3<-vector()
for (i in 1:(n-2))
```

```
 for (j in (i+1):(n-1))
   for (k in (j+1):n)
 {if (Gam3PEstdTetra(dat[i,],dat[j,],dat[k,],dat,r)==1)
   ind.gam3<-rbind(ind.gam3,c(i,j,k))}

ind.gam3

#or try
rv1<-rv.tetraCC(dat[1,],tetra)$rv; rv2<-rv.tetraCC(dat[2,],tetra)$rv;
rv3<-rv.tetraCC(dat[3,],tetra)$rv
Gam3PEstdTetra(dat[1,],dat[2,],dat[3,],dat,r,rv1,rv2,rv3)

#or try
rv1<-rv.tetraCC(dat[1,],tetra)$rv;
Gam3PEstdTetra(dat[1,],dat[2,],dat[3,],dat,r,rv1)

#or try
rv2<-rv.tetraCC(dat[2,],tetra)$rv
Gam3PEstdTetra(dat[1,],dat[2,],dat[3,],dat,r,rv2=rv2)

P1<-c(.1,.1,.1)
P2<-c(.3,.3,.3)
P3<-c(.4,.1,.2)
Gam3PEstdTetra(P1,P2,P3,dat,r)

dat.fr<-data.frame(a=dat)
Gam3PEstdTetra(P1,P2,P3,dat.fr,r)

Gam3PEstdTetra(dat[1,],c(1,1,1),dat[3,],dat,r,ch.data.pnts = TRUE)
#gives an error message since not all points, pt1, pt2 and pt3, are data points in Dt

## End(Not run)
```

`Gam3PEtetra` *The indicator for three 3D points constituting a dominating set for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one tetrahedron case*

Description

Returns I({`pt1,pt2,pt3`} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 3D data set `Dt` in the tetrahedron `th`, that is, returns 1 if {`pt1,pt2,pt3`} is a dominating set of PE-PCD, returns 0 otherwise.

Point, `pt1`, is in the region of vertex `rv1` (default is `NULL`), point, `pt2`, is in the region of vertex `rv2` (default is `NULL`); point, `pt3`), is in the region of vertex `rv3`) (default is `NULL`); vertices (and hence `rv1,rv2` and `rv3`) are labeled as 1,2,3,4 in the order they are stacked row-wise in `th`.

PE proximity region is constructed with respect to the tetrahedron `th` with expansion parameter $r \geq 1$ and vertex regions are based on center of mass CM (equivalent to circumcenter in this case).

ch.data.pnts is for checking whether points pt1, pt2 and pt3 are all data points in Dt or not (default is FALSE), so by default this function checks whether the points pt1, pt2 and pt3 would constitute a dominating set if they actually were all in the data set.

See also (Ceyhan (2005, 2010)).

Usage

```
Gam3PEtetra(
  pt1,
  pt2,
  pt3,
  Dt,
  r,
  th,
  M = "CM",
  rv1 = NULL,
  rv2 = NULL,
  rv3 = NULL,
  ch.data.pnts = FALSE
)
```

Arguments

pt1, pt2, pt3	Three 3D points to be tested for constituting a dominating set of the PE-PCD.
Dt	A set of 3D points which constitutes the vertices of the PE-PCD.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
th	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.
M	The center to be used in the construction of the vertex regions in the tetrahedron, th. Currently it only takes "CC" for circumcenter and "CM" for center of mass; default="CM".
rv1, rv2, rv3	The indices of the vertices whose regions contains pt1, pt2 and pt3, respectively. They take the vertex labels as 1,2,3,4 as in the row order of the vertices in th (default is NULL for all).
ch.data.pnts	A logical argument for checking whether points pt1 and pt2 are data points in Dt or not (default is FALSE).

Value

I({pt1,pt2,pt3} is a dominating set of the PE-PCD) where the vertices of the PE-PCD are the 3D data set Dt), that is, returns 1 if {pt1,pt2,pt3} is a dominating set of PE-PCD, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties,*

and Applications". Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

Gam3PEstdTetra

Examples

```
## Not run:
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-20
dat<-runif.tetra(n,tetra)$g

M<-"CM";  #try also M<-"CC";
r<-1.25

Gam3PEtetra(dat[1,],dat[2,],dat[3,],dat,r,tetra,M)

Gam3PEtetra(dat[1,],dat[2,],dat[2,],dat,r,tetra,M)

Gam3PEtetra(dat[1,],c(1,1,1),dat[3,],dat,r,tetra,M)

Gam3PEtetra(c(-1,1,1),c(1,1,1),c(1,1,-1),rbind(c(-1,1,1),c(1,1,1),c(1,1,-1)),r,tetra,M)

ind.gam3<-vector()
for (i in 1:(n-2))
 for (j in (i+1):(n-1))
   for (k in (j+1):n)
   {if (Gam3PEtetra(dat[i,],dat[j,],dat[k,],dat,r,tetra,M)==1)
     ind.gam3<-rbind(ind.gam3,c(i,j,k))}

ind.gam3

#or try
rv1<-rv.tetraCC(dat[1,],tetra)$rv; rv2<-rv.tetraCC(dat[2,],tetra)$rv;
rv3<-rv.tetraCC(dat[3,],tetra)$rv
Gam3PEtetra(dat[1,],dat[2,],dat[3,],dat,r,tetra,M,rv1,rv2,rv3)

#or try
rv1<-rv.tetraCC(dat[1,],tetra)$rv;
Gam3PEtetra(dat[1,],dat[2,],dat[3,],dat,r,tetra,M,rv1)

#or try
rv2<-rv.tetraCC(dat[2,],tetra)$rv
Gam3PEtetra(dat[1,],dat[2,],dat[3,],dat,r,tetra,M,rv2=rv2)

P1<-c(.1,.1,.1)
P2<-c(.3,.3,.3)
```

```
P3<-c(.4,.1,.2)
Gam3PEtetra(P1,P2,P3,dat,r,tetra,M)

dat.fr<-data.frame(a=dat)
Gam3PEtetra(P1,P2,P3,dat.fr,r,tetra,M)

Gam3PEtetra(dat[1,],c(1,1,1),dat[3,],dat,r,tetra,M,ch.data.pnts = TRUE)
#gives an error message since not all points, pt1, pt2 and pts, are data points in Dt

## End(Not run)
```

`in.circle` *Check whether a point is inside a circle*

Description

Checks if the point `pt` lies in the circle with center `cent` and radius `rad`, denoted as `C(cent,rad)`. So, it returns 1 or `TRUE` if `pt` is inside the circle, and 0 otherwise.

`boundary` is a logical argument (default=`FALSE`) to include boundary or not, so if it is `TRUE`, the function checks if the point, `pt`, lies in the closure of the circle (i.e. interior and boundary combined) else it checks if `pt` lies in the interior of the circle.

Usage

```
in.circle(pt, cent, rad, boundary = FALSE)
```

Arguments

`pt`	A 2D point to be checked whether it is inside the circle or not.
`cent`	A 2D point in Cartesian coordinates which serves as the center of the circle.
`rad`	A positive real number which serves as the radius of the circle.
`boundary`	A logical parameter (default=`FALSE`) to include boundary or not, so if it is `TRUE`, the function checks if the point, `pt`, lies in the closure of the circle (i.e. interior and boundary combined) else it checks if `pt` lies in the interior of the circle.

Value

Indicator for the point `pt` being inside the circle or not, i.e., returns 1 or `TRUE` if `pt` is inside the circle, and 0 otherwise.

See Also

`in.triangle`, `in.tetrahedron`, and `on.convex.hull` from the `interp` package for documentation for `in.convex.hull`

Examples

```
cent<-c(1,1); rad<-1; p<-c(1.4,1.2)
#try also cent<-runif(2); rad<-runif(1); p<-runif(2);

in.circle(p,cent,rad)

p<-c(.4,-.2)
in.circle(p,cent,rad)

p<-c(1,0)
in.circle(p,cent,rad)
in.circle(p,cent,rad,boundary=TRUE)

#for a NA entry point
p<-c(NA,.2)
in.circle(p,cent,rad)
```

---

in.tetrahedron *Check whether a point is inside a tetrahedron*

Description

Checks if the point p lies in the tetrahedron, th, using the barycentric coordinates, generally denoted as (α, β, γ). If all (normalized or non-normalized) barycentric coordinates are positive then the point p is inside the tetrahedron, if all are nonnegative with one or more are zero, then p falls on the boundary. If some of the barycentric coordinates are negative, then p falls outside the tetrahedron.

boundary is a logical argument (default=FALSE) to include boundary or not, so if it is TRUE, the function checks if the point, p, lies in the closure of the tetrahedron (i.e. interior and boundary combined) else it checks if p lies in the interior of the tetrahedron.

Usage

```
in.tetrahedron(p, th, boundary = FALSE)
```

Arguments

p	A 3D point to be checked whether it is inside the tetrahedron or not.
th	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.
boundary	A logical parameter (default=FALSE) to include boundary or not, so if it is TRUE, the function checks if the point, p, lies in the closure of the tetrahedron (i.e. interior and boundary combined) else it checks if p lies in the interior of the tetrahedron.

Value

A list with two elements

in.tetra	A logical output, if the point, p, is inside the tetrahedron, th, it is TRUE, else it is FALSE.
barycentric	The barycentric coordinates of the point p with respect to the tetrahedron, th.

See Also

in.triangle

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3); P<-c(.1,.1,.1)
tetra<-rbind(A,B,C,D)

in.tetrahedron(P,tetra,boundary=FALSE)

in.tetrahedron(C,tetra,boundary=FALSE)
in.tetrahedron(C,tetra,boundary=TRUE)

n1<-5; n2<-5; n<-n1+n2
Dt<-rbind(cbind(runif(n1),runif(n1,0,sqrt(3)/2),runif(n1,0,sqrt(6)/3)),
          runif.tetra(n2,tetra)$g)

in.tetra<-vector()
for (i in 1:n)
{in.tetra<-c(in.tetra,in.tetrahedron(Dt[i,],tetra,boundary=TRUE)$in.tetra) }

in.tetra
Dt.tet<-Dt[in.tetra,]
if (is.vector(Dt.tet)) {Dt.tet<-matrix(Dt.tet,nrow=1)}

Xlim<-range(tetra[,1],Dt[,1])
Ylim<-range(tetra[,2],Dt[,2])
Zlim<-range(tetra[,3],Dt[,3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(Dt[,1],Dt[,2],Dt[,3], phi=40,theta=40, bty = "g", pch = 20, cex = 1,
ticktype="detailed",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),zlim=Zlim+zd*c(-.05,.05))
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
plot3D::points3D(Dt.tet[,1],Dt.tet[,2],Dt.tet[,3],pch=4, add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3], labels=c("A","B","C","D"), add=TRUE)

in.tetrahedron(P,tetra) #this works fine
```

```
dat.fr<-data.frame(a=tetra) #this works fine
in.tetrahedron(P,dat.fr)
```

`in.triangle` *Check whether a point is inside a triangle*

Description

Checks if the point `p` lies in the triangle, `tri`, using the barycentric coordinates, generally denoted as (α, β, γ).

If all (normalized or non-normalized) barycentric coordinates are positive then the point `p` is inside the triangle, if all are nonnegative with one or more are zero, then `p` falls in the boundary. If some of the barycentric coordinates are negative, then `p` falls outside the triangle.

`boundary` is a logical argument (default=`FALSE`) to include boundary or not, so if it is `TRUE`, the function checks if the point, `p`, lies in the closure of the triangle (i.e. interior and boundary combined) else it checks if `p` lies in the interior of the triangle.

Usage

```
in.triangle(p, tri, boundary = FALSE)
```

Arguments

`p`	A 2D point to be checked whether it is inside the triangle or not.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`boundary`	A logical parameter (default=`FALSE`) to include boundary or not, so if it is `TRUE`, the function checks if the point, `p`, lies in the closure of the triangle (i.e. interior and boundary combined) else it checks if `p` lies in the interior of the triangle.

Value

A list with two elements

`in.tri`	A logical output, it is `TRUE`, if the point, `p`, is inside the triangle, `tri`, else it is `FALSE`.
`barycentric`	The barycentric coordinates (alpha, beta, gamma) of the point `p` with respect to the triangle, `tri`.

See Also

`inTriAll` and `on.convex.hull` from the `interp` package for documentation for `in.convex.hull`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2); p<-c(1.4,1.2)

Tr<-rbind(A,B,C)

in.triangle(p,Tr)

p<-c(.4,-.2)
in.triangle(p,Tr)

#for the vertex A
in.triangle(A,Tr)
in.triangle(A,Tr,boundary=TRUE)

#for a point on the edge AB
D3<-(A+B)/2
in.triangle(D3,Tr)
in.triangle(D3,Tr,boundary=TRUE)

#for a NA entry point
p<-c(NA,.2)
in.triangle(p,Tr)

dat.fr<-data.frame(a=Tr)
in.triangle(p,dat.fr)

## Not run:
#when triangle is degenerate
B<-A+2*(C-A);
Tr<-rbind(A,B,C)
p<-c(.4,.2)
in.triangle(p,Tr)

## End(Not run)
```

IncMatASMT *Incidence matrix for Arc Slice Proximity Catch Digraphs (AS-PCDs) - multiple triangle case*

Description

Returns the incidence matrix for the AS-PCD whose vertices are a given 2D numerical data set, Xp, in the convex hull of Yp which is partitioned by the Delaunay triangles based on Yp points.

AS proximity regions are defined with respect to the Delaunay triangles based on Yp points and vertex regions are based on the center M="CC" for circumcenter of each Delaunay triangle or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle; default is M="CC" i.e. circumcenter of each triangle. Loops are allowed, so the diagonal entries are all equal to 1.

See (Ceyhan (2005, 2010)) for more on AS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
IncMatASMT(Xp, Yp, M = "CC")
```

Arguments

Xp	A set of 2D points which constitute the vertices of the AS-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
M	The center of the triangle. "CC" stands for circumcenter of each Delaunay triangle `tri` or 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle; default is `M`="CC" i.e. the circumcenter of each triangle.

Value

Incidence matrix for the AS-PCD whose vertices are the 2D data set, `Xp`, and AS proximity regions are defined in the Delaunay triangles based on `Yp` points.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

`IncMatAStri`, `IncMatPEMT`, and `IncMatCSMT`

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-"CC"  #try also M<-c(1,1,1)
```

```
IM<-IncMatASMT(Xp,Yp,M)
IM
dom.greedy(IM)  #try also dom.exact(IM)  #this might take a long time for large  nx

IM<-IncMatASMT(Xp,Yp[1:3,],M)

IncMatASMT(Xp,rbind(Yp,Yp))

dat.fr<-data.frame(a=Xp)
IncMatASMT(dat.fr,Yp,M)
```

`IncMatAStri` *Incidence matrix for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns the incidence matrix for the AS-PCD whose vertices are the given 2D numerical data set, `dat`.

AS proximity regions are defined with respect to the triangle $tri = T(v = 1, v = 2, v = 3)$ and vertex regions based on the center `M`="CC" for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M`="CC" i.e. circumcenter of `tri`. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005, 2010)).

Usage

```
IncMatAStri(dat, tri, M = "CC")
```

Arguments

`dat`	A set of 2D points which constitute the vertices of AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. "CC" stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of `tri`; default is `M`="CC" i.e. the circumcenter of `tri`.

Value

Incidence matrix for the AS-PCD whose vertices are 2D data set, `dat`, and AS proximity regions are defined with respect to the triangle `tri` and vertex regions based on circumcenter.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

IncMatASMT, IncMatPEtri, and IncMatCStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

IM<-IncMatAStri(dat,Tr,M)
IM

dom.greedy(IM)
dom.exact(IM)

dat.fr<-data.frame(a=dat)
IncMatAStri(dat.fr,Tr,M)
```

IncMatCS1D *Incidence matrix for Central Similarity Proximity Catch Digraphs (CS-PCDs) for 1D data - multiple interval case*

Description

Returns the incidence matrix for the CS-PCD for a given 1D numerical data set, Xp, as the vertices of the digraph and Yp determines the end points of the intervals (in the multi-interval case). Loops are allowed, so the diagonal entries are all equal to 1.

CS proximity region is constructed with an expansion parameter $t > 0$ and a centrality parameter c in $(0, 1)$.

See also (Ceyhan (2016)).

Usage

```
IncMatCS1D(Xp, Yp, t, c)
```

Arguments

Xp	a set of 1D points which constitutes the vertices of the digraph.
Yp	a set of 1D points which constitutes the end points of the intervals that partition the real line.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

Incidence matrix for the CS-PCD with vertices being 1D data set, Xp, and Yp determines the end points of the intervals (the multi-interval case)

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

IncMatCS1D, IncMatPEtri, and IncMatPEMT

Examples

```
t<-2
c<-.4
a<-0; b<-10;
nx<-10; ny<-4

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

IM<-IncMatCS1D(Xp,Yp,t,c)
IM
dom.greedy(IM)

dom.exact(IM)  #might take a long time depending on nx

IndUBdom(IM,5)

Arcs<-ArcsCSMI(Xp,Yp,t,c)
Arcs
summary(Arcs)
plot(Arcs)
```

```
IncMatCS1D(Xp,Yp+10,t,c)

t<-2
c<-.4
a<-0; b<-10;
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

IncMatCS1D(Xp,Yp,t,c)
```

`IncMatCSMT` *Incidence matrix for Central Similarity Proximity Catch Digraphs (CS-PCDs) - multiple triangle case*

Description

Returns the incidence matrix of Central Similarity Proximity Catch Digraph (CS-PCD) whose vertices are the data points in `Xp` in the multiple triangle case.

CS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $t > 0$ and edge regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (unscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the incidence matrix loops are allowed, so the diagonal entries are all equal to 1.

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) for more on CS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
IncMatCSMT(Xp, Yp, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.

M — A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle, default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

Incidence matrix for the CS-PCD with vertices being 2D data set, Xp. CS proximity regions are constructed with respect to the Delaunay triangles and M-edge regions.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

IncMatCStri, IncMatCSTe, IncMatASMT, and IncMatPEMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

t<-1.5  #try also t<-2

IM<-IncMatCSMT(Xp,Yp,t,M)
IM
dom.greedy(IM)

dom.exact(IM)  #takes a very long time for large nx, try smaller nx
```

```
IndUBdom(IM,3)  #takes a very long time for large nx, try smaller nx


IncMatCSMT(Xp,Yp,t,M)
IncMatCSMT(Xp,Yp[1:3,],t,M)

IncMatCSMT(Xp,rbind(Yp,Yp),t,M)

dat.fr<-data.frame(a=Xp)
IncMatCSMT(dat.fr,Yp,t,M)

dat.fr<-data.frame(a=Yp)
IncMatCSMT(Xp,dat.fr,t,M)
```

`IncMatCSTe` *Incidence matrix for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns the incidence matrix for the CS-PCD whose vertices are the given 2D numerical data set, `dat`, in the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$.

CS proximity region is defined with respect to the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
IncMatCSTe(dat, t, M = c(1, 1, 1))
```

Arguments

`dat`	A set of 2D points which constitute the vertices of the CS-PCD.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates. which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e.

Value

Incidence matrix for the CS-PCD with vertices being 2D data set, `dat` and CS proximity regions are defined in the standard equilateral triangle T_e with `M`-edge regions.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

IncMatCStri, IncMatCSMT and IncMatPETe

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

NumArcsCSTe(dat,t=1.25)

inc.mat<-IncMatCSTe(dat,t=1.25,M)
inc.mat
sum(inc.mat)-n

dom.greedy(inc.mat)

dom.exact(inc.mat)  #might take a long time for large n

IndUBdom(inc.mat,1)

inc.mat<-IncMatCSTe(rbind(dat,c(0,1)),t=1.25,M)
inc.mat
sum(inc.mat)-(n+1)

IncMatCSTe(dat,t=1.5,M);

IncMatCSTe(rbind(dat,dat),t=1.5,M)

dat.fr<-data.frame(a=dat)
IncMatCSTe(dat.fr,t=1.5,M);
```

IncMatCStri *Incidence matrix for Central Similarity Proximity Catch Digraphs (CS-PCDs) - one triangle case*

Description

Returns the incidence matrix for the CS-PCD whose vertices are the given 2D numerical data set, dat, in the triangle $tri = T(v = 1, v = 2, v = 3)$.

CS proximity regions are constructed with respect to triangle tri with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
IncMatCStri(dat, tri, t, M = c(1, 1, 1))
```

Arguments

dat	A set of 2D points which constitute the vertices of CS-PCD.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
t	A positive real number which serves as the expansion parameter in CS proximity region.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.

Value

Incidence matrix for the CS-PCD with vertices being 2D data set, dat, in the triangle tri with edge regions based on center M

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

IncMatCSMT, IncMatPEtri, and IncMatAStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

IM<-IncMatCStri(dat,Tr,t=1.25,M)
IM
dom.greedy(IM)
dom.exact(IM)
IndUBdom(IM,3)

IncMatCStri(dat,Tr,t=1.5,M)
IncMatCStri(dat,Tr,t=2,M)

t<-2
IncMatCStri(dat,Tr,t,M)

dat.fr<-data.frame(a=dat)
IncMatCStri(dat.fr,Tr,t,M)

dat.fr<-data.frame(a=Tr)
IncMatCStri(dat,dat.fr,t,M)
```

IncMatPE1D — *Incidence matrix for Proportiona-Edge Proximity Catch Digraphs (PE-PCDs) for 1D data - multiple interval case*

Description

Returns the incidence matrix for the PE-PCD for a given 1D numerical data set, Xp, as the vertices of the digraph and Yp determines the end points of the intervals (in the multi-interval case). Loops are allowed, so the diagonal entries are all equal to 1.

PE proximity region is constructed with an expansion parameter $r \geq 1$ and a centrality parameter c in $(0, 1)$.

See also (Ceyhan (2012)).

Usage

```
IncMatPE1D(Xp, Yp, r, c)
```

Arguments

Xp	a set of 1D points which constitutes the vertices of the digraph.
Yp	a set of 1D points which constitutes the end points of the intervals that partition the real line.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

Incidence matrix for the PE-PCD with vertices being 1D data set, Xp, and Yp determines the end points of the intervals (in the multi-interval case)

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

IncMatCS1D, IncMatPEtri, and IncMatPEMT

Examples

```
r<-2
c<-.4
a<-0; b<-10;
nx<-10; ny<-4

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

IM<-IncMatPE1D(Xp,Yp,r,c)
IM

dom.greedy(IM)
IndUBdom(IM,6)
dom.exact(IM)

Arcs<-ArcsPEMI(Xp,Yp,r,c)
Arcs
summary(Arcs)
plot(Arcs)
```

```
IncMatPE1D(Xp,Yp+10,r,c)

r<-2
c<-.4
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)
IncMatPE1D(Xp,Yp,r,c)
```

`IncMatPEMT` *Incidence matrix for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - multiple triangle case*

Description

Returns the incidence matrix of Proportional Edge Proximity Catch Digraph (PE-PCD) whose vertices are the data points in `Xp` in the multiple triangle case.

PE proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $r \geq 1$ and vertex regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle).

Each Delaunay triangle is first converted to an (unscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the incidence matrix loops are allowed, so the diagonal entries are all equal to 1.

See (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)) for more on the PE-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
IncMatPEMT(Xp, Yp, r, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this argument should be set as `M="CC"`), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

Incidence matrix for the PE-PCD with vertices being 2D data set, Xp. PE proximity regions are constructed with respect to the Delaunay triangles and M-vertex regions.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

IncMatPEtri, IncMatPETe, IncMatASMT, and IncMatCSMT

Examples

```
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

r<-1.5  #try also r<-2

IM<-IncMatPEMT(Xp,Yp,r,M)
IM
dom.greedy(IM)

dom.exact(IM)  #might take a long time in this brute-force fashion ignoring the
#disconnected nature of the digraph inherent by the geometric construction of it

PEdomMTnd(Xp,Yp,r)
```

```
Arcs<-ArcsPEMT(Xp,Yp,r,M)
Arcs
summary(Arcs)
plot(Arcs)

IncMatPEMT(Xp,Yp,r,M)
IncMatPEMT(Xp,Yp[1:3,],r,M)

IncMatPEMT(Xp,rbind(Yp,Yp),r,M)

dat.fr<-data.frame(a=Xp)
IncMatPEMT(dat.fr,Yp,r,M)

dat.fr<-data.frame(a=Yp)
IncMatPEMT(Xp,dat.fr,r,M)
```

`IncMatPETe` *Incidence matrix for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard equilateral triangle case*

Description

Returns the incidence matrix for the PE-PCD whose vertices are the given 2D numerical data set, `dat`, in the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$.

PE proximity region is constructed with respect to the standard equilateral triangle T_e with expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005, 2010)).

Usage

```
IncMatPETe(dat, r, M = c(1, 1, 1))
```

Arguments

`dat`	A set of 2D points which constitute the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e.

Value

Incidence matrix for the PE-PCD with vertices being 2D data set, `dat` and PE proximity regions are defined in the standard equilateral triangle T_e with `M`-vertex regions.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

IncMatPEtri, IncMatPEMT and IncMatCSTe

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

NumArcsPETe(dat,r=1.25)

inc.mat<-IncMatPETe(dat,r=1.25,M)
inc.mat
sum(inc.mat)-n

dom.greedy(inc.mat)
IndUBdom(inc.mat,2)
dom.exact(inc.mat)

inc.mat<-IncMatPETe(rbind(dat,c(0,1)),r=1.25,M)
inc.mat
sum(inc.mat)-(n+1)

IncMatPETe(dat,r=1.5,M);

IncMatPETe(rbind(dat,dat),r=1.5,M)

dat.fr<-data.frame(a=dat)
IncMatPETe(dat.fr,r=1.5,M);
```

IncMatPEtetra *Incidence matrix for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one tetrahedron case*

Description

Returns the incidence matrix for the PE-PCD whose vertices are the given 3D numerical data set, dat, in the tetrahedron $th = T(v = 1, v = 2, v = 3, v = 4)$.

PE proximity regions are constructed with respect to tetrahedron th with expansion parameter $r \geq 1$ and vertex regions are based on the center M which is circumcenter ("CC") or center of mass ("CM") of th with default="CM". Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005, 2010)).

Usage

```
IncMatPEtetra(dat, th, r, M = "CM")
```

Arguments

dat	A set of 3D points which constitute the vertices of PE-PCD.
th	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	The center to be used in the construction of the vertex regions in the tetrahedron, th. Currently it only takes "CC" for circumcenter and "CM" for center of mass; default="CM".

Value

Incidence matrix for the PE-PCD with vertices being 3D data set, dat, in the tetrahedron th with vertex regions based on circumcenter or center of mass

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

IncMatPEtri, IncMatPE1D, and IncMatPEMT

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-10

dat<-runif.tetra(n,tetra)$g  #try also dat<-c(.5,.5,.5)

M<-"CM"  #try also M<-"CC"
r<-1.5

IM<-IncMatPEtetra(dat,tetra,r=1.25)  #uses the default M="CM"
IM<-IncMatPEtetra(dat,tetra,r=1.25,M)
IM
dom.greedy(IM)
IndUBdom(IM,3)
dom.exact(IM)  #this might take a long time for large n

IncMatPEtetra(dat,tetra,r=1.5)
IncMatPEtetra(dat,tetra,r=2)

r<-2
IncMatPEtetra(dat,tetra,r,M)

dat.fr<-data.frame(a=dat)
IncMatPEtetra(dat.fr,tetra,r,M)

dat.fr<-data.frame(a=tetra)
IncMatPEtetra(dat,dat.fr,r,M)
```

`IncMatPEtri` *Incidence matrix for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns the incidence matrix for the PE-PCD whose vertices are the given 2D numerical data set, `dat`, in the triangle $tri = T(v = 1, v = 2, v = 3)$.

PE proximity regions are constructed with respect to triangle `tri` with expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`. Loops are allowed, so the diagonal entries are all equal to 1.

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)).

Usage

```
IncMatPEtri(dat, tri, r, M = c(1, 1, 1))
```

Arguments

dat	A set of 2D points which constitute the vertices of PE-PCD.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.

Value

Incidence matrix for the PE-PCD with vertices being 2D data set, dat, in the triangle tri with vertex regions based on center M

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

IncMatPEMT, IncMatCStri, and IncMatAStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

IM<-IncMatPEtri(dat,Tr,r=1.25,M)

IM
dom.greedy(IM)
dom.exact(IM)
IndUBdom(IM,2)
```

```
IndUBdom(IM,3)

IncMatPEtri(dat,Tr,r=1.5,M)
IncMatPEtri(dat,Tr,r=2,M)

r<-2
IncMatPEtri(dat,Tr,r,M)

dat.fr<-data.frame(a=dat)
IncMatPEtri(dat.fr,Tr,r,M)

dat.fr<-data.frame(a=Tr)
IncMatPEtri(dat,dat.fr,r,M)
```

`ind.int.set` *Indices of the intervals where the 1D point(s) reside*

Description

Returns the indices of intervals for all the points in 1D data set, `dat`, as a vector.

Intervals are based on `Yp` and left end interval is labeled as 0, the next interval as 1, and so on.

Usage

```
ind.int.set(dat, Yp)
```

Arguments

`dat`	A set of 1D points for which the indices of intervals are to be determined.
`Yp`	A set of 1D points from which intervals are constructed.

Value

The vector of indices of the intervals in which points in the 1D data set, `dat`, reside

Examples

```
a<-0; b<-10; int<-c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1
Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)  #try also Yp<-runif(ny,a+1,b-1)
```

```
ind<-ind.int.set(Xp,Yp)
ind

jit<-.1
yjit<-runif(nx,-jit,jit)

Xlim<-range(a,b,Xp,Yp)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0), xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=3*c(-jit,jit),pch=".")
abline(h=0)
points(Xp, yjit,pch=".",cex=3)
abline(v=Yp,lty=2)
text(Xp,yjit,labels=factor(ind))

ind.int.set(3,5)
ind.int.set(6,5)

ind.int.set(Xp,Yp)
```

`IndASdomUBtri` *Indicator for an upper bound for the domination number of Arc Slice Proximity Catch Digraph (AS-PCD) by the exact algorithm - one triangle case*

Description

Returns I(domination number of AS-PCD whose vertices are the data points `Dt` is less than or equal to `k`), that is, returns 1 if the domination number of CS-PCD is less than the prespecified value `k`, returns 0 otherwise. It also provides the vertices (i.e. data points) in a dominating set of size `k` of AS-PCD.

AS proximity regions are constructed with respect to the triangle `tri` and vertex regions are based on the center `M`="CC" for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M`="CC" i.e. circumcenter of `tri`.

The vertices of `tri`angle, `tri`, are labeled as 1,2,3 according to the row number the vertex is recorded in `tri`. Loops are allowed in the digraph. It takes a long time for large number of vertices (i.e., large number of row numbers).

Usage

```
IndASdomUBtri(Dt, k, tri, M = "CC")
```

Arguments

`Dt`	A set of 2D points which constitute the vertices of the AS-PCD.

k	A positive integer to be tested for an upper bound for the domination number of AS-PCDs.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	The center of the triangle. "CC" stands for circumcenter of the triangle tri or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of tri; default is M="CC" i.e. the circumcenter of tri.

Value

A list with the elements

domUB	The suggested upper bound (to be checked) for the domination number of AS-PCD. It is prespecified as k in the function arguments.
IndUBdom	The indicator for the upper bound for domination number of AS-PCD being the specified value k or not. It returns 1 if the upper bound is k, and 0 otherwise.
ind.domset	The vertices (i.e., data points) in the dominating set of size k if it exists, otherwise it yields NULL.

See Also

IndCSdomUBtri, IndCSdomUBTe, IndUBdom, and dom.exact

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

IndASdomUBtri(dat,1,Tr)

for (k in 1:n)
  print(c(k,IndASdomUBtri(dat,k,Tr,M)))

IndASdomUBtri(dat,k=4,Tr,M)

P<-c(.4,.2)
IndASdomUBtri(P,1,Tr,M)

IndASdomUBtri(rbind(dat,dat),k=2,Tr,M)

dat.fr<-data.frame(a=dat)
IndASdomUBtri(dat.fr,1,Tr,M)
```

IndCS.Te.onesixth *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - first one sixth of the standard equilateral triangle case*

Description

Returns I(pt2 is in $NCS(pt1, t = 1)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NCS(pt1, t = 1)$, returns 0 otherwise, where $NCS(x, t = 1)$ is the CS proximity region for point x with expansion parameter $t = 1$.

CS proximity region is defined with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ and edge regions are based on the center of mass $CM = (1/2, \sqrt{3}/6)$. Here pt1 must lie in the first one-sixth of T_e, which is the triangle with vertices $T(A, D_3, CM) = T((0, 0), (1/2, 0), CM)$. If pt1 and pt2 are distint and pt1 is outside of $T(A, D_3, CM)$ or pt2 is outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

Usage

```
IndCS.Te.onesixth(pt1, pt2)
```

Arguments

pt1	A 2D point whose CS proximity region is constructed.
pt2	A 2D point. The function determines whether pt2 is inside the CS proximity region of pt1 or not.

Value

I(pt2 is in $NCS(pt1, t = 1)$) for pt1 in the first one-sixth of T_e, $T(A, D_3, CM)$, that is, returns 1 if pt2 is in $NCS(pt1, t = 1)$, returns 0 otherwise

See Also

IndCSTe

Examples

```
## Not run:
n<-5

set.seed(1)
dat<-rbind(runifTe.onesixth(n)$gen.points,runifTe(n)$gen.points)

IndCS.Te.onesixth(dat[1,],dat[2,])
IndCS.Te.onesixth(dat[1,],dat[1,])
IndCS.Te.onesixth(dat[1,],dat[9,])
IndCS.Te.onesixth(c(.2,.5),dat[2,])
```

```
IndCS.Te.onesixth(c(.2,.5),c(.2,.5))

set.seed(1)
dat<-runifTe(10)$gen.points

IndCS.Te.onesixth(dat[1,],dat[2,])

## End(Not run)
```

IndCSdomUBTe *The indicator for* k *being an upper bound for the domination number of Central Similarity Proximity Catch Digraph (CS-PCD) by the exact algorithm - standard equilateral triangle case*

Description

Returns I(domination number of CS-PCD is less than or equal to k) where the vertices of the CS-PCD are the data points Dt, that is, returns 1 if the domination number of CS-PCD is less than the prespecified value k, returns 0 otherwise. It also provides the vertices (i.e. data points) in a dominating set of size k of CS-PCD.

CS proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e (which is equivalent to the circumcenter of T_e).

Edges of T_e, AB, BC, AC, are also labeled as 3, 1, and 2, respectively. Loops are allowed in the digraph. It takes a long time for large number of vertices (i.e., large number of row numbers).

See also (Ceyhan (2012)).

Usage

```
IndCSdomUBTe(Dt, k, t, M = c(1, 1, 1))
```

Arguments

Dt	A set of 2D points which constitute the vertices of CS-PCD.
k	A positive integer representing an upper bound for the domination number of CS-PCD.
t	A positive real number which serves as the expansion parameter in CS proximity region in the standard equilateral triangle $T_e = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.

Value

A list with two elements

domUB	The upper bound k (to be checked) for the domination number of CS-PCD. It is prespecified as k in the function arguments.
IndUBdom	The indicator for the upper bound for domination number of CS-PCD being the specified value k or not. It returns 1 if the upper bound is k, and 0 otherwise.
ind.domset	The vertices (i.e., data points) in the dominating set of size k if it exists, otherwise it is NULL.

References

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

IndCSdomUBtri, IndUBdom, IndASdomUBtri, and dom.exact

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

t<-.5

IndCSdomUBTe(dat,1,t,M)

for (k in 1:n)
  print(c(k,IndCSdomUBTe(dat,k,t,M)$IndUBdom))
  print(c(k,IndCSdomUBTe(dat,k,t,M)$domUB))

IndCSdomUBTe(dat,k=4,t,M)

P<-c(.4,.2)
IndCSdomUBTe(P,1,t,M)

IndCSdomUBTe(rbind(dat,dat),3,t,M)

dat.fr<-data.frame(a=dat)
IndCSdomUBTe(dat.fr,1,t,M)
```

`IndCSdomUBtri` *Indicator for an upper bound for the domination number of Central Similarity Proximity Catch Digraph (CS-PCD) by the exact algorithm - one triangle case*

Description

Returns I(domination number of CS-PCD is less than or equal to k) where the vertices of the CS-PCD are the data points Dt, that is, returns 1 if the domination number of CS-PCD is less than the prespecified value k, returns 0 otherwise. It also provides the vertices (i.e. data points) in a dominating set of size k of CS-PCD.

CS proximity region is constructed with respect to the triangle $tri = T(A, B, C)$ with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.

Edges of tri, AB, BC, AC, are also labeled as 3, 1, and 2, respectively. Loops are allowed in the digraph.

See also (Ceyhan (2012)).

Caveat: It takes a long time for large number of vertices (i.e., large number of row numbers).

Usage

```
IndCSdomUBtri(Dt, k, t, tri, M = c(1, 1, 1))
```

Arguments

Dt	A set of 2D points which constitute the vertices of CS-PCD.
k	A positive integer to be tested for an upper bound for the domination number of CS-PCDs.
t	A positive real number which serves as the expansion parameter in CS proximity region in the triangle tri.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri; default is $M = (1, 1, 1)$, i.e. the center of mass of tri.

Value

A list with two elements

domUB	The upper bound k (to be checked) for the domination number of CS-PCD. It is prespecified as k in the function arguments.
IndUBdom	The indicator for the upper bound for domination number of CS-PCD being the specified value k or not. It returns 1 if the upper bound is k, and 0 otherwise.

`ind.domset` The vertices (i.e., data points) in the dominating set of size `k` if it exists, otherwise it is `NULL`.

References

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`IndCSdomUBTe`, `IndUBdom`, `IndASdomUBtri`, and `dom.exact`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

t<-.5

IndCSdomUBtri(dat,1,t,Tr,M)

for (k in 1:n)
  print(c(k,IndCSdomUBtri(dat,k,t,Tr,M)))

IndCSdomUBtri(dat,k=4,t,Tr,M)

P<-c(.4,.2)
IndCSdomUBtri(P,1,t,Tr,M)

IndCSdomUBtri(rbind(dat,dat),3,t,Tr,M)

dat.fr<-data.frame(a=dat)
IndCSdomUBtri(dat.fr,1,t,Tr,M)
```

`IndCSTe` *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns I(pt2 is in $NCS(pt1, t)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NCS(pt1, t)$, returns 0 otherwise, where $NCS(x, t)$ is the CS proximity region for point x with expansion parameter $t > 0$.

CS proximity region is defined with respect to the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e. re is the index of the edge region pt1 resides, with default=NULL.

If pt1 and pt2 are distint and either of them are outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
IndCSTe(pt1, pt2, t, M = c(1, 1, 1), re = NULL)
```

Arguments

pt1	A 2D point whose CS proximity region is constructed.
pt2	A 2D point. The function determines whether pt2 is inside the CS proximity region of pt1 or not.
t	A positive real number which serves as the expansion parameter in CS proximity region.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.
re	The index of the edge region in T_e containing the point, either 1, 2, 3 or NULL (default is NULL).

Value

I(pt2 is in $NCS(pt1, t)$) for pt1, that is, returns 1 if pt2 is in $NCS(pt1, t)$, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

IndNCStri and IndNPETe

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

t<-1

IndCSTe(dat[1,],dat[2,],t,M)
IndCSTe(dat[1,],dat[1,],t,M)

IndCSTe(dat[1,],dat[2,],t=4,M)
IndCSTe(dat[2,],dat[5,],t,M)
IndCSTe(c(.2,.5),dat[2,],t,M)

IndCSTe(c(.2,.5),c(.2,.5),t,M)

#or try
re<-reTeCM(dat[1,])$re
IndCSTe(dat[1,],dat[2,],t,M,re=re)

IndCSTe(dat[1,],dat[2,],t,M)
```

IndCSTe.domset — *The indicator for the set of points* S *being a dominating set or not for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns I(S a dominating set of the CS-PCD) where the vertices of the CS-PCD are the data set Dt), that is, returns 1 if S is a dominating set of CS-PCD, returns 0 otherwise.

CS proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e (which is equivalent to the circumcenter of T_e).

Edges of T_e, AB, BC, AC, are also labeled as 3, 1, and 2, respectively.

See also (Ceyhan (2012)).

Usage

```
IndCSTe.domset(S, Dt, t, M = c(1, 1, 1))
```

Arguments

S	A set of 2D points which is to be tested for being a dominating set for the CS-PCDs.
Dt	A set of 2D points which constitute the vertices of the CS-PCD.
t	A positive real number which serves as the expansion parameter in CS proximity region in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e.

Value

I(S a dominating set of CS-PCD), that is, returns 1 if S is a dominating set of CS-PCD, returns 0 otherwise, where CS proximity region is constructed in the standard equilateral triangle T_e

References

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

IndNCStri.domset and IndNPETe.domset

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

t<-.5

S<-rbind(dat[1,],dat[2,])
IndCSTe.domset(S,dat,t,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,])
IndCSTe.domset(S,dat,t,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IndCSTe.domset(S,dat,t,M)
```

```
IndCSTe.domset(c(.2,.5),dat,t,M)
IndCSTe.domset(c(.2,.5),c(.2,.5),t,M)
IndCSTe.domset(dat[5,],dat[2,],t,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,],c(.2,.5))
IndCSTe.domset(S,dat[3,],t,M)

IndCSTe.domset(dat,dat,t,M)

P<-c(.4,.2)
S<-dat[c(1,3,4),]
IndCSTe.domset(dat,P,t,M)
IndCSTe.domset(S,P,t,M)
IndCSTe.domset(S,dat,t,M)

IndCSTe.domset(rbind(S,S),dat,t,M)

dat.fr<-data.frame(a=dat)
IndCSTe.domset(S,dat.fr,t,M)
```

`IndCSTeSet` *The indicator for the presence of an arc from a point in set* `S` *to the point* `pt` *for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns I(`pt` in $NCS(x,t)$ for some x in `S`), that is, returns 1 if `pt` is in $\cup_{xinS} NCS(x,t)$, returns 0 otherwise, CS proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A,B,C) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with the expansion parameter $t > 0$ and edge regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e (which is equivalent to circumcenter of T_e).

Edges of T_e, AB, BC, AC, are also labeled as edges 3, 1, and 2, respectively. If `pt` is not in `S` and either `pt` or all points in `S` are outside T_e, it returns 0, but if `pt` is in `S`, then it always returns 1 regardless of its location (i.e., loops are allowed).

See also (Ceyhan (2012)).

Usage

```
IndCSTeSet(S, pt, t, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points. Presence of an arc from a point in `S` to point `pt` is checked by the function.

pt	A 2D point. Presence of an arc from a point in S to point pt is checked by the function.
t	A positive real number which serves as the expansion parameter in CS proximity region in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e.

Value

I(pt is in $\cup_{x in S} NCS(x,t)$), that is, returns 1 if pt is in S or inside $NCS(x,t)$ for at least one x in S, returns 0 otherwise. CS proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A,B,C) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with M-edge regions.

References

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

IndNCStriSet, IndCSTe, IndNCStri, and IndNPETeSet

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

t<-.5

S<-rbind(dat[1,],dat[2,])  #try also S<-c(.5,.5)
IndCSTeSet(S,dat[3,],t,M)
IndCSTeSet(S,dat[3,],t=1,M)
IndCSTeSet(S,dat[3,],t=1.5,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,])
IndCSTeSet(S,dat[3,],t,M)

IndCSTeSet(S,dat[6,],t,M)
IndCSTeSet(S,dat[6,],t=.25,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IndCSTeSet(S,dat[3,],t,M)
```

```
IndCSTeSet(c(.2,.5),dat[2,],t,M)
IndCSTeSet(dat,c(.2,.5),t,M)
IndCSTeSet(dat,dat[2,],t)
IndCSTeSet(c(.2,.5),c(.2,.5),t,M)
IndCSTeSet(dat[5,],dat[2,],t,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,],c(.2,.5))
IndCSTeSet(S,dat[3,],t,M)

P<-c(.4,.2)
S<-dat[c(1,3,4),]
IndCSTeSet(dat,P,t,M)

IndCSTeSet(rbind(S,S),P,t,M)

dat.fr<-data.frame(a=S)
IndCSTeSet(dat.fr,P,t,M)
```

`IndCSTet1` *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case with* $t = 1$

Description

Returns I(`pt2` is in $NCS(pt1, t = 1)$) for points `pt1` and `pt2`, that is, returns 1 if `pt2` is in $NCS(pt1, t = 1)$, returns 0 otherwise, where $NCS(x, t = 1)$ is the CS proximity region for point x with expansion parameter $t = 1$.

CS proximity region is defined with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ and edge regions are based on the center of mass $CM = (1/2, \sqrt{3}/6)$.

If `pt1` and `pt2` are distint and either are outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

Usage

```
IndCSTet1(pt1, pt2)
```

Arguments

`pt1`	A 2D point whose CS proximity region is constructed.
`pt2`	A 2D point. The function determines whether `pt2` is inside the CS proximity region of `pt1` or not.

Value

I(`pt2` is in $NCS(pt1, t = 1)$) for `pt1` in T_e that is, returns 1 if `pt2` is in $NCS(pt1, t = 1)$, returns 0 otherwise

See Also

IndCSTe

Examples

```
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

IndCSTet1(dat[1,],dat[2,])
IndCSTet1(dat[1,],dat[1,])

IndCSTet1(dat[2,],dat[5,])
IndCSTet1(c(.2,.5),dat[2,])

IndCSTet1(c(.2,.5),c(.2,.5))

IndCSTet1(dat[1,],dat[7,])

## End(Not run)
```

IndNASbastri *The indicator for the presence of an arc from a point to another for Arc Slice Proximity Catch Digraphs (AS-PCDs) - basic triangle case*

Description

Returns $I(pt2 \in N_{AS}(pt1))$ for points pt1 and pt2, that is, returns 1 if pt2 is in NAS(pt1), returns 0 otherwise, where $N_{AS}(x)$ is the AS proximity region for point x.

AS proximity region is constructed in the basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where c1 is in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Vertex regions are based on the center M="CC" for circumcenter of T_b; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_b; default is M="CC" i.e. circumcenter of T_b. rv is the index of the vertex region pt1 resides, with default=NULL.

If pt1 and pt2 are distint and either of them are outside T_b, the function returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
IndNASbastri(pt1, pt2, c1, c2, M = "CC", rv = NULL)
```

Arguments

pt1	A 2D point whose AS proximity region is constructed.
pt2	A 2D point. The function determines whether pt2 is inside the AS proximity region of pt1 or not.
c1, c2	Positive real numbers representing the top vertex in basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, c1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.
M	The center of the triangle. "CC" stands for circumcenter or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is M="CC" i.e. the circumcenter of T_b.
rv	The index of the M-vertex region in T_b containing the point, either 1, 2, 3 or NULL (default is NULL).

Value

$I(pt2 \in N_{AS}(pt1))$ for points pt1 and pt2, that is, returns 1 if pt2 is in NAS(pt1) (i.e., if there is an arc from pt1 to pt2), returns 0 otherwise.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

IndNAStri and NAStri

Examples

```
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.2)

P1<-as.numeric(runif.bastri(1,c1,c2)$g)
P2<-as.numeric(runif.bastri(1,c1,c2)$g)
IndNASbastri(P1,P2,c1,c2,M)
```

```
P1<-c(.3,.2)
P2<-c(.6,.2)
IndNASbastri(P1,P2,c1,c2,M)

#or try
Rv<-rv.bastriCC(P1,c1,c2)$rv
IndNASbastri(P1,P2,c1,c2,M,Rv)

P1<-c(.3,.2)
P2<-c(.8,.2)
IndNASbastri(P1,P2,c1,c2,M)

P3<-c(.5,.4)
IndNASbastri(P1,P3,c1,c2,M)

P4<-c(1.5,.4)
IndNASbastri(P1,P4,c1,c2,M)
IndNASbastri(P4,P4,c1,c2,M)

c1<-.4; c2<-.6;
P1<-c(.3,.2)
P2<-c(.6,.2)
IndNASbastri(P1,P2,c1,c2,M)
```

`IndNAStri` *The indicator for the presence of an arc from a point to another for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns $I(pt2 \in N_{AS}(pt1))$ for points `pt1` and `pt2`, that is, returns 1 if `pt2` is in NAS(`pt1`), returns 0 otherwise, where $N_{AS}(x)$ is the AS proximity region for point x.

AS proximity regions are constructed with respect to the triangle, $tri = T(A, B, C) = (rv = 1, rv = 2, rv = 3)$, and vertex regions are based on the center `M`="CC" for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M`="CC" i.e. circumcenter of `tri`. `rv` is the index of the vertex region `pt1` resides, with default=`NULL`.

If `pt1` and `pt2` are distint and either of them are outside `tri`, the function returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005, 2010)).

Usage

```
IndNAStri(pt1, pt2, tri, M = "CC", rv = NULL)
```

Arguments

pt1	A 2D point whose AS proximity region is constructed.
pt2	A 2D point. The function determines whether pt2 is inside the AS proximity region of pt1 or not.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	The center of the triangle. "CC" stands for circumcenter of the triangle tri or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of tri; default is M="CC" i.e. the circumcenter of tri.
rv	The index of the M-vertex region in tri containing the point, either 1, 2, 3 or NULL (default is NULL).

Value

$I(pt2 \in N_{AS}(pt1))$ for pt1, that is, returns 1 if pt2 is in NAS(pt1), returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

IndNASbastri, IndNPEtri, and IndNCStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

P1<-as.numeric(runif.tri(1,Tr)$g)
P2<-as.numeric(runif.tri(1,Tr)$g)
IndNAStri(P1,P2,Tr,M)

P1<-c(1.3,1.2)
P2<-c(1.8,.5)
IndNAStri(P1,P2,Tr,M)
```

```
IndNAStri(P1,P1,Tr,M)

#or try
Rv<-rv.triCC(P1,Tr)$rv
IndNAStri(P1,P2,Tr,M,Rv)

P3<-c(1.6,1.4)
IndNAStri(P1,P3,Tr,M)

P4<-c(1.5,1.0)
IndNAStri(P1,P4,Tr,M)

P5<-c(.5,1.0)
IndNAStri(P1,P5,Tr,M)
IndNAStri(P5,P5,Tr,M)

#or try
Rv<-rv.triCC(P5,Tr)$rv
IndNAStri(P5,P5,Tr,M,Rv)

dat.fr<-data.frame(a=Tr)
IndNAStri(P1,P2,dat.fr,M)
```

`IndNAStri.domset` *The indicator for the set of points* `S` *being a dominating set or not for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns I(`S` a dominating set of AS-PCD), that is, returns 1 if `S` is a dominating set of AS-PCD, returns 0 otherwise.

AS-PCD has vertex set `Dt` and AS proximity region is constructed with vertex regions based on the center `M`="CC" for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M`="CC" i.e. circumcenter of `tri` whose vertices are also labeled as edges 1, 2, and 3, respectively.

See also (Ceyhan (2005, 2010)).

Usage

```
IndNAStri.domset(S, Dt, tri, M = "CC")
```

Arguments

`S`	A set of 2D points which is to be tested for being a dominating set for the AS-PCDs.
`Dt`	A set of 2D points which constitute the vertices of the AS-PCD.

tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	The center of the triangle. "CC" stands for circumcenter of the triangle tri or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of tri; default is M="CC" i.e. the circumcenter of tri.

Value

I(S a dominating set of AS-PCD), that is, returns 1 if S is a dominating set of AS-PCD whose vertices are the data points in Dt; returns 0 otherwise, where AS proximity region is constructed in the triangle tri.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

IndNAStriSet, IndNPEtri.domset and IndNCStri.domset

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

S<-rbind(dat[1,],dat[2,])
IndNAStri.domset(S,dat,Tr,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,])
IndNAStri.domset(S,dat,Tr,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IndNAStri.domset(S,dat,Tr,M)
```

```
IndNAStri.domset(c(.2,.5),dat,Tr,M)
IndNAStri.domset(c(.2,.5),c(.2,.5),Tr,M)
IndNAStri.domset(dat[5,],dat[2,],Tr,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,],c(.2,.5))
IndNAStri.domset(S,dat[3,],Tr,M)

IndNAStri.domset(dat,dat,Tr,M)

P<-c(.4,.2)
S<-dat[c(1,3,4),]
IndNAStri.domset(dat,P,Tr,M)
IndNAStri.domset(S,P,Tr,M)
IndNAStri.domset(S,dat,Tr,M)

IndNAStri.domset(rbind(S,S),dat,Tr,M)

dat.fr<-data.frame(a=dat)
IndNAStri.domset(S,dat.fr,Tr,M)
```

`IndNAStriSet` *The indicator for the presence of an arc from a point in set* `S` *to the point* `pt` *for Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns I($pt \in N_{AS}(x)$ for some $x \in S$), that is, returns 1 if `pt` is in $\cup_{x \in S} NAS(x)$, returns 0 otherwise, where $N_{AS}(x)$ is the AS proximity region for point x.

AS proximity regions are constructed with respect to the triangle, $tri = T(A, B, C) = (rv = 1, rv = 2, rv = 3)$, and vertices of `tri` are also labeled as 1,2, and 3, respectively.

Vertex regions are based on the center `M`="CC" for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M`="CC" i.e. circumcenter of `tri`.

If `pt` is not in `S` and either `pt` or all points in `S` are outside `tri`, it returns 0, but if `pt` is in `S`, then it always returns 1 (i.e., loops are allowed).

See also (Ceyhan (2005, 2010)).

Usage

```
IndNAStriSet(S, pt, tri, M = "CC")
```

Arguments

`S`	A set of 2D points whose AS proximity regions are considered.
`pt`	A 2D point. The function determines whether `pt` is inside the union of AS proximity regions of points in `S` or not.

tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	The center of the triangle. "CC" stands for circumcenter of the triangle tri or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of tri; default is M="CC" i.e. the circumcenter of tri.

Value

I(pt in U_x in SNAS(x,r)), that is, returns 1 if pt is in S or inside NAS(x) for at least one x in S, returns 0 otherwise, where AS proximity region is constructed in tri

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

IndNAStri, IndNAStriSet, and IndNCStriSet

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$gen.points

S<-rbind(dat[1,],dat[2,])  #try also S<-c(1.5,1)

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

IndNAStriSet(S,dat[3,],Tr,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,])
IndNAStriSet(S,dat[3,],Tr,M)

IndNAStriSet(S,dat[6,],Tr,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IndNAStriSet(S,dat[3,],Tr,M)
```

```
IndNAStriSet(c(.2,.5),dat[2,],Tr,M)
IndNAStriSet(dat,c(.2,.5),Tr,M)
IndNAStriSet(dat,dat[2,],Tr,M)
IndNAStriSet(c(.2,.5),c(.2,.5),Tr,M)
IndNAStriSet(dat[5,],dat[2,],Tr,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,],c(.2,.5))
IndNAStriSet(S,dat[3,],Tr,M)

P<-c(.4,.2)
S<-dat[c(1,3,4),]
IndNAStriSet(dat,P,Tr,M)
IndNAStriSet(S,P,Tr,M)

IndNAStriSet(rbind(S,S),P,Tr,M)

dat.fr<-data.frame(a=S)
IndNAStriSet(dat.fr,P,Tr,M)
```

`IndNCSend1D` *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - end interval case*

Description

Returns I(`x2` in $NCS(x1,t)$) for points `x1` and `x2`, that is, returns 1 if `x2` is in $NCS(x1,t)$, returns 0 otherwise, where $NCS(x,t)$ is the CS proximity region for point x with expansion parameter $t > 0$ for the region outside the interval (a, b).

`rv` is the index of the end vertex region `x1` resides, with default=`NULL`, and $rv = 1$ for left end interval and $rv = 2$ for the right end interval. If `x1` and `x2` are distint and either of them are inside interval `int`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2016)).

Usage

```
IndNCSend1D(x1, x2, t, int, rv = NULL)
```

Arguments

`x1`	A 1D point for which the CS proximity region is constructed.
`x2`	A 1D point to check whether it is inside the proximity region or not.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`int`	A vector of two real numbers representing an interval.
`rv`	Index of the end interval containing the point, either 1, 2 or `NULL` (default=`NULL`).

Value

I(x2 in $NCS(x1, t)$) for points x1 and x2, that is, returns 1 if x2 is in $NCS(x1, t)$ (i.e., if there is an arc from x1 to x2), returns 0 otherwise

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

IndNCSmid1D, IndNPEmid1D, and IndNPEend1D

Examples

```
a<-0; b<-10; int<-c(a,b)
t<-2

IndNCSend1D(15,17,t,int)
IndNCSend1D(15,15,t,int)

IndNCSend1D(1.5,17,t,int)
IndNCSend1D(1.5,1.5,t,int)

IndNCSend1D(-15,17,t,int)

IndNCSend1D(-15,-17,t,int)

a<-0; b<-10; int<-c(a,b)
t<-.5

IndNCSend1D(15,17,t,int)
```

IndNCSint

The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - one interval case

---

Description

Returns I(x2 in $NCS(x1, t, c)$) for points x1 and x2, that is, returns 1 if x2 is in $NCS(x1, t, c)$, returns 0 otherwise, where $NCS(x, t, c)$ is the CS proximity region for point x with expansion parameter $t > 0$ and centrality parameter c in $(0, 1)$.

CS proximity region is constructed with respect to the interval (a, b). This function works whether x1 and x2 are inside or outside the interval int.

Vertex regions for middle intervals are based on the center associated with the centrality parameter c in $(0, 1)$. If x1 and x2 are identical, then it returns 1 regardless of their locations (i.e., loops are allowed in the digraph).

See also (Ceyhan (2016)).

Usage

```
IndNCSint(x1, x2, t, c = 0.5, int)
```

Arguments

x1	A 1D point for which the proximity region is constructed.
x2	A 1D point for which it is checked whether it resides in the proximity region of x1 or not.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$, and default=0.5.
int	A vector of two real numbers representing an interval.

Value

I(x2 in $NCS(x1, t, c)$) for x2, that is, returns 1 if x2 in $NCS(x1, t, c)$, returns 0 otherwise

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

IndNCSmid1D, IndNCSend1D and IndNPEint

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

IndNCSint(7,5,t,c,int)
IndNCSint(17,17,t,c,int)
IndNCSint(15,17,t,c,int)
IndNCSint(1,3,t,c,int)

IndNCSint(-17,17,t,c,int)

IndNCSint(3,5,t,c,int)
IndNCSint(3,3,t,c,int)
```

```
IndNCSint(4,5,t,c,int)
IndNCSint(a,5,t,c,int)

c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

IndNCSint(7,5,t,c,int)
```

`IndNCSmid1D` *The indicator for the presence of an arc from a point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs) - middle interval case*

Description

Returns I(x2 in $NCS(x1, t, c)$) for points x1 and x2, that is, returns 1 if x2 is in $NCS(x1, t, c)$, returns 0 otherwise, where $NCS(x, t, c)$ is the CS proximity region for point x and is constructed with expansion parameter $t > 0$ and centrality parameter c in $(0, 1)$ for the interval (a, b).

CS proximity regions are defined with respect to the middle interval int and vertex regions are based on the center associated with the centrality parameter c in $(0, 1)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$. rv is the index of the vertex region x1 resides, with default=NULL.

If x1 and x2 are distint and either of them are outside interval int, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., loops are allowed in the digraph).

See also (Ceyhan (2016)).

Usage

```
IndNCSmid1D(x1, x2, t, c, int, rv = NULL)
```

Arguments

x1, x2	1D points; x1 is the point for which the proximity region, $NCS(x1, t, c)$ is constructed and x2 is the point which the function is checking whether its inside $NCS(x1, t, c)$ or not.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside int.
int	A vector of two real numbers representing an interval.
rv	Index of the end interval containing the point, either 1, 2 or NULL (default is NULL).

Value

I(x2 in $NCS(x1, t, c)$) for points x1 and x2 that is, returns 1 if x2 is in $NCS(x1, t, c)$, returns 0 otherwise

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

IndNCSend1D, IndNPEmid1D, and IndNPEend1D

Examples

```
c<-.5
t<-2
a<-0; b<-10; int<-c(a,b)

IndNCSmid1D(7,5,t,c,int)
IndNCSmid1D(7,7,t,c,int)
IndNCSmid1D(7,5,t,c=.4,int)

IndNCSmid1D(1,3,t,c,int)

IndNCSmid1D(9,11,t,c,int)

IndNCSmid1D(19,1,t,c,int)
IndNCSmid1D(19,19,t,c,int)

IndNCSmid1D(3,5,t,c,int)

#or try
Rv<-rv.mid.int(3,c,int)$rv
IndNCSmid1D(3,5,t,c,int,rv=Rv)

IndNCSmid1D(7,5,t,c,int)
```

IndNCStri *The indicator for the presence of an arc from one point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs)*

Description

Returns I(pt2 is in $NCS(pt1, t)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NCS(pt1, \tau)$, returns 0 otherwise, where $NCS(x, \tau)$ is the CS proximity region for point x with the expansion parameter $\tau > 0$.

CS proximity region is constructed with respect to the triangle `tri` and edge regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of `tri` or based on the circumcenter of `tri`. `re` is the index of the edge region `pt` resides, with default=`NULL`

If `pt1` and `pt2` are distint and either of them are outside `tri`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
IndNCStri(pt1, pt2, tau, tri, M, re = NULL)
```

Arguments

`pt1`	A 2D point whose CS proximity region is constructed.
`pt2`	A 2D point. The function determines whether `pt2` is inside the CS proximity region of `pt1` or not.
`tau`	A positive real number which serves as the expansion parameter in CS proximity region.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.
`re`	Index of the `M`-edge region containing the point `pt`, either 1, 2, 3 or `NULL` (default is `NULL`).

Value

I(`pt2` is in $NCS(pt1, \tau)$) for `pt1`, that is, returns 1 if `pt2` is in $NCS(pt1, \tau)$, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`IndNAStri`, `IndNPEtri`, `IndNCStri`, and `IndCSTe`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
tau<-1.5

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

n<-10
set.seed(1)
dat<-runif.tri(n,Tr)$g

IndNCStri(dat[1,],dat[2,],tau,Tr,M)

P1<-as.numeric(runif.tri(1,Tr)$g)
P2<-as.numeric(runif.tri(1,Tr)$g)
IndNCStri(P1,P2,tau,Tr,M)

P1<-c(.4,.2)
P2<-c(1.8,.5)
IndNCStri(P1,P2,tau,Tr,M)
IndNCStri(P2,P1,tau,Tr,M)

IndNCStri(P1,P1,tau,Tr,M)
IndNCStri(P2,P2,tau,Tr,M)

P3<-c(1.7,.6)
IndNCStri(P2,P3,tau,Tr,M)
IndNCStri(P3,P2,tau,Tr,M)

#or try
re<-redges.tri.cent(P1,Tr,M)$re
IndNCStri(P1,P2,tau,Tr,M,re)

P2<-c(1.8,.5)
P3<-c(1.7,.6)
IndNCStri(P2,P3,tau,Tr,M)

dat.fr<-data.frame(a=Tr)
IndNCStri(P2,P3,tau,dat.fr,M)
```

IndNCStri.alt *An alternative function to the function IndNCStri which yields the indicator for the presence of an arc from one point to another for Central Similarity Proximity Catch Digraphs (CS-PCDs)*

Description

Returns I(pt2 is in $NCS(pt1, t)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NCS(pt1, t)$, returns 0 otherwise, where $NCS(x, t)$ is the CS proximity region for point x with the expansion parameter $t > 0$.

CS proximity region is constructed with respect to the triangle `tri` and edge regions are based on the center of mass, CM. `re` is the index of the CM-edge region `pt` resides, with default=`NULL` but must be provided as vertices (y_1, y_2, y_3) for $re = 3$ as rbind(y2,y3,y1) for $re = 1$ and as rbind(y1,y3,y2) for $re = 2$ for triangle $T(y_1, y_2, y_3)$.

If `pt1` and `pt2` are distint and either of them are outside `tri`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
IndNCStri.alt(pt1, pt2, t, tri, re = NULL)
```

Arguments

`pt1`	A 2D point whose CS proximity region is constructed.
`pt2`	A 2D point. The function determines whether `pt2` is inside the CS proximity region of `pt1` or not.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`re`	Index of the CM-edge region containing the point `pt`, either 1, 2, 3 or `NULL`, default=`NULL` but must be provided (row-wise) as vertices (y_1, y_2, y_3) for $re = 3$ as (y_2, y_3, y_1) for $re = 1$ and as (y_1, y_3, y_2) for $re = 2$ for triangle $T(y_1, y_2, y_3)$.

Value

I(`pt2` is in $NCS(pt1, t)$) for `pt1`, that is, returns 1 if `pt2` is in $NCS(pt1, t)$, returns 0 otherwise.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`IndNAStri`, `IndNPEtri`, `IndNCStri`, and `IndCSTe`

Examples

```
## Not run:
A<-c(1,1); B<-c(2,0); C<-c(1.6,2);
Tr<-rbind(A,B,C);
t<-1.5

P1<-c(.4,.2)
P2<-c(1.8,.5)
IndNCStri(P1,P2,t,Tr,M=c(1,1,1))
IndNCStri.alt(P1,P2,t,Tr)

IndNCStri(P1,P1,t,Tr,M=c(1,1,1))
IndNCStri.alt(P1,P1,t,Tr)

IndNCStri(P2,P1,t,Tr,M=c(1,1,1))
IndNCStri.alt(P2,P1,t,Tr)

IndNCStri(P2,P2,t,Tr,M=c(1,1,1))
IndNCStri.alt(P2,P2,t,Tr)

P3<-c(1.7,.6)
IndNCStri(P2,P3,t,Tr,M=c(1,1,1))
IndNCStri.alt(P2,P3,t,Tr)

IndNCStri(P3,P2,t,Tr,M=c(1,1,1))
IndNCStri.alt(P3,P2,t,Tr)

#or try
re<-redges.triCM(P1,Tr)$re
IndNCStri(P1,P2,t,Tr,M=c(1,1,1),re)
IndNCStri.alt(P1,P2,t,Tr,re)

P2<-c(1.85,.5); P3<-c(1.7,.6)
IndNCStri.alt(P2,P3,t,Tr)

dat.fr<-data.frame(a=Tr)
IndNCStri.alt(P2,P3,t,dat.fr)

## End(Not run)
```

`IndNCStri.domset` *The indicator for the set of points* `S` *being a dominating set or not for Central Similarity Proximity Catch Digraphs (CS-PCDs) - one triangle case*

Description

Returns I(`S` a dominating set of CS-PCD whose vertices are the data set `Dt`), that is, returns 1 if `S` is a dominating set of CS-PCD, returns 0 otherwise.

CS proximity region is constructed with respect to the triangle `tri` with the expansion parameter $\tau > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

The triangle `tri`=$T(A, B, C)$ has edges AB, BC, AC which are also labeled as edges 3, 1, and 2, respectively.

See also (Ceyhan (2012)).

Usage

```
IndNCStri.domset(S, Dt, tau, tri, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points which is to be tested for being a dominating set for the CS-PCDs.
`Dt`	A set of 2D points which constitute the vertices of the CS-PCD.
`tau`	A positive real number which serves as the expansion parameter in CS proximity region constructed in the triangle `tri`.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

Value

I(`S` a dominating set of CS-PCD), that is, returns 1 if `S` is a dominating set of CS-PCD whose vertices are the data points in `Dt`; returns 0 otherwise, where CS proximity region is constructed in the triangle `tri`

References

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`IndCSTe.domset`, `IndNPEtri.domset` and `IndNAStri.domset`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
```

```
dat<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

tau<-.5

S<-rbind(dat[1,],dat[2,])
IndNCStri.domset(S,dat,tau,Tr,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,])
IndNCStri.domset(S,dat,tau,Tr,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IndNCStri.domset(S,dat,tau,Tr,M)

IndNCStri.domset(c(.2,.5),dat,tau,Tr,M)
IndNCStri.domset(c(.2,.5),c(.2,.5),tau,Tr,M)
IndNCStri.domset(dat[5,],dat[2,],tau,Tr,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,],c(.2,.5))
IndNCStri.domset(S,dat[3,],tau,Tr,M)

IndNCStri.domset(dat,dat,tau,Tr,M)

P<-c(.4,.2)
S<-dat[c(1,3,4),]
IndNCStri.domset(dat,P,tau,Tr,M)
IndNCStri.domset(S,P,tau,Tr,M)
IndNCStri.domset(S,dat,tau,Tr,M)

IndNCStri.domset(rbind(S,S),dat,tau,Tr,M)

dat.fr<-data.frame(a=dat)
IndNCStri.domset(S,dat.fr,tau,Tr,M)
```

`IndNCStriSet` *The indicator for the presence of an arc from a point in set* `S` *to the point* `pt` *for Central Similarity Proximity Catch Digraphs (CS-PCDs) - one triangle case*

Description

Returns I(`pt` in $NCS(x, \tau)$ for some x in `S`), that is, returns 1 if `pt` in $\cup_{x in S} NCS(x, \tau)$, returns 0 otherwise.

CS proximity region is constructed with respect to the triangle `tri` with the expansion parameter $t > 0$ and edge regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

Edges of tri=$T(A, B, C)$, AB, BC, AC, are also labeled as edges 3, 1, and 2, respectively. If pt is not in S and either pt or all points in S are outside tri, it returns 0, but if pt is in S, then it always returns 1 regardless of its location (i.e., loops are allowed).

Usage

```
IndNCStriSet(S, pt, tau, tri, M = c(1, 1, 1))
```

Arguments

S	A set of 2D points. Presence of an arc from a point in S to point pt is checked by the function.
pt	A 2D point. Presence of an arc from a point in S to point pt is checked by the function.
tau	A positive real number which serves as the expansion parameter in CS proximity region constructed in the triangle tri.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.

Value

I(pt is in $\cup_{x in S} NCS(x, \tau)$), that is, returns 1 if pt is in S or inside $NCS(x, \tau)$ for at least one x in S, returns 0 otherwise where CS proximity region is constructed with respect to the triangle tri

See Also

IndCSTeSet, IndNCStri, IndCSTe, IndNAStriSet, and IndNPEtriSet

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$gen.points

S<-rbind(dat[1,],dat[2,])  #try also S<-c(1.5,1)

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

tau<-.5

IndNCStriSet(S,dat[3,],tau,Tr,M)
IndNCStriSet(S,dat[3,],tau=1,Tr,M)
IndNCStriSet(S,dat[3,],tau=1.5,Tr,M)
```

```
S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,])
IndNCStriSet(S,dat[3,],tau,Tr,M)

IndNCStriSet(S,dat[6,],tau,Tr,M)
IndNCStriSet(S,dat[6,],tau=.25,Tr,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IndNCStriSet(S,dat[3,],tau,Tr,M)

IndNCStriSet(c(.2,.5),dat[2,],tau,Tr,M)
IndNCStriSet(dat,c(.2,.5),tau,Tr,M)
IndNCStriSet(dat,dat[2,],tau,Tr,M)
IndNCStriSet(c(.2,.5),c(.2,.5),tau,Tr,M)
IndNCStriSet(dat[5,],dat[2,],tau,Tr,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,],c(.2,.5))
IndNCStriSet(S,dat[3,],tau,Tr,M)

P<-c(.4,.2)
S<-dat[c(1,3,4),]
IndNCStriSet(dat,P,tau,Tr,M)

IndNCStriSet(rbind(S,S),P,tau,Tr,M)

dat.fr<-data.frame(a=S)
IndNCStriSet(dat.fr,P,tau,Tr,M)
```

`IndNPEbastri` *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - basic triangle case*

Description

Returns I(`pt2` is in $NPE(pt1, r)$) for points `pt1` and `pt2`, that is, returns 1 if `pt2` is in $NPE(pt1, r)$, returns 0 otherwise, where $NPE(x, r)$ is the PE proximity region for point x with expansion parameter $r \geq 1$.

PE proximity region is defined with respect to the basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the basic triangle T_b or based on circumcenter of T_b; default is $M = (1, 1, 1)$ i.e. the center of mass of T_b. `rv` is the index of the vertex region `pt1` resides, with default=`NULL`.

If `pt1` and `pt2` are distint and either of them are outside T_b, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2006)).

Usage

```
IndNPEbastri(pt1, pt2, r, c1, c2, M = c(1, 1, 1), rv = NULL)
```

Arguments

pt1	A 2D point whose PE proximity region is constructed.
pt2	A 2D point. The function determines whether pt2 is inside the PE proximity region of pt1 or not.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1
c1, c2	Positive real numbers which constitute the vertex of the basic triangle adjacent to the shorter edges; c1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the basic triangle or circumcenter of T_b; default is $M = (1, 1, 1)$ i.e. the center of mass of T_b.
rv	The index of the vertex region in T_b containing the point, either 1, 2, 3 or NULL (default is NULL).

Value

I(pt2 is in $NPE(pt1, r)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NPE(pt1, r)$, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

IndNPEtri and IndNPETe

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);
```

```
M<-as.numeric(runif.bastri(1,c1,c2)$g)

r<-2

P1<-as.numeric(runif.bastri(1,c1,c2)$g)
P2<-as.numeric(runif.bastri(1,c1,c2)$g)
IndNPEbastri(P1,P2,r,c1,c2,M)

P1<-c(.4,.2)
P2<-c(.5,.26)
IndNPEbastri(P1,P2,r,c1,c2,M)
IndNPEbastri(P2,P1,r,c1,c2,M)
IndNPEbastri(P1,P1,r,c1,c2,M)

#or try
Rv<-rv.bastri.cent(P1,c1,c2,M)$rv
IndNPEbastri(P1,P2,r,c1,c2,M,Rv)

P1<-c(1.4,.2)
P2<-c(.5,.26)
IndNPEbastri(P1,P2,r,c1,c2,M)
IndNPEbastri(P1,P1,r,c1,c2,M)

#or try
Rv<-rv.bastri.cent(P1,c1,c2,M)$rv
IndNPEbastri(P1,P2,r,c1,c2,M,Rv)
IndNPEbastri(P1,P1,r,c1,c2,M,Rv)

P1<-c(.4,.2)
P2<-c(1.5,.26)
IndNPEbastri(P1,P2,r,c1,c2,M)

P1<-c(.4,.2)
P2<-c(.5,.26)
IndNPEbastri(P1,P2,r,c1,c2,M)
```

`IndNPEend1D` *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - end interval case*

Description

Returns I(`x2` in $NPE(x1, r)$) for points `x1` and `x2`, that is, returns 1 if `x2` is in $NPE(x1, r)$, returns 0 otherwise, where $NPE(x, r)$ is the PE proximity region for point x with expansion parameter $r \geq 1$ for the region outside the interval (a, b).

`rv` is the index of the end vertex region `x1` resides, with default=`NULL`, and $rv = 1$ for left end interval and $rv = 2$ for the right end interval. If `x1` and `x2` are distint and either of them are inside

interval int, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2012)).

Usage

```
IndNPEend1D(x1, x2, r, int, rv = NULL)
```

Arguments

x1	A 1D point whose PE proximity region is constructed.
x2	A 1D point. The function determines whether x2 is inside the PE proximity region of x1 or not.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
int	A vector of two real numbers representing an interval.
rv	Index of the end interval containing the point, either 1, 2 or NULL (default is NULL).

Value

I(x2 in $NPE(x1, r)$) for points x1 and x2, that is, returns 1 if x2 is in $NPE(x1, r)$ (i.e., if there is an arc from x1 to x2), returns 0 otherwise

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

IndNPEmid1D, IndNCSmid1D, and IndNCSend1D

Examples

```
a<-0; b<-10; int<-c(a,b)
r<-2

IndNPEend1D(15,17,r,int)
IndNPEend1D(15,15,r,int)

IndNPEend1D(1.5,17,r,int)

IndNPEend1D(-15,17,r,int)

IndNPEend1D(a,17,r,int)
IndNPEend1D(15,17,r=1.1,int)

a<-0; b<-10; int<-c(a,b)
```

```
r<-2
IndNPEend1D(15,17,r,int)
```

`IndNPEint` *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one interval case*

Description

Returns I(x2 in $NPE(x1, r, c)$) for points x1 and x2, that is, returns 1 if x2 is in $NPE(x1, r, c)$, returns 0 otherwise, where $NPE(x, r, c)$ is the PE proximity region for point x with expansion parameter $r \geq 1$ and centrality parameter c in $(0, 1)$.

PE proximity region is constructed with respect to the interval (a, b). This function works whether x1 and x2 are inside or outside the interval int.

Vertex regions for middle intervals are based on the center associated with the centrality parameter c in $(0, 1)$. If x1 and x2 are identical, then it returns 1 regardless of their locations (i.e., loops are allowed in the digraph).

See also (Ceyhan (2012)).

Usage

```
IndNPEint(x1, x2, r, c = 0.5, int)
```

Arguments

x1	A 1D point for which the proximity region is constructed.
x2	A 1D point for which it is checked whether it resides in the proximity region of x1 or not.
r	A positive real number which serves as the expansion parameter in PE proximity region must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$, and default=0.5
int	A vector of two real numbers representing an interval.

Value

I(x2 in $NPE(x1, r, c)$) for x2, that is, returns 1 if x2 in $NPE(x1, r, c)$, returns 0 otherwise

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

IndNPEmid1D, IndNPEend1D and IndNCSint

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

IndNPEint(7,5,r,c,int)
IndNPEint(17,17,r,c,int)
IndNPEint(15,17,r,c,int)
IndNPEint(1,3,r,c,int)

IndNPEint(-17,17,r,c,int)

IndNPEint(3,5,r,c,int)
IndNPEint(3,3,r,c,int)
IndNPEint(4,5,r,c,int)
IndNPEint(a,5,r,c,int)

c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)
IndNPEint(7,5,r,c,int)
```

IndNPEmid1D *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - middle interval case*

Description

Returns I(x2 in $NPE(x1, r, c)$) for points x1 and x2, that is, returns 1 if x2 is in $NPE(x1, r, c)$, returns 0 otherwise, where $NPE(x, r, c)$ is the PE proximity region for point x and is constructed with expansion parameter $r \geq 1$ and centrality parameter c in $(0, 1)$ for the interval (a, b).

PE proximity regions are defined with respect to the middle interval int and vertex regions are based on the center associated with the centrality parameter c in $(0, 1)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$. rv is the index of the vertex region x1 resides, with default=NULL. If x1 and x2 are distint and either of them are outside interval int, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., loops are allowed in the digraph).

See also (Ceyhan (2012, 2016)).

Usage

```
IndNPEmid1D(x1, x2, r, c, int, rv = NULL)
```

Arguments

x1, x2	1D points; x1 is the point for which the proximity region, $NPE(x1, r, c)$ is constructed and x2 is the point which the function is checking whether its inside $NPE(x1, r, c)$ or not.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
int	A vector of two real numbers representing an interval.
rv	The index of the vertex region x1 resides, with default=NULL.

Value

I(x2 in $NPE(x1, r, c)$) for points x1 and x2 that is, returns 1 if x2 is in $NPE(x1, r, c)$, returns 0 otherwise

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

IndNPEend1D, IndNCSmid1D, and IndNCSend1D

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

IndNPEmid1D(7,5,r,c,int)
IndNPEmid1D(17,17,r,c,int)
IndNPEmid1D(1,3,r,c,int)

IndNPEmid1D(3,5,r,c,int)
IndNPEmid1D(3,3,r,c,int)
IndNPEmid1D(4,5,r,c,int)
IndNPEmid1D(a,5,r,c,int)

c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)
IndNPEmid1D(7,5,r,c,int)
```

IndNPEstdtetra — *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard regular tetrahedron case*

Description

Returns I(pt2 is in $NPE(pt1, r)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NPE(pt1, r)$, returns 0 otherwise, where $NPE(x, r)$ is the PE proximity region for point x with expansion parameter $r \geq 1$.

PE proximity region is defined with respect to the standard regular tetrahedron $T_h = T(v = 1, v = 2, v = 3, v = 4)$=$T((0, 0, 0), (1, 0, 0), (1/2, \sqrt{3}/2, 0), (1/2, \sqrt{3}/6, \sqrt{6}/3))$ and vertex regions are based on the circumcenter (which is equivalent to the center of mass for standard regular tetrahedron) of T_h. rv is the index of the vertex region pt1 resides, with default=NULL.

If pt1 and pt2 are distint and either of them are outside T_h, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005, 2010)).

Usage

```
IndNPEstdtetra(pt1, pt2, r, rv = NULL)
```

Arguments

pt1	A 3D point whose PE proximity region is constructed.
pt2	A 3D point. The function determines whether pt2 is inside the PE proximity region of pt1 or not.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
rv	Index of the vertex region containing the point, either 1, 2, 3, 4 (defualt is NULL).

Value

I(pt2 is in $NPE(pt1, r)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NPE(pt1, r)$, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

IndNPEtetra, IndNPEtri and IndNPEint

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-10  #try also n<-20
dat<-runif.stdtetra(n)$g
r<-1.5
IndNPEstdtetra(dat[1,],dat[1,],r)

IndNPEstdtetra(dat[1,],dat[5,],r)

IndNPEstdtetra(c(.4,.4,.4),c(.5,.5,.5),r)

#or try
RV<-rv.tetraCC(dat[1,],tetra)$rv
IndNPEstdtetra(dat[1,],dat[5,],r,rv=RV)

IndNPEstdtetra(dat[1,],c(-1,-1,-1),r,rv=NULL)

IndNPEstdtetra(c(-1,-1,-1),dat[1,],r,rv=NULL)
IndNPEstdtetra(c(-1,-1,-1),c(-1,-1,-1),r)

IndNPEstdtetra(dat[1,],dat[5,],r)

P1<-c(.1,.1,.1)
P2<-c(.5,.5,.5)
IndNPEstdtetra(P1,P2,r)
```

IndNPETe

The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard equilateral triangle case

---

Description

Returns I(pt2 is in $NPE(pt1, r)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NPE(pt1, r)$, returns 0 otherwise, where $NPE(x, r)$ is the PE proximity region for point x with expansion parameter $r \geq 1$.

PE proximity region is defined with respect to the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e. rv is the index of the vertex region pt1 resides, with default=NULL.

If pt1 and pt2 are distint and either of them are outside T_e, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
IndNPETe(pt1, pt2, r, M = c(1, 1, 1), rv = NULL)
```

Arguments

pt1	A 2D point whose PE proximity region is constructed.
pt2	A 2D point. The function determines whether pt2 is inside the PE proximity region of pt1 or not.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.
rv	The index of the vertex region in T_e containing the point, either 1, 2, 3 or NULL (default is NULL).

Value

I(pt2 is in $NPE(pt1, r)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NPE(pt1, r)$, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

IndNPEtri, IndNPEbastri, and IndCSTe

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
n<-10

set.seed(1)
```

```
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

IndNPETe(dat[1,],dat[10,],r=2,M)
IndNPETe(c(0,1),dat[10,],r=2,M)

IndNPETe(dat[1,],dat[10,],r=1.5,M)
IndNPETe(dat[1,],dat[5,],r=2,M)
IndNPETe(dat[1,],dat[5,],r=3,M)

IndNPETe(c(1,1),dat[5,],r=3,M)
IndNPETe(c(1,1),c(1,1),r=3,M)

#or try
Rv<-rvTeCM(dat[1,])$rv
IndNPETe(dat[1,],dat[10,],r=2,M,Rv)

P1<-c(.4,.2)
P2<-c(.5,.26)
r<-2
IndNPETe(P1,P2,r,M)
```

`IndNPETe.domset` *The indicator for the set of points* `S` *being a dominating set or not for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard equilateral triangle case*

Description

Returns I(`S` a dominating set of PE-PCD whose vertices are the data points `Dt`), that is, returns 1 if `S` is a dominating set of PE-PCD, returns 0 otherwise.

PE proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e (which is equivalent to the circumcenter of T_e). Vertices of T_e are also labeled as 1, 2, and 3, respectively.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
IndNPETe.domset(S, Dt, r, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points whose PE proximity regions are considered.
`Dt`	A set of 2D points which constitutes the vertices of the PE-PCD.

r — A positive real number which serves as the expansion parameter in PE proximity region in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$; must be ≥ 1.

M — A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e.

Value

I(S a dominating set of PE-PCD), that is, returns 1 if S is a dominating set of PE-PCD, returns 0 otherwise, where PE proximity region is constructed in the standard equilateral triangle T_e

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

IndNPEtri.domset and IndCSTe.domset

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

r<-1.5

S<-rbind(dat[1,],dat[2,])
IndNPETe.domset(S,dat,r,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,])
IndNPETe.domset(S,dat,r,M)
```

```
S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IndNPETe.domset(S,dat,r,M)

IndNPETe.domset(c(.2,.5),dat,r,M)
IndNPETe.domset(c(.2,.5),c(.2,.5),r,M)
IndNPETe.domset(dat[5,],dat[2,],r,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,],c(.2,.5))
IndNPETe.domset(S,dat[3,],r,M)

IndNPETe.domset(dat,dat,r,M)

P<-c(.4,.2)
S<-dat[c(1,3,4),]
IndNPETe.domset(dat,P,r,M)

IndNPETe.domset(rbind(S,S),dat,r,M)

dat.fr<-data.frame(a=dat)
IndNPETe.domset(S,dat.fr,r,M)
```

`IndNPETeSet` *The indicator for the presence of an arc from a point in set* `S` *to the point* `pt` *for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard equilateral triangle case*

Description

Returns I(`pt` in $NPE(x, r)$ for some x in `S`), that is, returns 1 if `pt` is in $\cup_{x in S} NPE(x, r)$, returns 0 otherwise.

PE proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A, B, C) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with the expansion parameter $r \geq 1$ and vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e (which is equivalent to the circumcenter for T_e).

Vertices of T_e are also labeled as 1, 2, and 3, respectively. If `pt` is not in `S` and either `pt` or all points in `S` are outside T_e, it returns 0, but if `pt` is in `S`, then it always returns 1 regardless of its location (i.e., loops are allowed).

Usage

```
IndNPETeSet(S, pt, r, M = c(1, 1, 1))
```

Arguments

S	A set of 2D points. Presence of an arc from a point in S to point pt is checked by the function.
pt	A 2D point. Presence of an arc from a point in S to point pt is checked by the function.
r	A positive real number which serves as the expansion parameter in PE proximity region in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$; must be ≥ 1.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e.

Value

I(pt is in U_x in S NPE(x,r)), that is, returns 1 if pt is in S or inside $NPE(x,r)$ for at least one x in S, returns 0 otherwise. PE proximity region is constructed with respect to the standard equilateral triangle $T_e = T(A,B,C) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with M-vertex regions

See Also

IndNPEtriSet, IndNPETe, IndNPEtri, and IndCSTeSet

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

r<-1.5

S<-rbind(dat[1,],dat[2,])  #try also S<-c(.5,.5)
IndNPETeSet(S,dat[3,],r,M)
IndNPETeSet(S,dat[3,],r=1,M)
IndNPETeSet(S,dat[3,],r=1.5,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,])
IndNPETeSet(S,dat[3,],r,M)

IndNPETeSet(S,dat[6,],r,M)
IndNPETeSet(S,dat[6,],r=1.25,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IndNPETeSet(S,dat[3,],r,M)

IndNPETeSet(c(.2,.5),dat[2,],r,M)
```

```
IndNPETeSet(dat,c(.2,.5),r,M)
IndNPETeSet(dat,dat[2,],r,M)
IndNPETeSet(c(.2,.5),c(.2,.5),r,M)
IndNPETeSet(dat[5,],dat[2,],r,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,],c(.2,.5))
IndNPETeSet(S,dat[3,],r,M)

P<-c(.4,.2)
S<-dat[c(1,3,4),]
IndNPETeSet(dat,P,r,M)

IndNPETeSet(rbind(S,S),P,r,M)

dat.fr<-data.frame(a=S)
IndNPETeSet(dat.fr,P,r,M)
```

`IndNPEtetra` *The indicator for the presence of an arc from one 3D point to another 3D point for Proportional Edge Proximity Catch Digraphs (PE-PCDs)*

Description

Returns I(`pt2` is in $NPE(pt1, r)$) for 3D points `pt1` and `pt2`, that is, returns 1 if `pt2` is in $NPE(pt1, r)$, returns 0 otherwise, where NPE(x,r) is the PE proximity region for point x with the expansion parameter $r \geq 1$.

PE proximity region is constructed with respect to the tetrahedron `th` and vertex regions are based on the center `M` which is circumcenter ("CC") or center of mass ("CM") of `th` with default="CM". `rv` is the index of the vertex region `pt1` resides, with default=`NULL`.

If `pt1` and `pt2` are distint and either of them are outside `th`, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005, 2010)).

Usage

```
IndNPEtetra(pt1, pt2, r, th, M = "CM", rv = NULL)
```

Arguments

`pt1`	A 3D point whose PE proximity region is constructed.
`pt2`	A 3D point. The function determines whether `pt2` is inside the PE proximity region of `pt1` or not.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`th`	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.

M	The center to be used in the construction of the vertex regions in the tetrahedron, th. Currently it only takes "CC" for circumcenter and "CM" for center of mass; default="CM".
rv	Index of the M-vertex region containing the point, either 1, 2, 3, 4 (defualt is NULL).

Value

I(pt2 is in $NPE(pt1, r)$) for pt1, that is, returns 1 if pt2 is in $NPE(pt1, r)$, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

IndNPEstdtetra, IndNPEtri and IndNPEint

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-10  #try also n<-20

dat<-runif.tetra(n,tetra)$g

M<-"CM"  #try also M<-"CC"
r<-1.5

IndNPEtetra(dat[1,],dat[2,],r,tetra)  #uses the default M="CM"
IndNPEtetra(dat[1,],dat[2,],r,tetra,M)

IndNPEtetra(dat[1,],dat[1,],r,tetra,M)

IndNPEtetra(c(.4,.4,.4),c(.5,.5,.5),r,tetra,M)

#or try
RV<-rv.tetraCC(dat[1,],tetra)$rv
IndNPEtetra(dat[1,],dat[5,],r,tetra,M,rv=RV)

IndNPEtetra(dat[1,],c(-1,-1,-1),r,tetra,M,rv=NULL)

IndNPEtetra(c(-1,-1,-1),dat[1,],r,tetra,M,rv=NULL)
IndNPEtetra(c(-1,-1,-1),c(-1,-1,-1),r,tetra,M)

P1<-c(.1,.1,.1)
```

```
P2<-c(.5,.5,.5)
IndNPEtetra(P1,P2,r,tetra,M)
```

IndNPEtri — *The indicator for the presence of an arc from a point to another for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns I(pt2 is in $NPE(pt1, r)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NPE(pt1, r)$, returns 0 otherwise, where $NPE(x, r)$ is the PE proximity region for point x with the expansion parameter $r \geq 1$.

PE proximity region is constructed with respect to the triangle tri and vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of tri or based on the circumcenter of tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri. rv is the index of the vertex region pt1 resides, with default=NULL.

If pt1 and pt2 are distint and either of them are outside tri, it returns 0, but if they are identical, then it returns 1 regardless of their locations (i.e., it allows loops).

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)).

Usage

```
IndNPEtri(pt1, pt2, r, tri, M = c(1, 1, 1), rv = NULL)
```

Arguments

pt1	A 2D point whose PE proximity region is constructed.
pt2	A 2D point. The function determines whether pt2 is inside the PE proximity region of pt1 or not.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.
rv	Index of the M-vertex region containing the point, either 1, 2, 3 or NULL (default is NULL).

Value

I(pt2 is in $NPE(pt1, r)$) for points pt1 and pt2, that is, returns 1 if pt2 is in $NPE(pt1, r)$, returns 0 otherwise

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

IndNPEbastri, IndNPETe, IndNAStri, and IndNCStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0);

r<-1.5

n<-10
set.seed(1)
dat<-runif.tri(n,Tr)$g

IndNPEtri(dat[1,],dat[2,],r,Tr,M)

P1<-as.numeric(runif.tri(1,Tr)$g)
P2<-as.numeric(runif.tri(1,Tr)$g)
IndNPEtri(P1,P2,r,Tr,M)

P1<-c(.4,.2)
P2<-c(1.8,.5)
IndNPEtri(P1,P2,r,Tr,M)
IndNPEtri(P2,P1,r,Tr,M)
IndNPEtri(P1,P1,r,Tr,M)

IndNPEtri(P2,P2,r,Tr,M)

P3<-c(1.7,.6)
IndNPEtri(P2,P3,r,Tr,M)
IndNPEtri(P3,P2,r,Tr,M)

M<-c(1.3,1.3)
r<-2

P1<-c(1.4,1.2)
```

```
P2<-c(1.5,1.26)
IndNPEtri(P1,P2,r,Tr,M)
IndNPEtri(P2,P1,r,Tr,M)

#or try
Rv<-rv.tri.cent(P1,Tr,M)$rv
IndNPEtri(P1,P2,r,Tr,M,Rv)

P2<-c(1.8,.5)
P3<-c(1.7,.6)
IndNPEtri(P2,P3,r,Tr,M)

dat.fr<-data.frame(a=Tr)
IndNPEtri(P2,P3,r,dat.fr,M)
```

`IndNPEtri.domset` *The indicator for the set of points* `S` *being a dominating set or not for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns I(`S` a dominating set of PE-PCD whose vertices are the data set `Dt`), that is, returns 1 if `S` is a dominating set of PE-PCD, returns 0 otherwise.

PE proximity region is constructed with respect to the triangle `tri` with the expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`. The triangle `tri`=$T(A, B, C)$ has edges AB, BC, AC which are also labeled as edges 3, 1, and 2, respectively.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
IndNPEtri.domset(S, Dt, r, tri, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points which is to be tested for being a dominating set for the PE-PCDs.
`Dt`	A set of 2D points which constitute the vertices of the PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region constructed in the triangle `tri`; must be ≥ 1.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

Value

I(S a dominating set of PE-PCD), that is, returns 1 if S is a dominating set of PE-PCD whose vertices are the data points in Dt; returns 0 otherwise, where PE proximity region is constructed in the triangle tri

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

IndNPETe.domset, IndNPEtriSet, IndNCStri.domset and IndNAStri.domset

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5

S<-rbind(dat[1,],dat[2,])
IndNPEtri.domset(S,dat,r,Tr,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,])
IndNPEtri.domset(S,dat,r,Tr,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IndNPEtri.domset(S,dat,r,Tr,M)

IndNPEtri.domset(c(.2,.5),dat,r,Tr,M)
IndNPEtri.domset(c(.2,.5),c(.2,.5),r,Tr,M)

IndNPEtri.domset(dat[5,],dat[2,],r,Tr,M)
```

```
S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,],c(.2,.5))
IndNPEtri.domset(S,dat[3,],r,Tr,M)

IndNPEtri.domset(dat,dat,r,Tr,M)

P<-c(.4,.2)
S<-dat[c(1,3,4),]
IndNPEtri.domset(dat,P,r,Tr,M)

IndNPEtri.domset(rbind(S,S),dat,r,Tr,M)

dat.fr<-data.frame(a=dat)
IndNPEtri.domset(S,dat.fr,r,Tr,M)
```

`IndNPEtriSet` *The indicator for the presence of an arc from a point in set* `S` *to the point* `pt` *for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns I(`pt` in $NPE(x, r)$ for some x in `S`), that is, returns 1 if `pt` is in $\cup_{x in S} NPE(x, r)$, returns 0 otherwise.

PE proximity region is constructed with respect to the triangle `tri` with the expansion parameter $r \geq 1$ and vertex regions are based on the center, $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`. Vertices of `tri` are also labeled as 1, 2, and 3, respectively.

If `pt` is not in `S` and either `pt` or all points in `S` are outside `tri`, it returns 0, but if `pt` is in `S`, then it always returns 1 regardless of its location (i.e., loops are allowed).

Usage

```
IndNPEtriSet(S, pt, r, tri, M = c(1, 1, 1))
```

Arguments

`S`	A set of 2D points. Presence of an arc from a point in `S` to point `pt` is checked by the function.
`pt`	A 2D point. Presence of an arc from a point in `S` to point `pt` is checked by the function.
`r`	A positive real number which serves as the expansion parameter in PE proximity region constructed in the triangle `tri`; must be ≥ 1.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

M — A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

Value

I(`pt` is in U_x in `S` NPE(x,r)), that is, returns 1 if `pt` is in `S` or inside $NPE(x, r)$ for at least one x in `S`, returns 0 otherwise where PE proximity region is constructed with respect to the triangle `tri`

See Also

IndNPETeSet, IndNPEtri, IndNPETe, IndNAStriSet, and IndNCStriSet

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$gen.points

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5

S<-rbind(dat[1,],dat[2,])  #try also S<-c(1.5,1)

IndNPEtriSet(S,dat[3,],r,Tr,M)
IndNPEtriSet(S,dat[3,],r=1,Tr,M)
IndNPEtriSet(S,dat[3,],r=1.5,Tr,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,])
IndNPEtriSet(S,dat[3,],r,Tr,M)

IndNPEtriSet(S,dat[6,],r,Tr,M)
IndNPEtriSet(S,dat[6,],r=1.25,Tr,M)

S<-rbind(c(.1,.1),c(.3,.4),c(.5,.3))
IndNPEtriSet(S,dat[3,],r,Tr,M)

IndNPEtriSet(c(.2,.5),dat[2,],r,Tr,M)
IndNPEtriSet(dat,c(.2,.5),r,Tr,M)
IndNPEtriSet(dat,dat[2,],r,Tr,M)
IndNPEtriSet(c(.2,.5),c(.2,.5),r,Tr,M)
IndNPEtriSet(dat[5,],dat[2,],r,Tr,M)

S<-rbind(dat[1,],dat[2,],dat[3,],dat[5,],c(.2,.5))
IndNPEtriSet(S,dat[3,],r,Tr,M)

P<-c(.4,.2)
S<-dat[c(1,3,4),]
```

```
IndNPEtriSet(dat,P,r,Tr,M)

IndNPEtriSet(rbind(S,S),P,r,Tr,M)

dat.fr<-data.frame(a=S)
IndNPEtriSet(dat.fr,P,r,Tr,M)
```

IndUBdom *Indicator for an upper bound for the domination number by the exact algorithm*

Description

Returns 1 if the domination number is less than or equal to the prespecified value k and also the indices (i.e. row numbers) of a dominating set of size k based on the incidence matrix Inc.Mat of a graph or a digraph. Here the row number in the incidence matrix corresponds to the index of the vertex (i.e. index of the data point). The function works whether loops are allowed or not (i.e., whether the first diagonal is all 1 or all 0). It takes a rather long time for large number of vertices (i.e., large number of row numbers).

Usage

```
IndUBdom(Inc.Mat, k)
```

Arguments

Inc.Mat	A square matrix consisting of 0's and 1's which represents the incidence matrix of a graph or digraph.
k	A positive integer for the upper bound (to be checked) for the domination number.

Value

A list with two elements

domUB	The upper bound (to be checked) for the domination number. It is prespecified as k in the function arguments.
IndUBdom	The indicator for the upper bound for domination number of the graph or digraph being the specified value k or not. It returns 1 if the upper bound is k, and 0 otherwise based on the incidence matrix Inc.Mat of the graph or digraph.
ind.domset	Indices of the rows in the incidence matrix Inc.Mat that correspond to the vertices in the dominating set of size k if it exists, otherwise it yields NULL.

See Also

dom.exact and dom.greedy

Examples

```
n<-10
M<-matrix(sample(c(0,1),n^2,replace=TRUE),nrow=n)
diag(M)<-1

dom.greedy(M)
IndUBdom(M,2)

for (k in 1:n)
print(c(k,IndUBdom(M,k)))
```

`int.2lines` *The point of intersection of two lines*

Description

Returns the intersection of two lines, first line passing through points `p1` and `q1` and second line passing through points `p2` and `q2`. The points are chosen so that lines are well defined.

Usage

```
int.2lines(p1, q1, p2, q2)
```

Arguments

`p1, q1`	2D points that determine the first straight line (i.e., through which the first straight line passes).
`p2, q2`	2D points that determine the second straight line (i.e., through which the second straight line passes).

Value

The coordinates of the point of intersection of the two lines, first passing through points `p1` and `q1` and second passing through points `p2` and `q2`.

See Also

`int.circ.line` and `dp2l`

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75); C<-c(0,6); D<-c(3,-2)

ip<-int.2lines(A,B,C,D)
ip
pts<-rbind(A,B,C,D,ip)
xr<-range(pts)
```

```
xf<-abs(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=20)  #try also l=100
lnAB<-Line(A,B,x)
lnCD<-Line(C,D,x)

y1<-lnAB$y
y2<-lnCD$y
Xlim<-range(x,pts)
Ylim<-range(y1,y2,pts)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
pf<-c(xd,-yd)*.025

#plot of the line joining A and B
plot(rbind(A,B,C,D),pch=1,xlab="x",ylab="y",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
lines(x,y1,lty=1,col=1)
lines(x,y2,lty=1,col=2)
text(rbind(A+pf,B+pf),c("A","B"))
text(rbind(C+pf,D+pf),c("C","D"))
text(rbind(ip+pf),c("intersection\n point"))

int.2lines(c(1,2),c(1,3),C,D)
```

`int.circ.line` *The points of intersection of a line and a circle*

Description

Returns the intersection point(s) of a line and a circle. The line is determined by the two points `pt1` and `pt2` and the circle is centered at point `cent` and has radius `rad`. If the circle does not intersect the line, the function yields `NULL`; if the circle intersects at only one point, it yields only that point; otherwise it yields both intersection points as output. When there are two intersection points, they are listed in the order of the x-coordinates of `pt1` and `pt2`; and if the x-coordinates of `pt1` and `pt2` are equal, intersection points are listed in the order of y-coordinates of `pt1` and `pt2`.

Usage

```
int.circ.line(pt1, pt2, cent, rad)
```

Arguments

`pt1, pt2`	2D points that determine the straight line (i.e., through which the straight line passes).
`cent`	A 2D point representing the center of the circle.
`rad`	A positive real number representing the radius of the circle.

Value

point(s) of intersection between the circle and the line (if they do not intersect, the function yields NULL as the output)

See Also

int.2lines

Examples

```
P1<-c(.3,.2)*100
P2<-c(.6,.3)*100
cent<-c(1.1,1.1)*100
rad<-2*100

int.circ.line(P1,P2,cent,rad)
int.circ.line(P2,P1,cent,rad)
int.circ.line(P1,P1+c(0,1),cent,rad)
int.circ.line(P1+c(0,1),P1,cent,rad)

dp2l(cent,P1,P2)
rad2<-dp2l(cent,P1,P2)$d
int.circ.line(P1,P2,cent,rad2)
int.circ.line(P1,P2,cent,rad=.8)
int.circ.line(P1,P2,cent,rad=.78)

#plot of the line and the circle
A<-c(.3,.2); B<-c(.6,.3); cent<-c(1,1); rad<-2 #check dp2l(cent,A,B)$dis, .3

IPs<-int.circ.line(A,B,cent,rad)

xr<-range(A[1],B[1],cent[1])
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-rad-xf,xr[2]+rad+xf,l=20)  #try also l=100
lnAB<-Line(A,B,x)
y<-lnAB$y

Xlim<-range(x,cent[1])
Ylim<-range(y,A[2],B[2],cent[2]-rad,cent[2]+rad)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(rbind(A,B,cent),pch=1,asp=1,xlab="x",ylab="y",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
lines(x,y,lty=1)
interp::circles(cent[1],cent[2],rad)
IP.txt<-c()
if (!is.null(IPs))
{
  for (i in 1:(length(IPs)/2))
    IP.txt<-c(IP.txt,paste("I",i, sep = ""))
}
```

```
txt<-rbind(A,B,cent,IPs)
text(txt+cbind(rep(xd*.03,nrow(txt)),rep(-yd*.03,nrow(txt))),c("A","B","M",IP.txt))
```

`int.line.plane` *The point of intersection of a line and a plane*

Description

Returns the point of the intersection of the line determined by the 3D points `x1` and `x2` and the plane spanned by 3D points `x3`, `x4`, and `x5`.

Usage

```
int.line.plane(x1, x2, x3, x4, x5)
```

Arguments

`x1, x2`	3D points that determine the straight line (i.e., through which the straight line passes).
`x3, x4, x5`	3D points that determine the plane (i.e., through which the plane passes).

Value

The coordinates of the point of intersection the line determined by the 3D points `x1` and `x2` and the plane determined by 3D points `x3`, `x4`, and `x5`.

See Also

`int.2lines` and `int.circ.line`

Examples

```
L1<-c(2,4,6); L2<-c(1,3,5); A<-c(1,10,3); B<-c(1,1,3); C<-c(3,9,12)

Pint<-int.line.plane(L1,L2,A,B,C)
Pint
pts<-rbind(L1,L2,A,B,C,Pint)

tr<-max(Dist(L1,L2),Dist(L1,Pint),Dist(L2,Pint))
tf<-tr*1.1 #how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf,tf,l=20)  #try also l=100

lnAB3D<-Line3D(L1,L2,tsq)
xl<-lnAB3D$x
yl<-lnAB3D$y
zl<-lnAB3D$z
```

```
xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1 #how far to go at the lower and upper ends in the y-coordinate
xp<-seq(xr[1]-xf,xr[2]+xf,l=20)  #try also l=100
yp<-seq(yr[1]-yf,yr[2]+yf,l=20)  #try also l=100

plABC<-Plane(A,B,C,xp,yp)
z.grid<-plABC$z

res<-persp(xp,yp,z.grid, xlab="x",ylab="y",zlab="z",theta = -30, phi = 30, expand = 0.5,
col = "lightblue", ltheta = 120, shade = 0.05, ticktype = "detailed")
lines (trans3d(xl, yl, zl, pmat = res), col = 3)

Xlim<-range(xl,pts[,1])
Ylim<-range(yl,pts[,2])
Zlim<-range(zl,pts[,3])

xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::persp3D(z = z.grid, x = xp, y = yp, theta =225, phi = 30, ticktype = "detailed"
        ,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),zlim=Zlim+zd*c(-.1,.1))
        #plane spanned by points A, B, C
#add the defining points
plot3D::points3D(pts[,1],pts[,2],pts[,3], pch = ".", col = "black", bty = "f", cex = 5,add=TRUE)
plot3D::points3D(Pint[1],Pint[2],Pint[3], pch = "*", col = "red", bty = "f", cex = 5,add=TRUE)
plot3D::lines3D(xl, yl, zl, bty = "g", cex = 2, ticktype = "detailed",add=TRUE)
```

`inTriAll` *Check whether all points in a data set are inside the triangle*

Description

Checks if all the data points in the 2D data set, `Dt`, lie in the triangle, `tri`, using the barycentric coordinates, generally denoted as (α, β, γ).

If all (normalized or non-normalized) barycentric coordinates of a point are positive then the point is inside the triangle, if all are nonnegative with one or more are zero, then the point falls in the boundary. If some of the barycentric coordinates are negative, then the point falls outside the triangle.

`boundary` is a logical argument (default=`FALSE`) to include boundary or not, so if it is `TRUE`, the function checks if a point lies in the closure of the triangle (i.e. interior and boundary combined) else it checks if the point lies in the interior of the triangle.

Usage

```
inTriAll(Dt, tri, boundary = FALSE)
```

Arguments

Dt	A set of 2D points representing the set of data points.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
boundary	A logical parameter (default=FALSE) to include boundary or not, so if it is TRUE, the function checks if a point lies in the closure of the triangle (i.e. interior and boundary combined) else it checks if the point lies in the interior of the triangle.

Value

A logical output, if all data points in Dt are inside the triangle, tri, the output is TRUE, else it is FALSE.

See Also

in.triangle and on.convex.hull from the interp package for documentation for in.convex.hull

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2); p<-c(1.4,1.2)

Tr<-rbind(A,B,C)

inTriAll(p,Tr)

#for the vertex A
inTriAll(A,Tr)
inTriAll(A,Tr,boundary=TRUE)

#for a point on the edge AB
D3<-(A+B)/2
inTriAll(D3,Tr)
inTriAll(D3,Tr,boundary=TRUE)

#data set
n<-10
dat<-cbind(runif(n),runif(n))
inTriAll(dat,Tr,boundary=TRUE)

dat<-runifTe(n)$gen.points
inTriAll(dat,Tr)
inTriAll(dat,Tr,boundary=TRUE)

dat<-runif.tri(n,Tr)$g
inTriAll(dat,Tr)
inTriAll(dat,Tr,boundary=TRUE)

dat.fr<-data.frame(a=dat)
inTriAll(dat.fr,Tr)

dat.fr<-data.frame(a=Tr)
```

```
inTriAll(dat,dat.fr)
```

`is.in.data` *Check a point belong to a given data set*

Description

returns `TRUE` if the point `p` of any dimension is inside the data set `Dt` of the same dimension as `p`; otherwise returns `FALSE`.

Usage

```
is.in.data(p, Dt)
```

Arguments

`p`	A 2D point for which the function checks membership to the data set `Dt`.
`Dt`	A set of 2D points representing the set of data points.

Value

`TRUE` if `p` belongs to the data set `Dt`.

Examples

```
n<-10
dat<-cbind(runif(n),runif(n))

P<-dat[7,]
is.in.data(P,dat)
is.in.data(P,dat[7,])

P<-dat[7,]+10^(-7)
is.in.data(P,dat)

P<-dat[7,]+10^(-9)
is.in.data(P,dat)

is.in.data(P,P)

is.in.data(c(2,2),c(2,2))

#for 1D data
n<-10
dat<-runif(n)

## Not run:
P<-dat[7]
```

```
is.in.data(P,dat)  #Both arguments must be of the same dimension
#this does not work because both entries are treated as vectors of different dimensions

## End(Not run)
P<-dat[7]
is.in.data(P,dat[7])  #this works because both entries are treated as 1D vectors

dat<-as.matrix(dat)
is.in.data(P,dat)
#this works, because P is a 1D point, and dat is treated as a set of 10 1D points

P<-dat[7]+10^(-7)
is.in.data(P,dat)

P<-dat[7]+10^(-9)
is.in.data(P,dat)

is.in.data(P,P)

#for 3D data
n<-10
dat<-cbind(runif(n),runif(n),runif(n))

P<-dat[7,]
is.in.data(P,dat)
is.in.data(P,dat[7,])

P<-dat[7,]+10^(-7)
is.in.data(P,dat)

P<-dat[7,]+10^(-9)
is.in.data(P,dat)

is.in.data(P,P)

n<-10
dat<-cbind(runif(n),runif(n))
P<-dat[7,]
is.in.data(P,dat)

dat.fr<-data.frame(a=dat)
is.in.data(P,dat.fr)
```

`is.point` *Check the argument is a point of a given dimension*

Description

Returns `TRUE` if the arguement `p` is a numeric point of dimension `dim` (default is `dim=2`); otherwise returns `FALSE`.

Usage

```
is.point(p, dim = 2)
```

Arguments

p	A vector to be checked to see it is a point of dimension dim or not.
dim	A positive integer representing the dimension of the argument p.

Value

TRUE if p is a vector of dimension dim.

See Also

dimension

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75,4)
is.point(A)
is.point(A,2)
is.point(A,1)
is.point(A,3)

is.point(B)
is.point(B,3)

Dt<-rbind(A,B[1:2])
is.point(Dt)
is.point(as.vector(Dt),4)

lis<-list(a=A,b=B)
is.point(lis)

dat.fr<-data.frame(a=A,b=B[1:2])
is.point(dat.fr)

is.point(c("a","b"))
```

---

isStdEqTri *Check whether a triangle is a standard equilateral triangle*

Description

Checks whether the triangle, tri, is the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ or not.

Usage

```
isStdEqTri(tri)
```

Arguments

tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

TRUE if tri is a standard equilateral triangle, else FALSE.

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)  #try adding +10^(-16) to each vertex
isStdEqTri(Te)

isStdEqTri(Te)

Te<-rbind(B,C,A)
isStdEqTri(Te)

Tr<-rbind(A,B,-C)
isStdEqTri(Tr)

A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
isStdEqTri(Tr)

A<-c(0,0); B<-c(1,0); C<-c(.5,0.8660254);
Te<-rbind(A,B,C)
isStdEqTri(Te)

isStdEqTri(rbind(A,A,B))

dat.fr<-data.frame(a=Te)
isStdEqTri(dat.fr)
```

Kfr2vTbVRCC — *The* k *furthest points from vertices in each CC-vertex region in a basic triangle*

Description

An object of class "Extrema". Returns the k furthest data points among the data set, Dt, in each CC-vertex region from the vertex in the basic triangle $T_b = T(A = (0,0), B = (1,0), C = (c_1, c_2))$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

`ch.all.intri` is for checking whether all data points are inside T_b (default is `FALSE`). In the extrema, ext, in the output, the first `k` entries are the `k` furthest points from vertex 1, second `k` entries are `k` furthest points are from vertex 2, and last `k` entries are the `k` furthest points from vertex 3

Usage

```
Kfr2vTbVRCC(Dt, c1, c2, k, ch.all.intri = FALSE)
```

Arguments

`Dt`	A set of 2D points representing the set of data points.
`c1, c2`	Positive real numbers which constitute the vertex of the basic triangle. adjacent to the shorter edges; `c1` must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$
`k`	A positive integer. `k` furthest data points in each CC-vertex region are to be found if exists, else `NA` are provided for (some of) the `k` furthest points.
`ch.all.intri`	A logical argument for checking whether all data points are inside T_b (default is `FALSE`).

Value

A list with the elements

`txt1`	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
`txt2`	A shorter description of the distances as "Distances of `k` furthest points in the vertex regions to Vertices".
`type`	Type of the extrema points
`desc`	A short description of the extrema points
`mtitle`	The "main" title for the plot of the exrema
`ext`	The extrema points, here, `k` furthest points from vertices in each vertex region.
`X`	The input data, `Dt`, can be a matrix or data frame
`num.points`	The number of data points, i.e., size of `Dt`
`supp`	Support of the data points, here, it is T_b.
`cent`	The center point used for construction of edge regions.
`ncent`	Name of the center, `cent`, it is circumcenter "CC" for this function.
`regions`	Vertex regions inside the triangle, T_b, provided as a list.
`region.names`	Names of the vertex regions as "vr=1","vr=2","vr=3"
`region.centers`	Centers of mass of the vertex regions inside T_b.
`dist2ref`	Distances from `k` furthest points in each vertex region to the corresponding vertex (each row representing a vertex).

See Also

fr2vTbVRCC, fr2vVRCC, fr2eTeER, and Kfr2vVRCC

Examples

```
## Not run:
c1<-.4; c2<-.6;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)
n<-20
k<-3

set.seed(1)
dat<-runif.bastri(n,c1,c2)$g

Ext<-Kfr2vTbVRCC(dat,c1,c2,k)
Ext
summary(Ext)
plot(Ext)

Kfr2vTbVRCC(dat[1:k,],c1,c2,k)
Kfr2vTbVRCC(dat[1,],c1,c2,k)

kf2v<-Kfr2vTbVRCC(dat,c1,c2,k)

CC<-circ.cent.bastri(c1,c2)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1],dat[,1])
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(dat)
points(kf2v$ext,pch=4,col=2)

txt<-rbind(Tb,CC,Ds)
xc<-txt[,1]+c(-.03,.03,.02,.07,.06,-.05,.01)
yc<-txt[,2]+c(.02,.02,.03,-.02,.02,.03,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

Kfr2vTbVRCC(dat,c1,c2,k)

Kfr2vTbVRCC(c(.4,.2),c1,c2,k)

dat.fr<-data.frame(a=dat)
```

```
Kfr2vTbVRCC(dat.fr,c1,c2,k)

dat2<-rbind(dat,c(.2,.4))
Kfr2vTbVRCC(dat2,c1,c2,k)

Kfr2vTbVRCC(dat2,c1,c2,k,ch.all.intri = TRUE)
#gives an error message since not all points are in the basic triangle

## End(Not run)
```

Kfr2vVRCC *The* k *furthest points in a data set from vertices in each CC-vertex region in a triangle*

Description

An object of class "Extrema". Returns the k furthest data points among the data set, Dt, in each CC-vertex region from the vertex in the triangle, $tri = T(A, B, C)$, vertices are stacked row-wise. Vertex region labels/numbers correspond to the row number of the vertex in tri.

ch.all.intri is for checking whether all data points are inside tri (default is FALSE). If some of the data points are not inside tri and ch.all.intri=TRUE, then the function yields an error message. If some of the data points are not inside tri and ch.all.intri=FALSE, then the function yields the closest points to edges among the data points inside tri (yields NA if there are no data points inside tri).

In the extrema, ext, in the output, the first k entries are the k furthest points from vertex 1, second k entries are k furthest points are from vertex 2, and last k entries are the k furthest points from vertex 3. If data size does not allow, NA's are inserted for some or all of the k furthest points for each vertex.

Usage

```
Kfr2vVRCC(Dt, tri, k, ch.all.intri = FALSE)
```

Arguments

Dt	A set of 2D points representing the set of data points.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
k	A positive integer. k furthest data points in each CC-vertex region are to be found if exists, else NA are provided for (some of) the k furthest points.
ch.all.intri	A logical argument (default=FALSE) to check whether all data points are inside the triangle tri. So if it is TRUE, the function checks if all data points are inside the closure of the triangle (i.e. interior and boundary combined) else it does not.

Value

A list with the elements

txt1	Vertex labels are A=1, B=2, and C=3 (corresponds to row number in Extrema Points).
txt2	A shorter description of the distances as "Distances of k furthest points in the vertex regions to Vertices".
type	Type of the extrema points
desc	A short description of the extrema points
mtitle	The "main" title for the plot of the exrema
ext	The extrema points, here, k furthest points from vertices in each CC-vertex region in the triangle tri.
X	The input data, Dt, can be a matrix or data frame
num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, it is tri for this function.
cent	The center point used for construction of vertex regions
ncent	Name of the center, cent, it is circumcenter "CC" for this function.
regions	Vertex regions inside the triangle, tri, provided as a list
region.names	Names of the vertex regions as "vr=1","vr=2","vr=3"
region.centers	Centers of mass of the vertex regions inside T_b.
dist2ref	Distances from k furthest points in each vertex region to the corresponding vertex (each row representing a vertex in tri). Among the distances the first k entries are the distances from the k furthest points from vertex 1 to vertex 1, second k entries are distances from the k furthest points from vertex 2 to vertex 2, and the last k entries are the distances from the k furthest points from vertex 3 to vertex 3.

See Also

Kfr2vTbVRCC, fr2vTbVRCC, fr2vVRCC, and fr2eTeER

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20
k<-3

set.seed(1)
dat<-runif.tri(n,Tr)$g

Ext<-Kfr2vVRCC(dat,Tr,k)
Ext
summary(Ext)
plot(Ext)
```

```
Kfr2vVRCC(dat[1:k,],Tr,k)
Kfr2vVRCC(dat[1,],Tr,k)

dat2<-rbind(dat,c(.2,.4))
Kfr2vVRCC(dat2,Tr,k)  #try also Kfr2vVRCC(dat2,Tr,k,ch.all.intri = TRUE)

kf2v<-Kfr2vVRCC(dat,Tr,k)

CC<-circ.cent.tri(Tr)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
points(dat)
points(kf2v$ext,pch=4,col=2)

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.06,.08,.05,.12,-.1,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.04,.0,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

Kfr2vVRCC(dat,Tr,k)

dat.fr<-data.frame(a=dat)
Kfr2vVRCC(dat.fr,Tr,k)

dat.fr<-data.frame(a=Tr)
fr2vVRCC(dat,dat.fr)
```

`Line` *The line joining two distinct 2D points* `a` *and* `b`

Description

An object of class "Lines". Returns the `equation`, `slope`, `intercept`, and y-coordinates of the line crossing two distinct 2D points `a` and `b` with x-coordinates provided in vector `x`.

This function is different from the `line` function in the standard `stats` package in R in the sense that `Line(a,b,x)` fits the line passing through points `a` and `b` and returns various quantities (see

below) for this line and x is the x-coordinates of the points we want to find on the Line(a,b,x) while line(a,b) fits the line robustly whose x-coordinates are in a and y-coordinates are in b.

Line(a,b,x) and line(x,Line(A,B,x)$y) would yield the same straight line (i.e., the line with the same coefficients.)

Usage

```
Line(a, b, x)
```

Arguments

a, b	2D points that determine the straight line (i.e., through which the straight line passes).
x	A scalar or a vector of scalars representing the x-coordinates of the line.

Value

A list with the elements

desc	A description of the line
mtitle	The "main" title for the plot of the line
points	The input points a and b through which the straight line passes (stacked row-wise, i.e., row 1 is point a and row 2 is point b).
x	The input scalar or vector which constitutes the x-coordinates of the point(s) of interest on the line.
y	The output scalar or vector which constitutes the y-coordinates of the point(s) of interest on the line. If x is a scalar, then y will be a scalar and if x is a vector of scalars, then y will be a vector of scalars.
slope	Slope of the line, Inf is allowed, passing through points a and b
intercept	Intercept of the line passing through points a and b
equation	Equation of the line passing through points a and b

See Also

slope, paraline, perpline, line in the generic stats package and and Line3D

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75)

xr<-range(A,B);
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=20)  #try also l=100

lnAB<-Line(A,B,x)
lnAB
summary(lnAB)
plot(lnAB)
```

```
line(A,B)  #this takes A as the x points and B as the y points and fits the line

y<-lnAB$y
Xlim<-range(x,A,B)
if (!is.na(y[1])) {Ylim<-range(y,A,B)} else {Ylim<-range(A,B)}
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
pf<-c(xd,-yd)*.025

#plot of the line joining A and B
plot(rbind(A,B),pch=1,xlab="x",ylab="y",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
if (!is.na(y[1])) {lines(x,y,lty=1)} else {abline(v=A[1])}
text(rbind(A+pf,B+pf),c("A","B"))
int<-round(lnAB$intercept,2)  #intercept
sl<-round(lnAB$slope,2)  #slope
text(rbind((A+B)/2+pf*3),ifelse(is.na(int),paste("x=",A[1]),
ifelse(sl==0,paste("y=",int),
ifelse(sl==1,ifelse(sign(int)<0,paste("y=x",int),paste("y=x+",int)),
ifelse(sign(int)<0,paste("y=",sl,"x",int),paste("y=",sl,"x+",int))))))

Line(c(1,2),c(2,3),3)
```

`Line3D` *The line crossing 3D point* `r0` *in the direction of vector* `v` *(or if* `v` *is a point, in direction of* $v - r0$*)*

Description

An object of class "Lines3D". Returns the equation, x-, y-, and z-coordinates of the line crossing 3D point `r0` in the direction of vector `v` (of if `v` is a point, in the direction of $v - r0$) with the parameter `t` being provided in vector `t`.

Usage

```
Line3D(r0, v, t, dir.vec = TRUE)
```

Arguments

`r0`	A 3D point through which the straight line passes.
`v`	A 3D vector which determines the direction of the straight line (i.e., the straight line would be parallel to this vector) if the `dir.vec`=T, otherwise it is 3D point and $v - r0$ determines the direction of the the straight line.
`t`	A scalar or a vector of scalars representing the parameter of the coordinates of the line (for the form: $x = x_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $r_0 = (x_0, y_0, z_0)$ and $v = (a, b, c)$ if `dir.vec`=T, else $v - r0 = (a, b, c)$).
`dir.vec`	A logical argument about `v`, if `TRUE` `v` is treated as a vector, else `v` is treated as a point and so the direction vector is taken to be $v - r0$.

Value

A list with the elements

desc	A description of the line
mtitle	The "main" title for the plot of the line
pts	The input points that determine a line and/or a plane, NULL for this function.
pnames	The names of the input points that determine a line and/or a plane, NULL for this function.
vecs	The point r0 and the vector v (if dir.vec=T) or the point v (if dir.vec=F). The first row is r0 and the second row is v.
vec.names	The names of the point r0 and the vector v (if dir.vec=T) or the point v (if dir.vec=F).
x,y,z	The x-, y- and z-coordinates of the point(s) of interest on the line.
tsq	The scalar or the vector of the parameter in defining each coordinate of the line for the form: $x = x_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $r_0 = (x_0, y_0, z_0)$ and $v = (a, b, c)$ if dir.vec=T, else $v - r0 = (a, b, c)$.
equation	Equation of the line passing through point r0 in the direction of the vector v (if dir.vec=T) else in the direction of $v - r0$. The line equation is in the form: $x = x_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $r_0 = (x_0, y_0, z_0)$ and $v = (a, b, c)$ if dir.vec=T, else $v - r0 = (a, b, c)$.

See Also

line, paraline3D, and Plane

Examples

```
A<-c(1,10,3); B<-c(1,1,3);

vecs<-rbind(A,B)

Line3D(A,B,.1)
Line3D(A,B,.1,dir.vec=FALSE)

tr<-range(vecs);
tf<-(tr[2]-tr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=20)  #try also l=100

lnAB3D<-Line3D(A,B,tsq)
lnAB3D
summary(lnAB3D)
plot(lnAB3D)

Line3D(A,B,c(.1,.2))

x<-lnAB3D$x
y<-lnAB3D$y
```

```
z<-lnAB3D$z

zr<-range(z)
zf<-(zr[2]-zr[1])*.2
Bv<-B*tf*5

Xlim<-range(x)
Ylim<-range(y)
Zlim<-range(z)

xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

Dr<-A+min(tsq)*B

plot3D::lines3D(x, y, z, phi = 0, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),zlim=Zlim+zd*c(-.1,.1),
        pch = 20, cex = 2, ticktype = "detailed")
plot3D::arrows3D(Dr[1],Dr[2],Dr[3]+zf,Dr[1]+Bv[1],Dr[2]+Bv[2],Dr[3]+zf+Bv[3], add=TRUE)
plot3D::points3D(A[1],A[2],A[3],add=TRUE)
plot3D::arrows3D(A[1],A[2],A[3]-2*zf,A[1],A[2],A[3],lty=2, add=TRUE)
plot3D::text3D(A[1],A[2],A[3]-2*zf,labels="initial point",add=TRUE)
plot3D::text3D(A[1],A[2],A[3]+zf/2,labels=expression(r[0]),add=TRUE)
plot3D::arrows3D(Dr[1]+Bv[1]/2,Dr[2]+Bv[2]/2,Dr[3]+3*zf+Bv[3]/2,
Dr[1]+Bv[1]/2,Dr[2]+Bv[2]/2,Dr[3]+zf+Bv[3]/2,lty=2, add=TRUE)
plot3D::text3D(Dr[1]+Bv[1]/2,Dr[2]+Bv[2]/2,Dr[3]+3*zf+Bv[3]/2,
labels="direction vector",add=TRUE)
plot3D::text3D(Dr[1]+Bv[1]/2,Dr[2]+Bv[2]/2,Dr[3]+zf+Bv[3]/2,labels="v",add=TRUE)
plot3D::text3D(0,0,0,labels="O",add=TRUE)
```

NASbastri *The vertices of the Arc Slice (AS) Proximity Region in the basic triangle*

Description

Returns the end points of the line segments and arc-slices that constitute the boundary of AS proximity region for a point in the basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where c1 is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Vertex regions are based on the center M="CC" for circumcenter of T_b; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_b; default is M="CC" the circumcenter of T_b. rv is the index of the vertex region pt resides, with default=NULL.

If pt is outside T_b, it returns NULL for the proximity region. dec is the number of decimals (default is 4) to round the barycentric coordinates when checking whether the end points fall on the boundary of the triangle T_b or not (so as not to miss the intersection points due to precision in the decimals).

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
NASbastri(pt, c1, c2, M = "CC", rv = NULL, dec = 4)
```

Arguments

`pt`	A 2D point whose AS proximity region is to be computed.
`c1, c2`	Positive real numbers representing the top vertex in basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, `c1` must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.
`M`	The center of the triangle. "CC" stands for circumcenter of the triangle T_b or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M`="CC" i.e. the circumcenter of T_b.
`rv`	The index of the `M`-vertex region containing the point, either 1, 2, 3 or `NULL` (default is `NULL`).
`dec`	a positive integer the number of decimals (default is 4) to round the barycentric coordinates when checking whether the end points fall on the boundary of the triangle T_b or not.

Value

A list with the elements

`L,R`	The end points of the line segments on the boundary of the AS proximity region. Each row in `L` and `R` constitute a line segment on the boundary.
`Arc.Slices`	The end points of the arc-slices on the circular parts of the AS proximity region. Here points in row 1 and row 2 constitute the end points of one arc-slice, points on row 3 and row 4 constitute the end points for the next arc-slice and so on

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

NAStri and IndNASbastri

Examples

```
c1<-.4; c2<-.6  #try also c1<-.2; c2<-.2;
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C)

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.2)

P1<-as.numeric(runif.bastri(1,c1,c2)$g);  #try also P1<-c(.3,.2)
NASbastri(P1,c1,c2)  #default with M="CC"
NASbastri(P1,c1,c2,M)

#or try
Rv<-rv.bastriCC(P1,c1,c2)$rv
NASbastri(P1,c1,c2,M,Rv)

NASbastri(c(3,5),c1,c2,M)

P2<-c(.5,.4)
NASbastri(P2,c1,c2,M)

P3<-c(1.5,.4)
NASbastri(P3,c1,c2,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

#plot of the NAS region
P1<-as.numeric(runif.bastri(1,c1,c2)$g);
CC<-circ.cent.bastri(c1,c2)

if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
rv<-rv.bastriCC(P1,c1,c2)$rv
} else
{cent<-M
cent.name<-"M"
Ds<-cp2e.bastri(c1,c2,M)
D1<-Ds[1,]; D2<-Ds[2,]; D3<-Ds[3,]
rv<-rv.bastri.cent(P1,c1,c2,M)$rv
}
RV<-Tb[rv,]
rad<-Dist(P1,RV)

Int.Pts<-NASbastri(P1,c1,c2,M)
```

```
Xlim<-range(Tb[,1],P1[1]+rad,P1[1]-rad)
Ylim<-range(Tb[,2],P1[2]+rad,P1[2]-rad)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
points(rbind(Tb,P1,rbind(Int.Pts$L,Int.Pts$R)))
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
interp::circles(P1[1],P1[2],rad,lty=2)
L<-Int.Pts$L; R<-Int.Pts$R
segments(L[,1], L[,2], R[,1], R[,2], lty=1,col=2)
Arcs<-Int.Pts$A;
if (!is.null(Arcs))
{
  K<-nrow(Arcs)/2
  for (i in 1:K)
  {A1<-Arcs[2*i-1,]; A2<-Arcs[2*i,];
  angles<-angle.str2end(A1,P1,A2)$c

  plotrix::draw.arc(P1[1],P1[2],rad,angle1=angles[1],angle2=angles[2],col=2)
  }
}

#proximity region with the triangle (i.e., for labeling the vertices of the NAS)
IP.txt<-intpts<-c()
if (!is.null(Int.Pts$A))
{
  intpts<-unique(round(Int.Pts$A,7))
  #this part is for labeling the intersection points of the spherical
  for (i in 1:(length(intpts)/2))
    IP.txt<-c(IP.txt,paste("I",i+1, sep = ""))
}
txt<-rbind(Tb,P1,cent,intpts)
txt.str<-c("A","B","C","P1",cent.name,IP.txt)
text(txt+cbind(rep(xd*.02,nrow(txt)),rep(-xd*.03,nrow(txt))),txt.str)

c1<-.4; c2<-.6;
P1<-c(.3,.2)
NASbastri(P1,c1,c2,M)
```

NAStri *The vertices of the Arc Slice (AS) Proximity Region in a general triangle*

Description

Returns the end points of the line segments and arc-slices that constitute the boundary of AS proximity region for a point in the triangle $tri = T(A, B, C) = (rv = 1, rv = 2, rv = 3)$.

Vertex regions are based on the center M="CC" for circumcenter of tri; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle tri; default is M="CC" the circumcenter of tri. rv is the index of the vertex region pt1 resides, with default=NULL.

If pt is outside of tri, it returns NULL for the proximity region. dec is the number of decimals (default is 4) to round the barycentric coordinates when checking the points fall on the boundary of the triangle tri or not (so as not to miss the intersection points due to precision in the decimals).

See also (Ceyhan (2005, 2010)).

Usage

```
NAStri(pt, tri, M = "CC", rv = NULL, dec = 4)
```

Arguments

pt	A 2D point whose AS proximity region is to be computed.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	The center of the triangle. "CC" stands for circumcenter of the triangle tri or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri; default is M="CC" i.e. the circumcenter of tri.
rv	Index of the M-vertex region containing the point pt, either 1, 2, 3 or NULL (default is NULL).
dec	a positive integer the number of decimals (default is 4) to round the barycentric coordinates when checking whether the end points fall on the boundary of the triangle tri or not.

Value

A list with the elements

L,R	End points of the line segments on the boundary of the AS proximity region. Each row in L and R constitute a line segment on the boundary.
Arc.Slices	The end points of the arc-slices on the circular parts of the AS proximity region. Here points in row 1 and row 2 constitute the end points of one arc-slice, points on row 3 and row 4 constitute the end points for the next arc-slice and so on

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number

of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

NASbastri, NPEtri, NCStri and IndNAStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(.6,.2)

P1<-as.numeric(runif.tri(1,Tr)$g)  #try also P1<-c(1.3,1.2)
NAStri(P1,Tr,M)

#or try
Rv<-rv.triCC(P1,Tr)$rv
NAStri(P1,Tr,M,Rv)

NAStri(c(3,5),Tr,M)

P2<-c(1.5,1.4)
NAStri(P2,Tr,M)

P3<-c(1.5,.4)
NAStri(P3,Tr,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

CC<-circ.cent.tri(Tr)  #the circumcenter

if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
rv<-rv.triCC(P1,Tr)$rv
} else
{cent<-M
cent.name<-"M"
Ds<-cp2e.tri(Tr,M)
D1<-Ds[1,]; D2<-Ds[2,]; D3<-Ds[3,]
rv<-rv.tri.cent(P1,Tr,M)$rv
}
RV<-Tr[rv,]
rad<-Dist(P1,RV)

Int.Pts<-NAStri(P1,Tr,M)

#plot of the NAS region
```

```
Xlim<-range(Tr[,1],P1[1]+rad,P1[1]-rad)
Ylim<-range(Tr[,2],P1[2]+rad,P1[2]-rad)
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",asp=1,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
#asp=1 must be the case to have the arc properly placed in the figure
polygon(Tr)
points(rbind(Tr,P1,rbind(Int.Pts$L,Int.Pts$R)))
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
interp::circles(P1[1],P1[2],rad,lty=2)
L<-Int.Pts$L; R<-Int.Pts$R
segments(L[,1], L[,2], R[,1], R[,2], lty=1,col=2)
Arcs<-Int.Pts$A;
if (!is.null(Arcs))
{
  K<-nrow(Arcs)/2
  for (i in 1:K)
  {A1<-Int.Pts$Arc[2*i-1,]; A2<-Int.Pts$Arc[2*i,];
  angles<-angle.str2end(A1,P1,A2)$c

  test.ang1<-angles[1]+(.01)*(angles[2]-angles[1])
  test.Pnt<-P1+rad*c(cos(test.ang1),sin(test.ang1))
  if (!in.triangle(test.Pnt,Tr,boundary = T)$i) {angles<-c(min(angles),max(angles)-2*pi)}
  plotrix::draw.arc(P1[1],P1[2],rad,angle1=angles[1],angle2=angles[2],col=2)
  }
}

#proximity region with the triangle (i.e., for labeling the vertices of the NAS)
IP.txt<-intpts<-c()
if (!is.null(Int.Pts$A))
{
  intpts<-unique(round(Int.Pts$A,7))
  #this part is for labeling the intersection points of the spherical
  for (i in 1:(length(intpts)/2))
    IP.txt<-c(IP.txt,paste("I",i+1, sep = ""))
}
txt<-rbind(Tr,P1,cent,intpts)
txt.str<-c("A","B","C","P1",cent.name,IP.txt)
text(txt+cbind(rep(xd*.02,nrow(txt)),rep(-xd*.03,nrow(txt))),txt.str)

P1<-c(.3,.2)
NAStri(P1,Tr,M)
```

NCSint *The end points of the Central Similarity (CS) Proximity Region for a point - one interval case*

Description

Returns the end points of the interval which constitutess the CS proximity region for a point in the interval $int = (a, b) = (rv = 1, rv = 2)$.

CS proximity region is constructed with respect to the interval `int` with expansion parameter $t > 0$ and centrality parameter `c` in $(0, 1)$.

Vertex regions are based on the (parameterized) center, M_c, which is $M_c = a + c(b - a)$ for the interval, $int = (a, b)$. The CS proximity region is constructed whether `x` is inside or outside the interval `int`.

See also (Ceyhan (2016)).

Usage

```
NCSint(x, t, c = 0.5, int)
```

Arguments

`x`	A 1D point for which CS proximity region is constructed.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$, and default=0.5.
`int`	A vector of two real numbers representing an interval.

Value

The interval which constitutes the CS proximity region for the point `x`

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`NPEint` and `NCStri`

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

NCSint(7,t,c,int)
NCSint(17,t,c,int)
NCSint(1,t,c,int)
NCSint(-1,t,c,int)
```

```
NCSint(3,t,c,int)
NCSint(4,t,c,int)
NCSint(a,t,c,int)
```

NCStri — *The vertices of the Central Similarity (CS) Proximity Region in a general triangle*

Description

Returns the vertices of the CS proximity region (which is itself a triangle) for a point in the triangle $tri = T(A, B, C) = (rv = 1, rv = 2, rv = 3)$.

CS proximity region is defined with respect to the triangle `tri` with expansion parameter $t > 0$ and edge regions based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`.

Edge regions are labeled as 1,2,3 rowwise for the corresponding vertices of the triangle `tri`. `re` is the index of the edge region `pt` resides, with default=`NULL`. If `pt` is outside of `tri`, it returns `NULL` for the proximity region.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
NCStri(pt, tau, tri, M, re = NULL)
```

Arguments

`pt`	A 2D point whose CS proximity region is to be computed.
`tau`	A positive real number which serves as the expansion parameter in CS proximity region.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.
`re`	Index of the `M`-edge region containing the point `pt`, either 1, 2, 3 or `NULL` (default is `NULL`).

Value

Vertices of the triangular region which constitutes the CS proximity region with expansion parameter $\tau > 0$ and center `M` for a point `pt`

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

NPEtri, NAStri, and IndNCStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
tau<-1.5

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

n<-10
set.seed(1)
dat<-runif.tri(n,Tr)$g

NCStri(dat[7,],tau,Tr,M)

P1<-as.numeric(runif.tri(1,Tr)$g)  #try also P1<-c(.4,.2)
NCStri(P1,tau,Tr,M)

P2<-c(1.8,.5)
NCStri(P2,tau,Tr,M)

P3<-c(1.7,.6)
NCStri(P3,tau,Tr,M)

#or try
re<-redges.tri.cent(P2,Tr,M)$re
NCStri(P2,tau,Tr,M,re)

dat.fr<-data.frame(a=Tr)
NCStri(P2,tau,dat.fr,M)
```

NPEbastri — *The vertices of the Proportional Edge (PE) Proximity Region in a basic triangle*

Description

Returns the vertices of the PE proximity region (which is itself a triangle) for a point in the basic triangle $T_b = T((0,0),(1,0),(c1,c2)) = (rv = 1, rv = 2, rv = 3)$.

PE proximity region is defined with respect to the basic triangle T_b with expansion parameter $r \geq 1$ and vertex regions based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the basic triangle T_b or based on the circumcenter of T_b; default is $M = (1,1,1)$ i.e. the center of mass of T_b.

Vertex regions are labeled as 1,2,3 rowwise for the vertices of the triangle T_b. `rv` is the index of the vertex region `pt` resides, with default=`NULL`. If `pt` is outside of `tri`, it returns `NULL` for the proximity region.

See also (Ceyhan (2005, 2010)).

Usage

```
NPEbastri(pt, r, c1, c2, M = c(1, 1, 1), rv = NULL)
```

Arguments

`pt`	A 2D point whose PE proximity region is to be computed.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c1, c2`	Positive real numbers representing the top vertex in basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, `c1` must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the basic triangle T_b or the circumcenter of T_b; default is $M = (1,1,1)$ i.e. the center of mass of T_b.
`rv`	Index of the `M`-vertex region containing the point `pt`, either 1, 2, 3 or `NULL` (default is `NULL`).

Value

Vertices of the triangular region which constitutes the PE proximity region with expansion parameter `r` and center `M` for a point `pt`

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

NPEtri, NAStri, NCStri and IndNPEbastri

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.2)

r<-2

P1<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also P1<-c(.4,.2)
NPEbastri(P1,r,c1,c2,M)

#or try
Rv<-rv.bastri.cent(P1,c1,c2,M)$rv
NPEbastri(P1,r,c1,c2,M,Rv)

P2<-c(1.8,.5)
NPEbastri(P2,r,c1,c2,M)

P3<-c(1.7,.6)
NPEbastri(P3,r,c1,c2,M)

M<-c(1.3,1.3)
r<-2

## Not run:
P1<-c(1.4,1.2)
P2<-c(1.5,1.26)
NPEbastri(P1,r,c1,c2,M)
#gives an error since center is not the circumcenter or not in the interior of the triangle
NPEbastri(P2,r,c1,c2,M)
#gives an error since center is not the circumcenter or not in the interior of the triangle

## End(Not run)
```

NPEint *The end points of the Proportional Edge (PE) Proximity Region for a point - one interval case*

Description

Returns the end points of the interval which constitutess the PE proximity region for a point in the interval $int = (a, b) = (rv = 1, rv = 2)$. PE proximity region is constructed with respect to the interval `int` with expansion parameter $r \geq 1$ and centrality parameter `c` in $(0, 1)$.

Vertex regions are based on the (parameterized) center, M_c, which is $M_c = a + c(b-a)$ for the interval, $int = (a, b)$. The PE proximity region is constructed whether `x` is inside or outside the interval `int`.

See also (Ceyhan (2012)).

Usage

```
NPEint(x, r, c = 0.5, int)
```

Arguments

`x`	A 1D point for which PE proximity region is constructed.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b-a)$, and default=0.5
`int`	A vector of two real numbers representing an interval.

Value

The interval which constitutes the PE proximity region for the point `x`

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`NCSint`, `NPEtri` and `NPEtetra`

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

NPEint(7,r,c,int)
NPEint(17,r,c,int)
NPEint(1,r,c,int)
NPEint(-1,r,c,int)

NPEint(3,r,c,int)
NPEint(4,r,c,int)
NPEint(a,r,c,int)
```

NPEstdtetra *The vertices of the Proportional Edge (PE) Proximity Region in the standard regular tetrahedron*

Description

Returns the vertices of the PE proximity region (which is itself a tetrahedron) for a point in the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/3)) = (rv = 1, rv = 2, rv = 3, rv = 4)$.

PE proximity region is defined with respect to the tetrahedron T_h with expansion parameter $r \geq 1$ and vertex regions based on the circumcenter of T_h (which is equivalent to the center of mass in the standard regular tetrahedron).

Vertex regions are labeled as 1,2,3,4 rowwise for the vertices of the tetrahedron T_h. `rv` is the index of the vertex region `pt` resides, with default=`NULL`. If `pt` is outside of T_h, it returns `NULL` for the proximity region.

See also (Ceyhan (2005, 2010)).

Usage

```
NPEstdtetra(pt, r, rv = NULL)
```

Arguments

`pt`	A 3D point whose PE proximity region is to be computed.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`rv`	Index of the vertex region containing the point, either 1, 2, 3, 4 or `NULL` (defaut is `NULL`).

Value

Vertices of the tetrahedron which constitutes the PE proximity region with expansion parameter `r` and circumcenter (or center of mass) for a point `pt` in the standard regular tetrahedron

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`NPEtetra`, `NPEtri` and `NPEint`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-10  #try also n<-20
dat<-runif.stdtetra(n)$g
r<-1.5
NPEstdtetra(dat[1,],r)

NPEstdtetra(dat[5,],r)

NPEstdtetra(c(.4,.4,.4),r)
NPEstdtetra(c(.5,.5,.5),r)
NPEstdtetra(c(.5,.5,5),r)

#or try
RV<-rv.tetraCC(dat[1,],tetra)$rv
NPEstdtetra(dat[1,],r,rv=RV)

NPEstdtetra(c(-1,-1,-1),r,rv=NULL)

P1<-c(.1,.1,.1)
P2<-c(.5,.5,.5)
NPEstdtetra(P1,r)
```

NPEtetra — *The vertices of the Proportional Edge (PE) Proximity Region in a tetrahedron*

Description

Returns the vertices of the PE proximity region (which is itself a tetrahedron) for a point in the tetrahedron `th`.

PE proximity region is defined with respect to the tetrahedron `th` with expansion parameter $r \geq 1$ and vertex regions based on the center `M` which is circumcenter ("CC") or center of mass ("CM") of `th` with default="CM".

Vertex regions are labeled as 1,2,3,4 rowwise for the vertices of the tetrahedron `th`. `rv` is the index of the vertex region `pt` resides, with default=`NULL`. If `pt` is outside of `th`, it returns `NULL` for the proximity region.

See also (Ceyhan (2005, 2010)).

Usage

```
NPEtetra(pt, r, th, M = "CM", rv = NULL)
```

Arguments

pt	A 3D point whose PE proximity region is to be computed.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
th	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.
M	The center to be used in the construction of the vertex regions in the tetrahedron, th. Currently it only takes "CC" for circumcenter and "CM" for center of mass; default="CM".
rv	Index of the vertex region containing the point, either 1, 2, 3, 4 (default is NULL).

Value

Vertices of the tetrahedron which constitutes the PE proximity region with expansion parameter r and circumcenter (or center of mass) for a point pt in the tetrahedron

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

NPEstdtetra, NPEtri and NPEint

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-10  #try also n<-20

dat<-runif.tetra(n,tetra)$g

M<-"CM"  #try also M<-"CC"
r<-1.5

NPEtetra(dat[1,],r,tetra)  #uses the default M="CM"
NPEtetra(dat[1,],r,tetra,M)

NPEtetra(dat[5,],r,tetra,M)

NPEtetra(c(.4,.4,.4),r,tetra,M)
NPEtetra(c(.5,.5,.5),r,tetra,M)

#or try
```

```
RV<-rv.tetraCM(dat[1,],tetra)$rv
NPEtetra(dat[1,],r,tetra,M,rv=RV)

NPEtetra(c(-1,-1,-1),r,tetra,M,rv=NULL)

P1<-c(.1,.1,.1)
P2<-c(.5,.5,.5)
NPEtetra(P1,r,tetra,M)
```

NPEtri *The vertices of the Proportional Edge (PE) Proximity Region in a general triangle*

Description

Returns the vertices of the PE proximity region (which is itself a triangle) for a point in the triangle $tri = T(A, B, C) = (rv = 1, rv = 2, rv = 3)$.

PE proximity region is defined with respect to the triangle `tri` with expansion parameter $r \geq 1$ and vertex regions based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

Vertex regions are labeled as 1,2,3 rowwise for the vertices of the triangle `tri`. `rv` is the index of the vertex region `pt` resides, with default=`NULL`. If `pt` is outside of `tri`, it returns `NULL` for the proximity region.

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)).

Usage

```
NPEtri(pt, r, tri, M = c(1, 1, 1), rv = NULL)
```

Arguments

`pt`	A 2D point whose PE proximity region is to be computed.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.
`rv`	Index of the `M`-vertex region containing the point `pt`, either 1, 2, 3 or `NULL` (default is `NULL`).

Value

Vertices of the triangular region which constitutes the PE proximity region with expansion parameter `r` and center `M` for a point `pt`

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

`NPEbastri`, `NAStri`, `NCStri` and `IndNPEtri`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5

n<-10
set.seed(1)
dat<-runif.tri(n,Tr)$g

NPEtri(dat[7,],r,Tr,M)

P1<-as.numeric(runif.tri(1,Tr)$g)  #try also P1<-c(.4,.2)
NPEtri(P1,r,Tr,M)

P2<-c(1.8,.5)
NPEtri(P2,r,Tr,M)

P3<-c(1.7,.6)
NPEtri(P3,r,Tr,M)

M<-c(1.3,1.3)
r<-2

P1<-c(1.4,1.2)
P2<-c(1.5,1.26)
NPEtri(P1,r,Tr,M)
```

```
NPEtri(P2,r,Tr,M)

#or try
Rv<-rv.tri.cent(P1,Tr,M)$rv
NPEtri(P1,r,Tr,M,Rv)

dat.fr<-data.frame(a=Tr)
NPEtri(P2,r,dat.fr,M)
```

NumArcsASMT *Number of arcs of Arc Slice Proximity Catch Digraphs (AS-PCDs) - multiple triangle case*

Description

Returns the number of arcs of Arc Slice Proximity Catch Digraph (AS-PCD) whose vertices are the data points in Xp in the multiple triangle case.

AS proximity regions are defined with respect to the Delaunay triangles based on Yp points and vertex regions in each triangle are based on the center M="CC" for circumcenter of each Delaunay triangle or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle; default is M="CC" i.e. circumcenter of each triangle.

Convex hull of Yp is partitioned by the Delaunay triangles based on Yp points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of Yp points).

See (Ceyhan (2005, 2010)) for more on AS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
NumArcsASMT(Xp, Yp, M = "CC")
```

Arguments

Xp	A set of 2D points which constitute the vertices of the AS-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
M	The center of the triangle. "CC" stands for circumcenter of each Delaunay triangle tri or 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle; default is M="CC" i.e. the circumcenter of each triangle.

Value

A list with the elements

num.pts.in.CH	Number of Xp points in the convex hull of Yp points
num.arcs	Number of arcs in the AS-PCD for the Xp points inside the convex hull of Yp points
weight.vec	The vector of areas of the Delaunay triangles that have at least on Xp points in them

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

NumArcsAStri, NumArcsPEMT, and NumArcsCSMT

Examples

```
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny),runif(ny))

oldpar <- par(mfrow = c(1,2))
plotDeltri(Xp,Yp,xlab="",ylab="")
par(oldpar)

M<-"CC"  #try also M<-c(1,1,1)

NumArcsASMT(Xp,Yp,M)
NumArcsASMT(Xp,Yp[1:3,],M)

NumArcsASMT(c(.4,.2),Yp,M)

NumArcsASMT(Xp,rbind(Yp,Yp),M)

dat.fr<-data.frame(a=Xp)
NumArcsASMT(dat.fr,Yp,M)
```

NumArcsAStri *Number of arcs of Arc Slice Proximity Catch Digraphs (AS-PCDs) - one triangle case*

Description

Returns the number of arcs of Arc Slice Proximity Catch Digraphs (AS-PCDs) whose vertices are the 2D data set, dat. The data points could be inside or outside a general triangle $tri = T(A, B, C) = (rv = 1, rv = 2, rv = 3)$, with vertices of tri stacked row-wise.

AS proximity regions are defined with respect to the triangle tri and vertex regions are based on the center M="CC" for circumcenter of tri; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle tri; default is M="CC" i.e. circumcenter of tri. For the number of arcs, loops are not allowed, so arcs are only possible for points inside the triangle, tri.

See also (Ceyhan (2005, 2010)).

Usage

```
NumArcsAStri(dat, tri, M = "CC")
```

Arguments

dat	A set of 2D points which constitute the vertices of the digraph (i.e., AS-PCD).
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	The center of the triangle. "CC" stands for circumcenter of the triangle tri or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of tri; default is M="CC" i.e. the circumcenter of tri.

Value

The number of arcs of Arc Slice Proximity Catch Digraphs (AS-PCDs)

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

NumArcsASMT, NumArcsPEtri, and NumArcsCStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

NumArcsAStri(dat,Tr)  #with default M="CC"
NumArcsAStri(dat,Tr,M)

NumArcsAStri(rbind(dat,c(0,2)),Tr,M)

dat.fr<-data.frame(a=dat)
NumArcsAStri(dat.fr,Tr,M)
```

NumArcsCSend1D *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) - end interval case*

Description

Returns the number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) whose vertices are a 1D numerical data set, dat, outside the interval $int = (a, b)$.

CS proximity region is constructed only with expansion parameter $t > 0$ for points outside the interval (a, b).

End vertex regions are based on the end points of the interval, i.e., the corresponding end vertex region is an interval as $(-\infty, a)$ or (b, ∞) for the interval (a, b). For the number of arcs, loops are not allowed, so arcs are only possible for points outside the interval, int, for this function.

See also (Ceyhan (2016)).

Usage

```
NumArcsCSend1D(dat, t, int)
```

Arguments

dat	A vector of 1D points which constitute the vertices of the digraph.
t	A positive real number which serves as the expansion parameter in CS proximity region.
int	A vector of two real numbers representing an interval.

Value

Number of arcs for the CS-PCD with vertices being 1D data set, dat, expansion parameter, t, for the end intervals.

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

NumArcsCSmid1D, NumArcsPEmid1D, and NumArcsPEend1D

Examples

```
a<-0; b<-10; int<-c(a,b)

n<-5
datL<-runif(n,a-5,a)
datR<-runif(n,b,b+5)
dat<-c(datL,datR)

NumArcsCSend1D(dat,t=2,int)

NumArcsCSend1D(dat,t=1.2,int)

NumArcsCSend1D(dat,t=4,int)

NumArcsCSend1D(dat,t=2,int+5)
NumArcsCSend1D(dat,t=2,int=c(-5,15))

n<-10  #try also n<-20
dat2<-runif(n,a-5,b+5)
NumArcsCSend1D(dat2,t=2,int)

t<-.5
NumArcsCSend1D(dat,t,int)
```

NumArcsCSint — *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) - one interval case*

Description

Returns the number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) whose vertices are the 1D data set dat in the one-interval case.

The data points could be inside or outside the interval is $int = (a, b)$.

CS proximity region is constructed with an expansion parameter $t > 0$ and a centrality parameter c in $(0, 1)$. CS proximity region is constructed for both points inside and outside the interval, hence the arcs may exist for all points inside or outside the interval.

See also (Ceyhan (2016)).

Usage

```
NumArcsCSint(dat, t, c = 0.5, int)
```

Arguments

dat	A set of 1D points which constitute the vertices of CS-PCD.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$; default=0.5.
int	A vector of two real numbers representing an interval.

Value

Number of arcs for the CS-PCD whose vertices are the 1D data set, dat, with expansion parameter, $t > 0$, and centrality parameter, c in $(0, 1)$.

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

NumArcsCSmid1D, NumArcsCSend1D, and NumArcsPEint

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

n<-10
dat<-runif(n,a,b)
NumArcsCSint(dat,t,c,int)

NumArcsCSint(3,t,c,int)

NumArcsCSint(dat,t,c=.3,int)

NumArcsCSint(dat,t=1.5,c,int)

n<-10  #try also n<-20
```

```
dat<-runif(n,a,b)

NumArcsCSint(dat,t,c,int)

n<-10
dat<-runif(n,a,b)
NumArcsCSint(dat,t,c,int)
```

`NumArcsCSmid1D` *Number of Arcs of of Central Similarity Proximity Catch Digraphs (CS-PCDs) - middle interval case*

Description

Returns the number of arcs of of Central Similarity Proximity Catch Digraphs (CS-PCDs) whose vertices are the given 1D numerical data set, `dat`.

CS proximity region $NCS(x, t, c)$ is defined with respect to the interval $int = (a, b)$ for this function. CS proximity region is constructed with expansion parameter $t > 0$ and centrality parameter `c` in $(0, 1)$.

Vertex regions are based on the center associated with the centrality parameter `c` in $(0, 1)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$ and for the number of arcs, loops are not allowed so arcs are only possible for points inside the middle interval `int` for this function.

See also (Ceyhan (2016)).

Usage

```
NumArcsCSmid1D(dat, t, c, int)
```

Arguments

`dat`	A set or vector of 1D points which constitute the vertices of CS-PCD.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
`int`	A vector of two real numbers representing an interval.

Value

Number of arcs for the PE-PCD whose vertices are the 1D data set, `dat`, with expansion parameter, $r \geq 1$, and centrality parameter, `c` in $(0, 1)$. PE proximity regions are defined only for `dat` points inside the interval `int`, i.e. arcs are possible for such points only.

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

NumArcsCSend1D, NumArcsPEmid1D, and NumArcsPEend1D

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

n<-10
dat<-runif(n,a,b)
NumArcsCSmid1D(dat,t,c,int)

NumArcsCSmid1D(dat,t,c=.3,int)

NumArcsCSmid1D(dat,t=1.5,c,int)

NumArcsCSmid1D(dat,t,c,int+5)
NumArcsCSmid1D(dat,t,c,int+10)

n<-10  #try also n<-20
dat<-runif(n,a-5,b+5)
NumArcsCSint(dat,t,c,int)

dat<-runif(n,a+10,b+10)
NumArcsCSmid1D(dat,t,c,int)

n<-10
dat<-runif(n,a,b)
NumArcsCSmid1D(dat,t,c,int)
```

NumArcsCSMT *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) - multiple triangle case*

Description

Returns the number of arcs of Central Similarity Proximity Catch Digraph (CS-PCD) whose vertices are the data points in Xp in the multiple triangle case.

CS proximity regions are defined with respect to the Delaunay triangles based on Yp points with expansion parameter $t > 0$ and edge regions in each triangle is based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of

each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (unscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the number of arcs, loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) for more on CS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
NumArcsCSMT(Xp, Yp, t, M = c(1, 1, 1))
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle, default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

A list with the elements

`num.arcs`	Total number of arcs in all triangles
`num.in.conhull`	Number of `Xp` points in the convex hull of `Yp` points
`weight.vec`	The vector of the areas of Delaunay triangles based on `Yp` points

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

NumArcsCStri, NumArcsCSTe, NumArcsPEMT, and NumArcsASMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny),runif(ny))

oldpar <- par(mfrow = c(1,2))
plotDeltri(Xp,Yp,xlab="",ylab="")
par(oldpar)

M<-c(1,1,1)  #try also M<-c(1,2,3)

NumArcsCSMT(Xp,Yp,t=.5,M)
NumArcsCSMT(Xp,Yp,t=1.,M)
NumArcsCSMT(Xp,Yp,t=1.5,M)

NumArcsCSMT(c(.4,.2),Yp,t=.5,M)
NumArcsCSMT(c(.4,.2),Yp[1:3,],t=.5,M)

t<-2
NumArcsCSMT(Xp,Yp,t,M)
NumArcsCSMT(Xp,Yp[1:3,],t,M)

NumArcsCSMT(Xp,rbind(Yp,Yp),t,M)

dat.fr<-data.frame(a=Xp)
NumArcsCSMT(dat.fr,Yp,t,M)

dat.fr<-data.frame(a=Yp)
NumArcsCSMT(Xp,dat.fr,t,M)
```

NumArcsCSTe *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) - standard equilateral triangle case*

Description

Returns the number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) whose vertices are the given 2D numerical data set, dat.

CS proximity region NCS(x,r) is defined with respect to the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e. For the number of arcs, loops are not allowed so arcs are only possible for points inside T_e for this function.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
NumArcsCSTe(dat, t, M = c(1, 1, 1))
```

Arguments

dat	A set of 2D points which constitute the vertices of the digraph.
t	A positive real number which serves as the expansion parameter in CS proximity region.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates. which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1,1,1)$ i.e. the center of mass of T_e.

Value

Number of arcs for the CS-PCD with vertices being 2D data set, dat, in T_e with expansion parameter, $t > 0$, and center of mass CM. CS proximity regions are defined only for dat points inside T_e, i.e. arcs are possible for such points only.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

NumArcsCStri, NumArcsCSMT, and NumArcsPETe,

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
n<-10  #try also n<-20

set.seed(1)
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

NumArcsCSTe(dat,t=.5,M)
NumArcsCSTe(dat,t=3,M)
NumArcsCSTe(dat,t=1.5,M)

NumArcsCSTe(rbind(dat,c(0,1)),t=2,M)
NumArcsCSTe(c(.4,.2),t=.5,M)

NumArcsCSTe(dat,t=1.5,M);

NumArcsCSTe(rbind(dat,dat),t=1.5,M)

dat.fr<-data.frame(a=dat)
NumArcsCSTe(dat.fr,t=1.5,M);
```

`NumArcsCStri` *Number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) - one triangle case*

Description

Returns the number of arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) whose vertices are the given 2D numerical data set, `dat`.

CS proximity region NCS(x,r) is defined with respect to the triangle, `tri` with expansion parameter $t > 0$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`. For the number of arcs, loops are not allowed so arcs are only possible for points inside `tri` for this function.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
NumArcsCStri(Dt, tri, t, M = c(1, 1, 1))
```

Arguments

`Dt`	A set of 2D points which constitute the vertices of CS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

t	A positive real number which serves as the expansion parameter in CS proximity region.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.

Value

Number of arcs for the CS-PCD with vertices being 2D data set, dat, in tri with expansion parameter, $t > 0$, and center of mass CM. CS proximity regions are defined only for Dt points inside tri, i.e. arcs are possible for such points only.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

NumArcsCSTe, NumArcsCSMT, NumArcsPEtri, and NumArcsAStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

NumArcsCStri(dat,Tr,t=.5,M)
NumArcsCStri(dat,Tr,t=1,M)
NumArcsCStri(dat,Tr,t=1.5,M)

NumArcsCStri(c(1.4,.2),Tr,t=.5,M)

t<-.5
NumArcsCStri(dat,Tr,t,M)

dat.fr<-data.frame(a=dat)
NumArcsCStri(dat.fr,Tr,t,M)
```

```
dat.fr<-data.frame(a=Tr)
NumArcsCStri(dat,dat.fr,t,M)
```

`NumArcsPEend1D` *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) - end interval case*

Description

Returns the number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) whose vertices are a 1D numerical data set, `dat`, outside the interval $int = (a, b)$.

PE proximity region is constructed only with expansion parameter $r \geq 1$ for points outside the interval (a, b). End vertex regions are based on the end points of the interval, i.e., the corresponding vertex region is an interval as $(-\infty, a)$ or (b, ∞) for the interval (a, b). For the number of arcs, loops are not allowed, so arcs are only possible for points outside the interval, `int`, for this function.

See also (Ceyhan (2012)).

Usage

```
NumArcsPEend1D(dat, r, int)
```

Arguments

`dat`	A vector of 1D points which constitute the vertices of the digraph.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`int`	A vector of two real numbers representing an interval.

Value

Number of arcs for the PE-PCD with vertices being 1D data set, `dat`, expansion parameter, $r \geq 1$, for the end intervals.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

`NumArcsPEmid1D`, `NumArcsCSmid1D`, and `NumArcsCSend1D`

Examples

```
a<-0; b<-10; int<-c(a,b)

n<-5
datL<-runif(n,a-5,a)
datR<-runif(n,b,b+5)
dat<-c(datL,datR)

r<-2
NumArcsPEend1D(dat,r,int)

NumArcsPEend1D(dat,r=1.2,int)
NumArcsPEend1D(dat,r=4,int)

n<-10  #try also n<-20
dat2<-runif(n,a-5,b+5)
NumArcsPEend1D(dat2,r,int)

NumArcsPEend1D(dat,r,int)
```

NumArcsPEint *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one interval case*

Description

Returns the number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) whose vertices are the 1D data set dat in the one-interval case.

The data points could be inside or outside the interval is $int = (a, b)$. PE proximity region is constructed with an expansion parameter $r \geq 1$ and a centrality parameter c in $(0, 1)$.

The PE proximity region is constructed for both points inside and outside the interval, hence the arcs may exist for all points inside or outside the interval.

See also (Ceyhan (2012)).

Usage

```
NumArcsPEint(dat, r, c = 0.5, int)
```

Arguments

dat	A set of 1D points which constitute the vertices of PE-PCD.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$; default=0.5
int	A vector of two real numbers representing an interval.

Value

Number of arcs for the PE-PCD whose vertices are the 1D data set, dat, with expansion parameter, $r \geq 1$, and centrality parameter, c in $(0, 1)$.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

NumArcsPEmid1D, NumArcsPEend1D, and NumArcsCSint

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

n<-10
dat<-runif(n,a,b)
NumArcsPEint(dat,r,c,int)

NumArcsPEint(3,r,c,int)

NumArcsPEint(dat,r,c=.3,int)

NumArcsPEint(dat,r=1.5,c,int)

n<-10  #try also n<-20
dat<-runif(n,a,b)

NumArcsPEint(dat,r,c,int)

dat<-runif(n,a+10,b+10)
NumArcsPEint(dat,r,c,int)

n<-10
dat<-runif(n,a,b)
NumArcsPEint(dat,r,c,int)
```

NumArcsPEmid1D *Number of Arcs for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - middle interval case*

Description

Returns the number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) whose vertices are the given 1D numerical data set, dat. PE proximity region $NPE(x, r, c)$ is defined with respect to the interval $int = (a, b)$ for this function.

PE proximity region is constructed with expansion parameter $r \geq 1$ and centrality parameter c in $(0, 1)$.

Vertex regions are based on the center associated with the centrality parameter c in $(0, 1)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$ and for the number of arcs, loops are not allowed so arcs are only possible for points inside the middle interval int for this function.

See also (Ceyhan (2012)).

Usage

```
NumArcsPEmid1D(dat, r, c, int)
```

Arguments

dat	A set or vector of 1D points which constitute the vertices of PE-PCD.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
int	A vector of two real numbers representing an interval.

Value

Number of arcs for the PE-PCD whose vertices are the 1D data set, dat, with expansion parameter, $r \geq 1$, and centrality parameter, c in $(0, 1)$. PE proximity regions are defined only for dat points inside the interval int, i.e. arcs are possible for such points only.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

NumArcsPEend1D, NumArcsCSmid1D, and NumArcsCSend1D

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

n<-10
dat<-runif(n,a,b)
```

```
NumArcsPEmid1D(dat,r,c,int)

NumArcsPEmid1D(3,r,c,int)

NumArcsPEmid1D(dat,r,c=.3,int)

NumArcsPEmid1D(dat,r=1.5,c,int)

n<-10  #try also n<-20
dat<-runif(n,a-5,b+5)
NumArcsPEmid1D(dat,r,c,int)

dat<-runif(n,a+10,b+10)
NumArcsPEmid1D(dat,r,c,int)

n<-10
dat<-runif(n,a,b)
NumArcsPEmid1D(dat,r,c,int)
```

`NumArcsPEMT` *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) - multiple triangle case*

Description

Returns the number of arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) whose vertices are the data points in `Xp` in the multiple triangle case.

PE proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $r \geq 1$ and vertex regions in each triangle is based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (unscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). For the number of arcs, loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2006)) for more on PE-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
NumArcsPEMT(Xp, Yp, r, M = c(1, 1, 1))
```

Arguments

Xp	A set of 2D points which constitute the vertices of the PE-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this argument should be set as M="CC"), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

A list with the elements

num.arcs	Total number of arcs in all triangles
num.in.conhull	Number of Xp points in the convex hull of Yp points
weight.vec	The vector of the areas of Delaunay triangles based on Yp points

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

NumArcsPEtri, NumArcsPETe, NumArcsCSMT, and NumArcsASMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny),runif(ny))
```

```
oldpar <- par(mfrow = c(1,2))
plotDeltri(Xp,Yp,xlab="",ylab="")
par(oldpar)

M<-c(1,1,1)  #try also M<-c(1,2,3)

NumArcsPEMT(Xp,Yp,r=1.25,M)
NumArcsPEMT(Xp,Yp,r=1.5,M)
NumArcsPEMT(Xp,Yp,r=2,M)

NumArcsPEMT(c(.4,.2),Yp,r=1.25)

r<-2
NumArcsPEMT(Xp,Yp,r)
NumArcsPEMT(Xp,Yp[1:3,],r)

NumArcsPEMT(Xp,rbind(Yp,Yp),r)

dat.fr<-data.frame(a=Xp)
NumArcsPEMT(dat.fr,Yp,r)

dat.fr<-data.frame(a=Yp)
NumArcsPEMT(Xp,dat.fr,r)
```

`NumArcsPETe` *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) - standard equilateral triangle case*

Description

Returns the number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) whose vertices are the given 2D numerical data set, `dat`.

PE proximity region $NPE(x, r)$ is defined with respect to the standard equilateral triangle $T_e = T(v = 1, v = 2, v = 3) = T((0, 0), (1, 0), (1/2, \sqrt{3}/2))$ with expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e. For the number of arcs, loops are not allowed so arcs are only possible for points inside T_e for this function.

See also (Ceyhan et al. (2006)).

Usage

```
NumArcsPETe(dat, r, M = c(1, 1, 1))
```

Arguments

dat	A set of 2D points which constitute the vertices of the PE-PCD.
r	A positive real number which serves as the expansion parameter for PE proximity region; must be ≥ 1.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e; default is $M = (1, 1, 1)$ i.e. the center of mass of T_e.

Value

Number of arcs for the PE-PCD with vertices being 2D data set, dat, in T_e with expansion parameter, $r \geq 1$, and M-vertex regions. PE proximity regions are defined only for dat points inside T_e, i.e. arcs are possible for such points only.

References

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

NumArcsPEtri, NumArcsPEMT, and NumArcsCSTe

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
n<-10  #try also n<-20

set.seed(1)
dat<-runifTe(n)$gen.points

M<-c(.6,.2)  #try also M<-c(1,1,1)

NumArcsPETe(dat,r=1.25,M)
NumArcsPETe(dat,r=1.5,M)
NumArcsPETe(dat,r=2,M)

NumArcsPETe(rbind(dat,c(0,1)),r=2,M)
NumArcsPETe(c(.2,.3),r=2,M)

NumArcsPETe(dat,r=1.5,M);

dat.fr<-data.frame(a=dat)
NumArcsPETe(dat.fr,r=1.5,M);
```

NumArcsPEtri *Number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns the number of arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) whose vertices are the given 2D numerical data set, dat.

PE proximity region $NPE(x, r)$ is defined with respect to the triangle, tri for this function. PE proximity region is constructed with expansion parameter $r \geq 1$ and vertex regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle tri or based on circumcenter of tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri. For the number of arcs, loops are not allowed so arcs are only possible for points inside the triangle tri for this function.

See also (Ceyhan (2005); Ceyhan et al. (2006)).

Usage

```
NumArcsPEtri(dat, tri, r, M = c(1, 1, 1))
```

Arguments

dat	A set of 2D points which constitute the vertices of PE-PCD.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.

Value

Number of arcs of the PE-PCD with vertices being 2D data set, dat, in tri with expansion parameter, $r \geq 1$, and center M. PE proximity regions are defined only for dat points inside tri, i.e. arcs are possible for such points only.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

NumArcsPETe, NumArcsPEMT, NumArcsCStri, and NumArcsAStri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

NumArcsPEtri(dat,Tr,r=1.25,M)
NumArcsPEtri(dat,Tr,r=1.5,M)
NumArcsPEtri(dat,Tr,r=2.0,M)

NumArcsPEtri(rbind(dat,c(0,2)),Tr,r=1.25,M)

r<-2
NumArcsPEtri(dat,Tr,r,M)

dat.fr<-data.frame(a=dat)
NumArcsPEtri(dat.fr,Tr,r,M)

dat.fr<-data.frame(a=Tr)
NumArcsPEtri(dat,dat.fr,r,M)
```

NumDelTri *Number of Delaunay triangles based on a 2D data set*

Description

Returns the number of Delaunay triangles based on the 2D set of points Yp . See (Okabe et al. (2000); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
NumDelTri(Yp)
```

Arguments

Yp	A set of 2D points which constitute the vertices of Delaunay triangles.

Value

Number of Delaunay triangles based on Yp points.

References

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

plotDeltri

Examples

```
ny<-10

set.seed(1)
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

NumDelTri(Yp)
NumDelTri(Yp[1:3,])

dat.fr<-data.frame(a=Yp)
NumDelTri(dat.fr)
```

---

paraline *The line paralel to the line segment joining two distinct 2D points* a *and* b

---

Description

An object of class "Lines". Returns the `equation`, `slope`, `intercept`, and y-coordinates of the line crossing the point `p` and parallel to the line passing through the points `a` and `b` with x-coordinates are provided in vector `x`.

Usage

```
paraline(p, a, b, x)
```

Arguments

`p`	A 2D point at which the parallel line to line segment joining `a` and `b` crosses.
`a, b`	2D points that determine the line segment (the line will be parallel to this line segment).
`x`	A scalar or a vector of scalars representing the x-coordinates of the line parallel to ab and crossing `p`.

Value

A list with the elements

desc	Description of the line passing through point p and parallel to line segment joining a and b
mtitle	The "main" title for the plot of the line passing through point p and parallel to line segment joining a and b
points	The input points p, a, and b (stacked row-wise, i.e., point p is in row 1, point a is in row 2 and point b is in row 3). Line parallel to ab crosses p.
x	The input vector. It can be a scalar or a vector of scalars, which constitute the x-coordinates of the point(s) of interest on the line passing through point p and parallel to line segment joining a and b.
y	The output scalar or vector which constitutes the y-coordinates of the point(s) of interest on the line passing through point p and parallel to line segment joining a and b. If x is a scalar, then y will be a scalar and if x is a vector of scalars, then y will be a vector of scalars.
slope	Slope of the line, Inf is allowed, passing through point p and parallel to line segment joining a and b
intercept	Intercept of the line passing through point p and parallel to line segment joining a and b
equation	Equation of the line passing through point p and parallel to line segment joining a and b

See Also

slope, Line, and perpline, line in the generic stats package, paraline3D

Examples

```
A<-c(1.1,1.2); B<-c(2.3,3.4); p<-c(.51,2.5)

paraline(p,A,B,.45)
paraline(A,A,B,.45)

pts<-rbind(A,B,p)
xr<-range(pts)
xf<-(xr[2]-xr[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=20)  #try also l=100

plnAB<-paraline(p,A,B,x)
plnAB
summary(plnAB)
plot(plnAB)

y<-plnAB$y
Xlim<-range(x,pts[,1])
if (!is.na(y[1])) {Ylim<-range(y,pts[,2])} else {Ylim<-range(pts[,2])}
xd<-Xlim[2]-Xlim[1]
```

```
yd<-Ylim[2]-Ylim[1]
pf<-c(xd,-yd)*.025

plot(A,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(pts)
txt.str<-c("A","B","p")
text(pts+rbind(pf,pf,pf),txt.str)

segments(A[1],A[2],B[1],B[2],lty=2)
if (!is.na(y[1])) {lines(x,y,type="l",lty=1,xlim=Xlim,ylim=Ylim)} else {abline(v=p[1])}
tx<-(A[1]+B[1])/2;
if (!is.na(y[1])) {ty<-paraline(p,A,B,tx)$y} else {ty=p[2]}
text(tx,ty,"line parallel to AB\n and crossing p")
```

`paraline3D` *The line crossing the 3D point* `P` *and parallel to line joining 3D points* `A` *and* `B`

Description

An object of class "Lines3D". Returns the equation, x-, y-, and z-coordinates of the line crossing 3D point `P` and parallel to the line joining 3D points `A` and `B` (i.e., the line is in the direction of vector `B-A`) with the parameter `t` being provided in vector `t`.

Usage

```
paraline3D(P, A, B, t)
```

Arguments

`P`	A 3D point through which the straight line passes.
`A, B`	3D points which determine the straight line to which the line passing through point `P` would be parallel (i.e. B-A determines the direction of the straight line passing through `P`).
`t`	A scalar or a vector of scalars representing the parameter of the coordinates of the line (for the form: $x = x_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $P = (x_0, y_0, z_0)$ and $B - A = (a, b, c)$).

Value

A list with the elements

`desc`	A description of the line
`mtitle`	The "main" title for the plot of the line
`points`	The input points that determine the line to which the line crossing point `P` would be parallel.

pnames	The names of the input points that determine the line to which the line crossing point P would be parallel.
vecs	The points P, A, and B stacked row-wise in this order.
vec.names	The names of the points P, A, and B.
x,y,z	The x-, y- and z-coordinates of the point(s) of interest on the line parallel to the line determined by points A and B.
tsq	The scalar or the vector of the parameter in defining each coordinate of the line for the form: $x = x_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $P = (x_0, y_0, z_0)$ and $B - A = (a, b, c)$.
equation	Equation of the line passing through point P and parallel to the line joining points A and B (i.e., in the direction of the vector B-A). The line equation is in the form: $x = x_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $P = (x_0, y_0, z_0)$ and $B - A = (a, b, c)$.

See Also

Line3D, perp.ln2pl, and paraline

Examples

```
P<-c(1,10,4); A<-c(1,1,3); B<-c(3,9,12)

vecs<-rbind(P,B-A)
pts<-rbind(P,A,B)
paraline3D(P,A,B,.1)

tr<-range(pts,vecs);
tf<-(tr[2]-tr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=20)  #try also l=100

pln3D<-paraline3D(P,A,B,tsq)
pln3D
summary(pln3D)
plot(pln3D)

paraline3D(P,A,B,c(.1,.2))

x<-pln3D$x
y<-pln3D$y
z<-pln3D$z

zr<-range(z)
zf<-(zr[2]-zr[1])*.2
Av<-(B-A)*tf*5

Xlim<-range(x,pts[,1])
Ylim<-range(y,pts[,2])
Zlim<-range(z,pts[,3])
```

```
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

Dr<-P+min(tsq)*(B-A)

plot3D::lines3D(x, y, z, phi = 0, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),zlim=Zlim+zd*c(-.1,.1)+c(-zf,zf),
        pch = 20, cex = 2, ticktype = "detailed")
plot3D::arrows3D(Dr[1],Dr[2],Dr[3]+zf,Dr[1]+Av[1],Dr[2]+Av[2],Dr[3]+zf+Av[3], add=TRUE)
plot3D::points3D(pts[,1],pts[,2],pts[,3],add=TRUE)
plot3D::text3D(pts[,1],pts[,2],pts[,3],labels=c("P","A","B"),add=TRUE)
plot3D::arrows3D(P[1],P[2],P[3]-2*zf,P[1],P[2],P[3],lty=2, add=TRUE)
plot3D::text3D(P[1],P[2],P[3]-2*zf,labels="initial point",add=TRUE)
plot3D::arrows3D(Dr[1]+Av[1]/2,Dr[2]+Av[2]/2,Dr[3]+3*zf+Av[3]/2,Dr[1]+Av[1]/2,
Dr[2]+Av[2]/2,Dr[3]+zf+Av[3]/2,lty=2, add=TRUE)
plot3D::text3D(Dr[1]+Av[1]/2,Dr[2]+Av[2]/2,Dr[3]+3*zf+Av[3]/2,labels="direction vector",add=TRUE)
plot3D::text3D(Dr[1]+Av[1]/2,Dr[2]+Av[2]/2,Dr[3]+zf+Av[3]/2,labels="B-A",add=TRUE)
```

`paraplane` *The plane paralel to the plane spanned by three distinct 3D points* `a`*,* `b`*, and* `c`

Description

An object of class "Planes". Returns the equation and z-coordinates of the plane passing through point `p` and parallel to the plane spanned by three distinct 3D points `a`, `b`, and `c` with x- and y-coordinates are provided in vectors `x` and `y`, respectively.

Usage

```
paraplane(p, a, b, c, x, y)
```

Arguments

`p`	A 3D point which the plane parallel to the plane spanned by three distinct 3D points `a`, `b`, and `c` crosses.
`a, b, c`	3D points that determine the plane to which the plane crossing point `p` is parallel to.
`x, y`	A scalar or a vector of scalars representing the x- and y-coordinates of the plane parallel to the plane spanned by points `a`, `b`, and `c` and passing through point `p`.

Value

A list with the elements

desc	Description of the plane passing through point p and parallel to plane spanned by points a, b and c
points	The input points a, b, c, and p. Plane is parallel to the plane spanned by a, b, and c and passes through point p (stacked row-wise, i.e., row 1 is point a, row 2 is point b, row 3 is point c, and row 4 is point p).
x,y	The input vectors which constitutes the x- and y-coordinates of the point(s) of interest on the plane. x and y can be scalars or vectors of scalars.
z	The output vector which constitutes the z-coordinates of the point(s) of interest on the plane. If x and y are scalars, z will be a scalar and if x and y are vectors of scalars, then z needs to be matrix of scalars, containing the z-coordinate for each pair of x and y values.
coeff	Coefficients of the Plane (in the $z = Ax + By + C$ form).
equation	Equation of the plane in long form
equation2	Equation of the plane in short form, to be inserted on the plot

See Also

Plane

Examples

```
A<-c(1,10,3); B<-c(1,1,3); C<-c(3,9,12); P<-c(1,1,0)

Plane(A,B,C,.1,.2)

pts<-rbind(A,B,C,P)
paraplane(P,A,B,C,.1,.2)
paraplane(P,A,B,C,0,0)

xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.25 #how far to go at the lower and upper ends in the y-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=20)  #try also l=100
y<-seq(yr[1]-yf,yr[2]+yf,l=20)  #try also l=100

plP2ABC<-paraplane(P,A,B,C,x,y)
plP2ABC
summary(plP2ABC)
plot(plP2ABC)

paraplane(P,A,B,A+B,.1,.2)

z.grid<-plP2ABC$z

plABC<-Plane(A,B,C,x,y)
plABC
pl.grid<-plABC$z

zr<-max(z.grid)-min(z.grid)
```

```
Pts<-rbind(A,B,C,P)+rbind(c(0,0,zr*.1),c(0,0,zr*.1),c(0,0,zr*.1),c(0,0,zr*.1))
Mn.pts<-apply(Pts[1:3,],2,mean)

plot3D::persp3D(z = pl.grid, x = x, y = y, theta =225, phi = 30, ticktype = "detailed")
#plane spanned by points A, B, C
plot3D::persp3D(z = z.grid, x = x, y = y,add=TRUE)
#plane parallel to the original plane and passing thru point \code{P}

plot3D::persp3D(z = z.grid, x = x, y = y, theta =225, phi = 30, ticktype = "detailed")
#plane spanned by points A, B, C
#add the defining points
plot3D::points3D(Pts[,1],Pts[,2],Pts[,3], add=TRUE)
plot3D::text3D(Pts[,1],Pts[,2],Pts[,3], c("A","B","C","P"),add=TRUE)
plot3D::text3D(Mn.pts[1],Mn.pts[2],Mn.pts[3],plP2ABC$equation,add=TRUE)
plot3D::polygon3D(Pts[1:3,1],Pts[1:3,2],Pts[1:3,3], add=TRUE)

P<-c(1,1,1)
paraplane(P,A,B,C,.1,.2)
```

---

pcds *pcds: A package for Proximity Catch Digraphs and Their Applications*

Description

pcds is a package for generation, computation and visualization of proximity catch digraphs and tests based on them.

Details

The pcds package contains the functions for generating patterns of segregation, association, CSR (complete spatial randomness) and Uniform data in one, two and three dimensional cases, for testing these patterns based on two invariants of various families of the proximity catch digraphs (PCDs), (see (Ceyhan (2005)).

The graph invariants used in testing spatial point data are the domination number (Ceyhan (2011)) and arc density (Ceyhan et al. (2006); Ceyhan et al. (2007)) of for two-dimensional data for visualization of PCDs for one, two and three dimensional data. The PCD families considered are Arc-Slice PCDs, Proportional-Edge PCDs and Central Similarity PCDs.

The package also contains visualization tools for these digraphs for 1D-3D vertices. The AS-PCD related tools are provided for 1D and 2D data; PE-PCD related tools are provided for 1D-3D data, and CS-PCD tools are provided for 1D and 2D data.

The pcds functions

The pcds functions can be grouped as Auxiliary Functions, AS-PCD Functions, PE-PCD Functions, and CS-PCD Functions.

Auxiliary Functions

Contains the auxiliary functions used in PCD calculations, such as equation of lines for two points, distances between lines and points, generation of points from uniform, segregation and association patterns, checking points inside the triangle etc. In all these functions points are vectors, and data sets are either matrices or data frames.

Arc-Slice PCD Functions

Contains the functions used in AS-PCD calculations, such as generation of data in a given a triangle and estimation of gamma, arc density, etc.

Proportional-Edge PCD Functions

Contains the functions used in PE-PCD calculations, such as generation of data in a given interval, triangle and tetrahedron and estimation of gamma, arc density, etc.

Central-Similarity PCD Functions

Contains the functions used in CS-PCD calculations, such as generation of data in a given interval and triangle and estimation of gamma, arc density, etc.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

PEarcdens.tri — *Arc density of Proportional Edge Proximity Catch Digraphs (PE-PCDs) - one triangle case*

Description

Returns the arc density of PE-PCD whose vertex set is the given 2D numerical data set, `Xp`, (some of its members are) in the triangle `tri`.

PE proximity regions is defined with respect to `tri` with expansion parameter $r \geq 1$ and vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in

barycentric coordinates in the interior of the triangle `tri` or based on circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`. The function also provides arc density standardized by the mean and asymptotic variance of the arc density of PE-PCD for uniform data in the triangle `tri`. For the number of arcs, loops are not allowed.

`tri.cor` is a logical argument for triangle correction (default is `TRUE`), if `TRUE`, only the points inside the triangle are considered (i.e., digraph induced by these vertices are considered) in computing the arc density, otherwise all points are considered (for the number of vertices in the denominator of arc density).

Caveat: The standardized arc density is only correct when `M` is the center of mass in the current version.

See also (Ceyhan (2005); Ceyhan et al. (2006)).

Usage

```
PEarcdens.tri(Xp, tri, r, M = c(1, 1, 1), tri.cor = TRUE)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.
`tri.cor`	A logical argument for computing the arc density for only the points inside the triangle, `tri`. (default=`TRUE`), i.e., if `TRUE` only the induced digraph with the vertices inside `tri` are considered in the computation of arc density.

Value

A list with the elements

`arc.dens`	Arc density of PE-PCD whose vertices are the 2D numerical data set, `Xp`; PE proximity regions are defined with respect to the triangle `tri` and `M`-vertex regions
`std.arc.dens`	Arc density standardized by the mean and asymptotic variance of the arc density of PE-PCD for uniform data in the triangle `tri`.
`caveat`	The warning as "The standardized arc density is only correct when `M` is the center of mass in the current version".

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

ASarcdens.tri, CSarcdens.tri, and NumArcsPEtri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

NumArcsPEtri(dat,Tr,r=1.5,M)
PEarcdens.tri(dat,Tr,r=1.5,M)
PEarcdens.tri(dat,Tr,r=1.5,M,tri.cor = FALSE)

NumArcsPEtri(dat,Tr,r=1,M)
PEarcdens.tri(dat,Tr,r=1,M)

NumArcsPEtri(dat,Tr,r=1.5,M)
PEarcdens.tri(dat,Tr,r=1.5,M)

r<-2
PEarcdens.tri(dat,Tr,r,M)

dat.fr<-data.frame(a=dat)
PEarcdens.tri(dat.fr,Tr,r,M)

dat.fr<-data.frame(a=Tr)
PEarcdens.tri(dat,dat.fr,r,M)
```

PEdom.tetra — *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) - one tetrahedron case*

Description

Returns the domination number of PE-PCD whose vertices are the data points in Xp.

PE proximity region is defined with respect to the tetrahedron th with expansion parameter $r \geq 1$ and vertex regions are based on the center M which is circumcenter ("CC") or center of mass ("CM") of th with default="CM".

See also (Ceyhan (2005, 2010)).

Usage

```
PEdom.tetra(Xp, th, r, M = "CM")
```

Arguments

Xp	A set of 3D points which constitute the vertices of the digraph.
th	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	The center to be used in the construction of the vertex regions in the tetrahedron, th. Currently it only takes "CC" for circumcenter and "CM" for center of mass; default="CM".

Value

A list with two elements

dom.num	Domination number of PE-PCD with vertex set=Xp and expansion parameter $r \geq 1$ and center M
mds	A minimum dominating set of PE-PCD with vertex set=Xp and expansion parameter $r \geq 1$ and center M

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

PEdomtri

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-10  #try also n<-20

dat<-runif.tetra(n,tetra)$g

M<-"CM"  #try also M<-"CC"
r<-1.25
```

```
PEdom.tetra(dat,tetra,r,M)

PEdom.tetra(rbind(dat,c(5,5,5)),tetra,r,M)

P1<-c(.5,.5,.5)
PEdom.tetra(P1,tetra,r,M)
```

PEdom1D *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data*

Description

Returns the domination number of PE-PCD whose vertices are the 1D data set Xp.

Yp determines the end points of the intervals (i.e., partition the real line via intervalization).

PE proximity region is constructed with expansion parameter $r \geq 1$ and centrality parameter c in $(0, 1)$.

Usage

```
PEdom1D(Xp, Yp, r, c = 0.5)
```

Arguments

Xp	A set of 1D points which constitute the vertices of the PE-PCD.
Yp	A set of 1D points which constitute the end points of the intervals which partition the real line.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside int (default c=.5).

Value

A list with two elements

dom.num	Domination number of PE-PCD with vertex set=Xp and expansion parameter $r \geq 1$ and centrality parameter c in $(0, 1)$
mds	A minimum dominating set of PE-PCD with vertex set=Xp and expansion parameter $r \geq 1$ and centrality parameter c in $(0, 1)$

See Also

PEdomMTnd

Examples

```
a<-0; b<-10; int<-c(a,b)
c<-.4
r<-2

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

plotIntervals(Xp,Yp,main="Xp Points and Intervals Based on Yp Points")
plotPEregsMI(Xp,Yp,r,c,main="PE Proximity Regions for Xp points - Intervalization is by Yp Points")

PEdom1D(Xp,Yp,r,c)

PEdom1D(Xp,Yp,r,c=.25)
PEdom1D(Xp,Yp,r,c=.01)
PEdom1D(Xp,Yp,r=1.25,c)
```

PEdomMT *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) - multiple triangle case*

Description

Returns the domination number and a minimum dominating set of PE-PCD whose vertices are the data points in Xp in the multiple triangle case and the Delauany triangles based on Yp points.

PE proximity regions are defined with respect to the Delaunay triangles based on Yp points with expansion parameter $r \geq 1$ and vertex regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the inter#' The dominationior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (unscaled) basic triangle so that M will be the same type of center for each Delaunay triangle (this conversion is not necessary when M is CM).

Convex hull of Yp is partitioned by the Delaunay triangles based on Yp points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of Yp points). Loops are allowed for the domination number.

See (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)) for more on the domination number of PE-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
PEdomMT(Xp, Yp, r, M = c(1, 1, 1))
```

Arguments

Xp	A set of 2D points which constitute the vertices of the PE-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this argument should be set as M="CC"), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

Value

A list with two elements

dom.num	Domination number of the PE-PCD whose vertices are Xp points. PE proximity regions are constructed with respect to the Delaunay triangles based on the Yp points with expansion parameter $r \geq 1$.
mds	A minimum dominating set of the PE-PCD whose vertices are Xp points

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

PEdomtri, PEdom.tetra, dom.exact, and dom.greedy

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

r<-1.5  #try also r<-2

PEdomMT(Xp,Yp,r,M)

PEdomMT(Xp,Yp,r=1.4,M)
PEdomMT(Xp,Yp,r=2,M)

r<-1.5  #try also r<-2

PEdomMT(Xp,Yp,r,M)  #this may be different due to random selection of the center for r in (1,1.5)

PEdomMT(Xp,Yp,r,M)
PEdomMT(Xp,Yp[1:3,],r,M)

PEdomMT(Xp,rbind(Yp,Yp),r,M)

dat.fr<-data.frame(a=Xp)
PEdomMT(dat.fr,Yp,r,M)

dat.fr<-data.frame(a=Yp)
PEdomMT(Xp,dat.fr,r,M)
```

PEdomMTnd *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) with non-degeneracy centers - multiple triangle case*

Description

Returns the domination number and a minimum dominating set of PE-PCD whose vertices are the data points in Xp in the multiple triangle case and the Delauany triangles based on Yp points.

PE proximity regions are defined with respect to the Delaunay triangles based on Yp points with expansion parameter $r \geq 1$ and vertex regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center M where M is one of the 3 centers that renders the asymptotic distribution of domination number to be non-degenerate for a given value of r in $(1, 1.5)$ and M is center of mass for $r = 1.5$.

Convex hull of Yp is partitioned by the Delaunay triangles based on Yp points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of Yp points). Loops are allowed for the domination number.

See (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)) more on the domination number of PE-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
PEdomMTnd(Xp, Yp, r)
```

Arguments

Xp	A set of 2D points which constitute the vertices of the PE-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.

Value

A list with two elements

dom.num	Domination number of the PE-PCD whose vertices are Xp points. PE proximity regions are constructed with respect to the Delaunay triangles based on the Yp points with expansion parameter $r \geq 1$.
mds	A minimum dominating set of the PE-PCD whose vertices are Xp points

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

PEdomtri, PEdom.tetra, dom.exact, and dom.greedy

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

r<-1.5  #try also r<-2

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

PEdomMTnd(Xp,Yp,r)

PEdomMTnd(Xp,Yp,r=1.4)

r<-1.5  #try also  #r<-2
PEdomMTnd(Xp,Yp,r) #this may be different due to random selection of the center for r in (1,1.5)

PEdomMTnd(Xp,Yp[1:3,],r)

PEdomMTnd(Xp,rbind(Yp,Yp),r)

dat.fr<-data.frame(a=Xp)
PEdomMTnd(dat.fr,Yp,r)

dat.fr<-data.frame(a=Yp)
PEdomMTnd(Xp,dat.fr,r)
```

PEdomtri — *The domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) - one triangle case*

Description

Returns the domination number of PE-PCD whose vertices are the data points in Xp.

PE proximity region is defined with respect to the triangle tri with expansion parameter $r \geq 1$ and vertex regions are constructed with center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle tri or the circumcenter of tri.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011, 2012)).

Usage

```
PEdomtri(Xp, tri, r, M = c(1, 1, 1))
```

Arguments

Xp	A set of 2D points which constitute the vertices of the digraph.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri, default is (1,1,1) i.e. the center of mass.

Value

A list with two elements

dom.num	Domination number of PE-PCD with vertex set=Xp and expansion parameter $r \geq 1$ and center M
mds	A minimum dominating set of PE-PCD with vertex set=Xp and expansion parameter $r \geq 1$ and center M

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

PEdomMTnd, PEdomMT and PEdom1D

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2)
Tr<-rbind(A,B,C)
n<-10  #try also n<-20
```

```
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1,1,1)

r<-1.4

PEdomtri(dat,Tr,r,M)
IM<-IncMatPEtri(dat,Tr,r,M)
dom.greedy(IM)
dom.exact(IM)

gr.gam<-dom.greedy(IM)
gr.gam
dat[gr.gam$i,]

PEdomtri(rbind(dat,c(5,5)),Tr,r,M)

PEdomtri(dat,Tr,r,M=c(.4,.4))

PEdomtri(rbind(dat,c(5,5)),Tr,r,M=c(.4,.4))

P1<-c(.5,.5)
PEdomtri(P1,Tr,r,M)
```

`perp.ln2pl` *The line crossing the 3D point* `P` *and perpendicular to the plane spanned by 3D points* `A`, `B`, *and* `C`

Description

An object of class "Lines3D". Returns the equation, x-, y-, and z-coordinates of the line crossing 3D point `P` and perpendicular to the plane spanned by 3D points `A`, `B`, and `C` (i.e., the line is in the direction of normal vector of this plane) with the parameter `t` being provided in vector `t`.

Usage

```
perp.ln2pl(P, A, B, C, t)
```

Arguments

`P`	A 3D point through which the straight line passes.
`A, B, C`	3D points which determine the plane to which the line passing through point `P` would be perpendicular (i.e. the normal vector of this plane determines the direction of the straight line passing through `P`).
`t`	A scalar or a vector of scalars representing the parameter of the coordinates of the line (for the form: $x = x_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $P = (x_0, y_0, z_0)$ and normal vector=(a,b,c)).

Value

A list with the elements

desc	A description of the line
mtitle	The "main" title for the plot of the line
points	The input points that determine the line and plane, line crosses point P and plane is determined by 3D points A, B, and C.
pnames	The names of the input points that determine the line and plane; line would be perpendicular to the plane.
vecs	The point P and normal vector.
vec.names	The names of the point P and the second entry is "normal vector".
x,y,z	The x-, y- and z-coordinates of the point(s) of interest on the line perpendicular to the plane determined by points A, B, and C.
tsq	The scalar or the vector of the parameter in defining each coordinate of the line for the form: $x = x_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $P = (x_0, y_0, z_0)$ and normal vector=(a,b,c).
equation	Equation of the line passing through point P and perpendicular to the plane determined by points A, B, and C (i.e., line is in the direction of the normal vector N of the plane). The line equation is in the form: $x = x_0 + at$, $y = y_0 + bt$, and $z = z_0 + ct$ where $P = (x_0, y_0, z_0)$ and normal vector=(a,b,c).

See Also

Line3D, paraline3D and perpline

Examples

```
P<-c(1,1,1); A<-c(1,10,4); B<-c(1,1,3); C<-c(3,9,12)

cf<-as.numeric(Plane(A,B,C,1,1)$coeff)
a<-cf[1]; b<-cf[2]; c<- -1;

vecs<-rbind(A,c(a,b,c))
pts<-rbind(P,A,B,C)
perp.ln2pl(P,A,B,C,.1)

tr<-range(pts,vecs);
tf<-(tr[2]-tr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=20)  #try also l=100

pln2pl<-perp.ln2pl(P,A,B,C,tsq)
pln2pl
summary(pln2pl)
plot(pln2pl)

perp.ln2pl(P,A,B,C,c(.1,.2))
```

```
xc<-pln2pl$x
yc<-pln2pl$y
zc<-pln2pl$z

zr<-range(zc)
zf<-(zr[2]-zr[1])*.2
Bv<- -c(a,b,c)*zf*5

Dr<-(A+B+C)/3

pts2<-rbind(A,B,C)
xr<-range(pts2[,1],xc); yr<-range(pts2[,2],yc)
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1 #how far to go at the lower and upper ends in the y-coordinate
xs<-seq(xr[1]-xf,xr[2]+xf,l=20)  #try also l=100
ys<-seq(yr[1]-yf,yr[2]+yf,l=20)  #try also l=100

plABC<-Plane(A,B,C,xs,ys)
z.grid<-plABC$z

Xlim<-range(xc,xs,pts[,1])
Ylim<-range(yc,ys,pts[,2])
Zlim<-range(zc,z.grid,pts[,3])

xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::persp3D(z = z.grid, x = xs, y = ys, theta =225, phi = 30,
col="lightblue", ticktype = "detailed",
        xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),zlim=Zlim+zd*c(-.05,.05))
        #plane spanned by points A, B, C
plot3D::lines3D(xc, yc, zc, bty = "g",pch = 20, cex = 2,col="red", ticktype = "detailed",add=TRUE)
plot3D::arrows3D(Dr[1],Dr[2],Dr[3],Dr[1]+Bv[1],Dr[2]+Bv[2],Dr[3]+Bv[3], add=TRUE)
plot3D::points3D(pts[,1],pts[,2],pts[,3],add=TRUE)
plot3D::text3D(pts[,1],pts[,2],pts[,3],labels=c("P","A","B","C"),add=TRUE)
plot3D::arrows3D(P[1],P[2],P[3]-zf,P[1],P[2],P[3],lty=2, add=TRUE)
plot3D::text3D(P[1],P[2],P[3]-zf,labels="initial point",add=TRUE)
plot3D::text3D(P[1],P[2],P[3]+zf/2,labels="P",add=TRUE)
plot3D::arrows3D(Dr[1],Dr[2],Dr[3],Dr[1]+Bv[1]/2,Dr[2]+Bv[2]/2,Dr[3]+Bv[3]/2,lty=2, add=TRUE)
plot3D::text3D(Dr[1]+Bv[1]/2,Dr[2]+Bv[2]/2,Dr[3]+Bv[3]/2,labels="normal vector",add=TRUE)
```

`perpline` *The line passing thru a point and perpendicular to the line segment joining two points*

Description

An object of class "Lines". Returns the `equation`,`slope`,`intercept`, and y-coordinates of the line crossing the point `p` and perpendicular to the line passing through the points `a` and `b` with

x-coordinates are provided in vector x.

Usage

```
perpline(p, a, b, x)
```

Arguments

p	A 2D point at which the perpendicular line to line segment joining a and b crosses.
a, b	2D points that determine the line segment (the line will be perpendicular to this line segment).
x	A scalar or a vector of scalars representing the x-coordinates of the line perpendicular to line joing a and b and crossing p.

Value

A list with the elements

desc	Description of the line passing through point p and perpendicular to line joining a and b
mtitle	The "main" title for the plot of the line passing through point p and perpendicular to line joining a and b
points	The input points a and b (stacked row-wise, i.e., row 1 is point a and row 2 is point b). Line passing through point p is perpendicular to line joining a and b
x	The input vector, can be a scalar or a vector of scalars, which constitute the x-coordinates of the point(s) of interest on the line passing through point p and perpendicular to line joining a and b
y	The output vector which constitutes the y-coordinates of the point(s) of interest on the line passing through point p and perpendicular to line joining a and b. If x is a scalar, then y will be a scalar and if x is a vector of scalars, then y will be a vector of scalars.
slope	Slope of the line passing through point p and perpendicular to line joining a and b
intercept	Intercept of the line passing through point p and perpendicular to line joining a and b
equation	Equation of the line passing through point p and perpendicular to line joining a and b

See Also

slope, Line, and paraline

Examples

```
A<-c(1.1,1.2); B<-c(2.3,3.4); p<-c(.51,2.5)

perpline(p,A,B,.45)
perpline(A,A,B,.45)

pts<-rbind(A,B,p)
xr<-range(pts)
xf<-(xr[2]-xr[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=20)  #try also l=100

plnAB<-perpline(p,A,B,x)
plnAB
summary(plnAB)
plot(plnAB,asp=1)

y<-plnAB$y
Xlim<-range(x,pts[,1])
if (!is.na(y[1])) {Ylim<-range(y,pts[,2])} else {Ylim<-range(pts[,2])}
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
pf<-c(xd,-yd)*.025

plot(A,asp=1,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(pts)
txt.str<-c("A","B","p")
text(pts+rbind(pf,pf,pf),txt.str)

segments(A[1],A[2],B[1],B[2],lty=2)
if (!is.na(y[1])) {lines(x,y,type="l",lty=1,xlim=Xlim,ylim=Ylim)} else {abline(v=p[1])}
tx<-p[1]+abs(xr-p[1])/2;
if (!is.na(y[1])) {ty<-perpline(p,A,B,tx)$y} else {ty=p[2]}
text(tx,ty,"line perpendicular to AB\n and crossing p")
```

`PG2PE1D.asy` *The asymptotic probability of domination number=2 for Proportional Edge Proximity Catch Digraphs (PE-PCDs) - middle interval case*

Description

Returns the asymptotic P(domination number=2) for PE-PCD whose vertices are a uniform data set in a finite interval (a, b).

The PE proximity region $NPE(x, r, c)$ is defined with respect to (a,b) with centrality parameter c in $(0, 1)$ and expansion parameter $r = 1/\max(c, 1 - c)$.

Usage

```
PG2PE1D.asy(c)
```

Arguments

c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.

Value

The asymptotic P(domination number=2) for PE-PCD whose vertices are a uniform data set in a finite interval (a, b)

See Also

PG2PE1D and PG2PEtri

Examples

```
c<-.5

PG2PE1D.asy(c)

PG2PE1D.asy(c=1/1.5)
PG2PE1D(r=1.5,c=1/1.5,n=10)
PG2PE1D(r=1.5,c=1/1.5,n=100)
```

PG2PEtri

Asymptotic probability that domination number of Proportional Edge Proximity Catch Digraphs (PE-PCDs) equals 2 where vertices of the digraph are uniform points in a triangle

---

Description

Returns P(domination number=2) for PE-PCD for uniform data in a triangle, when the sample size n goes to infinity (i.e., asymptotic probability of domination number=2).

PE proximity regions are contructed with respect to the triangle with the expansion parameter $r \geq 1$ and M-vertex regions where M is the vertex that renders the asymptotic distribution of the domination number non-degenerate for the given value of r in $(1, 1.5]$.

See also (Ceyhan (2005); Ceyhan and Priebe (2007); Ceyhan (2011)).

Usage

```
PG2PEtri(r)
```

Arguments

r	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$ to attain non-degenerate asymptotic distribution for the domination number.

Value

P(domination number=2) for PE-PCD for uniform data on an triangle as the sample size n goes to infinity

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C (2007). "On the Distribution of the Domination Number of a New Family of Parametrized Random Digraphs." *Model Assisted Statistics and Applications*, **1(4)**, 231-255.

See Also

PG2PE1D

Examples

```
PG2PEtri(r=1.5)
PG2PEtri(r=1.4999999999)

PG2PEtri(r=1.5) / PG2PEtri(r=1.4999999999)

rseq<-seq(1.01,1.49999999999,l=20)  #try also l=100
lrseq<-length(rseq)

pg2<-vector()
for (i in 1:lrseq)
{
  pg2<-c(pg2,PG2PEtri(rseq[i]))
}

plot(rseq, pg2,type="l",xlab="r",ylab=expression(paste("P(", gamma, "=2)")),
     lty=1,xlim=range(rseq)+c(0,.01),ylim=c(0,1))
points(rbind(c(1.50,PG2PEtri(1.50))),pch=".",cex=3)
```

Plane *The plane passing through three distinct 3D points* a, b, *and* c

---

Description

An object of class "Planes". Returns the equation and z-coordinates of the plane passing through three distinct 3D points a, b, and c with x- and y-coordinates are provided in vectors x and y, respectively.

Usage

```
Plane(a, b, c, x, y)
```

Arguments

a, b, c	3D points that determine the plane (i.e., through which the plane is passing).
x, y	Scalars or vectors of scalars representing the x- and y-coordinates of the plane.

Value

A list with the elements

desc	A description of the plane
points	The input points a, b, and c through which the plane is passing (stacked row-wise, i.e., row 1 is point a, row 2 is point b and row 3 is point c).
x,y	The input vectors which constitutes the x- and y-coordinates of the point(s) of interest on the plane. x and y can be scalars or vectors of scalars.
z	The output vector which constitutes the z-coordinates of the point(s) of interest on the plane. If x and y are scalars, z will be a scalar and if x and y are vectors of scalars, then z needs to be matrix of scalars, containing the z-coordinate for each pair of x and y values.
coeff	Coefficients of the plane (in the $z = Ax + By + C$ form).
equation	Equation of the plane in long form
equation2	Equation of the plane in short form, to be inserted on the plot

See Also

paraplane

Examples

```
A<-c(1,10,3); B<-c(1,1,3); C<-c(3,9,12)

pts<-rbind(A,B,C)
Plane(A,B,C,.1,.2)

xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1 #how far to go at the lower and upper ends in the y-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=20)  #try also l=100
y<-seq(yr[1]-yf,yr[2]+yf,l=20)  #try also l=100

plABC<-Plane(A,B,C,x,y)
plABC
summary(plABC)
plot(plABC)
```

```
Plane(A,B,A+B,.1,.2)

z.grid<-plABC$z

persp(x,y,z.grid, xlab="x",ylab="y",zlab="z",
theta = -30, phi = 30, expand = 0.5, col = "lightblue",
      ltheta = 120, shade = 0.05, ticktype = "detailed")

zr<-max(z.grid)-min(z.grid)
Pts<-rbind(A,B,C)+rbind(c(0,0,zr*.1),c(0,0,zr*.1),c(0,0,zr*.1))
Mn.pts<-apply(Pts,2,mean)

plot3D::persp3D(z = z.grid, x = x, y = y, theta =225, phi = 30, ticktype = "detailed")
#plane spanned by points A, B, C
#add the defining points
plot3D::points3D(Pts[,1],Pts[,2],Pts[,3], add=TRUE)
plot3D::text3D(Pts[,1],Pts[,2],Pts[,3], c("A","B","C"),add=TRUE)
plot3D::text3D(Mn.pts[1],Mn.pts[2],Mn.pts[3],plABC$equation,add=TRUE)
plot3D::polygon3D(Pts[,1],Pts[,2],Pts[,3], add=TRUE)

Plane(A,B,C,.1,.2)
```

`plot.Extrema` *Plot an* `Extrema` *object*

Description

Plots the data points and extrema among these points together with the reference object (e.g., boundary of the support region)

Usage

```
## S3 method for class 'Extrema'
plot(x, asp = NA, xlab = " ", ylab = " ", zlab = "", ...)
```

Arguments

`x`	Object of class `Extrema`.
`asp`	A numeric value, giving the aspect ratio for y axis to x-axis y/x for the 2D case, it is redundant in the 3D case (default is `NA`), see the official help for `asp` by typing "? asp".
`xlab, ylab, zlab`	Titles for the x and y axes in the 2D case, and x, y, and z axes in the 3D case, respectively (default is "" for all).
`...`	Additional parameters for `plot`.

Value

None

See Also

print.Extrema, summary.Extrema, and print.summary.Extrema

Examples

```
n<-20
dat<-runifTe(n)$gen.points
Ext<-cl2edgesTe(dat)
Ext
plot(Ext,asp=1)
```

---

plot.Lines *Plot a* Lines *object*

Description

Plots the line together with the defining points.

Usage

```
## S3 method for class 'Lines'
plot(x, asp = NA, xlab = "x", ylab = "y", ...)
```

Arguments

x	Object of class Lines.
asp	A numeric value, giving the aspect ratio for y axis to x-axis y/x (default is NA), see the official help for asp by typing "? asp".
xlab, ylab	Titles for the x and y axes, respectively (default is xlab="x" and ylab="y").
...	Additional parameters for plot.

Value

None

See Also

print.Lines, summary.Lines, and print.summary.Lines

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75)
xr<-range(A,B);
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=100)

lnAB<-Line(A,B,x)
lnAB
plot(lnAB)
```

---

plot.Lines3D *Plot a* Lines3D *object*

Description

Plots the line together with the defining vectors (i.e., the initial and direction vectors).

Usage

```
## S3 method for class 'Lines3D'
plot(x, xlab = "x", ylab = "y", zlab = "z", ...)
```

Arguments

x	Object of class Lines3D.
xlab, ylab, zlab	Titles for the x, y and z axes, respectively (default is xlab="x", ylab="y" and zlab="z").
...	Additional parameters for plot.

Value

None

See Also

print.Lines3D, summary.Lines3D, and print.summary.Lines3D

Examples

```
A<-c(1,10,3); B<-c(1,1,3);
vecs<-rbind(A,B)
Line3D(A,B,.1)
Line3D(A,B,.1,dir.vec=FALSE)

tr<-range(vecs);
tf<-(tr[2]-tr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
```

```
tsq<-seq(-tf*10-tf,tf*10+tf,l=100)

lnAB3D<-Line3D(A,B,tsq)
lnAB3D
plot(lnAB3D)
```

`plot.Patterns` *Plot a* `Patterns` *object*

Description

Plots the points generated from the pattern (color coded for each class) together with the study window

Usage

```
## S3 method for class 'Patterns'
plot(x, asp = NA, xlab = "x", ylab = "y", ...)
```

Arguments

`x`	Object of class `Patterns`.
`asp`	A numeric value, giving the aspect ratio for y axis to x-axis y/x (default is `NA`), see the official help for `asp` by typing "? asp".
`xlab, ylab`	Titles for the x and y axes, respectively (default is `xlab`="x" and `ylab`="y").
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.Patterns`, `summary.Patterns`, and `print.summary.Patterns`

Examples

```
nx<-20; #try also 100 and 1000
ny<-3; #try also 1
e<-.15;
Y<-cbind(runif(ny),runif(ny)) #with default bounding box (i.e., unit square)

Xdt<-rseg.disc(nx,Y,e)
Xdt
plot(Xdt,asp=1)
```

`plot.PCDs` *Plot a* `PCDs` *object*

Description

Plots the vertices and the arcs of the PCD together with the vertices and boundaries of the partition cells (i.e., intervals in the 1D case and triangles in the 2D case)

Usage

```
## S3 method for class 'PCDs'
plot(x, Jit = 0.1, ...)
```

Arguments

`x`	Object of class `PCDs`.
`Jit`	A positive real number that determines the amount of jitter along the y-axis, default is 0.1, for the 1D case, the vertices of the PCD are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where `Jit` equals to the range of vertices and the interval end points; it is redundant in the 2D case.
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.PCDs`, `summary.PCDs`, and `print.summary.PCDs`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10
dat<-runif.tri(n,Tr)$g
M<-as.numeric(runif.tri(1,Tr)$g)
Arcs<-ArcsAStri(dat,Tr,M)
Arcs
plot(Arcs)
```

plot.Planes *Plot a* Planes *object*

Description

Plots the plane together with the defining 3D points.

Usage

```
## S3 method for class 'Planes'
plot(x, xlab = "x", ylab = "y", zlab = "z", ...)
```

Arguments

x	Object of class Planes.
xlab, ylab, zlab	Titles for the x, y and z axes, respectively (default is xlab="x", ylab="y", and zlab="z").
...	Additional parameters for plot.

Value

None

See Also

print.Planes, summary.Planes, and print.summary.Planes

Examples

```
A<-c(1,10,3); B<-c(1,1,3); C<-c(3,9,12)
pts<-rbind(A,B,C)

xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1 #how far to go at the lower and upper ends in the y-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=100)
y<-seq(yr[1]-yf,yr[2]+yf,l=100)

plABC<-Plane(A,B,C,x,y)
plABC
plot(plABC)
```

`plot.TriLines` *Plot a* `TriLines` *object*

Description

Plots the line together with the defining triangle.

Usage

```
## S3 method for class 'TriLines'
plot(x, xlab = "x", ylab = "y", ...)
```

Arguments

`x`	Object of class `TriLines`.
`xlab, ylab`	Titles for the x and y axes, respectively (default is `xlab`="x" and `ylab`="y").
`...`	Additional parameters for `plot`.

Value

None

See Also

`print.TriLines`, `summary.TriLines`, and `print.summary.TriLines`

Examples

```
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.01)

lnACM<-lA_CM.Te(x)
lnACM
plot(lnACM)

## End(Not run)
```

plot.Uniform *Plot a* Uniform *object*

Description

Plots the points generated from the uniform distribution together with the support region

Usage

```
## S3 method for class 'Uniform'
plot(x, asp = NA, xlab = "x", ylab = "y", zlab = "z", ...)
```

Arguments

x	Object of class Uniform.
asp	A numeric value, giving the aspect ratio for y axis to x-axis y/x for the 2D case, it is redundant in the 3D case (default is NA), see the official help for asp by typing "? asp".
xlab, ylab, zlab	Titles for the x and y axes in the 2D case, and x, y, and z axes in the 3D case, respectively (default is xlab="x", ylab="y", and zlab="z").
...	Additional parameters for plot.

Value

None

See Also

print.Uniform, summary.Uniform, and print.summary.Uniform

Examples

```
n<-10 #try also 100
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)

Xdt<-runif.tri(n,Tr)
Xdt
plot(Xdt,asp=1)
```

plotASarcsMT *The plot of the arcs of Arc Slice Proximity Catch Digraph (AS-PCD) for a 2D data set - multiple triangle case*

Description

Plots the arcs of AS-PCD whose vertices are the data points in Xp and Delaunay triangles based on Yp points.

AS proximity regions are constructed with respect to the Delaunay triangles based on Yp points, i.e., AS proximity regions are defined only for Xp points inside the convex hull of Yp points. That is, arcs may exist for Xp points only inside the convex hull of Yp points.

Vertex regions are based on the center M="CC" for circumcenter of each Delaunay triangle or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle; default is M="CC" i.e. circumcenter of each triangle.

See (Ceyhan (2005, 2010)) for more on AS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotASarcsMT(
  Xp,
  Yp,
  M = "CC",
  asp = NA,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

Xp	A set of 2D points which constitute the vertices of the AS-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangulation. The Delaunay triangles partition the convex hull of Yp points.
M	The center of the triangle. "CC" stands for circumcenter of each Delaunay triangle tri or 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle; default is M="CC" i.e. the circumcenter of each triangle.
asp	A numeric value, giving the aspect ratio for y axis to x-axis y/x (default is NA), see the official help for asp by typing "? asp".
main	An overall title for the plot (default="").

xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
...	Additional plot parameters.

Value

A plot of the arcs of the AS-PCD for a 2D data set Xp where AS proximity regions are defined with respect to the Delaunay triangles based on Yp points; also plots the Delaunay triangles based on Yp points

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

plotASarcsTri, plotPEarcsTri, plotPEarcsMT, plotCSarcsTri, and plotCSarcsMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

plotASarcsMT(Xp,Yp,M,xlab="",ylab="")
plotASarcsMT(Xp,Yp,M,asp=1,xlab="",ylab="")

plotASarcsMT(Xp,Yp[1:3,],M,xlab="",ylab="")

Xlim<-range(Xp[,1],Yp[,1])
```

```
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plotASarcsMT(Xp,Yp,M,xlab="",ylab="",xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
plotASarcsMT(Xp,Yp,M,asp=1,xlab="",ylab="",xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
```

`plotASarcsTri` *The plot of the arcs of Arc Slice Proximity Catch Digraph (AS-PCD) for a 2D data set - one triangle case*

Description

Plots the arcs of AS-PCD whose vertices are the data points, `Xp` and the triangle `tri`. AS proximity regions are constructed with respect to the triangle `tri`, i.e. only for `Xp` points inside the triangle `tri`.

Vertex regions are based on the center `M`="CC" for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M`="CC" the circumcenter of `tri`.

See also (Ceyhan (2005, 2010)).

Usage

```
plotASarcsTri(
  Xp,
  tri,
  M = "CC",
  asp = NA,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the AS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	The center of the triangle. "CC" stands for circumcenter of the triangle `tri` or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is `M`="CC" i.e. the circumcenter of `tri`.

asp	A numeric value, giving the aspect ratio for y axis to x-axis y/x (default is NA), see the official help for asp by typing "? asp".
main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
...	Additional plot parameters.

Value

A plot of the arcs of the AS-PCD for a 2D data set Xp where AS proximity regions are defined with respect to the triangle tri; also plots the triangle tri

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

plotASarcsMT, plotPEarcsTri, plotPEarcsMT, plotCSarcsTri, and plotCSarcsMT

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$g  #try also dat<-cbind(runif(n,1,2),runif(n,0,2))

M<-as.numeric(runif.tri(1,Tr)$g)  #try also  #M<-c(1.6,1.2)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

CC<-circ.cent.tri(Tr)
if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
```

```
} else
{cent<-M
cent.name<-"M"
Ds<-cp2e.tri(Tr,M)
D1<-Ds[1,]; D2<-Ds[2,]; D3<-Ds[3,]
}

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plotASarcsTri(dat,Tr,M,main="arcs of AS-PCD",xlab="",ylab="",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,cent,Ds)
xc<-txt[,1]+c(-.02,.03,.03,.03,.05,-0.03,-.01)
yc<-txt[,2]+c(.02,.02,.02,.07,.02,.05,-.06)
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)
```

`plotASregsMT` *The plot of the Arc Slice (AS) Proximity Regions for a 2D data set - multiple triangle case*

Description

Plots the `Xp` points in and outside of the convex hull of `Yp` points and also plots the AS proximity regions for `Xp` points and Delaunay triangles based on `Yp` points.

AS proximity regions are constructed with respect to the Delaunay triangles based on `Yp` points (these triangles partition the convex hull of `Yp` points), i.e., AS proximity regions are only defined for `Xp` points inside the convex hull of `Yp` points.

Vertex regions are based on the center `M`="CC" for circumcenter of each Delaunay triangle or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle; default is `M`="CC" i.e. circumcenter of each triangle.

See (Ceyhan (2005, 2010)) for more on AS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotASregsMT(
  Xp,
  Yp,
  M = "CC",
  main = "",
```

```
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

Xp	A set of 2D points for which AS proximity regions are constructed.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangulation. The Delaunay triangles partition the convex hull of Yp points.
M	The center of the triangle. "CC" stands for circumcenter of each Delaunay triangle tri or 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle; default is M="CC" i.e. the circumcenter of each triangle.
main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
...	Additional plot parameters.

Value

Plot of the Xp points, Delaunay triangles based on Yp and also the AS proximity regions for Xp points inside the convex hull of Yp points

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

plotASregsTri, plotPEregsTri, plotPEregsMT, plotCSregsTri, and plotCSregsMT

Examples

```
nx<-10 ; ny<-10

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

plotASregsMT(Xp,Yp,M,xlab="",ylab="")

Xlim<-range(Xp[,1],Yp[,1])
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plotASregsMT(Xp,Yp,M,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
plotASregsMT(Xp,Yp[1:3,],M,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))

Xp<-c(.5,.5)
Xlim<-Ylim<-range(Xp,Yp)
plotASregsMT(Xp,Yp,M,xlab="",ylab="",xlim=Xlim,ylim=Ylim)
```

---

plotASregsTri *The plot of the Arc Slice (AS) Proximity Regions for a 2D data set - one triangle case*

Description

Plots the points in and outside of the triangle `tri` and also the AS proximity regions for points in data set `Xp`.

AS proximity regions are defined with respect to the triangle `tri`, so AS proximity regions are defined only for points inside the triangle `tri` and vertex regions are based on the center `M`="CC" for circumcenter of `tri`; or $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is `M`="CC" the circumcenter of `tri`.

See also (Ceyhan (2005, 2010)).

Usage

```
plotASregsTri(
  Xp,
  tri,
  M = "CC",
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
```

```
  ylim = NULL,
  ...
)
```

Arguments

Xp	A set of 2D points for which AS proximity regions are constructed.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	The center of the triangle. "CC" stands for circumcenter of the triangle tri or a 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle T_b; default is M="CC" i.e. the circumcenter of tri.
main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
...	Additional plot parameters.

Value

Plot of the AS proximity regions for points inside the triangle tri (and only the points outside tri)

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

plotASregsMT, plotPEregsTri, plotPEregsMT, plotCSregsTri, and plotCSregsMT

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-1

set.seed(1)
dat<-runif.tri(n,Tr)$g
```

```
M<-as.numeric(runif.tri(1,Tr)$g)  #try also  #M<-c(1.6,1.2);

dat<-matrix(dat,ncol=2)
Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plotASregsTri(dat,Tr,M,main="Proximity Regions for AS-PCD",
xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))

plotASarcsTri(dat,Tr,main="arcs of AS-PCD", xlab="",ylab="",xlim=Xlim+c(-.05,.05),ylim=Ylim)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

CC<-circ.cent.tri(Tr)
if (isTRUE(all.equal(M,CC)) || identical(M,"CC"))
{cent<-CC
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
cent.name<-"CC"
} else
{cent<-M
cent.name<-"M"
Ds<-cp2e.tri(Tr,M)
D1<-Ds[1,]; D2<-Ds[2,]; D3<-Ds[3,]
}

plotASregsTri(dat,Tr,M,main="Proximity Regions for AS-PCD",xlab="",ylab="")
L<-rbind(cent,cent,cent); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,cent,Ds)
xc<-txt[,1]+c(-.02,.03,.03,.03,.05,-0.03,-.01)
yc<-txt[,2]+c(.02,.02,.02,.07,.02,.05,-.06)
txt.str<-c("A","B","C",cent.name,"D1","D2","D3")
text(xc,yc,txt.str)
```

`plotCSarcs1D` *The plot of the arcs of Central Similarity Proximity Catch Digraphs (CS-PCDs) for 1D data (vertices jittered along y-coordinate) - multiple interval case*

Description

Plots the arcs of CS-PCD whose vertices are the 1D points, `Xp`. CS proximity regions are constructed with expansion parameter $t > 0$ and centrality parameter `c` in $(0, 1)$ and the intervals are based on

Yp points (i.e. the intervalization is based on Yp points). That is, data set Xp constitutes the vertices of the digraph and Yp determines the end points of the intervals.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default for $Jit = .1$) is added to the y-direction where Jit equals to the range of Xp and Yp multiplied by Jit with default for $Jit = .1$).

centers is a logical argument, if TRUE, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.

See also (Ceyhan (2016)).

Usage

```
plotCSarcs1D(
  Xp,
  Yp,
  t,
  c,
  Jit = 0.1,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  centers = FALSE,
  ...
)
```

Arguments

Xp	A vector of 1D points constituting the vertices of the CS-PCD.
Yp	A vector of 1D points constituting the end points of the intervals.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
Jit	A positive real number that determines the amount of jitter along the y-axis, default is 0.1 and Xp points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where Jit equals to the range of Xp and Yp multiplied by Jit).
main	Title of the main heading of the plot.
xlab, ylab	Titles of the x and y axes in the plot (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
centers	A logical argument, if TRUE, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.
...	Additional plot parameters.

Value

A plot of the arcs of CS-PCD whose vertices are the 1D data set Xp in which vertices are jittered along y-axis for better visualization.

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

plotPEarcs1D

Examples

```
t<-1.5
c<-.4
a<-0; b<-10; int<-c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

Xlim=range(Xp,Yp)
Ylim=c(-.2,.2)

jit<-.1

plotCSarcs1D(Xp,Yp,t,c,jit,xlab="",ylab="",xlim=Xlim,ylim=Ylim)

set.seed(1)
plotCSarcs1D(Xp,Yp,t=1.5,c=.3,jit,main="t=1.5, c=.3",xlab="",ylab="",centers=TRUE)
set.seed(1)
plotCSarcs1D(Xp,Yp,t=2,c=.3,jit,main="t=2, c=.3",xlab="",ylab="",centers=TRUE)
set.seed(1)
plotCSarcs1D(Xp,Yp,t=1.5,c=.5,jit,main="t=1.5, c=.5",xlab="",ylab="",centers=TRUE)
set.seed(1)
plotCSarcs1D(Xp,Yp,t=2,c=.5,jit,main="t=2, c=.5",xlab="",ylab="",centers=TRUE)
```

`plotCSarcsMT` *The plot of the arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for a 2D data set - multiple triangle case*

Description

Plots the arcs of Central Similarity Proximity Catch Digraph (CS-PCD) whose vertices are the data points in `Xp` in the multiple triangle case and the Delauany triangles based on `Yp` points.

CS proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $\tau > 0$ and edge regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (unscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). Loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) more on the CS-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotCSarcsMT(
  Xp,
  Yp,
  tau,
  M = c(1, 1, 1),
  asp = NA,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`tau`	A positive real number which serves as the expansion parameter in CS proximity region.
`M`	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle, default for $M = (1, 1, 1)$ which is the center of mass of each triangle.

asp	A numeric value, giving the aspect ratio y/x (default is NA), see the official help for asp by typing "? asp"
main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both)
...	Additional plot parameters.

Value

A plot of the arcs of the CS-PCD whose vertices are the points in data set Xp and the Delaunay triangles based on Yp points

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

plotCSarcsTri, plotASarcsMT, and plotPEarcsMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

tau<-1.5  #try also tau<-2
```

```
Xlim<-range(Xp[,1],Yp[,1])
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plotCSarcsMT(Xp,Yp,tau,M,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))

plotCSarcsMT(Xp,Yp[1:3,],tau,M,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
```

`plotCSarcsTri` *The plot of the arcs of Central Similarity Proximity Catch Digraph (CS-PCD) for a 2D data set - one triangle case*

Description

Plots the arcs of CS-PCD whose vertices are the data points, `Xp` and the triangle `tri`. CS proximity regions are constructed with respect to the triangle `tri` with expansion parameter $t > 0$, i.e. arcs may exist only for `Xp` points inside the triangle `tri`.

Edge regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
plotCSarcsTri(
  Xp,
  tri,
  t,
  M = c(1, 1, 1),
  asp = NA,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

t	A positive real number which serves as the expansion parameter in CS proximity region.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.
asp	A numeric value, giving the aspect ratio y/x (default is NA), see the official help for asp by typing "? asp".
main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
...	Additional plot parameters.

Value

A plot of the arcs of the CS-PCD whose vertices are the points in data set Xp and the triangle tri

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

plotCSarcsMT, plotPEarcsTri and plotASarcsTri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

t<-1.5  #try also t<-2

dat<-matrix(dat,ncol=2)
Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
```

```
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

Ds<-cp2e.tri(Tr,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plotCSarcsTri(dat,Tr,t,M,main="arcs of CS-PCD with t=1.5",xlab="",ylab="",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,M,Ds)
xc<-txt[,1]+c(-.02,.03,.03,.03,.05,-0.03,-.01)
yc<-txt[,2]+c(.02,.02,.02,.07,.02,.05,-.06)
txt.str<-c("A","B","C","M","D1","D2","D3")
text(xc,yc,txt.str)
```

`plotCSregsInt` *The plot of the Central Similarity (CS) Proximity Regions for a general interval (vertices jittered along y-coordinate) - one interval case*

Description

Plots the points in and outside of the interval `int` and also the CS proximity regions (which are also intervals).

CS proximity regions are constructed with expansion parameter $t > 0$ and centrality parameter `c` in $(0, 1)$. For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default for $Jit = .1$ times range of proximity regions and `dat`) is added to the y-direction.

`centers` is a logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.

See also (Ceyhan (2016)).

Usage

```
plotCSregsInt(
  dat,
  t,
  c = 0.5,
  int,
  Jit = 0.1,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
```

```
  ylim = NULL,
  centers = FALSE,
  ...
)
```

Arguments

dat	A set of 1D points for which CS proximity regions are to be constructed.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$; default=0.5.
int	A vector of two real numbers representing an interval.
Jit	A positive real number that determines the amount of jitter along the y-axis, default is 0.1 and dat points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where Jit equals to the range of dat and proximity region intervals multiplied by Jit).
main	Title of the main heading of the plot.
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges.
centers	A logical argument, if TRUE, plot includes the centers of the intervals. as vertical lines in the plot, else centers of the intervals are not plotted.
...	Additional plot parameters.

Value

Plot of the CS proximity regions for 1D points in or outside the interval int

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

plotCSregsMI, plotPEregsInt, and plotPEregsMI

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

n<-10
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1
```

```
dat<-runif(n,a-xf,b+xf)  #try also dat<-runif(n,a-5,b+5)

plotCSregsInt(7,t,c,int)

plotCSregsInt(dat,t,c,int)

plotCSregsInt(17,t,c,int)
plotCSregsInt(1,t,c,int)
plotCSregsInt(4,t,c,int)

plotCSregsInt(-7,t,c,int)
```

`plotCSregsMI` *The plot of the Central Similarity (CS) Proximity Regions (vertices jittered along y-coordinate) - multiple interval case*

Description

Plots the points in and outside of the intervals based on `Yp` points and also the CS proximity regions (which are also intervals).

CS proximity region is constructed with expansion parameter $t > 0$ and centrality parameter `c` in $(0, 1)$. For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default for $Jit = .1$ times range of `Xp` and `Yp` and the proximity regions (intervals)) is added to the y-direction.

`centers` is a logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.

See also (Ceyhan (2016)).

Usage

```
plotCSregsMI(
  Xp,
  Yp,
  t,
  c,
  Jit = 0.1,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  centers = FALSE,
  ...
)
```

Arguments

Xp	A set of 1D points for which CS proximity regions are plotted.
Yp	A set of 1D points which constitute the end points of the intervals which partition the real line.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$; default=0.5.
Jit	A positive real number that determines the amount of jitter along the y-axis, default is 0.1 and Xp points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where Jit equals to the range of Xp and Yp and the proximity regions (intervals) multiplied by Jit).
main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
centers	A logical argument, if TRUE, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.
...	Additional plot parameters.

Value

Plot of the CS proximity regions for 1D points located in the middle or end intervals based on Yp points

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

plotCSregsInt and plotPEregsMI

Examples

```
t<-2
c<-.4
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1
```

```
Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

plotCSregsMI(Xp,Yp,t,c,xlab="",ylab="")
```

`plotCSregsMT` *The plot of the Central Similarity (CS) Proximity Regions for a 2D data set - multiple triangle case*

Description

Plots the points in and outside of the Delaunay triangles based on `Yp` points which partition the convex hull of `Yp` points and also plots the CS proximity regions for `Xp` points and the Delaunay triangles based on `Yp` points.

CS proximity regions are contructed with respect to the Delaunay triangles with the expansion parameter $t > 0$.

Edge regions in each triangle is based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle).

See (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)) more on the CS proximity regions. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotCSregsMT(
  Xp,
  Yp,
  t,
  M = c(1, 1, 1),
  asp = NA,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points for which CS proximity regions are constructed.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.

t	A positive real number which serves as the expansion parameter in CS proximity region.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri.
asp	A numeric value, giving the aspect ratio y/x (default is NA), see the official help for asp by typing "? asp".
main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
...	Additional plot parameters.

Value

Plot of the Xp points, Delaunay triangles based on Yp and also the CS proximity regions for Xp points inside the convex hull of Yp points

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

plotCSregsTri, plotASregsMT and plotPEregsMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
```

```
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

t<-1.5  #try also t<-2

Xlim<-range(Xp[,1],Yp[,1])
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plotCSregsMT(Xp,Yp,t,M,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))

plotCSregsMT(Xp,Yp[1:3,],t,M,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
```

`plotCSregsTri` *The plot of the Central Similarity (CS) Proximity Regions for a 2D data set - one triangle case Plots the points in and outside of the triangle* `tri` *and also the CS proximity regions which are also triangular for points inside the triangle* `tri` *with edge regions are based on the center of mass CM.*

Description

Plots the points in and outside of the triangle `tri` and also the CS proximity regions for points in data set `Xp`.

CS proximity regions are defined with respect to the triangle `tri` with expansion parameter $t > 0$, so CS proximity regions are defined only for points inside the triangle `tri`.

Edge regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
plotCSregsTri(
  Xp,
  tri,
  t,
  M = c(1, 1, 1),
  asp = NA,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
```

```
  ylim = NULL,
  ...
)
```

Arguments

Xp	A set of 2D points for which CS proximity regions are constructed.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
t	A positive real number which serves as the expansion parameter in CS proximity region.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri; default is $M = (1, 1, 1)$ i.e. the center of mass of tri.
asp	A numeric value, giving the aspect ratio y/x (default is NA), see the official help for asp by typing "? asp".
main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
...	Additional plot parameters.

Value

Plot of the CS proximity regions for points inside the triangle tri (and just the points outside tri)

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

plotCSregsMT, plotASregsTri and plotPEregsTri,

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10
```

```
set.seed(1)
dat<-runif.tri(n,Tr)$g
dat<-matrix(dat,ncol=2)

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

t<-.5  #try also t<-2

dat<-matrix(dat,ncol=2)
Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plotCSregsTri(dat[4,],Tr,t,M)
plotCSregsTri(dat,Tr,t,M)

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates
plotCSregsTri(dat,Tr,t,M,main="CS Proximity Regions with t=.5",
xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
L<-rbind(M,M,M); R<-Tr
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,M)
xc<-txt[,1]+c(-.02,.03,.03,.03)
yc<-txt[,2]+c(.02,.02,.02,.07)
txt.str<-c("A","B","C","M")
text(xc,yc,txt.str)
```

`plotDeltri` *The scatterplot of points from one class and plot of the Delaunay triangulation of the other class*

Description

Plots the scatter plot of `Xp` points together with the Delaunay triangles based on the `Yp` points. Both sets of points are of 2D.

See (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotDeltri(
  Xp,
  Yp,
  main = "",
```

```
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

Xp	A set of 2D points whose scatterplot is to be plotted.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both)
...	Additional plot parameters.

Value

A scatterplot of Xp points and the Delaunay triangulation of Yp points.

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

plot.triSht in interp package

Examples

```
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny),runif(ny))

oldpar <- par(mfrow = c(1,2))
plotDeltri(Xp,Yp,xlab="",ylab="",main="X points and Delaunay Triangulation of Y points")

P<-c(.6,.4)
plotDeltri(P,Yp,xlab="",ylab="",main="X points and Delaunay Triangulation of Y points")
```

```
plotDeltri(Xp,Yp,xlab="",ylab="")
plotDeltri(Xp,Yp[1:3,],xlab="",ylab="")

plotDeltri(Xp,rbind(Yp,Yp),xlab="",ylab="")

dat.fr<-data.frame(a=Xp)
plotDeltri(dat.fr,Yp,xlab="",ylab="")

dat.fr<-data.frame(a=Yp)
plotDeltri(Xp,dat.fr,xlab="",ylab="")
par(oldpar)
```

`plotIntervals` *The plot of the subintervals based on* `Yp` *points together with* `Xp` *points*

Description

Plots the `Xp` points and the intervals based on `Yp` points points.

Usage

```
plotIntervals(
  Xp,
  Yp,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 1D points whose scatter-plot is provided.
`Yp`	A set of 1D points which constitute the end points of the intervals which partition the real line.
`main`	An overall title for the plot (default="").
`xlab, ylab`	Titles for the x and y axes, respectively (default="" for both).
`xlim, ylim`	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`...`	Additional `plot` parameters.

Value

Plot of the intervals based on `Yp` points and also scatter plot of `Xp` points

See Also

plotPEregsMI and plotDeltri

Examples

```
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)

plotIntervals(Xp,Yp,xlab="",ylab="")
plotIntervals(Xp,Yp+10,xlab="",ylab="")
```

---

plotPEarcs1D — *The plot of the arcs of Proportional Edge Proximity Catch Digraphs (PE-PCDs) for 1D data (vertices jittered along y-coordinate) - multiple interval case*

Description

Plots the arcs of PE-PCD whose vertices are the 1D points, Xp. PE proximity regions are constructed with expansion parameter $r \geq 1$ and centrality parameter c in $(0, 1)$ and the intervals are based on Yp points (i.e. the intervalization is based on Yp points). That is, data set Xp constitutes the vertices of the digraph and Yp determines the end points of the intervals.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default for $Jit = .1$) is added to the y-direction where Jit equals to the range of Xp and Yp multiplied by Jit with default for $Jit = .1$). centers is a logical argument, if TRUE, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.

See also (Ceyhan (2012)).

Usage

```
plotPEarcs1D(
  Xp,
  Yp,
  r,
  c,
  Jit = 0.1,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
```

```
  ylim = NULL,
  centers = FALSE,
  ...
)
```

Arguments

Xp	A vector of 1D points constituting the vertices of the PE-PCD.
Yp	A vector of 1D points constituting the end points of the intervals.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
Jit	A positive real number that determines the amount of jitter along the y-axis, default is 0.1 and Xp points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where Jit equals to the range of Xp and Yp multiplied by Jit).
main	Title of the main heading of the plot.
xlab, ylab	Titles of the x and y axes in the plot (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
centers	A logical argument, if TRUE, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.
...	Additional plot parameters.

Value

A plot of the arcs of PE-PCD whose vertices are the 1D data set Xp in which vertices are jittered along y-axis for better visualization.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

plotCSarcs1D

Examples

```
r<-2
c<-.4
a<-0; b<-10; int<-c(a,b)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;
```

```
set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

Xlim=range(Xp,Yp)
Ylim=c(-.2,.2)

jit<-.1

plotPEarcs1D(Xp,Yp,r,c,jit,xlab="",ylab="",xlim=Xlim,ylim=Ylim)

set.seed(1)
plotPEarcs1D(Xp,Yp,r=1.5,c=.3,jit,main="r=1.5, c=.3",xlab="",ylab="",centers=TRUE)
set.seed(1)
plotPEarcs1D(Xp,Yp,r=2,c=.3,jit,main="r=2, c=.3",xlab="",ylab="",centers=TRUE)
set.seed(1)
plotPEarcs1D(Xp,Yp,r=1.5,c=.5,jit,main="r=1.5, c=.5",xlab="",ylab="",centers=TRUE)
set.seed(1)
plotPEarcs1D(Xp,Yp,r=2,c=.5,jit,main="r=2, c=.5",xlab="",ylab="",centers=TRUE)
```

`plotPEarcsMT` *The plot of the arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for a 2D data set - multiple triangle case*

Description

Plots the arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) whose vertices are the data points in `Xp` in the multiple triangle case and the Delauany triangles based on `Yp` points.

PE proximity regions are defined with respect to the Delaunay triangles based on `Yp` points with expansion parameter $r \geq 1$ and vertex regions in each triangle are based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle). Each Delaunay triangle is first converted to an (unscaled) basic triangle so that `M` will be the same type of center for each Delaunay triangle (this conversion is not necessary when `M` is CM).

Convex hull of `Yp` is partitioned by the Delaunay triangles based on `Yp` points (i.e., multiple triangles are the set of these Delaunay triangles whose union constitutes the convex hull of `Yp` points). Loops are not allowed so arcs are only possible for points inside the convex hull of `Yp` points.

See (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)) for more on the PE-PCDs. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotPEarcsMT(
  Xp,
  Yp,
  r,
  M = c(1, 1, 1),
  asp = NA,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

Xp	A set of 2D points which constitute the vertices of the PE-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 3D point in barycentric coordinates which serves as a center in the interior of each Delaunay triangle or circumcenter of each Delaunay triangle (for this argument should be set as M="CC"), default for $M = (1, 1, 1)$ which is the center of mass of each triangle.
asp	A numeric value, giving the aspect ratio y/x (default is NA), see the official help for asp by typing "? asp".
main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
...	Additional plot parameters.

Value

A plot of the arcs of the PE-PCD whose vertices are the points in data set Xp and the Delaunay triangles based on Yp points

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

plotPEarcsTri, plotASarcsMT, and plotCSarcsMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

M<-c(1,1,1)  #try also M<-c(1,2,3)

r<-1.5  #try also r<-2

Xlim<-range(Xp[,1],Yp[,1])
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plotPEarcsMT(Xp,Yp,r,M,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))

plotPEarcsMT(Xp,Yp[1:3,],r,M,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
```

---

plotPEarcsTri — *The plot of the arcs of Proportional Edge Proximity Catch Digraph (PE-PCD) for a 2D data set - one triangle case*

Description

Plots the arcs of PE-PCD whose vertices are the data points, `Xp` and the triangle `tri`. PE proximity regions are constructed with respect to the triangle `tri` with expansion parameter $r \geq 1$, i.e. arcs may exist only for `Xp` points inside the triangle `tri`.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)).

Usage

```
plotPEarcsTri(
  Xp,
  tri,
  r,
  M = c(1, 1, 1),
  asp = NA,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the PE-PCD.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.
`asp`	A numeric value, giving the aspect ratio y/x (default is `NA`), see the official help for `asp` by typing "? asp".
`main`	An overall title for the plot (default="").
`xlab, ylab`	Titles for the x and y axes, respectively (default="" for both).
`xlim, ylim`	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=`NULL` for both).
`...`	Additional `plot` parameters.

Value

A plot of the arcs of the PE-PCD whose vertices are the points in data set `Xp` and the triangle `tri`

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

plotPEarcsMT, plotASarcsTri and plotASarcsMT

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10  #try also n<-20

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5  #try also r<-2

ifelse(isTRUE(all.equal(M,circ.cent.tri(Tr))),
Ds<-rbind((B+C)/2,(A+C)/2,(A+B)/2),Ds<-cp2e.tri(Tr,M))

dat<-matrix(dat,ncol=2)

Xlim<-range(Tr[,1],dat[,1],M[1])
Ylim<-range(Tr[,2],dat[,2],M[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plotPEarcsTri(dat,Tr,r,M,main="arcs of PE-PCD with r=1.5",
xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,M,Ds)
xc<-txt[,1]+c(-.02,.03,.03,.03,.05,-0.03,-.01)
```

```
yc<-txt[,2]+c(.02,.02,.02,.07,.02,.05,-.06)
txt.str<-c("A","B","C","M","D1","D2","D3")
text(xc,yc,txt.str)
```

---

plotPEregsInt *The plot of the Proportional Edge (PE) Proximity Regions for a general interval (vertices jittered along y-coordinate) - one interval case*

Description

Plots the points in and outside of the interval `int` and also the PE proximity regions (which are also intervals). PE proximity regions are constructed with expansion parameter $r \geq 1$ and centrality parameter `c` in $(0, 1)$.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default for $Jit = .1$ times range of proximity regions and `dat`) is added to the y-direction. `centers` is a logical argument, if `TRUE`, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.

See also (Ceyhan (2012)).

Usage

```
plotPEregsInt(
  dat,
  r,
  c = 0.5,
  int,
  Jit = 0.1,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  centers = FALSE,
  ...
)
```

Arguments

`dat`	A set of 1D points for which PE proximity regions are to be constructed.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$; default=0.5
`int`	A vector of two real numbers representing an interval.

Jit	A positive real number that determines the amount of jitter along the y-axis, default is 0.1 and dat points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where Jit equals to the range of dat and proximity region intervals multiplied by Jit).
main	Title of the main heading of the plot.
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges.
centers	A logical argument, if TRUE, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.
...	Additional plot parameters.

Value

Plot of the PE proximity regions for 1D points in or outside the interval int

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

plotPEregsMI, plotCSregsInt, and plotCSregsMI

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

n<-10
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1

dat<-runif(n,a-xf,b+xf)  #try also dat<-runif(n,a-5,b+5)

plotPEregsInt(7,r,c,int)

plotPEregsInt(dat,r,c,int)

plotPEregsInt(17,r,c,int)
plotPEregsInt(1,r,c,int)
plotPEregsInt(4,r,c,int)

plotPEregsInt(-7,r,c,int)
```

plotPEregsMI *The plot of the Proportional Edge (PE) Proximity Regions (vertices jittered along y-coordinate) - multiple interval case*

Description

Plots the points in and outside of the intervals based on Yp points and also the PE proximity regions (i.e., intervals). PE proximity region is constructed with expansion parameter $r \geq 1$ and centrality parameter c in $(0, 1)$.

For better visualization, a uniform jitter from $U(-Jit, Jit)$ (default for $Jit = .1$ times range of Xp and Yp and the proximity regions (intervals)) is added to the y-direction.

centers is a logical argument, if TRUE, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted.

See also (Ceyhan (2012)).

Usage

```
plotPEregsMI(
  Xp,
  Yp,
  r,
  c,
  Jit = 0.1,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  centers = FALSE,
  ...
)
```

Arguments

Xp	A set of 1D points for which PE proximity regions are plotted.
Yp	A set of 1D points which constitute the end points of the intervals which partition the real line.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$; default=0.5.
Jit	A positive real number that determines the amount of jitter along the y-axis, default is 0.1 and Xp points are jittered according to $U(-Jit, Jit)$ distribution along the y-axis where Jit equals to the range of Xp and Yp and the proximity regions (intervals) multiplied by Jit).

main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
centers	A logical argument, if TRUE, plot includes the centers of the intervals as vertical lines in the plot, else centers of the intervals are not plotted (default is FALSE).
...	Additional plot parameters.

Value

Plot of the PE proximity regions for 1D points located in the middle or end intervals based on Yp points

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

See Also

plotPEregsMI, plotCSregsInt, and plotCSregsMI

Examples

```
r<-2
c<-.4
a<-0; b<-10;

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.1

Xp<-runif(nx,a-xf,b+xf)
Yp<-runif(ny,a,b)

plotPEregsMI(Xp,Yp,r,c,xlab="",ylab="")

plotPEregsMI(Xp,Yp+10,r,c,xlab="",ylab="")
```

plotPEregsMT *The plot of the Proportional Edge (PE) Proximity Regions for a 2D data set - multiple triangle case*

Description

Plots the points in and outside of the Delaunay triangles based on Yp points which partition the convex hull of Yp points and also plots the PE proximity regions for Xp points and the Delaunay triangles based on Yp points.

PE proximity regions are contructed with respect to the Delaunay triangles with the expansion parameter $r \geq 1$.

Vertex regions in each triangle is based on the center $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of each Delaunay triangle or based on circumcenter of each Delaunay triangle (default for $M = (1, 1, 1)$ which is the center of mass of the triangle).

See (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)) for more on the PE proximity regions. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
plotPEregsMT(
  Xp,
  Yp,
  r,
  M = c(1, 1, 1),
  asp = NA,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

Xp	A set of 2D points for which PE proximity regions are constructed.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri.
asp	A numeric value, giving the aspect ratio y/x (default is NA), see the official help for asp by typing "? asp".

main	An overall title for the plot (default="").
xlab, ylab	Titles for the x and y axes, respectively (default="" for both)
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
...	Additional plot parameters.

Value

Plot of the Xp points, Delaunay triangles based on Yp points and also the PE proximity regions for Xp points inside the convex hull of Yp points

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

plotPEregsTri, plotASregsMT and plotCSregsMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

Xlim<-range(Xp[,1],Yp[,1])
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
```

```
M<-c(1,1,1)  #try also M<-c(1,2,3)

r<-1.5  #try also r<-2

plotPEregsMT(Xp,Yp,r,M,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))

plotPEregsMT(Xp,Yp[1:3,],r,M,xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
```

`plotPEregsStdTH` *The plot of the Proportional Edge (PE) Proximity Regions for a 3D data set - standard regular tetrahedron case*

Description

Plots the points in and outside of the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2,\sqrt{3}/2,0),(1/2,\sqrt{3}/6,\sqrt{6}/$ and also the PE proximity regions for points in data set `Xp`.

PE proximity regions are defined with respect to the standard regular tetrahedron T_h with expansion parameter $r \geq 1$, so PE proximity regions are defined only for points inside T_h.

Vertex regions are based on circumcenter (which is equivalent to the center of mass for the standard regular tetrahedron) of T_h.

See also (Ceyhan (2005, 2010)).

Usage

```
plotPEregsStdTH(
  Xp,
  r,
  main = "",
  xlab = "",
  ylab = "",
  zlab = "",
  xlim = NULL,
  ylim = NULL,
  zlim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 3D points for which PE proximity regions are constructed.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`main`	An overall title for the plot (default="").

`xlab, ylab, zlab`
: titles for the x, y and z axes, respectively (default="" for all).

`xlim, ylim, zlim`
: Numeric vectors of length 2, giving the x-, y- and z-coordinate ranges (default=`NULL` for all).

`...`
: Additional `scatter3D` parameters.

Value

Plot of the PE proximity regions for points inside the standard regular tetrahedron T_h (and just the points outside T_h)

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`plotPEregsMT`, `plotASregsTri`, `plotASregsMT`, `plotCSregsTri`, and `plotCSregsMT`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
r<-1.5

n<-10  #try also n<-20
dat<-runif.stdtetra(n)$g  #try also dat[,1]<-dat[,1]+1

plotPEregsStdTH(dat[1,],r)

plotPEregsStdTH(dat[5,],r)

plotPEregsStdTH(c(.4,.4,.4),r)

plotPEregsStdTH(c(.5,.5,.5),r)

plotPEregsStdTH(dat[1:3,],r)

P1<-c(.1,.1,.1)
plotPEregsStdTH(rbind(P1,P1),r)
```

plotPEregsTH *The plot of the Proportional Edge (PE) Proximity Regions for a 3D data set - one tetrahedron case*

Description

Plots the points in and outside of the tetrahedron `th` and also the PE proximity regions (which are also tetrahedrons) for points inside the tetrahedron `th`.

PE proximity regions are constructed with respect to tetrahedron `th` with expansion parameter $r \geq 1$ and vertex regions are based on the center `M` which is circumcenter ("CC") or center of mass ("CM") of `th` with default="CM", so PE proximity regions are defined only for points inside the tetrahedron `th`.

See also (Ceyhan (2005, 2010)).

Usage

```
plotPEregsTH(
  Xp,
  r,
  th,
  M = "CM",
  main = "",
  xlab = "",
  ylab = "",
  zlab = "",
  xlim = NULL,
  ylim = NULL,
  zlim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 3D points for which PE proximity regions are constructed.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`th`	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.
`M`	The center to be used in the construction of the vertex regions in the tetrahedron, `th`. Currently it only takes "CC" for circumcenter and "CM" for center of mass; default="CM".
`main`	An overall title for the plot (default="").
`xlab, ylab, zlab`	Titles for the x, y and z axes, respectively (default="" for all).

`xlim, ylim, zlim`
: Numeric vectors of length 2, giving the x-, y- and z-coordinate ranges (default=NULL for all).

`...`
: Additional `scatter3D` parameters.

Value

Plot of the PE proximity regions for points inside the tetrahedron `th` (and just the points outside th)

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`plotPEregsStdTH`, `plotPEregsTri` and `plotPEregsInt`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-10  #try also n<-20

dat<-runif.tetra(n,tetra)$g  #try also dat[,1]<-dat[,1]+1

M<-"CM"  #try also M<-"CC"
r<-1.5

plotPEregsTH(dat[1,],r,tetra)  #uses the default M="CM"
plotPEregsTH(dat[1,],r,tetra,M)

plotPEregsTH(dat[5,],r,tetra,M)

plotPEregsTH(c(.4,.4,.4),r,tetra,M)

plotPEregsTH(c(.5,.5,.5),r,tetra,M)

plotPEregsTH(dat[1:3,],r,tetra,M)

P1<-c(.1,.1,.1)
plotPEregsTH(rbind(P1,P1),r,tetra,M)
```

plotPEregsTri *The plot of the Proportional Edge (PE) Proximity Regions for a 2D data set - one triangle case*

Description

Plots the points in and outside of the triangle `tri` and also the PE proximity regions for points in data set `Xp`.

PE proximity regions are defined with respect to the triangle `tri` with expansion parameter $r \geq 1$, so PE proximity regions are defined only for points inside the triangle `tri`.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` or based on the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.

See also (Ceyhan (2005); Ceyhan et al. (2006); Ceyhan (2011)).

Usage

```
plotPEregsTri(
  Xp,
  tri,
  r,
  M = c(1, 1, 1),
  asp = NA,
  main = "",
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`Xp`	A set of 2D points for which PE proximity regions are constructed.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`; default is $M = (1, 1, 1)$ i.e. the center of mass of `tri`.
`asp`	A numeric value, giving the aspect ratio y/x (default is `NA`), see the official help for `asp` by typing "? asp".
`main`	An overall title for the plot (default="").

xlab, ylab	Titles for the x and y axes, respectively (default="" for both).
xlim, ylim	Numeric vectors of length 2, giving the x- and y-coordinate ranges (default=NULL for both).
...	Additional plot parameters.

Value

Plot of the PE proximity regions for points inside the triangle tri (and just the points outside tri)

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

plotPEregsMT, plotASregsTri and plotCSregsTri

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10

set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

r<-1.5  #try also r<-2

ifelse(identical(M,circ.cent.tri(Tr)),Ds<-rbind((B+C)/2,(A+C)/2,(A+B)/2),Ds<-cp2e.tri(Tr,M))

plotPEregsTri(dat,Tr,r,M)
plotPEregsTri(dat[1,],Tr,r,M)

dat<-matrix(dat,ncol=2)
Xlim<-range(Tr[,1],dat[,1],M[1])
Ylim<-range(Tr[,2],dat[,2],M[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
```

```
#need to run this when M is given in barycentric coordinates
plotPEregsTri(dat,Tr,r,M,main="PE Proximity Regions with r=1.5",
xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,M,Ds)
xc<-txt[,1]+c(-.02,.03,.03,.03,.05,-0.03,-.01)
yc<-txt[,2]+c(.02,.02,.02,.07,.02,.05,-.06)
txt.str<-c("A","B","C","M","D1","D2","D3")
text(xc,yc,txt.str)
```

`print.Extrema` *Print a* `Extrema` *object*

Description

Prints the `call` of the object of class 'Extrema' and also the `type` (i.e. a brief description) of the extrema).

Usage

```
## S3 method for class 'Extrema'
print(x, ...)
```

Arguments

`x`	A `Extrema` object.
`...`	Additional arguments for the S3 method '`print`'.

Value

The `call` of the object of class 'Extrema' and also the `type` (i.e. a brief description) of the extrema).

See Also

`summary.Extrema`, `print.summary.Extrema`, and `plot.Extrema`

Examples

```
n<-20
dat<-runifTe(n)$gen.points
Ext<-cl2edgesTe(dat)
Ext
print(Ext)
```

print.Lines *Print a* Lines *object*

Description

Prints the call of the object of class 'Lines' and also the coefficients of the line (in the form: y = slope * x + intercept).

Usage

```
## S3 method for class 'Lines'
print(x, ...)
```

Arguments

x	A Lines object.
...	Additional arguments for the S3 method 'print'.

Value

The call of the object of class 'Lines' and the coefficients of the line (in the form: y = slope * x + intercept).

See Also

summary.Lines, print.summary.Lines, and plot.Lines

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75)
xr<-range(A,B);
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=100)

lnAB<-Line(A,B,x)
lnAB
print(lnAB)
```

print.Lines3D *Print a* Lines3D *object*

Description

Prints the call of the object of class 'Lines3D', the coefficients of the line (in the form: x=x0 + a*t, y=y0 + b*t, and z=z0 + c*t), and the initial point together with the direction vector.

Usage

```
## S3 method for class 'Lines3D'
print(x, ...)
```

Arguments

x	A Lines3D object.
...	Additional arguments for the S3 method 'print'.

Value

The call of the object of class 'Lines3D', the coefficients of the linee (in the form: x=x0 + a*t, y=y0 + b*t, and z=z0 + c*t), and the initial point together with the direction vector.

See Also

summary.Lines3D, print.summary.Lines3D, and plot.Lines3D

Examples

```
A<-c(1,10,3); B<-c(1,1,3);
vecs<-rbind(A,B)
Line3D(A,B,.1)
Line3D(A,B,.1,dir.vec=FALSE)

tr<-range(vecs);
tf<-(tr[2]-tr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=100)

lnAB3D<-Line3D(A,B,tsq)
lnAB3D
print(lnAB3D)
```

print.Patterns *Print a* Patterns *object*

Description

Prints the call of the object of class 'Patterns' and also the type (or description) of the pattern).

Usage

```
## S3 method for class 'Patterns'
print(x, ...)
```

Arguments

x	A Patterns object.
...	Additional arguments for the S3 method 'print'.

Value

The call of the object of class 'Patterns' and also the type (or description) of the pattern).

See Also

summary.Patterns, print.summary.Patterns, and plot.Patterns

Examples

```
nx<-20; #try also 100 and 1000
ny<-3; #try also 1
e<-.15;
Y<-cbind(runif(ny),runif(ny)) #with default bounding box (i.e., unit square)

Xdt<-rseg.disc(nx,Y,e)
Xdt
print(Xdt)
```

print.PCDs *Print a* PCDs *object*

Description

Prints the call of the object of class 'PCDs' and also the type (i.e. a brief description) of the proximity catch digraph (PCD)).

Usage

```
## S3 method for class 'PCDs'
print(x, ...)
```

Arguments

x	A PCDs object.
...	Additional arguments for the S3 method 'print'.

Value

The call of the object of class 'PCDs' and also the type (i.e. a brief description) of the proximity catch digraph (PCD)).

See Also

summary.PCDs, print.summary.PCDs, and plot.PCDs

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C); n<-10
dat<-runif.tri(n,Tr)$g
M<-as.numeric(runif.tri(1,Tr)$g)
Arcs<-ArcsAStri(dat,Tr,M)
Arcs
print(Arcs)
```

---

print.Planes *Print a* Planes *object*

Description

Prints the call of the object of class 'Planes' and also the coefficients of the plane (in the form: z = A*x + B*y + C).

Usage

```
## S3 method for class 'Planes'
print(x, ...)
```

Arguments

x	A Planes object.
...	Additional arguments for the S3 method 'print'.

Value

The call of the object of class 'Planes' and the coefficients of the plane (in the form: z = A*x + B*y + C).

See Also

summary.Planes, print.summary.Planes, and plot.Planes

Examples

```
A<-c(1,10,3); B<-c(1,1,3); C<-c(3,9,12)
pts<-rbind(A,B,C)

xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1 #how far to go at the lower and upper ends in the y-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=100)
y<-seq(yr[1]-yf,yr[2]+yf,l=100)

plABC<-Plane(A,B,C,x,y)
plABC
print(plABC)
```

---

print.summary.Extrema *Print a summary of a* Extrema *object*

Description

Prints some information about the object.

Usage

```
## S3 method for class 'summary.Extrema'
print(x, ...)
```

Arguments

x	object of class "summary.Extrema", generated by summary.Extrema.
...	Additional parameters for print.

Value

None

See Also

print.Extrema, summary.Extrema, and plot.Extrema

print.summary.Lines *Print a summary of a* Lines *object*

Description

Prints some information about the object.

Usage

```
## S3 method for class 'summary.Lines'
print(x, ...)
```

Arguments

x	object of class "summary.Lines", generated by summary.Lines.
...	Additional parameters for print.

Value

None

See Also

print.Lines, summary.Lines, and plot.Lines

print.summary.Lines3D *Print a summary of a* Lines3D *object*

Description

Prints some information about the object.

Usage

```
## S3 method for class 'summary.Lines3D'
print(x, ...)
```

Arguments

x	object of class "summary.Lines3D", generated by summary.Lines3D.
...	Additional parameters for print.

Value

None

See Also

print.Lines3D, summary.Lines3D, and plot.Lines3D

---

print.summary.Patterns *Print a summary of a* Patterns *object*

Description

Prints some information about the object.

Usage

```
## S3 method for class 'summary.Patterns'
print(x, ...)
```

Arguments

x	object of class "summary.Patterns", generated by summary.Patterns.
...	Additional parameters for print.

Value

None

See Also

print.Patterns, summary.Patterns, and plot.Patterns

---

print.summary.PCDs *Print a summary of a* PCDs *object*

Description

Prints some information about the object.

Usage

```
## S3 method for class 'summary.PCDs'
print(x, ...)
```

Arguments

x	object of class "summary.PCDs", generated by summary.PCDs.
...	Additional parameters for print.

Value

None

See Also

print.PCDs, summary.PCDs, and plot.PCDs

---

print.summary.Planes *Print a summary of a* Planes *object*

Description

Prints some information about the object.

Usage

```
## S3 method for class 'summary.Planes'
print(x, ...)
```

Arguments

x	object of class "summary.Planes", generated by summary.Planes.
...	Additional parameters for print.

Value

None

See Also

print.Planes, summary.Planes, and plot.Planes

---

print.summary.TriLines *Print a summary of a* TriLines *object*

Description

Prints some information about the object

Usage

```
## S3 method for class 'summary.TriLines'
print(x, ...)
```

Arguments

x	object of class "summary.TriLines", generated by summary.TriLines.
...	Additional parameters for print.

Value

None

See Also

print.TriLines, summary.TriLines, and plot.TriLines

---

print.summary.Uniform *Print a summary of a* Uniform *object*

Description

Prints some information about the object.

Usage

```
## S3 method for class 'summary.Uniform'
print(x, ...)
```

Arguments

x	object of class "summary.Uniform", generated by summary.Uniform.
...	Additional parameters for print.

Value

None

See Also

print.Uniform, summary.Uniform, and plot.Uniform

print.TriLines *Print a* TriLines *object*

Description

Prints the call of the object of class 'TriLines' and also the coefficients of the line (in the form: y = slope * x + intercept), and the vertices of the triangle with respect to which the line is defined.

Usage

```
## S3 method for class 'TriLines'
print(x, ...)
```

Arguments

x	A TriLines object.
...	Additional arguments for the S3 method 'print'.

Value

The call of the object of class 'TriLines' , the coefficients of the line (in the form: y = slope * x + intercept), and the vertices of the triangle with respect to which the line is defined.

See Also

summary.TriLines, print.summary.TriLines, and plot.TriLines

Examples

```
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.01)

lnACM<-lA_CM.Te(x)
lnACM
print(lnACM)

## End(Not run)
```

print.Uniform *Print a* Uniform *object*

Description

Prints the call of the object of class 'Uniform' and also the type (i.e. a brief description) of the uniform distribution).

Usage

```
## S3 method for class 'Uniform'
print(x, ...)
```

Arguments

x	A Uniform object.
...	Additional arguments for the S3 method 'print'.

Value

The call of the object of class 'Uniform' and also the type (i.e. a brief description) of the uniform distribution).

See Also

summary.Uniform, print.summary.Uniform, and plot.Uniform

Examples

```
n<-10 #try also 100
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)

Xdt<-runif.tri(n,Tr)
Xdt
print(Xdt)
```

radii *The radii of points from one class with respect to points from the other class*

Description

Returns the radii of the balls centered at x points where radius of an x point equals to the minimum distance to y points (i.e. distance to the closest y point). That is, for each x point $radius = \min_{y \in Y}(d(x, y))$. x and y points must be of the same dimension.

Usage

```
radii(x, y)
```

Arguments

x	A set of d-dimensional points for which the radii are computed. Radius of an x point equals to the distance to the closest y point.
y	A set of d-dimensional points representing the reference points for the balls. That is, radius of an x point is defined as the minimum distance to the y points.

Value

A list with three elements

rad	A vector whose entries are the radius values for the x points. Radius of an x point equals to the distance to the closest y point
index.of.clYp	A vector of indices of the closest y points to the x points. The i-th entry in this vector is the index of the closest y point to i-th x point.
closest.Yp	A vector of the closest y points to the x points. The i-th entry in this vector or i-th row in the matrix is the closest y point to i-th x point.

See Also

radius

Examples

```
nx<-10
ny<-5
X<-cbind(runif(nx),runif(nx))
Y<-cbind(runif(ny),runif(ny))
Rad<-radii(X,Y)
Rad
rd<-Rad$rad

Xlim<-range(X[,1]-rd,X[,1]+rd,Y[,1])
Ylim<-range(X[,2]-rd,X[,2]+rd,Y[,2])
```

```
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(rbind(Y),asp=1,pch=16,col=2,xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(rbind(X))
interp::circles(X[,1],X[,2],Rad$rad,lty=1,lwd=1,col=4)

nx<-5
ny<-1
X<-cbind(runif(nx),runif(nx))
Y<-matrix(c(runif(ny),runif(ny)),ncol=2)
Rad<-radii(X,Y)
Rad
radii(Y,X)

rd<-Rad$rad

Xlim<-range(X[,1]-rd,X[,1]+rd,Y[,1])
Ylim<-range(X[,2]-rd,X[,2]+rd,Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(rbind(Y),asp=1,pch=16,col=2,xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(rbind(X))
interp::circles(X[,1],X[,2],Rad$rad,lty=1,lwd=1,col=4)

radii(c(1,2),c(2,3))

## Not run:
nx<-10
ny<-5
X<-runif(nx)
Y<-runif(ny)
radii(X,Y)  #this does not work, as X and Y are treated as vectors (i.e. points)

## End(Not run)

nx<-10
ny<-5
X<-as.matrix(X)
Y<-as.matrix(Y)
radii(X,Y)  #this works as X and Y are treated as 1D data sets

nx<-10
ny<-5
X<-cbind(runif(nx),runif(nx),runif(nx))
Y<-cbind(runif(ny),runif(ny),runif(ny))
radii(X,Y)

dat.fr<-data.frame(a=X)
radii(dat.fr,Y)
```

```
radii(Y,dat.fr)
```

`radius` *The radius of a point from one class with respect to points from the other class*

Description

Returns the radius for the ball centered at point `pt` with radius=min distance to `Y` points. That is, for the point `pt` $radius = \min_{y \in Y} d(pt, y)$ (i.e. distance from `pt` to the closest `Y` point). The point `pt` and `Y` points must be of same dimension.

Usage

```
radius(pt, Y)
```

Arguments

`pt`	A d-dimensional point for which radius is computed. Radius of `pt` equals to the distance to the closest `Y` point to `pt`.
`Y`	A set of d-dimensional points representing the reference points for the balls. That is, radius of the point `pt` is defined as the minimum distance to the `Y` points.

Value

A list with three elements

`rad`	Radius value for the point, `pt` defined as $\min_{yinY} d(pt, y)$
`index.of.clYpnt`	Index of the closest `Y` points to the point `pt`
`closest.Ypnt`	The closest `Y` point to the point `pt`

See Also

radii

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);

ny<-10
Y<-cbind(runif(ny),runif(ny))
radius(A,Y)
radius(B,Y)
radius(C,Y)

radius(B,C)
```

```
nx<-10
X<-cbind(runif(nx),runif(nx))
rad<-rep(0,nx)
for (i in 1:nx)
rad[i]<-radius(X[i,],Y)$rad

Xlim<-range(X[,1]-rad,X[,1]+rad,Y[,1])
Ylim<-range(X[,2]-rad,X[,2]+rad,Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(rbind(Y),asp=1,pch=16,col=2,xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(rbind(X))
interp::circles(X[,1],X[,2],rad,lty=1,lwd=1,col=4)

dat.fr<-data.frame(a=Y)
radii(A,dat.fr)
```

`rasc.disc` *Generation of points associated (in a radial or circular fashion) with a given set of points*

Description

An object of class "Patterns". Generates `n` 2D points uniformly in $(a1 - e, a1 + e)x(a1 - e, a1 + e) \cap U_i B(y_i, e)$ where $Y = (y_1, y_2, ..., y_{n_y})$ with n_y being number of `Y` points for various values of `e` under the association pattern and $B(y_i, e)$ is the ball centered at y_i with radius `e`.

`e` must be positive and very large values of `e` provide patterns close to CSR. `a1` is defaulted to the minimum of the x-coordinates of the `Y` points, `a2` is defaulted to the maximum of the x-coordinates of the `Y` points, `b1` is defaulted to the minimum of the y-coordinates of the `Y` points, `b2` is defaulted to the maximum of the y-coordinates of the `Y` points. This function is also very similar to `rasc.matern`, where `rasc.disc` needs the study window to be specified, while `rasc.matern` does not.

Usage

```
rasc.disc(
  n,
  Y,
  e,
  a1 = min(Y[, 1]),
  a2 = max(Y[, 1]),
  b1 = min(Y[, 2]),
  b2 = max(Y[, 2])
)
```

Arguments

n	A positive integer representing the number of points to be generated.
Y	A set of 2D points representing the reference points. The generated points are associated (in a circular/radial fashion) with these points.
e	A positive real number representing the radius of the balls centered at Y points. Only these balls are allowed for the generated points (i.e., generated points would be in the union of these balls).
a1, a2	Real numbers representing the range of x-coordinates in the region (default is the range of x-coordinates of the Y points).
b1, b2	Real numbers representing the range of y-coordinates in the region (default is the range of y-coordinates of the Y points).

Value

A list with the elements

type	The type of the point pattern
mtitle	The "main" title for the plot of the point pattern
parameters	Radial attraction parameter of the association pattern
ref.points	The input set of attraction points Y, i.e., points with which generated points are associated.
gen.points	The output set of generated points associated with Y points
tri.Y	Logical output for triangulation based on Y points should be implemented or not. if TRUE triangulation based on Y points is to be implemented (default is set to FALSE).
desc.pat	Description of the point pattern
num.points	The vector of two numbers, which are the number of generated points and the number of attraction (i.e., Y) points.
xlimit,ylimit	The possible range of the x- and y-coordinates of the generated points.

See Also

rseg.disc, rascTe, rascIITe, rasc.matern, and rascMT

Examples

```
nx<-20; ny<-4;  #try also nx<-1000; ny<-10;

e<-.15;
#with default bounding box (i.e., unit square)
Y<-cbind(runif(ny),runif(ny))

Xdat<-rasc.disc(nx,Y,e)
Xdat
summary(Xdat)
plot(Xdat,asp=1)
```

```
Xdt<-rasc.disc(nx,Y,e)$gen.points
Xlim<-range(Xdt[,1],Y[,1]);
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,xlab="x",ylab="y",main="association of two classes",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),pch=16,col=2,lwd=2)
points(Xdt)

e<-.25; #pattern is very close to CSR!
#try also e<-.1;

#with default bounding box (i.e., unit square)
Y<-cbind(runif(ny),runif(ny))
Xdt<-rasc.disc(nx,Y,e)$gen.points

Xlim<-range(Xdt[,1],Y[,1]);
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,xlab="x",ylab="y",main="association of two classes",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),pch=16,col=2,lwd=2)
points(Xdt)

#with a rectangular bounding box
a1<-0; a2<-10;
b1<-0; b2<-5;
e<-1.1;  #try also e<-5; #pattern very close to CSR!

Y<-cbind(runif(ny,a1,a2),runif(ny,b1,b2))
#try also Y<-cbind(runif(ny,a1,a2/2),runif(ny,b1,b2/2))

Xdt<-rasc.disc(nx,Y,e,a1,a2,b1,b2)$gen.points
Xlim<-range(Xdt[,1],Y[,1]);
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,xlab="x",ylab="y",main="association of two classes",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),pch=16,col=2,lwd=2)
points(Xdt)

e<-.15
Y<-data.frame(yp=cbind(runif(ny),runif(ny)) )
Xdt<-rasc.disc(nx,Y,e)
```

`rasc.matern` *Generation of points associated (in a Matern-like fashion) to a given set of points*

Description

An object of class "Patterns". Generates `n` 2D points uniformly in $\cup B(y_i, e)$ where $Y = (y_1, y_2, ..., y_{n_y})$ with n_y being number of `Y` points for various values of `e` under the association pattern and $B(y_i, e)$ is the ball centered at y_i with radius `e`.

The pattern resembles the Matern cluster pattern (see `rMatClust` in the `spatstat` package for further information (Baddeley and Turner (2005)). `rMatClust(kappa,scale,mu,win)` in the simplest case generates a uniform Poisson point process of "parent" points with intensity `kappa`. Then each parent point is replaced by a random cluster of "offspring" points, the number of points per cluster being Poisson(`mu`) distributed, and their positions being placed and uniformly inside a disc of radius `scale` centred on the parent point. The resulting point pattern is a realisation of the classical "stationary Matern cluster process" generated inside the window `win`.

The main difference of `rasc.matern` and `rMatClust` is that the parent points are `Y` points which are given beforehand and we do not discard them in the end in `rasc.matern` and the offspring points are the points associated with the reference points, `Y`; `e` must be positive and very large values of `e` provide patterns close to CSR.

This function is also very similar to `rasc.disc`, where `rasc.disc` needs the study window to be specified, while `rasc.matern` does not.

Usage

```
rasc.matern(n, Y, e)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`Y`	A set of 2D points representing the reference points. The generated points are associated (in a Matern-cluster like fashion) with these points.
`e`	A positive real number representing the radius of the balls centered at `Y` points. Only these balls are allowed for the generated points (i.e., generated points would be in the union of these balls).

Value

A list with the elements

`type`	The type of the point pattern
`mtitle`	The "main" title for the plot of the point pattern
`parameters`	Radial (i.e. circular) attraction parameter of the association pattern.
`ref.points`	The input set of attraction points `Y`, i.e., points with which generated points are associated.
`gen.points`	The output set of generated points associated with `Y` points.

tri.Y	Logical output for triangulation based on Y points should be implemented or not. if TRUE triangulation based on Y points is to be implemented (default is set to FALSE).
desc.pat	Description of the point pattern
num.points	The vector of two numbers, which are the number of generated points and the number of attraction (i.e., Y) points.
xlimit,ylimit	The possible ranges of the x- and y-coordinates of the generated points.

References

Baddeley AJ, Turner R (2005). "spatstat: An R Package for Analyzing Spatial Point Patterns." *Journal of Statistical Software*, **12(6)**, 1-42.

See Also

rasc.disc, rascTe, rascIITe, rascMT, rseg.disc, and rMatClust in the spatstat package

Examples

```
nx<-20; ny<-4;  #try also nx<-1000; ny<-10;

e<-.15;  #try also e<-1.1; #closer to CSR than association, as e is large

#Y points uniform in unit square
Y<-cbind(runif(ny),runif(ny))

Xdt<-rasc.matern(nx,Y,e)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xdt<-rasc.matern(nx,Y,e)$gen.points
Xlim<-range(Xdt[,1],Y[,1]);
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,xlab="x",ylab="y",main="Association of two classes",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),pch=16,col=2,lwd=2)
points(Xdt)

a1<-0; a2<-10;
b1<-0; b2<-5;
e<-1.1;

#Y points uniform in a rectangle
Y<-cbind(runif(ny,a1,a2),runif(ny,b1,b2))
#try also Y<-cbind(runif(ny,a1,a2/2),runif(ny,b1,b2/2))

Xdt<-rasc.matern(nx,Y,e)$gen.points
Xlim<-range(Xdt[,1],Y[,1]);
```

```
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,xlab="x",ylab="y",main="association of two classes",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),pch=16,col=2,lwd=2)
points(Xdt)

e<-.15
Y<-data.frame(yp=cbind(runif(ny),runif(ny)) )
Xdt<-rasc.matern(nx,Y,e)
```

`rasc.tri` *Generation of points associated (in a Type I fashion) with the vertices of a triangle*

Description

An object of class "Patterns". Generates `k` points uniformly in the support for Type I association in a given triangle, `tri`. `delta` is the parameter of association (that is, only $\delta 100$ % area around each vertex in the triangle is allowed for point generation). `delta` corresponds to `eps` in the standard equilateral triangle T_e as $delta = 4eps^2/3$ (see `rsegTe` function).

See (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)) for more on the association pattern.

Usage

```
rasc.tri(k, tri, delta)
```

Arguments

`k`	A positive integer representing the number of points to be generated from the association pattern in the triangle, `tri`.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`delta`	A positive real number in (0,4/9). `delta` is the parameter of association (that is, only $\delta 100$ % area around each vertex in the triangle is allowed for point generation).

Value

A list with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The "main" title for the plot of the point pattern
`parameters`	Attraction parameter, `delta`, of the Type I association pattern. `delta` is in (0,4/9) only $\delta 100$ % of the area around each vertex in the triangle `tri` is allowed for point generation.

ref.points	The input set of points, i.e., vertices of tri; reference points, i.e., points with which generated points are associated.
gen.points	The output set of generated points associated with the vertices of tri.
tri.Y	Logical output, TRUE if triangulation based on Yp points should be implemented.
desc.pat	Description of the point pattern
num.points	The vector of two numbers, which are the number of generated points and the number of reference (i.e., Yp) points.
xlimit,ylimit	The ranges of the x- and y-coordinates of the reference points, which are the Yp points

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

rseg.tri, rascTe, rascIITe, and rascMT

Examples

```
n<-10  #try also n<-100
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)
del<-.4

Xdt<-rasc.tri(n,Tr,del)
Xdt
summary(Xdt)
plot(Xdt)

dat<-rasc.tri(n,Tr,del)$g
Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat)
xc<-Tr[,1]+c(-.01,.01,.01)
yc<-Tr[,2]+c(.02,.02,.02)
```

```
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

dat.fr<-data.frame(a=Tr)
rasc.tri(n,dat.fr,del)
```

`rascIITe` *Generation of points associated (in a Type II fashion) with the edges of* T_e

---

Description

An object of class "Patterns". Generates `k` points uniformly in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ under the type II association alternative for `eps` in $(0, \sqrt{3}/6 = 0.2886751]$.

In the type II association, the annular allowed regions around the edges are determined by the parameter `eps` where $\sqrt{3}/6 - eps$ is the distance from the interior triangle (i.e., forbidden region for association) to T_e (see examples for a sample plot.)

Usage

```
rascIITe(k, eps)
```

Arguments

`k`	A positive integer representing the number of points to be generated.
`eps`	A positive real number representing the parameter of type II association (where $\sqrt{3}/6 - eps$ is the distance from the interior triangle distance from the interior triangle to T_e).

Value

A list with the elements

`type`	The type of the point pattern
`mtitle`	The "main" title for the plot of the point pattern
`parameters`	The attraction parameter, `eps`, of the association pattern, where $\sqrt{3}/6 - eps$ is the distance from the interior triangle to T_e
`ref.points`	The input set of points `Y`; reference points, i.e., points with which generated points are associated (i.e., vertices of T_e).
`gen.points`	The output set of generated points associated with `Y` points (i.e., edges of T_e).
`tri.Y`	Logical output for triangulation based on `Y` points should be implemented or not. if `TRUE` triangulation based on `Y` points is to be implemented (default is set to `FALSE`).
`desc.pat`	Description of the point pattern

num.points	The vector of two numbers, which are the number of generated points and the number of reference (i.e., Y) points, which is 3 here.
xlimit,ylimit	The ranges of the x- and y-coordinates of the reference points, which are the vertices of T_e here

See Also

rseg.disc, rasc.disc, rsegIITe, and rsegMT

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10  #try also n<-20 or n<-100 or 1000
eps<-.2  #try also .25, .1

set.seed(1)
Xdt<-rascIITe(n,eps)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

dat<-rascIITe(n,eps)$gen.points
plot(Te,pch=".",xlab="",ylab="",
main="Type II association in the \n standard equilateral triangle",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(dat)

#The support for the Type II association alternative
A1<-c(1/2-eps*sqrt(3),sqrt(3)/6-eps); B1<-c(1/2+eps*sqrt(3),sqrt(3)/6-eps);
C1<-c(1/2,sqrt(3)/6+2*eps);
supp<-rbind(A1,B1,C1)

plot(Te,asp=1,pch=".",xlab="",ylab="",main="Support of the Type II Association",
xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te,col=5)
polygon(supp,col=0)
points(dat)
```

---

rascMT *Generation of points associated (in a Type I fashion) with a given set of points*

Description

An object of class "Patterns". Generates n points uniformly in the support for Type I association in the convex hull of set of points, Yp. delta is the parameter of association (that is, only $\delta 100$ % area around each vertex in each Delaunay triangle is allowed for point generation).

delta corresponds to eps in the standard equilateral triangle T_e as $delta = 4eps^2/3$ (see rsegTe function).

If Yp consists only of 3 points, then the function behaves like the function rasc.tri.

DTmesh must be the Delaunay triangulation of Yp and DTr must be the corresponding Delaunay triangles (both DTmesh and DTr are NULL by default). If NULL, DTmesh is computed via tri.mesh and DTr is computed via triangles function in interp package.

tri.mesh function yields the triangulation nodes with their neigbours, and creates a triangulation object, and triangles function yields a triangulation data structure from the triangulation object created by tri.mesh (the first three colums are the vertex indices of the Delaunay triangles).

See (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)) for more on the association pattern. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
rascMT(n, Yp, delta, DTmesh = NULL, DTr = NULL)
```

Arguments

n	A positive integer representing the number of points to be generated.
Yp	A set of 2D points from which Delaunay triangulation is constructed.
delta	A positive real number in (0,4/9). delta is the parameter of association (that is, only $\delta 100$ % area around each vertex in each Delaunay triangle is allowed for point generation).
DTmesh	Delaunay triangulation of Yp, default is NULL, which is computed via tri.mesh function in interp package. tri.mesh function yields the triangulation nodes with their neigbours, and creates a triangulation object.
DTr	Delaunay triangles based on Yp, default is NULL, which is computed via tri.mesh function in interp package. triangles function yields a triangulation data structure from the triangulation object created by tri.mesh.

Value

A list with the elements

type	The type of the pattern from which points are to be generated
mtitle	The "main" title for the plot of the point pattern
parameters	Attraction parameter, delta, of the Type I association pattern. delta is in (0,4/9) only $\delta 100$ % of the area around each vertex in each Delaunay triangle is allowed for point generation.
ref.points	The input set of points Yp; reference points, i.e., points with which generated points are associated.

gen.points	The output set of generated points associated with Yp points.
tri.Y	Logical output, TRUE if triangulation based on Yp points should be implemented.
desc.pat	Description of the point pattern
num.points	The vector of two numbers, which are the number of generated points and the number of reference (i.e., Yp) points.
xlimit,ylimit	The ranges of the x- and y-coordinates of the reference points, which are the Yp points

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

rasc.disc, rascTe, rascIITe, and rsegMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Yp<-cbind(runif(ny),runif(ny))
del<-.4

Xdt<-rascMT(nx,Yp,del)
Xdt
summary(Xdt)
plot(Xdt)

Yp<-cbind(runif(ny),runif(ny))
del<-.3 #try .5, .75, .85
```

```
dat<-rascMT(nx,Yp,del)  #data under CSR in the convex hull of Ypoints

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points
dat<-rascMT(nx,Yp,del,DTY)  #data under CSR in the convex hull of Ypoints

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points
TRY<-interp::triangles(DTY)[,1:3];
dat<-rascMT(nx,Yp,del,DTr=TRY)  #data under CSR in the convex hull of Ypoints

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points
TRY<-interp::triangles(DTY)[,1:3];
dat<-rascMT(nx,Yp,del,DTY,TRY)$g #data under CSR in the convex hull of Ypoints

Xlim<-range(Yp[,1])
Ylim<-range(Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

#plot of the data in the convex hull of Y points together with the Delaunay triangulation
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points

plot(dat,main=" ", xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),type="n")
interp::plot.triSht(DTY, add=TRUE, do.points=TRUE,col="blue")
points(dat,pch=".",cex=3)

Yp<-rbind(c(.3,.2),c(.4,.5),c(.14,.15))
rascMT(nx,Yp,del)

dat.fr<-data.frame(a=Yp)
rascMT(nx,dat.fr,del)
```

`rascTe` *Generation of points associated (in a Type I fashion) with the vertices of T_e*

Description

An object of class "Patterns". Generates `k` points uniformly in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ under the type I association alternative for `eps` in $(0, \sqrt{3}/3 = 0.5773503]$. The allowed triangular regions around the vertices are determined by the parameter `eps`.

In the type I association, the triangular support regions around the vertices are determined by the parameter `eps` where $\sqrt{3}/3 - eps$ serves as the height of these triangles (see examples for a sample plot.)

See also (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)).

Usage

```
rascTe(k, eps)
```

Arguments

k	A positive integer representing the number of points to be generated.
eps	A positive real number representing the parameter of type I association (where $\sqrt{3}/3 - eps$ serves as the height of the triangular support regions around the vertices).

Value

A list with the elements

type	The type of the point pattern
mtitle	The "main" title for the plot of the point pattern
parameters	The attraction parameter of the association pattern, eps, where $\sqrt{3}/3 - eps$ serves as the height of the triangular support regions around the vertices
ref.points	The input set of points Y; reference points, i.e., points with which generated points are associated (i.e., vertices of T_e).
gen.points	The output set of generated points associated with Y points (i.e., vertices of T_e).
tri.Y	Logical output for triangulation based on Y points should be implemented or not. if TRUE triangulation based on Y points is to be implemented (default is set to FALSE).
desc.pat	Description of the point pattern.
num.points	The vector of two numbers, which are the number of generated points and the number of reference (i.e., Y) points.
xlimit,ylimit	The ranges of the x- and y-coordinates of the reference points, which are the vertices of T_e here

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

rseg.disc, rasc.disc, rsegIITe, and rsegMT

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10  #try also n<-20 or n<-100 or 1000
eps<-.25  #try also .15, .5, .75

set.seed(1)
Xdt<-rascTe(n,eps)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

dat<-rascTe(n,eps)$gen.points
plot(Te,pch=".",xlab="",ylab="",
main="Type I association in the \n standard equilateral triangle",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(dat)

#The support for the Type I association alternative
sr<-(sqrt(3)/3-eps)/(sqrt(3)/2)
C1<-C+sr*(A-C); C2<-C+sr*(B-C)
A1<-A+sr*(B-A); A2<-A+sr*(C-A)
B1<-B+sr*(A-B); B2<-B+sr*(C-B)
supp<-rbind(A1,B1,B2,C2,C1,A2)

plot(Te,asp=1,pch=".",xlab="",ylab="",
main="Support of the Type I Association",
xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
if (sr<=.5)
{
  polygon(Te,col=5)
  polygon(supp,col=0)
} else
{
  polygon(Te,col=0,lwd=2.5)
  polygon(rbind(A,A1,A2),col=5,border=NA)
  polygon(rbind(B,B1,B2),col=5,border=NA)
  polygon(rbind(C,C1,C2),col=5,border=NA)
}
points(dat)
```

`re.bastri.cent` *The index of the edge region in a basic triangle that contains a point*

Description

Returns the index of the edge whose region contains point, pt, in the basic triangle $T_b = T(A = (0,0), B = (1,0), C = (c_1, c_2))$ and edge regions based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the basic triangle T_b.

Edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. If the point, pt, is not inside tri, then the function yields NA as output. Edge region 1 is the triangle $T(B, C, M)$, edge region 2 is $T(A, C, M)$, and edge region 3 is $T(A, B, M)$. In the basic triangle T_b c1 is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
re.bastri.cent(pt, c1, c2, M)
```

Arguments

pt	A 2D point for which M-edge region it resides in is to be determined in the basic triangle T_b.
c1, c2	Positive real numbers which constitute the upper vertex of the basic triangle (i.e., the vertex adjacent to the shorter edges of T_b); c1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the basic triangle T_b.

Value

A list with three elements

re	Index of the M-edge region that contains point, pt in the basic triangle T_b.
tri	The vertices of the triangle, where row labels are A, B, and C with edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.
desc	Description of the edge labels

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**,

299-334.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

re.triCM, re.tri.cent, re.bastri.cent, reTeCM, and redge.triCM

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);
M<-c(.6,.2)

P<-c(.4,.2)
re.bastri.cent(P,c1,c2,M)

P<-c(1.4,.2)
re.bastri.cent(P,c1,c2,M)

c1<-.5; c2<-.8
P<-c(.4,.2)
re.bastri.cent(P,c1,c2,M)

P<-c(.8,.2)
re.bastri.cent(P,c1,c2,M)

c1<-.4; c2<-.6
A<-c(0,0);B<-c(1,0);C<-c(c1,c2);
Tb<-rbind(A,B,C)

re.bastri.cent(A,c1,c2,M)
re.bastri.cent(B,c1,c2,M)
re.bastri.cent(C,c1,c2,M)
re.bastri.cent(M,c1,c2,M)

n<-10  #try also n<-20
dat<-runif.bastri(n,c1,c2)$g

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.2)

re<-vector()
for (i in 1:n)
  re<-c(re,re.bastri.cent(dat[i,],c1,c2,M)$re)
re

Xlim<-range(Tb[,1],dat[,1])
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
```

```
plot(Tb,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(dat,pch=".")
polygon(Tb)
L<-Tb; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
text(dat,labels=factor(re))

txt<-rbind(Tb,M)
xc<-txt[,1]+c(-.03,.03,.02,0)
yc<-txt[,2]+c(.02,.02,.02,-.03)
txt.str<-c("A","B","C","M")
text(xc,yc,txt.str)
```

`re.bastriCM` *The index of the CM-edge region in a basic triangle that contains a point*

Description

Returns the index of the edge whose region contains point, `pt`, in the basic triangle $T_b = T(A = (0,0), B = (1,0), C = (c_1, c_2)$ where `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$ with edge regions based on center of mass $CM = (A + B + C)/3$.

Edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. If the point, `pt`, is not inside `tri`, then the function yields `NA` as output. Edge region 1 is the triangle $T(B, C, CM)$, edge region 2 is $T(A, C, CM)$, and edge region 3 is $T(A, B, CM)$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
re.bastriCM(pt, c1, c2)
```

Arguments

`pt`	A 2D point for which CM-edge region it resides in is to be determined in the basic triangle T_b.
`c1, c2`	Positive real numbers which constitute the upper vertex of the basic triangle (i.e., the vertex adjacent to the shorter edges of T_b); `c1` must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Value

A list with three elements

`re`	Index of the CM-edge region that contains point, `pt` in the basic triangle T_b
`tri`	The vertices of the triangle, where row labels are $A = (0, 0)$, $B = (1, 0)$, and $C = (c_1, c_2)$ with edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.
`desc`	Description of the edge labels

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`re.triCM`, `re.tri.cent`, `re.bastri.cent`, `reTeCM`, and `redge.triCM`

Examples

```
c1<-.4; c2<-.6
P<-c(.4,.2)
re.bastriCM(P,c1,c2)

c1<-.5; c2<-.8
P<-c(.4,.2)
re.bastriCM(P,c1,c2)

P<-c(.8,.2)
re.bastriCM(P,c1,c2)

P<-c(1.8,.2)
re.bastriCM(P,c1,c2)

c1<-.4; c2<-.6
A<-c(0,0);B<-c(1,0);C<-c(c1,c2);
Tb<-rbind(A,B,C)
CM<-(A+B+C)/3
```

```
re.bastriCM(A,c1,c2)
re.bastriCM(B,c1,c2)
re.bastriCM(C,c1,c2)
re.bastriCM(CM,c1,c2)

n<-10  #try also n<-20
dat<-runif.bastri(n,c1,c2)$g

re<-vector()
for (i in 1:n)
  re<-c(re,re.bastriCM(dat[i,],c1,c2)$re)
re

Xlim<-range(Tb[,1],dat[,1])
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(dat,pch=".")
polygon(Tb)
L<-Tb; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
text(dat,labels=factor(re))

txt<-rbind(Tb,CM)
xc<-txt[,1]+c(-.03,.03,.02,0)
yc<-txt[,2]+c(.02,.02,.02,-.04)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)
```

`re.tri.cent` *The index of the edge region in a triangle that contains the point*

Description

Returns the index of the edge whose region contains point, `pt`, in the triangle $tri = T(A, B, C)$ with edge regions based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`.

Edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. If the point, `pt`, is not inside `tri`, then the function yields `NA` as output. Edge region 1 is the triangle $T(B, C, M)$, edge region 2 is $T(A, C, M)$, and edge region 3 is $T(A, B, M)$.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
re.tri.cent(pt, tri, M)
```

Arguments

pt	A 2D point for which M-edge region it resides in is to be determined in the triangle tri.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri.

Value

A list with three elements

re	Index of the M-edge region that contains point, pt in the triangle tri.
tri	The vertices of the triangle, where row labels are A, B, and C with edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.
desc	Description of the edge labels

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

re.triCM, re.bastriCM, re.bastri.cent, reTeCM, and redge.triCM

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(1.4,1.2)
M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

re.tri.cent(P,Tr,M)

P<-c(.8,.2)
re.tri.cent(P,Tr,M)
```

```
P<-c(1.5,1.61)
re.tri.cent(P,Tr,M)

re.tri.cent(A,Tr,M)
re.tri.cent(B,Tr,M)
re.tri.cent(C,Tr,M)
re.tri.cent(M,Tr,M)

n<-10  #try also n<-20
dat<-runif.tri(n,Tr)$g

re<-vector()
for (i in 1:n)
  re<-c(re,re.tri.cent(dat[i,],Tr,M)$re)
re

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}

plot(Tr,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".")
L<-Tr; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
text(dat,labels=factor(re))

txt<-rbind(Tr,M)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("A","B","C","M")
text(xc,yc,txt.str)

p1<-(A+B+M)/3
p2<-(B+C+M)/3
p3<-(A+C+M)/3

plot(Tr,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-Tr; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,M,p1,p2,p3)
xc<-txt[,1]+c(-.02,.02,.02,.02)
yc<-txt[,2]+c(.02,.02,.04,.05)
txt.str<-c("A","B","C","M","re=3","re=1","re=2")
text(xc,yc,txt.str)

dat.fr<-data.frame(a=Tr)
```

```
re.tri.cent(P,dat.fr,M)
```

`re.triCM` *The index of the CM-edge region in a triangle that contains the point*

Description

Returns the index of the edge whose region contains point, `pt`, in the triangle $tri = T(A, B, C)$ with edge regions based on center of mass $CM = (A + B + C)/3$.

Edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. If the point, `pt`, is not inside `tri`, then the function yields `NA` as output. Edge region 1 is the triangle $T(B, C, CM)$, edge region 2 is $T(A, C, CM)$, and edge region 3 is $T(A, B, CM)$.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
re.triCM(pt, tri)
```

Arguments

`pt`	A 2D point for which CM-edge region it resides in is to be determined in the triangle `tri`.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

A list with three elements

`re`	Index of the CM-edge region that contains point, `pt` in the triangle `tri`.
`tri`	The vertices of the triangle, where row labels are A, B, and C with edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.
`desc`	Description of the edge labels

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**,

299-334.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

re.tri.cent, re.bastriCM, re.bastri.cent, reTeCM, and redge.triCM

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
P<-c(1.4,1.2)
re.triCM(P,Tr)

P<-c(.8,.2)
re.triCM(P,Tr)

P<-c(1.5,1.61)
re.triCM(P,Tr)

CM<-(A+B+C)/3

re.triCM(A,Tr)
re.triCM(B,Tr)
re.triCM(C,Tr)
re.triCM(CM,Tr)

n<-10  #try also n<-20
dat<-runif.tri(n,Tr)$g

re<-vector()
for (i in 1:n)
  re<-c(re,re.triCM(dat[i,],Tr)$re)
re

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(dat,pch=".")
polygon(Tr)
L<-Tr; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
text(dat,labels=factor(re))

txt<-rbind(Tr,CM)
xc<-txt[,1]
yc<-txt[,2]
```

```
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)

p1<-(A+B+CM)/3
p2<-(B+C+CM)/3
p3<-(A+C+CM)/3

plot(Tr,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-Tr; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,CM,p1,p2,p3)
xc<-txt[,1]+c(-.02,.02,.02,.02)
yc<-txt[,2]+c(.02,.02,.04,.05)
txt.str<-c("A","B","C","CM","re=3","re=1","re=2")
text(xc,yc,txt.str)

A<-c(0,0); B<-c(1,0); C<-c(0.5,.8);
Tr<-rbind(A,B,C);
P<-c(.4,.2)
re.triCM(P,Tr)

dat.fr<-data.frame(a=Tr)
re.triCM(P,dat.fr)
```

`redge.triCM` *The vertices of the CM-edge region in a triangle that contains the point*

Description

Returns the edge whose region contains point, `pt`, in the triangle $tri = T(A, B, C)$ with edge regions based on center of mass $CM = (A + B + C)/3$.

This function is related to `re.triCM`, but unlike `re.triCM` the related edges are given as vertices ABC for $re = 3$, as BCA for $re = 1$ and as CAB for $re = 2$ where edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. The vertices are given one vertex in each row in the output, e.g., ABC is printed as $rbind(A, B, C)$, where row 1 has the entries of vertex A, row 2 has the entries of vertex B, and row 3 has the entries of vertex C.

If the point, `pt`, is not inside `tri`, then the function yields `NA` as output.

Edge region for BCA is the triangle $T(B, C, CM)$, edge region CAB is $T(A, C, CM)$, and edge region ABC is $T(A, B, CM)$.

See also (Ceyhan (2005, 2010)).

Usage

```
redge.triCM(pt, tri)
```

Arguments

pt	A 2D point for which CM-edge region it resides in is to be determined in the triangle tri.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

The CM-edge region that contains point, pt in the triangle tri. The related edges are given as vertices ABC for $re = 3$, as BCA for $re = 1$ and as CAB for $re = 2$ where edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

re.tri.cent, re.triCM, re.bastriCM, re.bastri.cent, reTeCM, and redge.triCM

Examples

```
## Not run:
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(.4,.2)  #try also P<-as.numeric(runif.tri(1,Tr)$g)
redge.triCM(P,Tr)

P<-c(.8,.2)
redge.triCM(P,Tr)

P<-c(.5,.61)
redge.triCM(P,Tr)

CM<-(A+B+C)/3
p1<-(A+B+CM)/3
p2<-(B+C+CM)/3
p3<-(A+C+CM)/3

Xlim<-range(Tr[,1])
```

```
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-Tr; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,CM,p1,p2,p3)
xc<-txt[,1]+c(-.02,.02,.02,.05,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,.02,0,0,0)
txt.str<-c("A","B","C","CM","re=y1y2y3","re=y2y3y1","re=y3y1y2")
text(xc,yc,txt.str)

dat.fr<-data.frame(a=Tr)
redge.triCM(P,dat.fr)

## End(Not run)
```

`redges.tri.cent` *The indices of the* `M`*-edge regions in a triangle that contains the points in a give data set*

Description

Returns the indices of the edges whose regions contain the points in data set `Dt` in a triangle $tri = T(A, B, C)$ and edge regions are based on the center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri` (see the plots in the example for illustrations).

The vertices of the triangle `tri` are labeled as 1=A,2=B, and 3=C also according to the row number the vertex is recorded in `tri` and the corresponding edges are 1=BC, 2=AC, and 3=AB.

If a point in `Dt` is not inside `tri`, then the function yields `NA` as output for that entry. The corresponding edge region is the polygon with the vertex, `M`, and vertices other than the non-adjacent vertex, i.e., edge region 1 is the triangle $T(B, M, C)$, edge region 2 is $T(A, M, C)$ and edge region 3 is $T(A, B, M)$.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
redges.tri.cent(Dt, tri, M)
```

Arguments

`Dt`	A set of 2D points representing the set of data points for which indices of the edge regions containing them are to be determined.

`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.

Value

A list with the elements

`re`	Indices (i.e., a vector of indices) of the edges whose region contains points in `Dt` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index opposite to edge whose index is given in re.
`desc`	Description of the edge labels as "Edge labels are AB=1, BC=2, and AC=3".

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

`redges.triCM`, `rverts.tri.cent` and `rverts.tri.nd`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

M<-c(1.6,1.2)

P<-c(.4,.2)
redges.tri.cent(P,Tr,M)

P<-c(1.8,.5)
redges.tri.cent(P,Tr,M)

P<-c(10.5,1.6)
redges.tri.cent(P,Tr,M)
```

```
n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.2)

re<-redges.tri.cent(dat,Tr,M)
re

D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".",col=1)
L<-Tr; R<-rbind(M,M,M)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

xc<-Tr[,1]+c(-.02,.03,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(.05,.06,-.05,-.02)
yc<-txt[,2]+c(.03,.03,.05,-.08)
txt.str<-c("M","re=2","re=3","re=1")
text(xc,yc,txt.str)
text(dat,labels=factor(re$re))

redges.tri.cent(dat,Tr,M)

dat.fr<-data.frame(a=dat)
redges.tri.cent(dat.fr,Tr,M)

dat.fr<-data.frame(a=Tr)
redges.tri.cent(dat,dat.fr,M)
```

`redges.triCM` *The indices of the CM-edge regions in a triangle that contains the points in a give data set*

Description

Returns the indices of the edges whose regions contain the points in data set `Dt` in a triangle `tri`=(A, B, C) and edge regions are based on the center of mass CM of `tri`. (see the plots in the example for illustrations).

The vertices of the triangle `tri` are labeled as 1=A,2=B, and 3=C also according to the row number the vertex is recorded in `tri` and the corresponding edges are 1=BC, 2=AC, and 3=AB.

If a point in `Dt` is not inside `tri`, then the function yields `NA` as output for that entry. The corresponding edge region is the polygon with the vertex, CM, and vertices other than the non-adjacent vertex, i.e., edge region 1 is the triangle $T(B, CM, C)$, edge region 2 is $T(A, CM, C)$ and edge region 3 is $T(A, B, CM)$.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
redges.triCM(Dt, tri)
```

Arguments

`Dt`	A set of 2D points representing the set of data points for which indices of the edge regions containing them are to be determined.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

A list with the elements

`re`	Indices (i.e., a vector of indices) of the edges whose region contains points in `Dt` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.
`desc`	Description of the edge labels as "Edge labels are AB=1, BC=2, and AC=3".

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

redges.tri.cent, rverts.tri.cent and rverts.tri.nd

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(.4,.2)
redges.triCM(P,Tr)

P<-c(1.8,.5)
redges.triCM(P,Tr)

P<-c(10.5,1.6)
redges.triCM(P,Tr)

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

re<-redges.triCM(dat,Tr)
re
CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".",col=1)
L<-Tr; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

xc<-Tr[,1]+c(-.02,.03,.02)
yc<-Tr[,2]+c(.02,.02,.04)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)

txt<-rbind(CM,Ds)
```

```
xc<-txt[,1]+c(.05,.06,-.05,-.02)
yc<-txt[,2]+c(.03,.03,.05,-.08)
txt.str<-c("CM","re=2","re=3","re=1")
text(xc,yc,txt.str)
text(dat,labels=factor(re$re))

redges.triCM(dat,Tr)

dat.fr<-data.frame(a=dat)
redges.triCM(dat.fr,Tr)

dat.fr<-data.frame(a=Tr)
redges.triCM(dat,dat.fr)
```

`rel.six.Te` *Region index inside the Gamma-1 region*

Description

Returns the region index of the point `Pt` for the 6 regions in sandard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$, starting with 1 on the first one-sixth of the triangle, and numbering follows the counter-clockwise direction (see the plot in the examples). These regions are in the inner hexagon which is the Gamma-1 region for CS-PCD with $t = 1$ if `Pt` is not in any of the 6 regions the function returns `NA`.

Usage

```
rel.six.Te(Pt)
```

Arguments

`Pt`	A 2D point whose index for the 6 regions in sandard equilateral triangle T_e is determined.

Value

rel An integer between 1-6 (inclusive) or `NA`

See Also

`runifTe.onesixth`

Examples

```
## Not run:
P<-c(.4,.2)
rel.six.Te(P)

P<-c(.4,.3)
rel.six.Te(P)

P<-c(.8,.8)
rel.six.Te(P)

P<-c(.5,.61)
rel.six.Te(P)

A<-c(0,0); B<-c(1,0); C<-c(0.5,sqrt(3)/2);
Te<-rbind(A,B,C)
CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

h1<-c(1/2,sqrt(3)/18); h2<-c(2/3, sqrt(3)/9); h3<-c(2/3, 2*sqrt(3)/9);
h4<-c(1/2, 5*sqrt(3)/18); h5<-c(1/3, 2*sqrt(3)/9); h6<-c(1/3, sqrt(3)/9);

r1<-(h1+h6+CM)/3;r2<-(h1+h2+CM)/3;r3<-(h2+h3+CM)/3;
r4<-(h3+h4+CM)/3;r5<-(h4+h5+CM)/3;r6<-(h5+h6+CM)/3;

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-Te; R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
polygon(rbind(h1,h2,h3,h4,h5,h6))

txt<-rbind(h1,h2,h3,h4,h5,h6)
xc<-txt[,1]+c(-.02,.02,.02,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,0,0,0)
txt.str<-c("h1","h2","h3","h4","h5","h6")
text(xc,yc,txt.str)

txt<-rbind(Te,CM,r1,r2,r3,r4,r5,r6)
xc<-txt[,1]+c(-.02,.02,.02,0,0,0,0,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,0,0,0,0,0,0,0)
txt.str<-c("A","B","C","CM","1","2","3","4","5","6")
text(xc,yc,txt.str)

n<-10  #try also n<-40
dat<-runifTe(n)$gen.points
```

```
Xlim<-range(Te[,1],dat[,1])
Ylim<-range(Te[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

rsix<-vector()
for (i in 1:n)
  rsix<-c(rsix,rel.six.Te(dat[i,]))
rsix

plot(A,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(dat,pch=".")
L<-Te; R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
polygon(rbind(h1,h2,h3,h4,h5,h6))
text(dat,labels=factor(rsix))

txt<-rbind(Te,CM)
xc<-txt[,1]+c(-.02,.02,.02,0)
yc<-txt[,2]+c(.02,.02,.02,-.05)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)

## End(Not run)
```

`reTeCM` *The index of the edge region in the standard equilateral triangle that contains a point*

Description

Returns the index of the edge whose region contains point, `pt`, in the standard equilateral triangle $T_e = T(A = (0,0), B = (1,0), C = (1/2, \sqrt{3}/2))$ with edge regions based on center of mass $CM = (A + B + C)/3$.

Edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC. If the point, `pt`, is not inside `tri`, then the function yields `NA` as output. Edge region 1 is the triangle $T(B, C, M)$, edge region 2 is $T(A, C, M)$, and edge region 3 is $T(A, B, M)$.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2007)).

Usage

```
reTeCM(pt)
```

Arguments

`pt`	A 2D point for which CM-edge region it resides in is to be determined in the the standard equilateral triangle T_e.

Value

A list with three elements

re	Index of the CM-edge region that contains point, pt in the standard equilateral triangle T_e
tri	The vertices of the standard equilateral triangle T_e, where row labels are A, B, and C with edges are labeled as 3 for edge AB, 1 for edge BC, and 2 for edge AC.
desc	Description of the edge labels

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

re.triCM, re.tri.cent, re.bastriCM, re.bastri.cent, and redge.triCM

Examples

```
P<-c(.4,.2)
reTeCM(P)

P<-c(.8,.2)
reTeCM(P)

P<-c(.8,.8)
reTeCM(P)

P<-c(.5,.61)
reTeCM(P)

A<-c(0,0); B<-c(1,0); C<-c(0.5,sqrt(3)/2);
Te<-rbind(A,B,C)
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
reTeCM(A)
reTeCM(B)
reTeCM(C)
```

```
reTeCM(D1)
reTeCM(D2)
reTeCM(D3)

CM<-(A+B+C)/3
reTeCM(CM)

n<-10  #try also n<-20
dat<-runifTe(n)$gen.points

re<-vector()
for (i in 1:n)
  re<-c(re,reTeCM(dat[i,])$re)
re

Xlim<-range(Te[,1],dat[,1])
Ylim<-range(Te[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,asp=1,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
points(dat,pch=".")
polygon(Te)
L<-Te; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
text(dat,labels=factor(re))

txt<-rbind(Te,CM)
xc<-txt[,1]+c(-.03,.03,.03,-.06)
yc<-txt[,2]+c(.02,.02,.02,.03)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)

p1<-(A+B+CM)/3
p2<-(B+C+CM)/3
p3<-(A+C+CM)/3

plot(Te,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
L<-Te; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Te,CM,p1,p2,p3)
xc<-txt[,1]+c(-.03,.03,.03,-.06,0,0,0)
yc<-txt[,2]+c(.02,.02,.02,.03,0,0,0)
txt.str<-c("A","B","C","CM","re=3","re=1","re=2")
text(xc,yc,txt.str)
```

`rseg.disc` *Generation of points segregated (in a radial or circular fashion) from a given set of points*

Description

An object of class "Patterns". Generates `n` 2D points uniformly in $(a1 - e, a1 + e)x(a1 - e, a1 + e) \setminus B(y_i, e)$ where $Y = (y_1, y_2, ..., y_{n_y})$ with n_y being number of `Y` points for various values of `e` under the segregation pattern and $B(y_i, e)$ is the ball centered at y_i with radius `e`.

Positive values of `e` yield realizations from the segregation pattern and nonpositive values of `e` provide a type of complete spatial randomness (CSR), `e` should not be too large to make the support of generated points empty, `a1` is defaulted to the minimum of the x-coordinates of the `Y` points, `a2` is defaulted to the maximum of the x-coordinates of the `Y` points, `b1` is defaulted to the minimum of the y-coordinates of the `Y` points, `b2` is defaulted to the maximum of the y-coordinates of the `Y` points.

Usage

```
rseg.disc(
  n,
  Y,
  e,
  a1 = min(Y[, 1]),
  a2 = max(Y[, 1]),
  b1 = min(Y[, 2]),
  b2 = max(Y[, 2])
)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`Y`	A set of 2D points representing the reference points. The generated points are segregated (in a circular/radial fashion) from these points.
`e`	A positive real number representing the radius of the balls centered at `Y` points. These balls are forbidden for the generated points (i.e., generated points would be in the complement of union of these balls).
`a1, a2`	Real numbers representing the range of x-coordinates in the region (default is the range of x-coordinates of the `Y` points).
`b1, b2`	Real numbers representing the range of y-coordinates in the region (default is the range of y-coordinates of the `Y` points).

Value

A list with the elements

`type`	The type of the point pattern
`mtitle`	The "main" title for the plot of the point pattern

parameters	Radial (i.e. circular) exclusion parameter of the segregation pattern
ref.points	The input set of reference points Y, i.e., points from which generated points are segregated.
gen.points	The output set of generated points segregated from Y points
tri.Y	Logical output for triangulation based on Y points should be implemented or not. if TRUE triangulation based on Y points is to be implemented (default is set to FALSE).
desc.pat	Description of the point pattern
num.points	The vector of two numbers, which are the number of generated points and the number of reference (i.e., Y) points.
xlimit,ylimit	The possible ranges of the x- and y-coordinates of the generated points

See Also

rasc.disc, rsegTe, rsegIITe, and rsegMT

Examples

```
nx<-20; ny<-4;  #try also nx<-1000; ny<-10
e<-.15;
#with default bounding box (i.e., unit square)
Y<-cbind(runif(ny),runif(ny))

Xdt<-rseg.disc(nx,Y,e)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

e<-.15;  #try also e<- -.1; #this provides a CSR realization

#with default bounding box (i.e., unit square)
Y<-cbind(runif(ny),runif(ny))
Xdt<-rseg.disc(nx,Y,e)$gen.points
Xlim<-range(Xdt[,1],Y[,1]);
Ylim<-range(Xdt[,2],Y[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Y,asp=1,pch=16,col=2,lwd=2, xlab="x",ylab="y",main="segregation of two classes",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
points(Xdt)

#with a rectangular bounding box
a1<-0; a2<-10;
b1<-0; b2<-5;
e<-1.5;
Y<-cbind(runif(ny,a1,a2),runif(ny,b1,b2))
#try also Y<-cbind(runif(ny,a1,a2/2),runif(ny,b1,b2/2))

Xdt<-rseg.disc(nx,Y,e,a1,a2,b1,b2)$gen.points
```

```
Xlim<-range(Xdt[,1],Y[,1]); Ylim<-range(Xdt[,2],Y[,2])

plot(Y,pch=16,asp=1,col=2,lwd=2, xlab="x",ylab="y",main="segregation of two classes",
     xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(Xdt)
```

`rseg.tri` *Generation of points segregated (in a Type I fashion) from the vertices of a triangle*

Description

An object of class "Patterns". Generates `k` points uniformly in the support for Type I segregation in a given triangle, `tri`.

`delta` is the parameter of segregation (that is, $\delta 100$ % of the area around each vertex in the triangle is forbidden for point generation). `delta` corresponds to `eps` in the standard equilateral triangle T_e as $delta = 4eps^2/3$ (see `rsegTe` function).

See (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)) for more on the segregation pattern.

Usage

```
rseg.tri(k, tri, delta)
```

Arguments

`k`	A positive integer representing the number of points to be generated from the segregation pattern in the triangle, `tri`.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`delta`	A positive real number in (0,4/9). `delta` is the parameter of segregation (that is, $\delta 100$ % area around each vertex in each Delaunay triangle is forbidden for point generation).

Value

A list with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The "main" title for the plot of the point pattern
`parameters`	Exclusion parameter, `delta`, of the Type I segregation pattern. `delta` is in (0,4/9) $\delta 100$ % area around each vertex in the triangle `tri` is forbidden for point generation.
`ref.points`	The input set of points, i.e., vertices of `tri`; reference points, i.e., points from which generated points are segregated.

gen.points	The output set of generated points segregated from the vertices of tri.
tri.Y	Logical output, if TRUE the triangle tri is also plotted when the corresponding plot function from the Patterns object is called.
desc.pat	Description of the point pattern
num.points	The vector of two numbers, which are the number of generated points and the number of reference (i.e., vertice of tri, which is 3 here).
xlimit,ylimit	The ranges of the x- and y-coordinates of the reference points, which are the vertice so the triangle tri

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

rasc.tri, rsegTe, rsegIITe, and rsegMT

Examples

```
n<-10  #try also n<-100
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)
del<-.4

Xdt<-rseg.tri(n,Tr,del)
Xdt
summary(Xdt)
plot(Xdt)

dat<-rseg.tri(n,Tr,del)$g
Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat)
xc<-Tr[,1]+c(-.01,.01,.01)
yc<-Tr[,2]+c(.02,.02,.02)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)
```

```
dat.fr<-data.frame(a=Tr)
rseg.tri(n,dat.fr,del)
```

`rsegIITe` *Generation of points segregated (in a Type II fashion) from the vertices of T_e*

Description

An object of class "Patterns". Generates `k` points uniformly in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ under the type II segregation alternative for `eps` in $(0, \sqrt{3}/6 = 0.2886751]$.

In the type II segregation, the annular forbidden regions around the edges are determined by the parameter `eps` which is the distance from the interior triangle (i.e., support for the segregation) to T_e (see examples for a sample plot.)

Usage

```
rsegIITe(k, eps)
```

Arguments

`k`	A positive integer representing the number of points to be generated.
`eps`	A positive real number representing the parameter of type II segregation (which is the distance from the interior triangle to T_e).

Value

A list with the elements

`type`	The type of the point pattern
`mtitle`	The "main" title for the plot of the point pattern
`parameters`	The exclusion parameter, `eps`, of the segregation pattern, which is the distance from the interior triangle to T_e
`ref.points`	The input set of points `Y`; reference points, i.e., points from which generated points are segregated (i.e., vertices of T_e).
`gen.points`	The output set of generated points segregated from `Y` points (i.e., vertices of T_e).
`tri.Y`	Logical output for triangulation based on `Y` points should be implemented or not. if `TRUE` triangulation based on `Y` points is to be implemented (default is set to `FALSE`).
`desc.pat`	Description of the point pattern
`num.points`	The vector of two numbers, which are the number of generated points and the number of reference (i.e., `Y`) points, which is 3 here.
`xlimit,ylimit`	The ranges of the x- and y-coordinates of the reference points, which are the vertices of T_e here

See Also

rseg.disc, rasc.disc, rsegTe, and rsegMT

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10  #try also n<-20 or n<-100 or 1000
eps<-.15  #try also .2

set.seed(1)
Xdt<-rsegIITe(n,eps)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

dat<-rsegIITe(n,eps)$gen.points

plot(Te,pch=".",xlab="",ylab="",main="Type II segregation in the \n standard equilateral triangle",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(dat)

#The support for the Type II segregation alternative
C1<-c(1/2,sqrt(3)/2-2*eps);
A1<-c(eps*sqrt(3),eps); B1<-c(1-eps*sqrt(3),eps);
supp<-rbind(A1,B1,C1)

plot(Te,asp=1,pch=".",xlab="",ylab="",main="Support of the Type II Segregation",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
  polygon(Te)
  polygon(supp,col=5)
points(dat)
```

---

rsegMT *Generation of points segregated (in a Type I fashion) from a given set of points*

Description

An object of class "Patterns". Generates n points uniformly in the support for Type I segregation in the convex hull of set of points, Yp.

`delta` is the parameter of segregation (that is, $\delta 100$ % of the area around each vertex in each Delaunay triangle is forbidden for point generation). `delta` corresponds to `eps` in the standard equilateral triangle T_e as $delta = 4eps^2/3$ (see `rsegTe` function).

If `Yp` consists only of 3 points, then the function behaves like the function `rseg.tri`.

`DTmesh` must be the Delaunay triangulation of `Yp` and `DTr` must be the corresponding Delaunay triangles (both `DTmesh` and `DTr` are `NULL` by default). If `NULL`, `DTmesh` is computed via `tri.mesh` and `DTr` is computed via `triangles` function in `interp` package.

`tri.mesh` function yields the triangulation nodes with their neigbours, and creates a triangulation object, and `triangles` function yields a triangulation data structure from the triangulation object created by `tri.mesh` (the first three colums are the vertex indices of the Delaunay triangles.)

See (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)) for more on the segregation pattern. Also see (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
rsegMT(n, Yp, delta, DTmesh = NULL, DTr = NULL)
```

Arguments

`n`	A positive integer representing the number of points to be generated.
`Yp`	A set of 2D points from which Delaunay triangulation is constructed.
`delta`	A positive real number in (0,4/9). `delta` is the parameter of segregation (that is, $\delta 100$ area around each vertex in each Delaunay triangle is forbidden for point generation).
`DTmesh`	Delaunay triangulation of `Yp`, default is `NULL`, which is computed via `tri.mesh` function in `interp` package. `tri.mesh` function yields the triangulation nodes with their neigbours, and creates a triangulation object.
`DTr`	Delaunay triangles based on `Yp`, default is `NULL`, which is computed via `tri.mesh` function in `interp` package. `triangles` function yields a triangulation data structure from the triangulation object created by `tri.mesh`.

Value

A list with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The "main" title for the plot of the point pattern
`parameters`	Exclusion parameter, `delta`, of the Type I segregation pattern. `delta` is in (0,4/9) $\delta 100$ % area around each vertex in each Delaunay triangle is forbidden for point generation.
`ref.points`	The input set of points `Yp`; reference points, i.e., points from which generated points are segregated.
`gen.points`	The output set of generated points segregated from `Yp` points.
`tri.Y`	Logical output, `TRUE` if triangulation based on `Yp` points should be implemented.

desc.pat	Description of the point pattern
num.points	The vector of two numbers, which are the number of generated points and the number of reference (i.e., Yp) points.
xlimit,ylimit	The ranges of the x- and y-coordinates of the reference points, which are the Yp points

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

rseg.disc, rsegTe, rsegIITe, and rascMT

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-1000; ny<-10;

set.seed(1)
Yp<-cbind(runif(ny),runif(ny))
del<-.4

Xdt<-rsegMT(nx,Yp,del)
Xdt
summary(Xdt)
plot(Xdt)

Yp<-cbind(runif(ny),runif(ny))
del<-.3 #try .5, .75, .85
dat<-rsegMT(nx,Yp,del)  #data under CSR in the convex hull of Ypoints

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points
```

```
dat<-rsegMT(nx,Yp,del,DTY)  #data under CSR in the convex hull of Ypoints

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points
TRY<-interp::triangles(DTY)[,1:3];
dat<-rsegMT(nx,Yp,del,DTr=TRY)  #data under CSR in the convex hull of Ypoints

#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points
TRY<-interp::triangles(DTY)[,1:3];
dat<-rsegMT(nx,Yp,del,DTY,TRY)$gen.points #data under CSR in the convex hull of Ypoints

Xlim<-range(Yp[,1])
Ylim<-range(Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

#plot of the data in the convex hull of Y points together with the Delaunay triangulation
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points

oldpar <- par(mfrow = c(1,2))
par(pty="s")
plot(dat,main=" ", xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),type="n")
interp::plot.triSht(DTY, add=TRUE, do.points=TRUE,col="blue")
points(dat,pch=".",cex=3)
par(oldpar)

Yp<-rbind(c(.3,.2),c(.4,.5),c(.14,.15))
rsegMT(nx,Yp,del)

dat.fr<-data.frame(a=Yp)
rsegMT(nx,dat.fr,del)
```

`rsegTe` *Generation of points segregated (in a Type I fashion) from the vertices of* T_e

---

Description

An object of class "Patterns". Generates `k` points uniformly in the standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ under the type I segregation alternative for `eps` in $(0, \sqrt{3}/3 = 0.5773503]$.

In the type I segregation, the triangular forbidden regions around the vertices are determined by the parameter `eps` which serves as the height of these triangles (see examples for a sample plot.)

See also (Ceyhan et al. (2006); Ceyhan et al. (2007); Ceyhan (2011)).

Usage

```
rsegTe(k, eps)
```

Arguments

k	A positive integer representing the number of points to be generated.
eps	A positive real number representing the parameter of type I segregation (which is the height of the triangular forbidden regions around the vertices).

Value

A list with the elements

type	The type of the point pattern
mtitle	The "main" title for the plot of the point pattern
parameters	The exclusion parameter, eps, of the segregation pattern, which is the height of the triangular forbidden regions around the vertices
ref.points	The input set of points Y; reference points, i.e., points from which generated points are segregated (i.e., vertices of T_e).
gen.points	The output set of generated points segregated from Y points (i.e., vertices of T_e).
tri.Y	Logical output for triangulation based on Y points should be implemented or not. if TRUE triangulation based on Y points is to be implemented (default is set to FALSE).
desc.pat	Description of the point pattern
num.points	The vector of two numbers, which are the number of generated points and the number of reference (i.e., Y) points, which is 3 here.
xlimit,ylimit	The ranges of the x- and y-coordinates of the reference points, which are the vertices of T_e here

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

rseg.disc, rasc.disc, rsegIITe, and rsegMT

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10  #try also n<-20 or n<-100 or 1000
eps<-.3  #try also .15, .5, .75
```

```
set.seed(1)
Xdt<-rsegTe(n,eps)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

set.seed(1)
dat<-rsegTe(n,eps)$gen.points

plot(Te,asp=1,pch=".",xlab="",ylab="",
main="Type I segregation in the \n standard equilateral triangle",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(dat)

#The support for the Type I segregation alternative
sr<-eps/(sqrt(3)/2)
C1<-C+sr*(A-C); C2<-C+sr*(B-C)
A1<-A+sr*(B-A); A2<-A+sr*(C-A)
B1<-B+sr*(A-B); B2<-B+sr*(C-B)
supp<-rbind(A1,B1,B2,C2,C1,A2)

plot(Te,asp=1,pch=".",xlab="",ylab="",main="Support of the Type I Segregation",
     xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
if (sr<=.5)
{
  polygon(Te)
  polygon(supp,col=5)
} else
{
  polygon(Te,col=5,lwd=2.5)
  polygon(rbind(A,A1,A2),col=0,border=NA)
  polygon(rbind(B,B1,B2),col=0,border=NA)
  polygon(rbind(C,C1,C2),col=0,border=NA)
}
points(dat)
```

`runif.bastri` *Generation of Uniform Points in the basic triangle*

Description

An object of class "Uniform". Generates `k` points uniformly in the basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan et al. (2006)). Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

Usage

```
runif.bastri(k, c1, c2)
```

Arguments

k	A positive integer representing the number of uniform points to be generated in the basic triangle.
c1, c2	Positive real numbers representing the top vertex in basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$, c1 must be in $[0,1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Value

A list with the elements

type	The type of the pattern from which points are to be generated
mtitle	The "main" title for the plot of the point pattern
tess.points	The vertices of the support of the uniformly generated points, it is the basic triangle T_b for this function
gen.points	The output set of generated points uniformly in the basic triangle
out.region	The outer region which contains the support region, NULL for this function.
desc.pat	Description of the point pattern from which points are to be generated
num.points	The vector of two numbers, which are the number of generated points and the number of vertices of the support points (here it is 3).
txt4pnts	Description of the two numbers in num.points.
xlimit,ylimit	The ranges of the x- and y-coordinates of the support, Tb

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

runifTe, runif.tri, and runifMT

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);
n<-10  #try also n<-100

set.seed(1)
runif.bastri(1,c1,c2)
Xdt<-runif.bastri(n,c1,c2)
Xdt
summary(Xdt)
plot(Xdt)

dat<-runif.bastri(n,c1,c2)$g

Xlim<-range(Tb[,1])
Ylim<-range(Tb[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,xlab="",ylab="",xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),type="n")
polygon(Tb)
points(dat)
```

`runif.stdtetra` *Generation of Uniform Points in the Standard Regular Tetrahedron* T_h

---

Description

An object of class "Uniform". Generates k points uniformly in the standard regular tetrahedron $T_h = T((0,0,0),(1,0,0),(1/2, sqrt(3)/2, 0),(1/2, \sqrt{3}/6, \sqrt{6}/3))$.

Usage

```
runif.stdtetra(k)
```

Arguments

k	A positive integer representing the number of uniform points to be generated in the standard regular tetrahedron T_h.

Value

A list with the elements

type	The type of the pattern from which points are to be generated
mtitle	The "main" title for the plot of the point pattern

`tess.points`	The vertices of the support region of the uniformly generated points, it is the standard regular tetrahedron T_h for this function
`gen.points`	The output set of generated points uniformly in the standard regular tetrahedron T_h.
`out.region`	The outer region which contains the support region, `NULL` for this function.
`desc.pat`	Description of the point pattern from which points are to be generated
`num.points`	The vector of two numbers, which are the number of generated points and the number of vertices of the support points (here it is 4).
`txt4pnts`	Description of the two numbers in `num.points`
`xlimit,ylimit,zlimit`	The ranges of the x-, y-, and z-coordinates of the support, T_h

See Also

`runif.tetra`, `runif.tri` and `runifMT`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)
n<-10  #try also n<-100

set.seed(1)
Xdt<-runif.stdtetra(n)
Xdt
summary(Xdt)
plot(Xdt)

Dt<-runif.stdtetra(n)$g

Xlim<-range(tetra[,1])
Ylim<-range(tetra[,2])
Zlim<-range(tetra[,3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(Dt[,1],Dt[,2],Dt[,3],
phi =20,theta=15, bty = "g", pch = 20, cex = 1, ticktype = "detailed",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05))
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)

plot3D::text3D(tetra[,1]+c(.05,0,0,0),tetra[,2],tetra[,3],
labels=c("A","B","C","D"), add=TRUE)
```

```
## Not run:
#need to install scatterplot3d package and call "library(scatterplot3d)"
s3d<-scatterplot3d(Dt, highlight.3d=TRUE,xlab="x",ylab="y",zlab="z",
              col.axis="blue", col.grid="lightblue",
               main="3D Scatterplot of the data", pch=20)
s3d$points3d(tetra,pch=20,col="blue")

## End(Not run)
```

`runif.tetra` *Generation of Uniform Points in a tetrahedron*

Description

An object of class "Uniform". Generates `k` points uniformly in the general tetrahedron `th` whose vertices are stacked row-wise.

Usage

```
runif.tetra(k, th)
```

Arguments

`k`	A positive integer representing the number of uniform points to be generated in the tetrahedron.
`th`	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.

Value

A list with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The "main" title for the plot of the point pattern
`tess.points`	The vertices of the support of the uniformly generated points, it is the tetrahedron `th` for this function
`gen.points`	The output set of generated points uniformly in the tetrahedron, `th`.
`out.region`	The outer region which contains the support region, `NULL` for this function.
`desc.pat`	Description of the point pattern from which points are to be generated
`num.points`	The vector of two numbers, which are the number of generated points and the number of vertices of the support points (here it is 4).
`txt4pnts`	Description of the two numbers in `num.points`
`xlimit, ylimit, zlimit`	The ranges of the x-, y- and z-coordinates of the support, `th`

See Also

runif.stdtetra and runif.tri

Examples

```
A<-sample(1:12,3); B<-sample(1:12,3); C<-sample(1:12,3); D<-sample(1:12,3)
tetra<-rbind(A,B,C,D)

n<-10  #try also n<-100

set.seed(1)
Xdt<-runif.tetra(n,tetra)
Xdt
summary(Xdt)
plot(Xdt)

Dt<-Xdt$g

Xlim<-range(tetra[,1],Dt[,1])
Ylim<-range(tetra[,2],Dt[,2])
Zlim<-range(tetra[,3],Dt[,3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(Dt[,1],Dt[,2],Dt[,3], theta =225, phi = 30, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
          pch = 20, cex = 1, ticktype = "detailed")
#add the vertices of the tetrahedron
plot3D::points3D(tetra[,1],tetra[,2],tetra[,3], add=TRUE)
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3], labels=c("A","B","C","D"), add=TRUE)

runif.tetra(n,tetra)

dat.fr<-data.frame(a=tetra)
runif.tetra(n,dat.fr)


## Not run:
#need to install scatterplot3d package and call "library(scatterplot3d)"
s3d<-scatterplot3d(Dt, highlight.3d=TRUE,xlab="x",ylab="y",zlab="z",
                   col.axis="blue", col.grid="lightblue",
                   main="3D Scatterplot of the data", pch=20)
 s3d$points3d(tetra,pch=20,col="blue")

## End(Not run)
```

runif.tri *Generation of Uniform Points in a triangle*

Description

An object of class "Uniform". Generates k points uniformly in a given triangle, tri

Usage

```
runif.tri(k, tri)
```

Arguments

k	A positive integer representing the number of uniform points to be generated in the triangle.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

A list with the elements

type	The type of the pattern from which points are to be generated
mtitle	The "main" title for the plot of the point pattern
tess.points	The vertices of the support of the uniformly generated points, it is the triangle tri for this function
gen.points	The output set of generated points uniformly in the triangle, tri.
out.region	The outer region which contains the support region, NULL for this function.
desc.pat	Description of the point pattern from which points are to be generated
num.points	The vector of two numbers, which are the number of generated points and the number of vertices of the support points (here it is 3).
txt4pnts	Description of the two numbers in num.points
xlimit,ylimit	The ranges of the x- and y-coordinates of the support, tri

See Also

runifTe, runif.bastri, and runifMT

Examples

```
n<-10  #try also n<-100
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)

Xdt<-runif.tri(n,Tr)
Xdt
```

```
summary(Xdt)
plot(Xdt)

dat<-runif.tri(n,Tr)$g
Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat)
xc<-Tr[,1]+c(-.01,.01,.01)
yc<-Tr[,2]+c(.02,.02,.02)
txt.str<-c("A","B","C")
text(xc,yc,txt.str)
```

`runifMT` *Generation of Uniform Points in the Convex Hull of Points*

Description

An object of class "Uniform". Generates `n` points uniformly in the Convex Hull of set of points, `Yp`. That is, generates uniformly in each of the triangles in the Delaunay triangulation of `Yp`, i.e., in the multiple triangles partitioning the convex hull of `Yp`.

If `Yp` consists only of 3 points, then the function behaves like the function `runif.tri`.

`DTmesh` is the Delaunay triangulation of `Yp`, default is `DTmesh=NULL`. `DTmesh` yields triangulation nodes with neigbours (result of `tri.mesh` function from `interp` package).

See (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
runifMT(n, Yp, DTmesh = NULL)
```

Arguments

`n`	A positive integer representing the number of uniform points to be generated in the convex hull of the point set `Yp`.
`Yp`	A set of 2D points whose convex hull is the support of the uniform points to be generated.
`DTmesh`	Triangulation nodes with neigbours (result of `tri.mesh` function from `interp` package).

Value

A list with the elements

type	The type of the pattern from which points are to be generated
mtitle	The "main" title for the plot of the point pattern
tess.points	The points which constitute the vertices of the triangulation and whose convex hull determines the support of the generated points.
gen.points	The output set of generated points uniformly in the convex hull of Yp
out.region	The outer region which contains the support region, NULL for this function.
desc.pat	Description of the point pattern from which points are to be generated
num.points	The vector of two numbers, which are the number of generated points and the number of vertices in the triangulation (i.e., size of Yp) points.
txt4pnts	Description of the two numbers in num.points
xlimit,ylimit	The ranges of the x- and y-coordinates of the points in Yp

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

runif.tri, runifTe, and runif.bastri,

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-1000; ny<-10;
set.seed(1)
Yp<-cbind(runif(ny,0,10),runif(ny,0,10))

Xdt<-runifMT(nx,Yp)  #data under CSR in the convex hull of Ypoints
Xdt
summary(Xdt)
plot(Xdt)

dat<-runifMT(nx,Yp)$g #data under CSR in the convex hull of Ypoints
#or use
DTY<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove") #Delaunay triangulation based on Y points
dat<-runifMT(nx,Yp,DTY)$g #data under CSR in the convex hull of Ypoints

Xlim<-range(Yp[,1])
Ylim<-range(Yp[,2])
```

```
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

#plot of the data in the convex hull of Y points together with the Delaunay triangulation
plot(dat,main=" ", xlab=" ", ylab=" ",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),type="n")
interp::plot.triSht(DTY, add=TRUE, do.points = TRUE,pch=16,col="blue")
points(dat,pch=".",cex=3)

Yp<-rbind(c(.3,.2),c(.4,.5),c(.14,.15))
runifMT(nx,Yp)

dat.fr<-data.frame(a=Yp)
runifMT(nx,dat.fr)
```

`runifTe` *Generation of Uniform Points in the Standard Equilateral Triangle*

Description

An object of class "Uniform". Generates `k` points uniformly in the standard equilateral triangle $T_e = T(A, B, C)$ with vertices $A = (0, 0)$, $B = (1, 0)$, and $C = (1/2, sqrt3/2)$.

Usage

```
runifTe(k)
```

Arguments

`k`	A positive integer representing the number of uniform points to be generated in the standard equilateral triangle T_e.

Value

A list with the elements

`type`	The type of the pattern from which points are to be generated
`mtitle`	The "main" title for the plot of the point pattern
`tess.points`	The vertices of the support region of the uniformly generated points, it is the standard equilateral triangle T_e for this function
`gen.points`	The output set of generated points uniformly in the standard equilateral triangle T_e.
`out.region`	The outer region which contains the support region, `NULL` for this function.
`desc.pat`	Description of the point pattern from which points are to be generated
`num.points`	The vector of two numbers, which are the number of generated points and the number of vertices of the support points (here it is 3).
`txt4pnts`	Description of the two numbers in `num.points`
`xlimit,ylimit`	The ranges of the x- and y-coordinates of the support, T_e

See Also

runif.bastri, runif.tri, and runifMT

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
n<-10  #try also n<-100

set.seed(1)
Xdt<-runifTe(n)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

dat<-runifTe(n)$gen.points
plot(Te,asp=1,pch=".",xlab="",ylab="",xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01))
polygon(Te)
points(dat)
```

---

runifTe.onesixth *Generation of Uniform Points in a part of standard equilateral triangle*

Description

An object of class "Uniform". Generates k points uniformly in the first 1/6th of the standard equilateral triangle $T_e = (A, B, C)$ with vertices with $A = (0, 0)$; $B = (1, 0)$; $C = (1/2, sqrt3/2)$ (see the examples below). The first 1/6th of the standard equilateral triangle is the triangle with vertices $A = (0, 0)$; (1/2,0); $C = (1/2, \sqrt{3}/6)$.

Usage

```
runifTe.onesixth(k)
```

Arguments

k	a positive integer representing number of uniform points to be generated in the first one sixth of T_e.

Value

A list with the elements

type	The type of the point pattern
mtitle	The "main" title for the plot of the point pattern
support	The vertices of the support of the uniformly generated points
gen.points	The output set of uniformly generated points in the first 1/6th of the standard equilateral triangle.
out.region	The outer region for the one-sixth of T_e, which is just Te here.
desc.pat	Description of the point pattern
num.points	The vector of two numbers, which are the number of generated points and the number of vertices of the support (i.e., Y) points.
txt4pnts	Description of the two numbers in num.points.
xlimit,ylimit	The ranges of the x- and y-coordinates of the generated, support and outer region points

See Also

runifTe, runif.bastri, runif.tri, and runifMT

Examples

```
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
CM<-(A+B+C)/3;
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
nx<-100  #try also nx<-100

#data generation step
set.seed(1)
Xdt<-runifTe.onesixth(nx)
Xdt
summary(Xdt)
plot(Xdt,asp=1)

Xd<-runifTe.onesixth(nx)$gen.points

#plot of the data with the regions in the equilateral triangle
Xlim<-range(Te[,1])
Ylim<-range(Te[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,asp=1,pch=".",xlim=Xlim+xd*c(-.01,.01),ylim=Ylim+yd*c(-.01,.01),xlab=" ",ylab=" ",
     main="first 1/6th of the \n standard equilateral triangle")
polygon(Te)
L<-Te; R<-Ds
```

```
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
polygon(rbind(A,D3,CM),col=5)
points(Xd)

#letter annotation of the plot
txt<-rbind(A,B,C,CM,D1,D2,D3)
xc<-txt[,1]+c(-.02,.02,.02,.04,.05,-.03,0)
yc<-txt[,2]+c(.02,.02,.02,.03,0,.03,-.03)
txt.str<-c("A","B","C","CM","D1","D2","D3")
text(xc,yc,txt.str)

## End(Not run)
```

`rv.bastri.cent` *The index of the vertex region in a basic triangle that contains a given point*

Description

Returns the index of the related vertex in the basic triangle whose region contains point `p`. The basic triangle is $T_b = T((0,0),(1,0),(c_1,c_2))$ where `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Vertex regions are based on the general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the basic triangle T_b. Vertices of the basic triangle T_b are labeled according to the row number the vertex is recorded, i.e., as 1=(0,0), 2=(1,0),and 3=(c_1, c_2).

If the point, `p`, is not inside T_b, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, `M`, and projections from `M` to the edges on the lines joining vertices and `M`. That is, $rv = 1$ has vertices (0,0),D3,M,D2; $rv = 2$ has vertices (1,0),D1,M,D3; and $rv = 3$ has vertices (c_1, c_2),D2,M,D1 (see the illustration in the examples).

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
rv.bastri.cent(p, c1, c2, M)
```

Arguments

`p`	A 2D point for which `M`-vertex region it resides in is to be determined in the basic triangle T_b.
`c1, c2`	Positive real numbers which constitute the vertex of the basic triangle adjacent to the shorter edges; `c1` must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the basic triangle.

Value

A list with two elements

rv	Index of the vertex whose region contains point, p; index of the vertex is the same as the row number in the basic triangle, T_b
tri	The vertices of the basic triangle, T_b, where row number corresponds to the vertex index rv with $rv = 1$ is row 1=(0,0), $rv = 2$ is row 2=(1,0), and $rv = 3$ is row 3=(c_1, c_2).

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

rv.bastriCM, rv.tri.cent, rv.triCC, rv.bastriCC, rv.triCM, and rvTeCM

Examples

```
c1<-.4; c2<-.6
A<-c(0,0); B<-c(1,0); C<-c(c1,c2);
Tb<-rbind(A,B,C);
M<-c(.6,.2)

P<-c(.4,.2)
rv.bastri.cent(P,c1,c2,M)

P<-c(1.8,.5)
rv.bastri.cent(P,c1,c2,M)

P<-c(.5,.26)
rv.bastri.cent(P,c1,c2,M)

n<-10  #try also n<-20
set.seed(1)
dat<-runif.bastri(n,c1,c2)$g

M<-as.numeric(runif.bastri(1,c1,c2)$g)  #try also M<-c(.6,.2)

Rv<-vector()
for (i in 1:n)
{ Rv<-c(Rv,rv.bastri.cent(dat[i,],c1,c2,M)$rv)}
```

```
Rv

Ds<-cp2e.bastri(c1,c2,M)

Xlim<-range(Tb[,1],dat[,1])
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tb)}
#need to run this when M is given in barycentric coordinates

plot(Tb,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.1,.1),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
points(dat,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

xc<-Tb[,1]+c(-.04,.05,.04)
yc<-Tb[,2]+c(.02,.02,.03)
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.04,-.03,0)
yc<-txt[,2]+c(-.02,.02,.02,-.03)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)

text(dat,labels=factor(Rv))

P<-c(.4,.2)
rv.bastri.cent(P,c1,c2,M)
```

`rv.bastriCC` *The index of the CC-vertex region in a basic triangle that contains a point*

Description

Returns the index of the vertex whose region contains point `p` in the basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$ and vertex regions are based on the circumcenter CC of T_b. (see the plots in the example for illustrations).

The vertices of the basic triangle T_b are labeled as 1=(0,0),2=(1,0),and 3=(c_1, c_2) also according to the row number the vertex is recorded in T_b. If the point, `p`, is not inside T_b, then the function yields `NA` as output. The corresponding vertex region is the polygon whose interior points are closest to that vertex.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See also (Ceyhan (2005, 2010)).

Usage

```
rv.bastriCC(p, c1, c2)
```

Arguments

p	A 2D point for which CC-vertex region it resides in is to be determined in the basic triangle T_b.
c1, c2	Positive real numbers which constitute the upper vertex of the basic triangle (i.e., the vertex adjacent to the shorter edges of T_b); c1 must be in $[0, 1/2]$, $c_2 > 0$ and $(1 - c_1)^2 + c_2^2 \leq 1$.

Value

A list with two elements

rv	Index of the CC-vertex region that contains point, p in the basic triangle T_b
tri	The vertices of the triangle, where row number corresponds to the vertex index in rv with row 1=(0,0), row 2=(1,0), and row 3=(c_1, c_2).

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

rv.triCM, rv.tri.cent, rv.triCC, rv.bastriCM, rv.bastri.cent, and rvTeCM

Examples

```
c1<-.4; c2<-.6;  #try also c1<-.5; c2<-.5;

P<-c(.3,.2)
rv.bastriCC(P,c1,c2)

P<-c(.6,.2)
```

```
rv.bastriCC(P,c1,c2)

P<-c(.5,.4)
rv.bastriCC(P,c1,c2)

P<-c(1.5,.4)
rv.bastriCC(P,c1,c2)

A<-c(0,0);B<-c(1,0);C<-c(c1,c2);
Tb<-rbind(A,B,C)
CC<-circ.cent.bastri(c1,c2)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tb[,1])
Ylim<-range(Tb[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,asp=1,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tb,CC,Ds)
xc<-txt[,1]+c(-.03,.03,0.02,-.01,.05,-.05,.01)
yc<-txt[,2]+c(.02,.02,.03,.06,.03,.03,-.03)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

RV1<-(A+D3+CC+D2)/4
RV2<-(B+D3+CC+D1)/4
RV3<-(C+D2+CC+D1)/4

txt<-rbind(RV1,RV2,RV3)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

n<-10  #try also n<-20
dat<-runif.bastri(n,c1,c2)$g

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rv.bastriCC(dat[i,],c1,c2)$rv)
Rv

Xlim<-range(Tb[,1],dat[,1])
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
```

```
plot(Tb,asp=1,xlab="",pch=".",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(dat,pch=".")
polygon(Tb)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
text(dat,labels=factor(Rv))

txt<-rbind(Tb,CC,Ds)
xc<-txt[,1]+c(-.03,.03,0.02,-.01,.05,-.05,.01)
yc<-txt[,2]+c(.02,.02,.03,.06,.03,.03,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

P<-c(.3,.2)
rv.bastriCC(P,c1,c2)
```

`rv.bastriCM` *The index of the CM-vertex region in a basic triangle that contains a point*

Description

Returns the index of the vertex whose region contains point `p` in the basic triangle $T_b = T((0,0),(1,0),(c_1,c_2))$ where `c1` is in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$ and vertex regions are based on the center of mass CM=((1+c1)/3,c2/3) of T_b. (see the plots in the example for illustrations).

The vertices of the basic triangle T_b are labeled as 1=(0,0),2=(1,0),and 3=(c_1, c_2) also according to the row number the vertex is recorded in T_b. If the point, `p`, is not inside T_b, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, CM, and midpoints of the edges adjacent to the vertex.

Any given triangle can be mapped to the basic triangle by a combination of rigid body motions (i.e., translation, rotation and reflection) and scaling, preserving uniformity of the points in the original triangle. Hence basic triangle is useful for simulation studies under the uniformness hypothesis.

See also (Ceyhan (2005, 2010); Ceyhan et al. (2006))

Usage

```
rv.bastriCM(p, c1, c2)
```

Arguments

`p`	A 2D point for which CM-vertex region it resides in is to be determined in the basic triangle T_b.
`c1, c2`	Positive real numbers which constitute the upper vertex of the basic triangle (i.e., the vertex adjacent to the shorter edges of T_b); `c1` must be in $[0, 1/2]$, $c_2 > 0$ and $(1-c_1)^2 + c_2^2 \leq 1$.

Value

A list with two elements

rv	Index of the CM-vertex region that contains point, p in the basic triangle T_b
tri	The vertices of the triangle, where row number corresponds to the vertex index in rv with row 1=(0,0), row 2=(1,0), and row 3=(c_1, c_2).

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

rv.triCM, rv.tri.cent, rv.triCC, rv.bastriCC, rv.bastri.cent, and rvTeCM

Examples

```
c1<-.4; c2<-.6
P<-c(.4,.2)
rv.bastriCM(P,c1,c2)

c1<-.5; c2<-.8
P<-c(.4,.2)
rv.bastriCM(P,c1,c2)

P<-c(.8,.2)
rv.bastriCM(P,c1,c2)

P<-c(1,.2)
rv.bastriCM(P,c1,c2)

c1<-.4; c2<-.6
A<-c(0,0);B<-c(1,0);C<-c(c1,c2);
Tb<-rbind(A,B,C)
CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)
```

```
rv.bastriCM(A,c1,c2)
rv.bastriCM(CM,c1,c2)
rv.bastriCM(D1,c1,c2)
rv.bastriCM(D2,c1,c2)
rv.bastriCM(D3,c1,c2)

n<-10  #try also n<-20
dat<-runif.bastri(n,c1,c2)$g

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rv.bastriCM(dat[i,],c1,c2)$rv)
Rv

Xlim<-range(Tb[,1],dat[,1])
Ylim<-range(Tb[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tb,xlab="",ylab="",axes="T",pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
points(dat,pch=".")
polygon(Tb)
L<-Ds; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
text(dat,labels=factor(Rv))

txt<-rbind(Tb,CM,Ds)
xc<-txt[,1]+c(-.03,.03,.02,-.01,.06,-.05,.0)
yc<-txt[,2]+c(.02,.02,.02,.04,.02,.02,-.03)
txt.str<-c("A","B","C","CM","D1","D2","D3")
text(xc,yc,txt.str)

plot(Tb,xlab="",ylab="",axes="T",pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tb)
L<-Ds; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

RV1<-(A+D3+CM+D2)/4
RV2<-(B+D3+CM+D1)/4
RV3<-(C+D2+CM+D1)/4

txt<-rbind(RV1,RV2,RV3)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(Tb,CM,Ds)
xc<-txt[,1]+c(-.03,.03,.02,-.01,.04,-.03,.0)
yc<-txt[,2]+c(.02,.02,.02,.04,.02,.02,-.03)
txt.str<-c("A","B","C","CM","D1","D2","D3")
text(xc,yc,txt.str)
```

`rv.end.int` *The index of the vertex region in an end-interval that contains a given point*

Description

Returns the index of the vertex in the interval, `int`, whose end interval contains the 1D point `pt`, that is, it finds the index of the vertex for the point, `pt`, outside the interval $int = (a, b) = (vertex1, vertex2)$; vertices of interval are labeled as 1 and 2 according to their order in the interval.

If the point, `pt`, is inside `int`, then the function yields `NA` as output. The corresponding vertex region is an interval as $(-\infty, a)$ or (b, ∞) for the interval (a, b). Then if `pt`<a, then $rv = 1$ and if `pt`>b, then rv=2. Unlike `rv.mid.int`, centrality parameter (i.e., center of the interval is not relevant for `rv.end.int`.)

See also (Ceyhan (2012, 2016)).

Usage

```
rv.end.int(pt, int)
```

Arguments

`pt`	A 1D point whose end interval region is provided by the function.
`int`	A vector of two real numbers representing an interval.

Value

A list with two elements

`rv`	Index of the end vertex whose region contains point, `pt`.
`int`	The vertices of the interval as a vector where position of the vertex corresponds to the vertex index as $int = (rv = 1, rv = 2)$.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`rv.mid.int`

Examples

```
a<-0; b<-10; int<-c(a,b)

rv.end.int(-6,int)
rv.end.int(16,int)

n<-5
xr<-range(a,b)
xf<-(xr[2]-xr[1])*.5
datL<-runif(n,a-xf,a)
datR<-runif(n,b,b+xf)
dat<-c(datL,datR)
rv.end.int(dat[1],int)

Rv<-vector()
for (i in 1:length(dat))
  Rv<-c(Rv,rv.end.int(dat[i],int)$rv)
Rv

Xlim<-range(a,b,dat)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),xlab="",pch=".",xlim=Xlim+xd*c(-.05,.05))
abline(h=0)
abline(v=c(a,b),col=1,lty=2)
points(cbind(dat,0))
text(cbind(dat,0.1),labels=factor(Rv))
text(cbind(c(a,b),-0.1),c("rv=1","rv=2"))

jit<-.1
yjit<-runif(length(dat),-jit,jit)

Xlim<-range(a,b,dat)
xd<-Xlim[2]-Xlim[1]

plot(cbind(a,0),main="vertex region indices for the points\n in the end intervals",
     xlab=" ", ylab=" ",pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=3*range(yjit))
points(dat, yjit,xlim=Xlim+xd*c(-.05,.05),pch=".",cex=3)
abline(h=0)
abline(v=c(a,b),lty=2)
text(dat,yjit,labels=factor(Rv))
text(cbind(c(a,b),-.01),c("rv=1","rv=2"))
```

`rv.mid.int` *The index of the vertex region in a middle interval that contains a given point*

Description

Returns the index of the vertex whose region contains point `pt` in the interval $int = (a, b) = (vertex1, vertex2)$ with (parameterized) center M_c associated with the centrality parameter `c` in $(0, 1)$; vertices of interval are labeled as 1 and 2 according to their order in the interval `int`. If the point, `pt`, is not inside `int`, then the function yields `NA` as output. The corresponding vertex region is the interval (a, b) as (a, M_c) and (M_c, b) where $M_c = a + c(b - a)$.

See also (Ceyhan (2012, 2016)).

Usage

```
rv.mid.int(pt, c, int)
```

Arguments

`pt`	A 1D point. The vertex region `pt` resides is to be found.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
`int`	A vector of two real numbers representing an interval.

Value

A list with two elements

`rv`	Index of the vertex in the interval `int` whose region contains point, `pt`.
`int`	The vertices of the interval as a vector where position of the vertex corresponds to the vertex index as $int = (rv = 1, rv = 2)$.

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

`rv.end.int`

Examples

```
c<-.4
a<-0; b<-10; int<-c(a,b)

Mc<-centMc(int,c)

rv.mid.int(6,c,int)

rv.mid.int(3,c,int)
```

```
rv.mid.int(13,c,int)

rv.mid.int(4,c,int)
rv.mid.int(0,c,int)
rv.mid.int(-3,c,int)

n<-10  #try also n<-20
xr<-range(a,b,Mc)
xf<-(xr[2]-xr[1])*.5
dat<-runif(n,a,b)

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rv.mid.int(dat[i],c,int)$rv)
Rv

jit<-.1
yjit<-runif(n,-jit,jit)

Xlim<-range(a,b,dat)
xd<-Xlim[2]-Xlim[1]

plot(cbind(Mc,0),main="vertex region indices for the points", xlab=" ", ylab=" ",
     xlim=Xlim+xd*c(-.05,.05),ylim=3*range(yjit),pch=".",cex=3)
abline(h=0)
points(dat,yjit)
abline(v=c(a,b,Mc),lty=2,col=c(1,1,2))
text(dat,yjit,labels=factor(Rv))
text(cbind(c(a,b,Mc),.02),c("rv=1","rv=2","Mc"))

rv.mid.int(6,c,int)
```

`rv.tetraCC` *The index of the CC-vertex region in a tetrahedron that contains a point*

Description

Returns the index of the vertex whose region contains point `p` in a tetrahedron $th = T(A, B, C, D)$ and vertex regions are based on the circumcenter CC of `th`. (see the plots in the example for illustrations).

The vertices of the tetrahedron `th` are labeled as 1=A, 2=B, 3=C, and 4=C also according to the row number the vertex is recorded in `th`.

If the point, `p`, is not inside `th`, then the function yields `NA` as output. The corresponding vertex region is the polygon whose interior points are closest to that vertex. If `th` is regular tetrahedron, then CC and CM (center of mass) coincide.

See also (Ceyhan (2005, 2010)).

Usage

```
rv.tetraCC(p, th)
```

Arguments

p	A 3D point for which CC-vertex region it resides in is to be determined in the tetrahedron th.
th	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.

Value

A list with two elements

rv	Index of the CC-vertex region that contains point, p in the tetrahedron th
tri	The vertices of the tetrahedron, where row number corresponds to the vertex index in rv.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

rv.tetraCM and rv.triCC

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-10  #try also n<-20

Dt<-runif.stdtetra(n)$g

rv.tetraCC(Dt[1,],tetra)
rv.tetraCC(Dt[5,],tetra)
rv.tetraCC(c(2,2,2),tetra)

Rv<-vector()
for (i in 1:n)
 Rv<-c(Rv,rv.tetraCC(Dt[i,],tetra)$rv)
Rv
```

```
CC<-circ.cent.tetra(tetra)
CC

Xlim<-range(tetra[,1],Dt[,1],CC[1])
Ylim<-range(tetra[,2],Dt[,2],CC[2])
Zlim<-range(tetra[,3],Dt[,3],CC[3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

plot3D::scatter3D(tetra[,1],tetra[,2],tetra[,3], phi =0,theta=40, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
          pch = 20, cex = 1, ticktype = "detailed")
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)
#add the data points
plot3D::points3D(Dt[,1],Dt[,2],Dt[,3], add=TRUE)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3], labels=c("A","B","C","D"), add=TRUE)
plot3D::text3D(CC[1],CC[2],CC[3], labels=c("CC"), add=TRUE)

D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-matrix(rep(CC,6),ncol=3,byrow=TRUE)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty=2)

F1<-int.line.plane(A,CC,B,C,D)
L<-matrix(rep(F1,4),ncol=3,byrow=TRUE); R<-rbind(D4,D5,D6,CC)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=2, add=TRUE,lty=2)

F2<-int.line.plane(B,CC,A,C,D)
L<-matrix(rep(F2,4),ncol=3,byrow=TRUE); R<-rbind(D2,D3,D6,CC)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=3, add=TRUE,lty=2)

F3<-int.line.plane(C,CC,A,B,D)
L<-matrix(rep(F3,4),ncol=3,byrow=TRUE); R<-rbind(D3,D5,D6,CC)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=4, add=TRUE,lty=2)

F4<-int.line.plane(D,CC,A,B,C)
L<-matrix(rep(F4,4),ncol=3,byrow=TRUE); R<-rbind(D1,D2,D4,CC)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=5, add=TRUE,lty=2)

plot3D::text3D(Dt[,1],Dt[,2],Dt[,3], labels=factor(Rv), add=TRUE)

P<-c(.1,.1,.1)
rv.tetraCC(P,tetra)

dat.fr<-data.frame(a=tetra)
rv.tetraCC(P,dat.fr)
```

`rv.tetraCM` *The index of the CM-vertex region in a tetrahedron that contains a point*

Description

Returns the index of the vertex whose region contains point `p` in a tetrahedron $th = T(A, B, C, D)$ and vertex regions are based on the center of mass CM=(A+B+C+D)/4 of `th`. (see the plots in the example for illustrations).

The vertices of the tetrahedron `th` are labeled as 1=A, 2=B, 3=C, and 4=C also according to the row number the vertex is recorded in `th`.

If the point, `p`, is not inside `th`, then the function yields `NA` as output. The corresponding vertex region is the simplex with the vertex, CM, and midpoints of the edges adjacent to the vertex.

See also (Ceyhan (2005, 2010)).

Usage

```
rv.tetraCM(p, th)
```

Arguments

`p`	A 3D point for which CM-vertex region it resides in is to be determined in the tetrahedron `th`.
`th`	Four 3D points, stacked row-wise, each row representing a vertex of the tetrahedron.

Value

A list with two elements

`rv`	Index of the CM-vertex region that contains point, `p` in the tetrahedron `th`
`th`	The vertices of the tetrahedron, where row number corresponds to the vertex index in `rv`.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

See Also

`rv.tetraCC` and `rv.triCM`

Examples

```
A<-c(0,0,0); B<-c(1,0,0); C<-c(1/2,sqrt(3)/2,0); D<-c(1/2,sqrt(3)/6,sqrt(6)/3)
tetra<-rbind(A,B,C,D)

n<-10  #try also n<-20

Dt<-runif.stdtetra(n)$g

rv.tetraCM(Dt[1,],tetra)
rv.tetraCM(Dt[5,],tetra)
rv.tetraCM(c(2,2,2),tetra)

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv, rv.tetraCM(Dt[i,],tetra)$rv )
Rv

Xlim<-range(tetra[,1],Dt[,1])
Ylim<-range(tetra[,2],Dt[,2])
Zlim<-range(tetra[,3],Dt[,3])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]
zd<-Zlim[2]-Zlim[1]

CM<-apply(tetra,2,mean)

plot3D::scatter3D(tetra[,1],tetra[,2],tetra[,3], phi =0,theta=40, bty = "g",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05), zlim=Zlim+zd*c(-.05,.05),
          pch = 20, cex = 1, ticktype = "detailed")
L<-rbind(A,A,A,B,B,C); R<-rbind(B,C,D,C,D,D)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lwd=2)
#add the data points
plot3D::points3D(Dt[,1],Dt[,2],Dt[,3], add=TRUE)

plot3D::text3D(tetra[,1],tetra[,2],tetra[,3], labels=c("A","B","C","D"), add=TRUE)
plot3D::text3D(CM[1],CM[2],CM[3], labels=c("CM"), add=TRUE)

D1<-(A+B)/2; D2<-(A+C)/2; D3<-(A+D)/2; D4<-(B+C)/2; D5<-(B+D)/2; D6<-(C+D)/2;
L<-rbind(D1,D2,D3,D4,D5,D6); R<-matrix(rep(CM,6),ncol=3,byrow=TRUE)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3], add=TRUE,lty=2)

F1<-int.line.plane(A,CM,B,C,D)
L<-matrix(rep(F1,4),ncol=3,byrow=TRUE); R<-rbind(D4,D5,D6,CM)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=2, add=TRUE,lty=2)

F2<-int.line.plane(B,CM,A,C,D)
L<-matrix(rep(F2,4),ncol=3,byrow=TRUE); R<-rbind(D2,D3,D6,CM)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=3, add=TRUE,lty=2)

F3<-int.line.plane(C,CM,A,B,D)
L<-matrix(rep(F3,4),ncol=3,byrow=TRUE); R<-rbind(D3,D5,D6,CM)
```

```
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=4, add=TRUE,lty=2)

F4<-int.line.plane(D,CM,A,B,C)
L<-matrix(rep(F4,4),ncol=3,byrow=TRUE); R<-rbind(D1,D2,D4,CM)
plot3D::segments3D(L[,1], L[,2], L[,3], R[,1], R[,2],R[,3],col=5, add=TRUE,lty=2)

plot3D::text3D(Dt[,1],Dt[,2],Dt[,3], labels=factor(Rv), add=TRUE)

P<-c(.1,.1,.1)
rv.tetraCM(P,tetra)

dat.fr<-data.frame(a=tetra)
rv.tetraCM(P,dat.fr)
```

`rv.tri.cent` *The index of the vertex region in a triangle that contains a given point*

Description

Returns the index of the related vertex in the triangle, `tri`, whose region contains point `p`.

Vertex regions are based on the general center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle `tri`. Vertices of the triangle `tri` are labeled according to the row number the vertex is recorded.

If the point, `p`, is not inside `tri`, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, `M`, and projections from `M` to the edges on the lines joining vertices and `M`. (see the illustration in the examples).

See also (Ceyhan (2005, 2010)).

Usage

```
rv.tri.cent(p, tri, M)
```

Arguments

`p`	A 2D point for which `M`-vertex region it resides in is to be determined in the triangle `tri`.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri`.

Value

A list with two elements

rv	Index of the vertex whose region contains point, p; index of the vertex is the same as the row number in the triangle, tri
tri	The vertices of the triangle, tri, where row number corresponds to the vertex index rv with $rv = 1$ is row 1, $rv = 2$ is row 2, and $rv = 3$ is is row 3.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

rv.triCM, rv.triCC, rv.bastriCC, rv.bastriCM, rv.bastri.cent, and rvTeCM

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
M<-c(1.6,1.0)

P<-c(.4,.2)
rv.tri.cent(P,Tr,M)

P<-c(1.8,.5)
rv.tri.cent(P,Tr,M)

P<-c(1.5,1.6)
rv.tri.cent(P,Tr,M)
#try also rv.tri.cent(P,Tr,M=c(2,2))  #center is not in the interior of the triangle

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also M<-c(1.6,1.0)

Rv<-vector()
for (i in 1:n)
{Rv<-c(Rv,rv.tri.cent(dat[i,],Tr,M)$rv)}
Rv
```

```
Ds<-cp2e.tri(Tr,M)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

xc<-Tr[,1]
yc<-Tr[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(-.02,.04,-.04,0)
yc<-txt[,2]+c(-.02,.04,.05,-.08)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)

text(dat,labels=factor(Rv))

P<-c(1.8,.5)
rv.tri.cent(P,Tr,M)

dat.fr<-data.frame(a=Tr)
rv.tri.cent(P,dat.fr,M)
```

`rv.triCC` *The index of the CC-vertex region in a triangle that contains a point*

Description

Returns the index of the vertex whose region contains point `p` in a triangle `tri`=(A, B, C) and vertex regions are based on the circumcenter CC of `tri`. (see the plots in the example for illustrations).

The vertices of the triangle `tri` are labeled as 1=A,2=B, and 3=C also according to the row number the vertex is recorded in `tri`. If the point, `p`, is not inside `tri`, then the function yields `NA` as output. The corresponding vertex region is the polygon whose interior points are closest to that vertex. If `tri` is equilateral triangle, then CC and CM (center of mass) coincide.

See also (Ceyhan (2005, 2010)).

Usage

```
rv.triCC(p, tri)
```

Arguments

p	A 2D point for which CC-vertex region it resides in is to be determined in the triangle tri.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

A list with two elements

rv	Index of the CC-vertex region that contains point, p in the triangle tri
tri	The vertices of the triangle, where row number corresponds to the vertex index in rv.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

rv.tri.cent, rv.triCM, rv.bastriCM, rv.bastriCC, rv.bastri.cent, and rvTeCM

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(1.3,1.2)
rv.triCC(P,Tr)

P<-c(1.8,.5)
rv.triCC(P,Tr)

P<-c(1.6,1.4)
rv.triCC(P,Tr)

P<-c(.5,.8)
```

```
rv.triCC(P,Tr)

CC<-circ.cent.tri(Tr)  #the circumcenter
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],CC[1])
Ylim<-range(Tr[,2],CC[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,asp=1,xlab="",ylab="",pch=".",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Tr,CC,Ds)
xc<-txt[,1]+c(-.07,.08,.06,.12,-.1,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.03,.0,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

RV1<-(A+.5*(D3-A)+A+.5*(D2-A))/2
RV2<-(B+.5*(D3-B)+B+.5*(D1-B))/2
RV3<-(C+.5*(D2-C)+C+.5*(D1-C))/2

txt<-rbind(RV1,RV2,RV3)
xc<-txt[,1]
yc<-txt[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

n<-10  #try also n<-20
dat<-runif.tri(n,Tr)$g

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rv.triCC(dat[i,],Tr)$rv)
Rv

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,asp=1,xlab="",ylab="",pch=".",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".")
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
text(dat,labels=factor(Rv))

txt<-rbind(Tr,CC,Ds)
```

```
xc<-txt[,1]+c(-.07,.08,.06,.12,-.1,-.1,-.09)
yc<-txt[,2]+c(.02,-.02,.03,.0,.02,.06,-.04)
txt.str<-c("A","B","C","CC","D1","D2","D3")
text(xc,yc,txt.str)

P<-c(1.3,1.2)
rv.triCC(P,Tr)

dat.fr<-data.frame(a=Tr)
rv.triCC(P,dat.fr)
```

`rv.triCM` *The index of the CM-vertex region in a triangle that contains a given point*

Description

Returns the index of the vertex whose region contains point `p` in the triangle `tri`=(y_1, y_2, y_3) with vertex regions are constructed with center of mass $CM = (y_1 + y_2 + y_3)/3$ (see the plots in the example for illustrations).

The vertices of triangle, `tri`, are labeled as 1,2,3 according to the row number the vertex is recorded in `tri`. If the point, `p`, is not inside `tri`, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, CM, and midpoints of the edges adjacent to the vertex.

See (Ceyhan (2005, 2010))

Usage

```
rv.triCM(p, tri)
```

Arguments

`p`	A 2D point for which CM-vertex region it resides in is to be determined in the triangle `tri`.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

A list with two elements

`rv`	Index of the CM-vertex region that contains point, `p` in the triangle `tri`.
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

rv.tri.cent, rv.triCC, rv.bastriCM, rv.bastriCC, rv.bastri.cent, and rvTeCM

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.6,2);
Tr<-rbind(A,B,C);
P<-c(1.4,1.2)
rv.triCM(P,Tr)

P<-c(.8,.2)
rv.triCM(P,Tr)

P<-c(1.5,1.6)
rv.triCM(P,Tr)

n<-10  #try also n<-20
dat<-runif.tri(n,Tr)$g

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rv.triCM(dat[i,],Tr)$rv)
Rv

CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,xlab="",ylab="",axes=TRUE,pch=".",xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".")
L<-Ds; R<-matrix(rep(CM,3),ncol=2,byrow=TRUE)
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
```

```
text(dat,labels=factor(Rv))

txt<-rbind(Tr,CM,D1,D2,D3)
xc<-txt[,1]+c(-.02,.02,.02,-.02,.02,-.01,-.01)
yc<-txt[,2]+c(-.02,-.04,.06,-.02,.02,.06,-.06)
txt.str<-c("rv=1","rv=2","rv=3","CM","D1","D2","D3")
text(xc,yc,txt.str)

rv.triCM(A,Tr)
rv.triCM(CM,Tr)
rv.triCM(D1,Tr)
rv.triCM(D2,Tr)
rv.triCM(D3,Tr)

#right triangle
A<-c(1,1); B<-c(1,2); C<-c(1.5,2);
T3<-rbind(A,B,C);
P<-c(1.1,1.5)
rv.triCM(P,T3)

#isosceles triangle
A<-c(1,1); B<-c(2,1); C<-c(1.5,2);
Tr<-rbind(A,B,C);
P<-c(1.5,1.1)
rv.triCM(P,Tr)

dat.fr<-data.frame(a=Tr)
rv.triCM(P,dat.fr)
```

`rverts.tri.cent` *The indices of the vertex regions in a triangle that contains the points in a give data set*

Description

Returns the indices of the vertices whose regions contain the points in data set `Dt` in a triangle $tri = T(A, B, C)$.

Vertex regions are based on center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of the triangle to the edges on the extension of the lines joining `M` to the vertices or based on the circumcenter of `tri`. Vertices of triangle `tri` are labeled as 1,2,3 according to the row number the vertex is recorded.

If a point in `Dt` is not inside `tri`, then the function yields `NA` as output for that entry. The corresponding vertex region is the polygon with the vertex, `M`, and projection points from `M` to the edges crossing the vertex (as the output of cp2e.tri(Tr,M)) or CC-vertex region. (see the examples for an illustration).

See also (Ceyhan (2005, 2011)).

Usage

```
rverts.tri.cent(Dt, tri, M)
```

Arguments

Dt	A set of 2D points representing the set of data points for which indices of the vertex regions containing them are to be determined.
tri	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
M	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle tri or the circumcenter of tri.

Value

A list with two elements

rv	Indices of the vertices whose regions contains points in Dt.
tri	The vertices of the triangle, where row number corresponds to the vertex index in rv.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

rverts.triCM, rverts.triCC and rverts.tri.nd

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
M<-c(1.6,1.0)

P<-c(.4,.2)
rverts.tri.cent(P,Tr,M)
```

```
P<-c(1.8,.5)
rverts.tri.cent(P,Tr,M)

P<-c(1.5,1.6)
rverts.tri.cent(P,Tr,M)

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-as.numeric(runif.tri(1,Tr)$g)  #try also  #M<-c(1.6,1.0)

rverts.tri.cent(dat,Tr,M)
rverts.tri.cent(rbind(dat,c(2,2)),Tr,M)

rv<-rverts.tri.cent(dat,Tr,M)
rv

ifelse(identical(M,circ.cent.tri(Tr)),
Ds<-rbind((B+C)/2,(A+C)/2,(A+B)/2),Ds<-cp2e.tri(Tr,M))

Xlim<-range(Tr[,1],M[1],dat[,1])
Ylim<-range(Tr[,2],M[2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Tr)}
#need to run this when M is given in barycentric coordinates

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

xc<-Tr[,1]
yc<-Tr[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(.02,.04,-.03,0)
yc<-txt[,2]+c(.07,.04,.05,-.07)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)

text(dat,labels=factor(rv$rv))

P<-c(1.4,1.0)
rverts.tri.cent(P,Tr,M)
rverts.tri.cent(dat,Tr,M)
```

```
rverts.tri.cent(rbind(dat,dat),Tr,M)

dat.fr<-data.frame(a=dat)
rverts.tri.cent(dat.fr,Tr,M)

dat.fr<-data.frame(a=Tr)
rverts.tri.cent(dat,dat.fr,M)
```

`rverts.tri.cent.alt` *The alternative function for the indices of the vertex regions in a triangle that contains the points in a give data set*

Description

An alternative function to the function `rverts.tri.cent`

Usage

```
rverts.tri.cent.alt(Dt, tri, M)
```

Arguments

`Dt`	A set of 2D points representing the set of data points for which indices of the vertex regions containing them are to be determined.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the triangle `tri` or the circumcenter of `tri`.

Value

A list with two elements

`rv`	Indices of the vertices whose regions contains points in `Dt`.
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

References

There are no references for Rd macro \insertAllCites on this help page.

See Also

`rverts.tri.cent`

Examples

```
## Not run:
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
M<-c(1.6,1.0)

P<-c(.4,.2)
rverts.tri.cent.alt(P,Tr,M)

P<-c(1.8,.5)
rverts.tri.cent.alt(P,Tr,M)

P<-c(1.5,1.6)
rverts.tri.cent.alt(P,Tr,M)

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

M<-c(1.6,1.0)  #try also M<-c(1.3,1.3)

rv<-rverts.tri.cent.alt(dat,Tr,M)
rv

Ds<-cp2e.tri(Tr,M)

Xlim<-range(Tr[,1])
Ylim<-range(Tr[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

xc<-Tr[,1]+c(-.03,.05,.05)
yc<-Tr[,2]+c(-.06,.02,.05)
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(.02,.04,-.03,0)
yc<-txt[,2]+c(.07,.03,.05,-.07)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)

text(dat,labels=factor(rv$rv))

P<-c(1.4,1.0)
rverts.tri.cent.alt(P,Tr,M)
```

```
rverts.tri.cent.alt(dat,Tr,M)

rverts.tri.cent.alt(rbind(dat,dat),Tr,M)

dat.fr<-data.frame(a=dat)
rverts.tri.cent.alt(dat.fr,Tr,M)

dat.fr<-data.frame(a=Tr)
rverts.tri.cent.alt(dat,dat.fr,M)

## End(Not run)
```

`rverts.tri.nd` *The indices of the vertex regions in a triangle that contains the points in a give data set*

Description

Returns the indices of the vertices whose regions contain the points in data set `Dt` in a triangle `tri`=(A, B, C) and vertex regions are based on the center `cent` which yields nondegenerate asymptotic distribution of the domination number of PE-PCD for uniform data in `tri` for expansion parameter `r` in $(1, 1.5]$.

Vertices of triangle `tri` are labeled as 1,2,3 according to the row number the vertex is recorded if a point in `Dt` is not inside `tri`, then the function yields `NA` as output for that entry. The corresponding vertex region is the polygon with the vertex, `cent`, and projection points on the edges. The center label `cent` values 1,2,3 correspond to the vertices M_1, M_2, and M_3; with default 1 (see the examples for an illustration).

See also (Ceyhan (2005, 2011)).

Usage

```
rverts.tri.nd(Dt, tri, r, cent = 1)
```

Arguments

`Dt`	A set of 2D points representing the set of data points for which indices of the vertex regions containing them are to be determined.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$ for this function.
`cent`	Index of the center (as 1,2,3 corresponding to M1,M2,M3) which gives nondegenerate asymptotic distribution of the domination number of PE-PCD for uniform data in `tri` for expansion parameter `r` in $(1, 1.5]$; default `cent`=1.

Value

A list with two elements

`rv`	Indices (i.e., a vector of indices) of the vertices whose region contains points in `Dt` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rverts.triCM`, `rverts.triCC` and `rverts.tri.cent`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
r<-1.35
cent<-2

P<-c(1.4,1.0)
rverts.tri.nd(P,Tr,r,cent)

P<-c(1.8,.5)
rverts.tri.nd(P,Tr,r,cent)

P<-c(10.5,1.6)
rverts.tri.nd(P,Tr,r,cent)

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

rverts.tri.nd(dat,Tr,r,cent)
rverts.tri.nd(rbind(dat,c(2,2)),Tr,r,cent)
```

```
rv<-rverts.tri.nd(dat,Tr,r,cent)

M<-cent.nondeg(Tr,r)[cent,];
Ds<-cp2edges.nd(Tr,r,cent)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

xc<-Tr[,1]+c(-.03,.05,.05)
yc<-Tr[,2]+c(-.06,.02,.05)
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(M,Ds)
xc<-txt[,1]+c(.02,.04,-.03,0)
yc<-txt[,2]+c(.07,.03,.05,-.07)
txt.str<-c("M","D1","D2","D3")
text(xc,yc,txt.str)

text(dat,labels=factor(rv$rv))

P<-c(1.4,1.0)
rverts.tri.nd(P,Tr,r,cent)
rverts.tri.nd(dat,Tr,r,cent)

rverts.tri.nd(rbind(dat,dat),Tr,r,cent)

dat.fr<-data.frame(a=dat)
rverts.tri.nd(dat.fr,Tr,r,1)

dat.fr<-data.frame(a=Tr)
rverts.tri.nd(dat,dat.fr,r,1)
```

`rverts.triCC` *The indices of the CC-vertex regions in a triangle that contains the points in a give data set.*

Description

Returns the indices of the vertices whose regions contain the points in data set `Dt` in a triangle `tri`=(A, B, C) and vertex regions are based on the circumcenter CC of `tri`. (see the plots in the

example for illustrations).

The vertices of the triangle `tri` are labeled as 1=A,2=B, and 3=C also according to the row number the vertex is recorded in `tri`. If a point in `Dt` is not inside `tri`, then the function yields `NA` as output. The corresponding vertex region is the polygon whose interior points are closest to that vertex. If `tri` is equilateral triangle, then CC and CM (center of mass) coincide.

See also (Ceyhan (2005, 2010)).

Usage

```
rverts.triCC(Dt, tri)
```

Arguments

`Dt`	A set of 2D points representing the set of data points for which indices of the vertex regions containing them are to be determined.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

A list with two elements

`rv`	Indices (i.e., a vector of indices) of the vertices whose region contains points in `Dt` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

`rverts.triCM`, `rverts.tri.cent`, and `rverts.tri.nd`

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
```

```
P<-c(.4,.2)
rverts.triCC(P,Tr)

P<-c(1.8,.5)
rverts.triCC(P,Tr)

P<-c(10.5,1.6)
rverts.triCC(P,Tr)

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

rverts.triCC(dat,Tr)
rverts.triCC(rbind(dat,c(2,2)),Tr)

rv<-rverts.triCC(dat,Tr)
rv

CC<-circ.cent.tri(Tr)
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],dat[,1],CC[1])
Ylim<-range(Tr[,2],dat[,2],CC[2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",asp=1,xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".",col=1)
L<-matrix(rep(CC,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

xc<-Tr[,1]
yc<-Tr[,2]
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(CC,Ds)
xc<-txt[,1]+c(.04,.04,-.03,0)
yc<-txt[,2]+c(-.07,.04,.06,-.08)
txt.str<-c("CC","D1","D2","D3")
text(xc,yc,txt.str)

text(dat,labels=factor(rv$rv))

P<-c(1.8,.5)
rverts.triCC(P,Tr)

dat.fr<-data.frame(a=dat)
rverts.triCC(dat.fr,Tr)
```

`rverts.triCM` *The indices of the CM-vertex regions in a triangle that contains the points in a give data set*

Description

Returns the indices of the vertices whose regions contain the points in data set `Dt` in a triangle `tri`=(A, B, C) and vertex regions are based on the center of mass CM of `tri`. (see the plots in the example for illustrations).

The vertices of the triangle `tri` are labeled as 1=A,2=B, and 3=C also according to the row number the vertex is recorded in `tri`. If a point in `Dt` is not inside `tri`, then the function yields `NA` as output for that entry. The corresponding vertex region is the polygon with the vertex, CM, and midpoints the edges crossing the vertex.

See also (Ceyhan (2005, 2010)).

Usage

```
rverts.triCM(Dt, tri)
```

Arguments

`Dt`	A set of 2D points representing the set of data points for which indices of the vertex regions containing them are to be determined.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

A list with two elements

`rv`	Indices (i.e., a vector of indices) of the vertices whose region contains points in `Dt` in the triangle `tri`
`tri`	The vertices of the triangle, where row number corresponds to the vertex index in `rv`.

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

rverts.tri.cent, rverts.triCC and rverts.tri.nd

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);

P<-c(.4,.2)
rverts.triCM(P,Tr)

P<-c(1.8,.5)
rverts.triCM(P,Tr)

P<-c(10.5,1.6)
rverts.triCM(P,Tr)

n<-10  #try also n<-20
set.seed(1)
dat<-runif.tri(n,Tr)$g

rverts.triCM(dat,Tr)
rverts.triCM(rbind(dat,c(2,2)),Tr)

rv<-rverts.triCM(dat,Tr)
rv

CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Tr[,1],dat[,1])
Ylim<-range(Tr[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Tr,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
points(dat,pch=".",col=1)
L<-matrix(rep(CM,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

xc<-Tr[,1]+c(-.04,.05,.05)
yc<-Tr[,2]+c(-.05,.05,.03)
txt.str<-c("rv=1","rv=2","rv=3")
text(xc,yc,txt.str)

txt<-rbind(CM,Ds)
xc<-txt[,1]+c(.04,.04,-.03,0)
yc<-txt[,2]+c(-.07,.04,.06,-.08)
txt.str<-c("CM","D1","D2","D3")
text(xc,yc,txt.str)
```

```
text(dat,labels=factor(rv$rv))

P<-c(1.8,.5)
rverts.triCM(P,Tr)

dat.fr<-data.frame(a=dat)
rverts.triCM(dat.fr,Tr)

dat.fr<-data.frame(a=Tr)
rverts.triCM(dat,dat.fr)
```

`rvTe.cent` *The index of the vertex region in the standard equilateral triangle that contains a given point*

Description

Returns the index of the vertex whose region contains point `pt` in standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with vertex regions are constructed with center $M = (m_1, m_2)$ in Cartesian coordinates or $M = (\alpha, \beta, \gamma)$ in barycentric coordinates in the interior of T_e. (see the plots in the example for illustrations).

The vertices of triangle, T_e, are labeled as 1,2,3 according to the row number the vertex is recorded in T_e. If the point, `pt`, is not inside T_e, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, `M`, and projections from `M` to the edges on the lines joining vertices and `M`.

See also (Ceyhan (2005, 2010)).

Usage

```
rvTe.cent(pt, M)
```

Arguments

`pt`	A 2D point for which `M`-vertex region it resides in is to be determined in the standard equilateral triangle T_e.
`M`	A 2D point in Cartesian coordinates or a 3D point in barycentric coordinates which serves as a center in the interior of the standard equilateral triangle T_e.

Value

A list with two elements

`rv`	Index of the vertex whose region contains point, `pt`
`tri`	The vertices of the triangle, T_e, where row number corresponds to the vertex index in `rv` with row 1=(0,0), row 2=(1,0), and row 3=(1/2,sqrt(3)/2).

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

rvTeCM, rv.tri.cent, rv.triCC, rv.bastriCC, rv.triCM, and rv.bastri.cent

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
n<-10  #try also n<-20

set.seed(1)
dat<-runifTe(n)$gen.points

M<-as.numeric(runifTe(1)$g)  #try also M<-c(.6,.2)

rvTe.cent(dat[1,],M)
rvTe.cent(c(.7,.2),M)
rvTe.cent(c(0,1),M)

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rvTe.cent(dat[i,],M)$rv)
Rv

Ds<-cp2e.tri(Te,M)

Xlim<-range(Te[,1],dat[,1])
Ylim<-range(Te[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

if (dimension(M)==3) {M<-bary2cart(M,Te)}
#need to run this when M is given in barycentric coordinates

plot(Te,asp=1,pch=".",xlab="",ylab="",axes=TRUE,
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(dat,pch=".",col=1)
L<-rbind(M,M,M); R<-Ds
```

```
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Te,M)
xc<-txt[,1]+c(-.02,.03,.02,0)
yc<-txt[,2]+c(.02,.02,.03,.05)
txt.str<-c("A","B","C","M")
text(xc,yc,txt.str)

text(dat,labels=factor(Rv))

rvTe.cent(c(.7,.2),M)
```

`rvTeCM` *The index of the CM-vertex region in the standard equilateral triangle that contains a given point*

Description

Returns the index of the vertex whose region contains point `p` in standard equilateral triangle $T_e = T((0,0),(1,0),(1/2,\sqrt{3}/2))$ with vertex regions are constructed with center of mass CM (see the plots in the example for illustrations).

The vertices of triangle, T_e, are labeled as 1,2,3 according to the row number the vertex is recorded in T_e. If the point, `p`, is not inside T_e, then the function yields `NA` as output. The corresponding vertex region is the polygon with the vertex, CM, and midpoints of the edges adjacent to the vertex.

See also (Ceyhan (2005, 2010)).

Usage

```
rvTeCM(pt)
```

Arguments

`pt`	A 2D point for which CM-vertex region it resides in is to be determined in the standard equilateral triangle T_e.

Value

A list with two elements

`rv`	Index of the vertex whose region contains point, `p`.
`tri`	The vertices of the triangle, T_e, where row number corresponds to the vertex index in `rv` with row 1=(0,0), row 2=(1,0), and row 3=(1/2,sqrt(3)/2).

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Ceyhan E (2012). "An investigation of new graph invariants related to the domination number of random proximity catch digraphs." *Methodology and Computing in Applied Probability*, **14(2)**, 299-334.

See Also

rv.bastriCM, rv.tri.cent, rv.triCC, rv.bastriCC, rv.triCM, and rv.bastri.cent

Examples

```
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)

n<-10  #try also n<-20

set.seed(1)
dat<-runifTe(n)$gen.points

rvTeCM(dat[1,])
rvTeCM(c(.7,.2))
rvTeCM(c(0,1))

Rv<-vector()
for (i in 1:n)
  Rv<-c(Rv,rvTeCM(dat[i,])$rv)
Rv

CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

Xlim<-range(Te[,1],dat[,1])
Ylim<-range(Te[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(Te,asp=1,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
points(dat,pch=".",col=1)
L<-matrix(rep(CM,3),ncol=2,byrow=TRUE); R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)

txt<-rbind(Te,CM)
```

```
xc<-txt[,1]+c(-.02,.03,.02,0)
yc<-txt[,2]+c(.02,.02,.03,.05)
txt.str<-c("A","B","C","CM")
text(xc,yc,txt.str)

text(dat,labels=factor(Rv))

rvTeCM(c(.7,.2))
```

`seg.tri.supp` *The auxiliary triangle to define the support of type I segregation*

Description

Returns the triangle whose intersection with a general triangle gives the support for type I segregation given the `delta` (i.e., $\delta 100$ % area of a triangle around the vertices is chopped off). See the plot in the examples.

Caveat: the vertices of this triangle may be outside the triangle, `tri`, depending on the value of `delta` (i.e., for small values of `delta`).

Usage

```
seg.tri.supp(delta, tri)
```

Arguments

`delta`	A positive real number between 0 and 1 that determines the percentage of area of the triangle around the vertices forbidden for point generation.
`tri`	Three 2D points, stacked row-wise, each row representing a vertex of the triangle.

Value

the vertices of the triangle (stacked row-wise) whose intersection with a general triangle gives the support for type I segregation for the given `delta`

See Also

`rsegTe` and `rsegMT`

Examples

```
## Not run:
#the standard equilateral triangle
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C);
delta<-.3  #try also .5,.75,.85
```

```
seg.tri.supp(delta,Te)

Tseg<-seg.tri.supp(delta,Te)

Xlim<-range(Te[,1],Tseg[,1])
Ylim<-range(Te[,2],Tseg[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

oldpar <- par(mfrow = c(1,2))
par(pty="s")
plot(Te,pch=".",xlab="",ylab="",
main="segregation support is the intersection\n of these two triangles"
     ,axes=T,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
polygon(Tseg,lty=2)

txt<-rbind(Te,Tseg)
xc<-txt[,1]+c(-.03,.03,.03,.05,.04,-.04)
yc<-txt[,2]+c(.02,.02,.04,-.03,0,0)
txt.str<-c("A","B","C","T1","T2","T3")
text(xc,yc,txt.str)
par(oldpar)

#for a general triangle
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
delta<-.3  #try also .5,.75,.85
Tseg<-seg.tri.supp(delta,Tr)

Xlim<-range(Tr[,1],Tseg[,1])
Ylim<-range(Tr[,2],Tseg[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

oldpar <- par(mfrow = c(1,2))
par(pty="s")
plot(Tr,pch=".",xlab="",ylab="",
main="segregation support is the intersection\n of these two triangles"
     ,axes=T,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Tr)
polygon(Tseg,lty=2)

txt<-rbind(Tr,Tseg)
xc<-txt[,1]+c(-.03,.03,.03,.06,.04,-.04)
yc<-txt[,2]+c(.02,.02,.04,-.03,0,0)
txt.str<-c("A","B","C","T1","T2","T3")
text(xc,yc,txt.str)
par(oldpar)

dat.fr<-data.frame(a=Tr)
seg.tri.supp(delta,dat.fr)
```

```
## End(Not run)
```

---

six.ext *The closest points among a data set in the standard equilateral triangle to the median lines in 6 half edge regions*

Description

An object of class "Extrema". Returns the six closest points among the data set, Dt, in the standard equilateral triangle $T_e = T(A = (0,0), B = (1,0), C = (1/2, \sqrt{3}/2))$ in half edge regions. In particular, in regions r_1 and r_6, it finds the closest point in each region to the line segment [A,CM] in regions r_2 and r_3, it finds the closest point in each region to the line segment [B,CM] and in regions r_4 and r_5, it finds the closest point in each region to the line segment [C,CM] where $CM = (A + B + C)/3$ is the center of mass.

See the example for this function or example for rel.six.Te function. If there is no data point in region r_i, then it returns "NA NA" for ith row in the extrema. ch.all.intri is for checking whether all data points are in T_e (default is FALSE).

Usage

```
six.ext(Dt, ch.all.intri = FALSE)
```

Arguments

Dt	A set of 2D points among which the closest points in the standard equilateral triangle to the median lines in 6 half edge regions.
ch.all.intri	A logical argument for checking whether all data points are in T_e (default is FALSE).

Value

A list with the elements

txt1	Region labels as r1-r6 (corresponds to row number in Extrema Points).
txt2	A short description of the distances as "Distances to Line Segments (A,CM), (B,CM), and (C,CM) in the six regions r1-r6".
type	Type of the extrema points
mtitle	The "main" title for the plot of the exrema
ext	The extrema points, here, closest points in each of regions r1-r6 to the line segments joining vertices to the center of mass, CM.
X	The input data, Dt, can be a matrix or data frame
num.points	The number of data points, i.e., size of Dt
supp	Support of the data points, here, it is T_e.
cent	The center point used for construction of edge regions.

ncent	Name of the center, cent, it is center of mass "CM" for this function.
regions	The six regions, r1-r6 and edge regions inside the triangle, T_e, provided as a list.
region.names	Names of the regions as "r1"-"r6" and names of the edge regions as "er=1","er=2","er=3".
region.centers	Centers of mass of the regions r1-r6 and of edge regions inside T_e.
dist2ref	Distances from closest points in each of regions r1-r6 to the line segments joining vertices to the center of mass, CM.

See Also

rel.six.Te and cl2edgesTe

Examples

```
## Not run:
n<-10  #try also n<-100
dat<-runifTe(n)$gen.points

Ext<-six.ext(dat)
Ext
summary(Ext)
plot(Ext)

six.ext(dat[1:5,])$ext
sixt<-six.ext(dat)

A<-c(0,0); B<-c(1,0); C<-c(0.5,sqrt(3)/2);
Te<-rbind(A,B,C)
CM<-(A+B+C)/3
D1<-(B+C)/2; D2<-(A+C)/2; D3<-(A+B)/2;
Ds<-rbind(D1,D2,D3)

h1<-c(1/2,sqrt(3)/18); h2<-c(2/3, sqrt(3)/9); h3<-c(2/3, 2*sqrt(3)/9);
h4<-c(1/2, 5*sqrt(3)/18); h5<-c(1/3, 2*sqrt(3)/9); h6<-c(1/3, sqrt(3)/9);

r1<-(h1+h6+CM)/3;r2<-(h1+h2+CM)/3;r3<-(h2+h3+CM)/3;
r4<-(h3+h4+CM)/3;r5<-(h4+h5+CM)/3;r6<-(h5+h6+CM)/3;

Xlim<-range(Te[,1],dat[,1])
Ylim<-range(Te[,2],dat[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

plot(A,pch=".",xlab="",ylab="",axes=TRUE,xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05))
polygon(Te)
L<-Te; R<-Ds
segments(L[,1], L[,2], R[,1], R[,2], lty=2)
polygon(rbind(h1,h2,h3,h4,h5,h6))
points(dat)
points(sixt$ext,pty=2,pch=4,col="red")

txt<-rbind(Te,r1,r2,r3,r4,r5,r6)
```

```
xc<-txt[,1]+c(-.02,.02,.02,0,0,0,0,0,0)
yc<-txt[,2]+c(.02,.02,.03,0,0,0,0,0,0)
txt.str<-c("A","B","C","1","2","3","4","5","6")
text(xc,yc,txt.str)

dat.fr<-data.frame(a=dat)
six.ext(dat.fr)

dat2<-rbind(dat,c(.8,.8))
six.ext(dat2)

six.ext(dat2,ch.all.intri = TRUE)
#gives an error message since not all data points are in the std equilateral triangle

## End(Not run)
```

`slope` *The slope of a line*

Description

Returns the slope of the line joining two distinct 2D points `a` and `b`.

Usage

```
slope(a, b)
```

Arguments

`a, b`	2D points that determine the straight line (i.e., through which the straight line passes).

Value

Slope of the line joining 2D points `a` and `b`

See Also

`Line`, `paraline`, and `perpline`

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75)
slope(A,B)

slope(c(1,2),c(2,3))
slope(c(1,2),c(1,3))
```

summary.Extrema *Return a summary of a* Extrema *object*

Description

Returns the below information about the object:

call of the function defining the object, the type of the extrema (i.e. the description of the extrema), extrema points, distances from extrema to the reference object (e.g. boundary of a triangle), some of the data points (from which extrema is found).

Usage

```
## S3 method for class 'Extrema'
summary(object, ...)
```

Arguments

object	Object of class Extrema.
...	Additional parameters for summary.

Value

The call of the object of class 'Extrema', the type of the extrema (i.e. the description of the extrema), extrema points, distances from extrema to the reference object (e.g. boundary of a triangle), some of the data points (from which extrema is found).

See Also

print.Extrema, print.summary.Extrema, and plot.Extrema

Examples

```
n<-20
dat<-runifTe(n)$gen.points
Ext<-cl2edgesTe(dat)
Ext
summary(Ext)
```

summary.Lines *Return a summary of a* Lines *object*

Description

Returns the below information about the object:

call of the function defining the object, the defining points, selected x and y points on the line, equation of the line, and coefficients of the line.

Usage

```
## S3 method for class 'Lines'
summary(object, ...)
```

Arguments

object	Object of class Lines.
...	Additional parameters for summary.

Value

The call of the object of class 'Lines', the defining points, selected x and y points on the line, equation of the line, and coefficients of the line (in the form: y = slope * x + intercept).

See Also

print.Lines, print.summary.Lines, and plot.Lines

Examples

```
A<-c(-1.22,-2.33); B<-c(2.55,3.75)
xr<-range(A,B);
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=100)

lnAB<-Line(A,B,x)
lnAB
summary(lnAB)
```

summary.Lines3D *Return a summary of a* Lines3D *object*

Description

Returns the below information about the object:

call of the function defining the object, the defining vectors (i.e., initial and direction vectors), selected x, y and z points on the line, equation of the line (in parametric form), and coefficients of the line.

Usage

```
## S3 method for class 'Lines3D'
summary(object, ...)
```

Arguments

object	Object of class Lines3D.
...	Additional parameters for summary.

Value

call of the function defining the object, the defining vectors (i.e., initial and direction vectors), selected x, y and z points on the line, equation of the line (in parametric form), and coefficients of the line (for the form: x=x0 + a*t, y=y0 + b*t, and z=z0 + c*t).

See Also

print.Lines3D, print.summary.Lines3D, and plot.Lines3D

Examples

```
A<-c(1,10,3); B<-c(1,1,3);
vecs<-rbind(A,B)
Line3D(A,B,.1)
Line3D(A,B,.1,dir.vec=FALSE)

tr<-range(vecs);
tf<-(tr[2]-tr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
tsq<-seq(-tf*10-tf,tf*10+tf,l=100)

lnAB3D<-Line3D(A,B,tsq)
lnAB3D
summary(lnAB3D)
```

summary.Patterns *Return a summary of a* Patterns *object*

Description

Returns the below information about the object:

call of the function defining the object, the type of the pattern, parameters of the pattern, study window, some sample points from the generated pattern, reference points (if any for the bivariate pattern), and number of points for each class

Usage

```
## S3 method for class 'Patterns'
summary(object, ...)
```

Arguments

object	Object of class Patterns.
...	Additional parameters for summary.

Value

The call of the object of class 'Patterns', the type of the pattern, parameters of the pattern, study window, some sample points from the generated pattern, reference points (if any for the bivariate pattern), and number of points for each class

See Also

print.Patterns, print.summary.Patterns, and plot.Patterns

Examples

```
nx<-20; #try also 100 and 1000
ny<-3; #try also 1
e<-.15;
Y<-cbind(runif(ny),runif(ny)) #with default bounding box (i.e., unit square)

Xdt<-rseg.disc(nx,Y,e)
Xdt
summary(Xdt)
```

summary.PCDs *Return a summary of a* PCDs *object*

Description

Returns the below information about the object:

call of the function defining the object, the type of the proximity catch digraph (PCD), (i.e. the description of the PCD), some of the partition (i.e. intervalization in the 1D case and triangulation in the 2D case) points (i.e., vertices of the intervals or the triangles), some of the tails (or source points) and the heads (or end points) of the arcs of the PCD, parameter of the PCD, and various quantitites (number of vertices, number of arcs and arc density of the PCDs, number of vertices for the partition and number of partition cells (i.e., intervals or triangles)).

Usage

```
## S3 method for class 'PCDs'
summary(object, ...)
```

Arguments

object	Object of class PCDs.
...	Additional parameters for summary.

Value

The call of the object of class 'PCDs', the type of the proximity catch digraph (PCD), (i.e. the description of the PCD), some of the partition (i.e. intervalization in the 1D case and triangulation in the 2D case) points (i.e., vertices of the intervals or the triangles), some of the tails (or source points) and the heads (or end points) of the arcs of the PCD, parameter of the PCD, and various quantitites (number of vertices, number of arcs and arc density of the PCDs, number of vertices for the partition and number of partition cells (i.e., intervals or triangles)).

See Also

print.PCDs, print.summary.PCDs, and plot.PCDs

Examples

```
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C);
n<-10
dat<-runif.tri(n,Tr)$g
M<-as.numeric(runif.tri(1,Tr)$g)
Arcs<-ArcsAStri(dat,Tr,M)
Arcs
summary(Arcs)
```

summary.Planes *Return a summary of a* Planes *object*

Description

Returns the below information about the object:

call of the function defining the object, the defining 3D points, selected x, y and z points on the plane, equation of the plane, and coefficients of the plane.

Usage

```
## S3 method for class 'Planes'
summary(object, ...)
```

Arguments

object	Object of class Planes.
...	Additional parameters for summary.

Value

The call of the object of class 'Planes', the defining 3D points, selected x, y and z points on the plane, equation of the plane, and coefficients of the plane (in the form: z = A*x + B*y + C).

See Also

print.Planes, print.summary.Planes, and plot.Planes

Examples

```
A<-c(1,10,3); B<-c(1,1,3); C<-c(3,9,12)
pts<-rbind(A,B,C)

xr<-range(pts[,1]); yr<-range(pts[,2])
xf<-(xr[2]-xr[1])*.1 #how far to go at the lower and upper ends in the x-coordinate
yf<-(yr[2]-yr[1])*.1 #how far to go at the lower and upper ends in the y-coordinate
x<-seq(xr[1]-xf,xr[2]+xf,l=100)
y<-seq(yr[1]-yf,yr[2]+yf,l=100)

plABC<-Plane(A,B,C,x,y)
plABC
summary(plABC)
```

summary.TriLines *Return a summary of a* TriLines *object*

Description

Returns the below information about the object:

call of the function defining the object, the defining points, selected x and y points on the line, equation of the line, together with the vertices of the triangle, and coefficients of the line.

Usage

```
## S3 method for class 'TriLines'
summary(object, ...)
```

Arguments

object	Object of class TriLines.
...	Additional parameters for summary.

Value

The call of the object of class 'TriLines', the defining points, selected x and y points on the line, equation of the line, together with the vertices of the triangle, and coefficients of the line (in the form: y = slope * x + intercept).

See Also

print.TriLines, print.summary.TriLines, and plot.TriLines

Examples

```
## Not run:
A<-c(0,0); B<-c(1,0); C<-c(1/2,sqrt(3)/2);
Te<-rbind(A,B,C)
xfence<-abs(A[1]-B[1])*.25 #how far to go at the lower and upper ends in the x-coordinate
x<-seq(min(A[1],B[1])-xfence,max(A[1],B[1])+xfence,by=.01)

lnACM<-lA_CM.Te(x)
lnACM
summary(lnACM)

## End(Not run)
```

summary.Uniform *Return a summary of a* Uniform *object*

Description

Returns the below information about the object:

call of the function defining the object, the type of the pattern (i.e. the description of the uniform distribution), study window, vertices of the support of the Uniform distribution, some sample points generated from the uniform distribution, and the number of points (i.e., number of generated points and the number of vertices of the support of the uniform distribution.)

Usage

```
## S3 method for class 'Uniform'
summary(object, ...)
```

Arguments

object	Object of class Uniform.
...	Additional parameters for summary.

Value

The call of the object of class 'Uniform', the type of the pattern (i.e. the description of the uniform distribution), study window, vertices of the support of the Uniform distribution, some sample points generated from the uniform distribution, and the number of points (i.e., number of generated points and the number of vertices of the support of the uniform distribution.)

See Also

print.Uniform, print.summary.Uniform, and plot.Uniform

Examples

```
n<-10 #try also 100
A<-c(1,1); B<-c(2,0); C<-c(1.5,2);
Tr<-rbind(A,B,C)

Xdt<-runif.tri(n,Tr)
Xdt
summary(Xdt)
```

`swamptrees` *Tree Species in a Swamp Forest*

Description

Locations and species classification of trees in a plot in the Savannah River, SC, USA. Locations are given in meters, rounded to the nearest 0.1 decimal. The data come from a one-hectar (200-by-50m) plot in the Savannah River Site. The 734 mapped stems included 156 Carolina ashes (Fraxinus caroliniana), 215 Water tupelos (Nyssa aquatica), 205 Swamp tupelos (Nyssa sylvatica), 98 Bald cypresses (Taxodium distichum) and 60 stems from 8 additional three species (labeled as Others (OT)). The plots were set up by Bill Good and their spatial patterns described in (Good and Whipple (1982)), the plots have been maintained and resampled by Rebecca Sharitz and her colleagues of the Savannah River Ecology Laboratory. The data and some of its description are borrowed from the swamp data entry in the dixon package in the CRAN repository.

See also (Good and Whipple (1982); Jones et al. (1994); Dixon (2002)).

Usage

```
data(swamptrees)
```

Format

A data frame with 734 rows and 4 variables

Details

Text describing the variable (i.e., column) names in the data set.

- x,y: x and y (i.e., Cartesian) coordinates of the trees
- live: a categorical variable that indicates the tree is alive (labeled as 1) or dead (labeled as 0)
- sp: species label of the trees:
  - FX: Carolina ash (Fraxinus caroliniana)
  - NS: Swamp tupelo (Nyssa sylvatica)
  - NX: Water tupelo (Nyssa aquatica)
  - TD: Bald cypress (Taxodium distichum)
  - OT: Other species

Source

Prof. Philip Dixon's website

References

Dixon PM (2002). "Nearest-neighbor contingency table analysis of spatial segregation for several species." *Ecoscience*, **9(2)**, 142-151.

Good BJ, Whipple SA (1982). "Tree spatial patterns: South Carolina bottomland and swamp forests." *Bulletin of the Torrey Botanical Club*, **109(4)**, 529-536.

Jones RH, Sharitz RR, James SM, Dixon PM (1994). "Tree population dynamics in seven South Carolina mixed-species forests." *Bulletin of the Torrey Botanical Club*, **121(4)**, 360-368.

Examples

```
data(swamptrees)
plot(swamptrees$x,swamptrees$y, col=as.numeric(swamptrees$sp),pch=19,
xlab='',ylab='',main='Swamp Trees')
```

`TSArcDensCS1D` *A test of uniformity of 1D data in a given interval based on Central Similarity Proximity Catch Digraph (CS-PCD)*

Description

An object of class "htest" (i.e., hypothesis test) function which performs a hypothesis test of uniformity of 1D data in one interval based on the normal approximation of the arc density of the CS-PCD with expansion parameter $t > 0$ and centrality parameter `c` in $(0, 1)$.

The function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is the arc density), and method and name of the data set used.

The null hypothesis is that data is uniform in a finite interval (i.e. arc density of CS-PCD equals to its expected value under uniform distribution) and `alternative` could be two-sided, or left-sided (i.e. data is accumulated around the end points) or right-sided (i.e. data is accumulated around the mid point or center M_c).

See also (Ceyhan (2016)).

Usage

```
TSArcDensCS1D(
  dat,
  t,
  c,
  int,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

dat	A set or vector of 1D points which constitute the vertices of CS-PCD.
t	A positive real number which serves as the expansion parameter in CS proximity region.
c	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
int	A vector of two real numbers representing an interval.
alternative	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater".
conf.level	Level of the confidence interval, default is 0.95, for the arc density of CS-PCD based on the 1D data set dat.

Value

A list with the elements

statistic	Test statistic
p.value	The p-value for the hypothesis test for the corresponding alternative
conf.int	Confidence interval for the arc density at the given level conf.level and depends on the type of alternative.
estimate	Estimate of the parameter, i.e., arc density
null.value	Hypothesized value for the parameter, i.e., the null arc density, which is usually the mean arc density under uniform distribution.
alternative	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater"
method	Description of the hypothesis test
data.name	Name of the data set

References

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

TSArcDensPE1D

Examples

```
c<-.4
t<-2
a<-0; b<-10; int<-c(a,b)

n<-10
dat<-runif(n,a,b)

NumArcsCSmid1D(dat,t,c,int)
```

```
TSArcDensCS1D(dat,t,c,int)

NumArcsCSmid1D(dat,t,c=.3,int)
TSArcDensCS1D(dat,t,c=.3,int)

NumArcsCSmid1D(dat,t=1.5,c,int)
TSArcDensCS1D(dat,t=1.5,c,int)

dat<-runif(n,a-1,b+1)
NumArcsCSmid1D(dat,t,c,int)
TSArcDensCS1D(dat,t,c,int)

c<-.4
t<-.5
a<-0; b<-10; int<-c(a,b)
n<-10  #try also n<-20
dat<-runif(n,a,b)

TSArcDensCS1D(dat,t,c,int)
```

`TSArcDensCSMT` *A test of segregation/association based on arc density of Central Similarity Proximity Catch Digraph (CS-PCD) for 2D data*

Description

An object of class "htest" (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of `Xp` points in the convex hull of `Yp` points against the alternatives of segregation (where `Xp` points cluster away from `Yp` points) and association (where `Xp` points cluster around `Yp` points) based on the normal approximation of the arc density of the CS-PCD for uniform 2D data in the convex hull of `Yp` points.

The function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is the arc density), and method and name of the data set used.

Under the null hypothesis of uniformity of `Xp` points in the convex hull of `Yp` points, arc density of CS-PCD whose vertices are `Xp` points equals to its expected value under the uniform distribution and `alternative` could be two-sided, or left-sided (i.e. data is accumulated around the `Yp` points, or association) or right-sided (i.e. data is accumulated around the centers of the triangles, or segregation).

CS proximity region is constructed with the expansion parameter $t > 0$ and CM-edge regions (i.e., the test is not available for a general center M at this version of the function).

This test is more appropriate when supports of `Xp` and `Yp` has a substantial overlap.

`ch.cor` is for convex hull correction (default is "no convex hull correction", i.e., `ch.cor=FALSE`) which is recommended when both `Xp` and `Yp` have the same rectangular support.

See also (Ceyhan (2005); Ceyhan et al. (2007); Ceyhan (2014)).

Usage

```
TSArcDensCSMT(
  Xp,
  Yp,
  t,
  ch.cor = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`Xp`	A set of 2D points which constitute the vertices of the CS-PCD.
`Yp`	A set of 2D points which constitute the vertices of the Delaunay triangles.
`t`	A positive real number which serves as the expansion parameter in CS proximity region.
`ch.cor`	A logical argument for convex hull correction, default `ch.cor=FALSE`, recommended when both `Xp` and `Yp` have the same rectangular support.
`alternative`	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater".
`conf.level`	Level of the confidence interval, default is 0.95, for the arc density of CS-PCD based on the 2D data set `Xp`.

Value

A list with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the arc density at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., arc density
`null.value`	Hypothesized value for the parameter, i.e., the null arc density, which is usually the mean arc density under uniform distribution.
`alternative`	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater"
`method`	Description of the hypothesis test
`data.name`	Name of the data set

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E (2014). "Comparison of Relative Density of Two Random Geometric Digraph Families in Testing Spatial Clustering." *TEST*, **23(1)**, 100-134.

Ceyhan E, Priebe C, Marchette D (2007). "A new family of random graphs for testing spatial segregation." *Canadian Journal of Statistics*, **35(1)**, 27-50.

See Also

TSArcDensPEMT and TSArcDensCS1D

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-40; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny),runif(ny))

oldpar <- par(mfrow = c(1,2))
plotDeltri(Xp,Yp,xlab="",ylab = "")
par(oldpar)

NumArcsCSMT(Xp,Yp,t=.5)

TSArcDensCSMT(Xp,Yp,t=.5)
TSArcDensCSMT(Xp,Yp,t=.5,ch=TRUE)

NumArcsCSMT(Xp,Yp,t=1.0)
TSArcDensCSMT(Xp,Yp,t=1.0)
TSArcDensCSMT(Xp,Yp,t=1.0,ch=TRUE)

NumArcsCSMT(Xp,Yp,t=1.5)
TSArcDensCSMT(Xp,Yp,t=1.5)
TSArcDensCSMT(Xp,Yp,t=1.5,ch=TRUE)

t<-2
TSArcDensCSMT(Xp,Yp,t)

Xp<-runif.tri(nx,Yp[1:3,])$g
TSArcDensCSMT(Xp,Yp[1:3,],t)

TSArcDensCSMT(Xp,rbind(Yp,Yp),t)

dat.fr<-data.frame(a=Xp)
TSArcDensCSMT(dat.fr,Yp,t)

dat.fr<-data.frame(a=Yp)
TSArcDensCSMT(Xp,dat.fr,t)

## Not run:
TSArcDensCSMT(c(.4,.2),Yp,t=.5)
#gives an error message since not enough points in the convex hull of non-target points to
#compute arc density of the target points
```

```
## End(Not run)
```

`TSArcDensPE1D` *A test of uniformity of 1D data in a given interval based on Proportional Edge Proximity Catch Digraph (PE-PCD)*

Description

An object of class "htest". This is an "htest" (i.e., hypothesis test) function which performs a hypothesis test of uniformity of 1D data in one interval based on the normal approximation of the arc density of the PE-PCD with expansion parameter $r \geq 1$ and centrality parameter `c` in $(0, 1)$.

The function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is the arc density), and method and name of the data set used.

The null hypothesis is that data is uniform in a finite interval (i.e. arc density of PE-PCD equals to its expected value under uniform distribution) and `alternative` could be two-sided, or left-sided (i.e. data is accumulated around the end points) or right-sided (i.e. data is accumulated around the mid point or center M_c).

See also (Ceyhan (2012, 2016)).

Usage

```
TSArcDensPE1D(
  dat,
  r,
  c,
  int,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`dat`	A set or vector of 1D points which constitute the vertices of PE-PCD.
`r`	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
`c`	A positive real number in $(0, 1)$ parameterizing the center inside $int = (a, b)$. For the interval, $int = (a, b)$, the parameterized center is $M_c = a + c(b - a)$.
`int`	A vector of two real numbers representing an interval.
`alternative`	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater".
`conf.level`	Level of the confidence interval, default is 0.95, for the arc density of PE-PCD based on the 1D data set `dat`.

Value

A list with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the arc density at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., arc density
`null.value`	Hypothesized value for the parameter, i.e., the null arc density, which is usually the mean arc density under uniform distribution.
`alternative`	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater"
`method`	Description of the hypothesis test
`data.name`	Name of the data set

References

Ceyhan E (2012). "The Distribution of the Relative Arc Density of a Family of Interval Catch Digraph Based on Uniform Data." *Metrika*, **75(6)**, 761-793.

Ceyhan E (2016). "Density of a Random Interval Catch Digraph Family and its Use for Testing Uniformity." *REVSTAT*, **14(4)**, 349-394.

See Also

TSArcDensCS1D

Examples

```
c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)

n<-10  #try also n<-20
dat<-runif(n,a,b)

NumArcsPEint(dat,r,c,int)
TSArcDensPE1D(dat,r,c,int)
TSArcDensPE1D(dat,r,c,int,alt="g")
TSArcDensPE1D(dat,r,c,int,alt="l")

NumArcsPEint(dat,r,c=.3,int)
TSArcDensPE1D(dat,r,c=.3,int)

NumArcsPEint(dat,r=1.5,c,int)
TSArcDensPE1D(dat,r=1.5,c,int)

dat<-runif(n,a-1,b+1)
NumArcsPEint(dat,r,c,int)
```

```
TSArcDensPE1D(dat,r,c,int)

c<-.4
r<-2
a<-0; b<-10; int<-c(a,b)
n<-10  #try also n<-20
dat<-runif(n,a,b)
TSArcDensPE1D(dat,r,c,int)
```

TSArcDensPEMT *A test of segregation/association based on arc density of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D data*

Description

An object of class "htest" (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of Xp points in the convex hull of Yp points against the alternatives of segregation (where Xp points cluster away from Yp points) and association (where Xp points cluster around Yp points) based on the normal approximation of the arc density of the PE-PCD for uniform 2D data in the convex hull of Yp points.

The function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is the arc density), and method and name of the data set used.

Under the null hypothesis of uniformity of Xp points in the convex hull of Yp points, arc density of PE-PCD whose vertices are Xp points equals to its expected value under the uniform distribution and alternative could be two-sided, or left-sided (i.e. data is accumulated around the Yp points, or association) or right-sided (i.e. data is accumulated around the centers of the triangles, or segregation).

PE proximity region is constructed with the expansion parameter $r \geq 1$ and CM-vertex regions (i.e., the test is not available for a general center M at this version of the function). This test is more appropriate when supports of Xp and Yp has a substantial overlap,

ch.cor is for convex hull correction (default is "no convex hull correction", i.e., ch.cor=FALSE) which is recommended when both Xp and Yp have the same rectangular support.

See also (Ceyhan (2005); Ceyhan et al. (2006)) for more on the test based on the arc density of PE-PCDs.

Usage

```
TSArcDensPEMT(
  Xp,
  Yp,
  r,
  ch.cor = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

Xp	A set of 2D points which constitute the vertices of the PE-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be ≥ 1.
ch.cor	A logical argument for convex hull correction, default ch.cor=FALSE, recommended when both Xp and Yp have the same rectangular support.
alternative	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater".
conf.level	Level of the confidence interval, default is 0.95, for the arc density of PE-PCD based on the 2D data set Xp.

Value

A list with the elements

statistic	Test statistic
p.value	The p-value for the hypothesis test for the corresponding alternative
conf.int	Confidence interval for the arc density at the given confidence level conf.level and depends on the type of alternative.
estimate	Estimate of the parameter, i.e., arc density
null.value	Hypothesized value for the parameter, i.e., the null arc density, which is usually the mean arc density under uniform distribution.
alternative	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater"
method	Description of the hypothesis test
data.name	Name of the data set

References

Ceyhan E (2005). *An Investigation of Proximity Catch Digraphs in Delaunay Tessellations, also available as technical monograph titled "Proximity Catch Digraphs: Auxiliary Tools, Properties, and Applications"*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD, 21218.

Ceyhan E, Priebe C, Wierman J (2006). "Relative density of the random r-factor proximity catch digraphs for testing spatial patterns of segregation and association." *Computational Statistics \& Data Analysis*, **50(8)**, 1925-1964.

See Also

TSArcDensCSMT and TSArcDensPE1D

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
```

```
Xp<-cbind(runif(nx),runif(nx))
Yp<-cbind(runif(ny),runif(ny))

oldpar <- par(mfrow = c(1,2))
plotDeltri(Xp,Yp,xlab="",ylab="")
par(oldpar)

NumArcsPEMT(Xp,Yp,r=1.25)
TSArcDensPEMT(Xp,Yp,r=1.25)
TSArcDensPEMT(Xp,Yp,r=1.25,ch=TRUE)

NumArcsPEMT(Xp,Yp,r=1.5)
TSArcDensPEMT(Xp,Yp,r=1.5,alt="l")
TSArcDensPEMT(Xp,Yp,r=1.5,ch=TRUE,alt="l")

NumArcsPEMT(Xp,Yp,r=2)
TSArcDensPEMT(Xp,Yp,r=2)
TSArcDensPEMT(Xp,Yp,r=2,ch=TRUE)

r<-2
TSArcDensPEMT(Xp,Yp,r)

Xp<-runif.tri(nx,Yp[1:3,])$g
TSArcDensPEMT(Xp,Yp[1:3,],r)


TSArcDensPEMT(Xp,rbind(Yp,Yp),r)

dat.fr<-data.frame(a=Xp)
TSArcDensPEMT(dat.fr,Yp,r)

dat.fr<-data.frame(a=Yp)
TSArcDensPEMT(Xp,dat.fr,r)
```

`TSDomPEBin` *A test of segregation/association based on domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D data - Binomial Approximation*

Description

An object of class "htest" (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of `Xp` points in the convex hull of `Yp` points against the alternatives of segregation (where `Xp` points cluster away from `Yp` points i.e. cluster around the centers of the Delaunay triangles) and association (where `Xp` points cluster around `Yp` points) based on the (asymptotic) binomial distribution of the domination number of PE-PCD for uniform 2D data in the convex hull of `Yp` points.

The function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is Pr(Domination Number=3)), and method and name of the data set used.

Under the null hypothesis of uniformity of Xp points in the convex hull of Yp points, probability of success (i.e. Pr(Domination Number=3)) equals to its expected value under the uniform distribution) and alternative could be two-sided, or left-sided (i.e. data is accumulated around the Yp points, or association) or right-sided (i.e. data is accumulated around the centers of the triangles, or segregation).

PE proximity region is constructed with the expansion parameter $r \geq 1$ and M-vertex regions where M is a center that yields non-degenerate asymptotic distribution of the domination number.

The test statistic is based on the binomial distribution, when domination number is scaled to have value 0 and 1 in the one triangle case (i.e., Domination Number - 2 for the one triangle case). That is, the test statistic is based on the domination number for Xp points inside convex hull of Yp points for the PE-PCD and default convex hull correction, ch.cor, is FALSE where M is the center that yields nondegenerate asymptotic distribution for the domination number. For this approximation to work, Yp must be at least 10 (i.e. about 15 or more Delaunay triangles) and Xp must be at least 7 times more than Yp points.

See also (Ceyhan (2011)).

Usage

```
TSDomPEBin(
  Xp,
  Yp,
  r,
  ch.cor = F,
  nt = NULL,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

Xp	A set of 2D points which constitute the vertices of the PE-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$.
ch.cor	A logical argument for convex hull correction, default ch.cor=FALSE, recommended when both Xp and Yp have the same rectangular support.
nt	Number of Delaunay triangles based on Yp points, default is NULL.
alternative	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater".
conf.level	Level of the confidence interval, default is 0.95, for the probability of success (i.e. Pr(domination number=3) for PE-PCD whose vertices are the 2D data set Xp.

Value

A list with the elements

`statistic`	Test statistic
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for Pr(Domination Number=3) at the given level `conf.level` and depends on the type of `alternative`.
`estimate`	A vector with two entries: first is is the estimate of the parameter, i.e., Pr(Domination Number=3) and second is the domination number
`null.value`	Hypothesized value for the parameter, i.e., the null value for Pr(Domination Number=3)
`alternative`	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater"
`method`	Description of the hypothesis test
`data.name`	Name of the data set

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

TSDomPENor

Examples

```
nx<-20; ny<-4  #try also nx<-1000; ny<-10
r<-1.4  #try also r<-1.5

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

oldpar <- par(mfrow = c(1,2))
plotDeltri(Xp,Yp,xlab="",ylab="")
par(oldpar)

PEdomMTnd(Xp,Yp,r)

TSDomPEBin(Xp,Yp,r,alt="t")
TSDomPEBin(Xp,Yp,r,alt="l")
TSDomPEBin(Xp,Yp,r,alt="g")
TSDomPEBin(Xp,Yp,r,ch=TRUE)
TSDomPEBin(Xp,Yp,r=1.25)

#or try
ndt<-NumDelTri(Yp)
TSDomPEBin(Xp,Yp,r,nt=ndt)
#values might differ due to the random of choice of the three centers M1,M2,M3
```

```
#for the non-degenerate asymptotic distribution of the domination number

TSDomPEBin(Xp,Yp,r)
TSDomPEBin(Xp,Yp[1:3,],r)

TSDomPEBin(Xp,rbind(Yp,Yp),r)

dat.fr<-data.frame(a=Xp)
TSDomPEBin(dat.fr,Yp,r)

dat.fr<-data.frame(a=Yp)
TSDomPEBin(Xp,dat.fr,r)
```

TSDomPEBin1D — *A test of segregation/association based on domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) for 1D data - Binomial Approximation*

Description

An object of class "htest" (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of Xp points within the subintervals based on Yp points (both residing in the interval (a, b)).

If Yp=NULL the support interval (a, b) is partitioned as Yp=(b-a)*(0:nint)/nint where nint=round(sqrt(nx),0) and nx is number of Xp points, and the test is for testing the uniformity of Xp points in the interval (a, b). If Yp points are given, the test is for testing the spatial interaction between Xp and Yp points.

In either case, the null hypothesis is uniformness of Xp points on (a,b). Yp determines the end points of the intervals (i.e., partition the real line via intervalization) where end points are the order statistics of Yp points.

The alternatives are segregation (where Xp points cluster away from Yp points i.e. cluster around the centers of the subintervals) and association (where Xp points cluster around Yp points). The test is based on the (asymptotic) binomial distribution of the domination number of PE-PCD for uniform 1D data in the subintervals based on Yp points.

The function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is Pr(Domination Number=2)), and method and name of the data set used.

Under the null hypothesis of uniformity of Xp points in the interval based on Yp points, probability of success (i.e. Pr(Domination Number=2)) equals to its expected value under the uniform distribution) and alternative could be two-sided, or left-sided (i.e. data is accumulated around the Yp points, or association) or right-sided (i.e. data is accumulated around the centers of the subintervals, or segregation).

PE proximity region is constructed with the expansion parameter $r \geq 1$ and centrality parameter c which yields M-vertex regions. More precisely $M = a + c(b - a)$ for the centrality parameter c and for a given c in $(0, 1)$, the expansion parameter r is taken to be $1/\max(c, 1 - c)$ which yields non-degenerate asymptotic distribution of the domination number.

The test statistic is based on the binomial distribution, when domination number is scaled to have value 0 and 1 in the one interval case (i.e., Domination Number - 1 for the one interval case). That is, the test statistic is based on the domination number for Xp points inside the interval based on Yp points for the PE-PCD . For this approach to work, Xp must be large for each subinterval, but 5 or more per subinterval seems to work fine in practice. Probability of success is chosen in the following way for various parameter choices.

asy.bin is a logical argument for the use of aymptotic probability of success for the binomial distribution, default is asy.bin=FALSE. It is an option only when Yp is not provided. When Yp is provided or when Yp is not provided but asy.bin=T, aymptotic probability of success for the binomial distribution is used. When Yp is not provided and asy.bin=F, the finite sample aymptotic probability of success for the binomial distribution is used with number of trials equals to expected number of Xp points per subinterval.

Usage

```
TSDomPEBin1D(
  Xp,
  Yp = NULL,
  int,
  c = 0.5,
  asy.bin = FALSE,
  end.int.cor = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

Xp	A set of 1D points which constitute the vertices of the PE-PCD.
Yp	A set of 1D points which constitute the end points of the subintervals, default is NULL. When Yp=NULL, the support interval (a, b) is partitioned as Yp=(b-a)*(0:nint)/nint where nint=round(sqrt(nx),0) and nx is the number of Xp points.
int	Support interval (a, b) with $a < b$. Uniformness of Xp points in this interval is tested.
c	A positive real number which serves as the centrality parameter in PE proximity region; must be in $(0, 1)$ (default c=.5).
asy.bin	A logical argument for the use of aymptotic probability of success for the binomial distribution, default asy.bin=FALSE. It is an option only when Yp is not provided. When Yp is provided or when Yp is not provided but asy.bin=T, aymptotic probability of success for the binomial distribution is used. When Yp is not provided and asy.bin=F, the finite sample aymptotic probability of success for the binomial distribution is used with number of trials equals to expected number of Xp points per subinterval.
end.int.cor	A logical argument for end interval correction, default is FALSE, recommended when both Xp and Yp have the same interval support.
alternative	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater".

conf.level	Level of the confidence interval, default is 0.95, for the probability of success (i.e. Pr(domination number=2) for PE-PCD whose vertices are the 1D data set Xp.

Value

A list with the elements

statistic	Test statistic
p.value	The p-value for the hypothesis test for the corresponding alternative
conf.int	Confidence interval for Pr(Domination Number=2) at the given level conf.level and depends on the type of alternative.
estimate	A vector with two entries: first is is the estimate of the parameter, i.e., Pr(Domination Number=2) and second is the domination number
null.value	Hypothesized value for the parameter, i.e., the null value for Pr(Domination Number=2)
alternative	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater"
method	Description of the hypothesis test
data.name	Name of the data set

References

There are no references for Rd macro \insertAllCites on this help page.

See Also

TSDomPEBin and PEdom1D

Examples

```
a<-0; b<-10; int<-c(a,b)
c<-.4

r<-1/max(c,1-c)

#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-runif(nx,a,b)
Yp<-runif(ny,a,b)
PEdom1D(Xp,Yp,r,c)

plotIntervals(Xp,Yp,xlab="",ylab="")
plotPEregsMI(Xp,Yp,r,c)

TSDomPEBin1D(Xp,Yp,int,c,alt="t")
TSDomPEBin1D(Xp,int=int,c=c,alt="t")
```

```
TSDomPEBin1D(Xp,Yp,int,c,alt="l")
TSDomPEBin1D(Xp,Yp,int,c,alt="g")
TSDomPEBin1D(Xp,Yp,int,c,end=TRUE)
TSDomPEBin1D(Xp,Yp,int,c=.25)
```

`TSDomPENor` *A test of segregation/association based on domination number of Proportional Edge Proximity Catch Digraph (PE-PCD) for 2D data - Normal Approximation*

Description

An object of class "htest" (i.e., hypothesis test) function which performs a hypothesis test of complete spatial randomness (CSR) or uniformity of `Xp` points in the convex hull of `Yp` points against the alternatives of segregation (where `Xp` points cluster away from `Yp` points i.e. cluster around the centers of the Delaunay triangles) and association (where `Xp` points cluster around `Yp` points) based on the normal approximation to the binomial distribution of the domination number of PE-PCD for uniform 2D data in the convex hull of `Yp` points

The function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is Pr(Domination Number=3)), and method and name of the data set used.

Under the null hypothesis of uniformity of `Xp` points in the convex hull of `Yp` points, probability of success (i.e. Pr(Domination Number=3)) equals to its expected value under the uniform distribution) and `alternative` could be two-sided, or left-sided (i.e. data is accumulated around the `Yp` points, or association) or right-sided (i.e. data is accumulated around the centers of the triangles, or segregation).

PE proximity region is constructed with the expansion parameter $r \geq 1$ and M-vertex regions where M is a center that yields non-degenerate asymptotic distribution of the domination number.

The test statistic is based on the normal approximation to the binomial distribution, when domination number is scaled to have value 0 and 1 in the one triangle case (i.e., Domination Number - 1 for the one triangle case). That is, the test statistic is based on the domination number for `Xp` points inside convex hull of `Yp` points for the PE-PCD and default convex hull correction, `ch.cor`, is `FALSE` where `M` is the center that yields nondegenerate asymptotic distribution for the domination number.

For this approximation to work, `Yp` must be at least 10 (actually about 15 or more Delaunay triangles) and `Xp` must be at least 10 times more than `Yp` points.

See also (Ceyhan (2011)).

Usage

```
TSDomPENor(
  Xp,
  Yp,
  r,
```

```
  ch.cor = F,
  nt = NULL,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

Xp	A set of 2D points which constitute the vertices of the PE-PCD.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.
r	A positive real number which serves as the expansion parameter in PE proximity region; must be in $(1, 1.5]$.
ch.cor	A logical argument for convex hull correction, default ch.cor=FALSE, recommended when both Xp and Yp have the same rectangular support.
nt	Number of Delaunay triangles based on Yp points, default is NULL.
alternative	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater".
conf.level	Level of the confidence interval, default is 0.95, for the domination number of PE-PCD whose vertices are the 2D data set Xp.

Value

A list with the elements

statistic	Test statistic
p.value	The p-value for the hypothesis test for the corresponding alternative
conf.int	Confidence interval for the domination number at the given level conf.level and depends on the type of alternative.
estimate	A vector with two entries: first is the domination number, and second is the estimate of the parameter, i.e., Pr(Domination Number=3)
null.value	Hypothesized value for the parameter, i.e., the null value for expected domination number
alternative	Type of the alternative hypothesis in the test, one of "two.sided", "less", "greater"
method	Description of the hypothesis test
data.name	Name of the data set

References

Ceyhan E (2011). "Spatial Clustering Tests Based on Domination Number of a New Random Digraph Family." *Communications in Statistics - Theory and Methods*, **40(8)**, 1363-1395.

See Also

TSDomPEBin

Examples

```
nx<-20; ny<-4  #try also nx<-1000; ny<-10
r<-1.5  #try also r<-2 or r<-1.25

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

oldpar <- par(mfrow = c(1,2))
plotDeltri(Xp,Yp,xlab="",ylab="")
par(oldpar)

PEdomMTnd(Xp,Yp,r)

TSDomPENor(Xp,Yp,r)
TSDomPENor(Xp,Yp,r)

TSDomPENor(Xp,Yp,1.25,ch=TRUE)

#or try
ndt<-NumDelTri(Yp)
TSDomPENor(Xp,Yp,r,nt=ndt)
#values might differ due to the random of choice of the three centers M1,M2,M3
#for the non-degenerate asymptotic distribution of the domination number

TSDomPENor(Xp,Yp,r)
TSDomPENor(Xp,Yp[1:3,],r)

TSDomPENor(Xp,rbind(Yp,Yp),r)

dat.fr<-data.frame(a=Xp)
TSDomPENor(dat.fr,Yp,r)

dat.fr<-data.frame(a=Yp)
TSDomPENor(Xp,dat.fr,r)
```

`XinCHY` *Points from one class inside the convex hull of the points from the other class*

Description

Given two 2D data sets, `Xp` and `Yp`, it returns the `Xp` points inside the convex hull of `Yp` points.

See (Okabe et al. (2000); Ceyhan (2010); Sinclair (2016)) for more on Delaunay triangulation and the corresponding algorithm.

Usage

```
XinCHY(Xp, Yp)
```

Arguments

Xp	A set of 2D points which constitute the data set.
Yp	A set of 2D points which constitute the vertices of the Delaunay triangles.

Value

Xp points inside the convex hull of Yp points

References

Ceyhan E (2010). "Extension of One-Dimensional Proximity Regions to Higher Dimensions." *Computational Geometry: Theory and Applications*, **43(9)**, 721-748.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York.

Sinclair D (2016). "S-hull: a fast radial sweep-hull routine for Delaunay triangulation." 1604.01428.

See Also

plotDeltri

Examples

```
#nx is number of X points (target) and ny is number of Y points (nontarget)
nx<-20; ny<-4;  #try also nx<-40; ny<-10 or nx<-1000; ny<-10;

set.seed(1)
Xp<-cbind(runif(nx,0,1),runif(nx,0,1))
Yp<-cbind(runif(ny,0,1),runif(ny,0,1))

DT<-interp::tri.mesh(Yp[,1],Yp[,2],duplicate="remove")

Xlim<-range(Xp[,1],Yp[,1])
Ylim<-range(Xp[,2],Yp[,2])
xd<-Xlim[2]-Xlim[1]
yd<-Ylim[2]-Ylim[1]

Xch<-XinCHY(Xp,Yp)

plot(Xp,main=" ", xlab=" ", ylab=" ",
xlim=Xlim+xd*c(-.05,.05),ylim=Ylim+yd*c(-.05,.05),pch=".",cex=3)
interp::convex.hull(DT,plot.it = TRUE, add = TRUE)  # or try polygon(Yp[ch$i,])
points(Xch,pch=4,col="red")

XinCHY(Xp,Yp)
XinCHY(Xp,Yp[1:3,])

XinCHY(Xp,rbind(Yp,Yp))

dat.fr<-data.frame(a=Xp)
```

```
XinCHY(dat.fr,Yp)

dat.fr<-data.frame(a=Yp)
XinCHY(Xp,dat.fr)
```

Index

∗Topic **datasets**
swamptrees, 535
.onAttach, 8
.onLoad, 8

angle.str2end, 9, *11*
angle3pnts, *10*, 11
ArcsASMT, 12, *16*, *27*, *40*
ArcsAStri, *14*, 14, *29*, *43*
ArcsCS1D, 17, *20–22*, *24*, *32*, *34*, *38*
ArcsCSend1D, *18*, 19, *22*, *24*, *32*, *34*, *38*
ArcsCSMI, *17*, *18*, 21, *36*
ArcsCSmid1D, *18*, *20*, *22*, 23, *24*, *32*, *34*, *38*
ArcsCSMT, *14*, *16*, 25, *29*, *40*
ArcsCStri, *14*, *16*, *27*, 28, *43*
ArcsPE1D, *18*, *20*, *24*, 31, *34–36*, *38*
ArcsPEend1D, *18*, *20*, *24*, *32*, 33, *36*, *38*
ArcsPEMI, *22*, *31*, *32*, 35
ArcsPEmid1D, *18*, *20*, *32*, *34*, *36*, 37
ArcsPEMT, *14*, *16*, *27*, 39, *43*
ArcsPEtri, *14*, *16*, *29*, *40*, 41
area.polygon, 44
as.bastri, 45
ASarcdens.tri, 46, *48*, *88*, *341*
asyvarCS1D, *130*
asyvarCS1D *(*funsMuVarCS1D*)*, 124
asyvarCS2D, *131*
asyvarCS2D *(*funsMuVarCS2D*)*, 125
asyvarCSend1D, *133*
asyvarCSend1D *(*funsMuVarCSend1D*)*, 127
asyvarPE1D, *124*
asyvarPE1D *(*funsMuVarPE1D*)*, 129
asyvarPE2D, *126*
asyvarPE2D *(*funsMuVarPE2D*)*, 131
asyvarPEend1D, *128*
asyvarPEend1D *(*funsMuVarPEend1D*)*, 132

bary2cart *(*funsCartBary*)*, 109

cart2bary *(*funsCartBary*)*, 109
cent.nondeg, 48
centersMc, 50, *51*
centMc, *50*, 51
circ.cent.bastri, 52, *56*
circ.cent.tetra, 54
circ.cent.tri, *53*, *54*, 55
cl2CC.TbVR, 57, *60*, *80*
cl2CC.VR, *58*, 59, *80*
cl2edgesTe, 62, *66*, *68*, *70*, *75*, *98*, *525*
cl2eTbVRcent, *58*, *60*, *63*, 64, *68*, *70*, *75*
cl2eVRCC, 67, *75*
cl2eVRcent, *58*, *60*, *63*, *66*, *68*, 69, *72*, *73*, *75*
cl2eVRcent.alt, 72
cl2eVRCM, *58*, *60*, *63*, *66*, *68*, *70*, 74
cl2fVRth, 76
cl2Mc.int, 79
cp2e.bastri, 81, *83*, *85*
cp2e.tri, *82*, 83, *85*
cp2edges.nd, *82*, *83*, 84
CSarcdens.tri, *48*, 87, *341*

dim, *89*
dimension, 89, *281*
Dist, 90, *95*, *96*
dist, *90*
dist.pt2set, 91, *95*, *96*
dom.exact, 92, *94*, *218*, *221*, *223*, *272*, *345*, *348*
dom.greedy, *93*, 93, *272*, *345*, *348*
dp2l, *91*, 94, *96*, *273*
dp2pl, *91*, *95*, 96
draw.arc, *9*

fr2eTeER, *58*, *60*, *63*, *78*, 97, *100*, *103*, *284*, *286*
fr2vTbVRCC, *98*, 99, *103*, *284*, *286*
fr2vVRCC, *78*, *98*, *100*, 102, *284*, *286*
funsAB2CMTe, 104
funsAB2MTe, 106
funsCartBary, 109

funsCSEdgeRegs, 111
funsCSGamTe, 114
funsCSt1EdgeRegs, 118
funsIndDelTri, 121
funsMuVarCS1D, 123
funsMuVarCS2D, 125
funsMuVarCSend1D, 127
funsMuVarPE1D, 129
funsMuVarPE2D, 131
funsMuVarPEend1D, 132
funsPG2PE1D, 134
funsRankOrderTe, 138
funsTbMid2CC, 140
fvar1 *(*funsMuVarPE1D*)*, 129
fvar2 *(*funsMuVarPE1D*)*, 129

Gam1ASbastri, 143, *148*, *161*
Gam1AStri, *145*, 146, *161*, *170*
Gam1CS.Te.onesixth, 149
Gam1CS1D, 152
Gam1CSTe, *115*, *150*, 153, *157*
Gam1CSTet1, *150*, *155*, 156
Gam1PE1D, *152*, 158
Gam1PEbastri, *145*, 160, *164*, *167*, *170*
Gam1PEstdTetra, 163, *167*
Gam1PEtetra, *164*, 165
Gam1PEtri, *159*, *164*, *167*, 168
Gam2ASbastri, 171, *175*, *179*
Gam2AStri, *173*, 174, *179*, *187*
Gam2CS.Te.onesixth, 176
Gam2CSTe, *177*
Gam2CSTe *(*funsCSGamTe*)*, 114
Gam2PEbastri, 178, *181*, *184*, *187*
Gam2PEstdTetra, 180, *184*
Gam2PEtetra, *115*, *181*, 183, *187*
Gam2PEtri, *115*, *179*, *181*, *184*, 185
Gam3CSTe *(*funsCSGamTe*)*, 114
Gam3PEstdTetra, 188, *192*
Gam3PEtetra, *189*, 190
Gam4CSTe *(*funsCSGamTe*)*, 114
Gam5CSTe *(*funsCSGamTe*)*, 114
Gam6CSTe *(*funsCSGamTe*)*, 114

in.circle, 193
in.tetrahedron, *193*, 194
in.triangle, *193*, *195*, 196, *278*
IncMatASMT, 197, *200*, *203*, *210*
IncMatAStri, *198*, 199, *207*, *215*
IncMatCS1D, 200, *201*, *208*
IncMatCSMT, *198*, 202, *205*, *207*, *210*
IncMatCSTe, *203*, 204, *212*
IncMatCStri, *200*, *203*, *205*, 206, *215*
IncMatPE1D, 207, *213*
IncMatPEMT, *198*, *201*, *203*, *208*, 209, *212*, *213*, *215*
IncMatPETe, *205*, *210*, 211
IncMatPEtetra, 213
IncMatPEtri, *200*, *201*, *207*, *208*, *210*, *212*, *213*, 214
ind.Del.tri *(*funsIndDelTri*)*, 121
ind.int.set, 216
IndASdomUBtri, 217, *221*, *223*
IndCS.Te.onesixth, 219
IndCSdomUBTe, *218*, 220, *223*
IndCSdomUBtri, *93*, *94*, *218*, *221*, 222
IndCSTe, *219*, 223, *228*, *230*, *243*, *245*, *249*, *259*
IndCSTe.domset, 225, *247*, *261*
IndCSTeRAB, *119*
IndCSTeRAB *(*funsCSEdgeRegs*)*, 111
IndCSTeRABt1, *112*
IndCSTeRABt1 *(*funsCSt1EdgeRegs*)*, 118
IndCSTeRAC, *119*
IndCSTeRAC *(*funsCSEdgeRegs*)*, 111
IndCSTeRACt1, *112*
IndCSTeRACt1 *(*funsCSt1EdgeRegs*)*, 118
IndCSTeRBC, *119*
IndCSTeRBC *(*funsCSEdgeRegs*)*, 111
IndCSTeRBCt1, *112*
IndCSTeRBCt1 *(*funsCSt1EdgeRegs*)*, 118
IndCSTeSet, 227, *249*, *263*
IndCSTet1, 229
indices.Del.tri *(*funsIndDelTri*)*, 121
IndNASbastri, 230, *233*, *293*
IndNAStri, *231*, 232, *237*, *243*, *245*, *267*, *296*
IndNAStri.domset, 234, *247*, *269*
IndNAStriSet, *235*, 236, *237*, *249*, *271*
IndNCSend1D, 238, *240*, *242*, *253*, *256*
IndNCSint, 239, *255*
IndNCSmid1D, *239*, *240*, 241, *253*, *256*
IndNCStri, *225*, *228*, *233*, 242, *243*–*245*, *249*, *267*, *300*
IndNCStri.alt, 244
IndNCStri.domset, *226*, *235*, 246, *269*
IndNCStriSet, *228*, *237*, 248, *271*
IndNPEbastri, 250, *259*, *267*, *302*
IndNPEend1D, *239*, *242*, 252, *255*, *256*

IndNPEint, *240*, 254, *258*, *265*
IndNPEmid1D, *239*, *242*, *253*, *255*, 255
IndNPEstdtetra, 257, *265*
IndNPETe, *225*, *251*, 258, *263*, *267*, *271*
IndNPETe.domset, *226*, 260, *269*
IndNPETeSet, *228*, 262, *271*
IndNPEtetra, *258*, 264
IndNPEtri, *233*, *243*, *245*, *251*, *258*, *259*, *263*, *265*, 266, *271*, *308*
IndNPEtri.domset, *235*, *247*, *261*, 268
IndNPEtriSet, *249*, *263*, *269*, 270
IndUBdom, *218*, *221*, *223*, 272
int.2lines, 273, *275*, *276*
int.circ.line, *273*, 274, *276*
int.line.plane, 276
inTriAll, *196*, 277
is.in.data, 279
is.point, *89*, 280
isStdEqTri, 281

Kfr2vTbVRCC, *78*, *98*, *100*, *103*, 282, *286*
Kfr2vVRCC, *78*, *98*, *100*, *103*, *284*, 285

lA_CM.Te, *107*, *142*
lA_CM.Te (funsAB2CMTe), 104
lA_M.Te, *105*, *142*
lA_M.Te (funsAB2MTe), 106
lB_CM.Te, *107*, *142*
lB_CM.Te (funsAB2CMTe), 104
lB_M.Te, *105*, *142*
lB_M.Te (funsAB2MTe), 106
lC_M.Te, *105*, *142*
lC_M.Te (funsAB2MTe), 106
lD1CCinTb (funsTbMid2CC), 141
lD2CCinTb (funsTbMid2CC), 141
Line, 287, *333*, *353*, *526*
line, *287*, *288*, *290*, *333*
Line3D, *288*, 289, *335*, *351*

mu1PE1D (funsMuVarPE1D), 129
muCS1D, *130*
muCS1D (funsMuVarCS1D), 124
muCS2D, *131*
muCS2D (funsMuVarCS2D), 125
muCSend1D, *133*
muCSend1D (funsMuVarCSend1D), 127
muPE1D, *124*
muPE1D (funsMuVarPE1D), 129
muPE2D, *126*
muPE2D (funsMuVarPE2D), 131
muPEend1D, *128*
muPEend1D (funsMuVarPEend1D), 132

NASbastri, 291, *296*
NAStri, *231*, *293*, 294, *300*, *302*, *308*
NCSint, 297, *303*
NCStri, *296*, *298*, 299, *302*, *308*
NPEbastri, 300, *308*
NPEint, *298*, 302, *304*, *306*
NPEstdtetra, 304, *306*
NPEtetra, *303*, *304*, 305
NPEtri, *296*, *300*, *302*–*304*, *306*, 307
NumArcsASMT, 309, *312*, *318*, *327*
NumArcsAStri, *48*, *310*, 311, *321*, *331*
NumArcsCSend1D, 312, *314*, *316*, *322*, *325*
NumArcsCSint, 313, *324*
NumArcsCSmid1D, *313*, *314*, 315, *322*, *325*
NumArcsCSMT, *310*, 316, *319*, *321*, *327*
NumArcsCSTe, *318*, 318, *321*, *329*
NumArcsCStri, *88*, *312*, *318*, *319*, 320, *331*
NumArcsPEend1D, *313*, *316*, 322, *324*, *325*
NumArcsPEint, *314*, 323
NumArcsPEmid1D, *313*, *316*, *322*, *324*, 324
NumArcsPEMT, *310*, *318*, 326, *329*, *331*
NumArcsPETe, *319*, *327*, 328, *331*
NumArcsPEtri, *312*, *321*, *327*, *329*, 330, *341*
NumDelTri, 331

on.convex.hull, *193*, *196*, *278*
order.d2e.Te (funsRankOrderTe), 138

paraline, *288*, 332, *335*, *353*, *526*
paraline3D, *290*, *333*, 334, *351*
paraplane, 336, *357*
pcds, 338
PEarcdens.tri, *88*, 339
PEdom.tetra, 341, *345*, *348*
PEdom1D, *93*, *94*, 343, *349*, *550*
PEdomMT, 344, *349*
PEdomMTnd, *93*, *94*, *343*, 346, *349*
PEdomtri, *93*, *94*, *342*, *345*, *348*, 348
perp.ln2pl, *335*, 350
perpline, *288*, *333*, *351*, 352, *526*
PG2A (funsPG2PE1D), 134
PG2AI (funsPG2PE1D), 134
PG2AII (funsPG2PE1D), 134
PG2AIII (funsPG2PE1D), 134
PG2AIV (funsPG2PE1D), 134

PG2Asym *(*funsPG2PE1D*)*, 134
PG2B *(*funsPG2PE1D*)*, 134
PG2BIII *(*funsPG2PE1D*)*, 134
PG2Bsym *(*funsPG2PE1D*)*, 134
PG2C *(*funsPG2PE1D*)*, 134
PG2CIV *(*funsPG2PE1D*)*, 134
PG2Csym *(*funsPG2PE1D*)*, 134
PG2PE1D, *355, 356*
PG2PE1D *(*funsPG2PE1D*)*, 134
PG2PE1D.asy, *136*, 354
PG2PEtri, *136*, *355*, 355
Plane, *290*, *337*, 356
plot.Extrema, 358, *411*, *416*, *527*
plot.Lines, 359, *412*, *417*, *528*
plot.Lines3D, 360, *413*, *418*, *529*
plot.Patterns, 361, *414*, *418*, *530*
plot.PCDs, 362, *415*, *419*, *531*
plot.Planes, 363, *416*, *419*, *532*
plot.TriLines, 364, *420, 421*, *533*
plot.triSht, *390*
plot.Uniform, 365, *420*, *422*, *534*
plotASarcsMT, 366, *369*, *378*, *396*, *398*
plotASarcsTri, *367*, 368, *380*, *398*
plotASregsMT, 370, *373*, *386*, *404*, *406*
plotASregsTri, *371*, 372, *388*, *406*, *410*
plotCSarcs1D, 374, *393*
plotCSarcsMT, *367*, *369*, 377, *380*, *396*
plotCSarcsTri, *367*, *369*, *378*, 379
plotCSregsInt, 381, *384*, *400*, *402*
plotCSregsMI, *382*, 383, *400*, *402*
plotCSregsMT, *371*, *373*, 385, *388*, *404*, *406*
plotCSregsTri, *371*, *373*, *386*, 387, *406*, *410*
plotDeltri, *332*, 389, *392*, *554*
plotIntervals, 391
plotPEarcs1D, *376*, 392
plotPEarcsMT, *367*, *369*, *378*, 394, *398*
plotPEarcsTri, *367*, *369*, *380*, *396*, 396
plotPEregsInt, *382*, 399, *408*
plotPEregsMI, *382*, *384*, *392*, *400*, 401, *402*
plotPEregsMT, *371*, *373*, *386*, 403, *406*, *410*
plotPEregsStdTH, 405, *408*
plotPEregsTH, 407
plotPEregsTri, *371*, *373*, *388*, *404*, *408*, 409
print.Extrema, *359*, 411, *416*, *527*
print.Lines, *359*, 412, *417*, *528*
print.Lines3D, *360*, 413, *418*, *529*
print.Patterns, *361*, 414, *418*, *530*
print.PCDs, *362*, 414, *419*, *531*
print.Planes, *363*, 415, *419*, *532*
print.summary.Extrema, *359*, *411*, 416, *527*
print.summary.Lines, *359*, *412*, 417, *528*
print.summary.Lines3D, *360*, *413*, 417, *529*
print.summary.Patterns, *361*, *414*, 418, *530*
print.summary.PCDs, *362*, *415*, 418, *531*
print.summary.Planes, *363*, *416*, 419, *532*
print.summary.TriLines, *364*, 419, *421*, *533*
print.summary.Uniform, *365*, 420, *422*, *534*
print.TriLines, *364*, *420*, 421, *533*
print.Uniform, *365*, *420*, 422, *534*

radii, 423, *425*
radius, *423*, 425
rank.d2e.Te *(*funsRankOrderTe*)*, 138
rasc.disc, 426, *429, 430*, *434*, *436*, *438*, *462*, *466*, *470*
rasc.matern, *426, 427*, 428
rasc.tri, 431, *435*, *464*
rascIITe, *427*, *430*, *432*, 433, *436*
rascMT, *427*, *430*, *432*, 434, *468*
rascTe, *427*, *430*, *432*, *436*, 437
re.bastri.cent, 439, *441*, *443*, *445*, *448*, *450*, *459*
re.bastriCM, 442, *445*, *448*, *450*, *459*
re.tri.cent, *441*, *443*, 444, *448*, *450*, *459*
re.triCM, *441*, *443*, *445*, 447, *449, 450*, *459*
redge.triCM, *441*, *443*, *445*, *448*, 449, *450*, *459*
redges.tri.cent, 451, *455*
redges.triCM, *452*, 453
rel.six.Te, 456, *525*
reTeCM, *441*, *443*, *445*, *448*, *450*, 458
rMatClust, *429, 430*
rseg.disc, *427*, *430*, *434*, *438*, 460, *466*, *468*, *470*
rseg.tri, *432*, 463, *467*
rsegIITe, *434*, *438*, *462*, *464*, 465, *468*, *470*
rsegMT, *434*, *436*, *438*, *462*, *464*, *466*, 466, *470*, *522*
rsegTe, *462*, *464*, *466*, *468*, 469, *522*
runif.bastri, 471, *477*, *479*, *481, 482*
runif.stdtetra, 473, *476*
runif.tetra, *474*, 475
runif.tri, *472*, *474*, *476*, 477, *478, 479*, *481, 482*
runifMT, *472*, *474*, *477*, 478, *481, 482*

runifTe, *472*, *477*, *479*, 480, *482*
runifTe.onesixth, *456*, 481
rv.bastri.cent, 483, *486*, *489*, *500*, *502*, *505*, *519*, *521*
rv.bastriCC, *484*, 485, *489*, *500*, *502*, *505*, *519*, *521*
rv.bastriCM, *484*, *486*, 488, *500*, *502*, *505*, *521*
rv.end.int, 491, *493*
rv.mid.int, *491*, 492
rv.tetraCC, 494, *497*
rv.tetraCM, *495*, 497
rv.tri.cent, *484*, *486*, *489*, 499, *502*, *505*, *519*, *521*
rv.triCC, *484*, *486*, *489*, *495*, *500*, 501, *505*, *519*, *521*
rv.triCM, *484*, *486*, *489*, *497*, *500*, *502*, 504, *519*, *521*
rverts.tri.cent, *452*, *455*, 506, *509*, *512*, *514*, *517*
rverts.tri.cent.alt, 509
rverts.tri.nd, *452*, *455*, *507*, 511, *514*, *517*
rverts.triCC, *507*, *512*, 513, *517*
rverts.triCM, *507*, *512*, *514*, 516
rvTe.cent, 518
rvTeCM, *484*, *486*, *489*, *500*, *502*, *505*, *519*, 520

seg.tri.supp, 522
six.ext, 524
slope, *288*, *333*, *353*, 526
summary.Extrema, *359*, *411*, *416*, 527
summary.Lines, *359*, *412*, *417*, 528
summary.Lines3D, *360*, *413*, *418*, 529
summary.Patterns, *361*, *414*, *418*, 530
summary.PCDs, *362*, *415*, *419*, 531
summary.Planes, *363*, *416*, *419*, 532
summary.TriLines, *364*, *420, 421*, 533
summary.Uniform, *365*, *420*, *422*, 534
swamptrees, 535

tri.mesh, *121*, *435*, *467*, *478*
triangles, *121*, *435*, *467*
TSArcDensCS1D, 536, *540*, *542*
TSArcDensCSMT, 538, *544*
TSArcDensPE1D, *537*, 541, *544*
TSArcDensPEMT, *540*, 543
TSDomPEBin, 545, *550*, *552*
TSDomPEBin1D, 548
TSDomPENor, *547*, 551

XinCHY, 553