Package 'GriegSmith'

February 19, 2015

Type Package

Title Uses Grieg-Smith method on 2 dimentional spatial data

Version 1.0

Date 2011-03-18

Author Brian McGuire

Maintainer Brian McGuire <mcguirbc@gmail.com>

Description The function GriegSmith accepts either quadrat count data, a point process object(ppp) or a matrix of x and y coordinates. The function calculates a nested analysis of variance and simulation envelopes.

Depends spatstat

License GPL-2

LazyLoad yes

Repository CRAN

Date/Publication 2013-03-14 08:23:34

NeedsCompilation no

R topics documented:

addem *addem*

Description

Used by GriegSmith function

Usage

```
addem(startingvals, xmatlen, ymatlen, data)
```

Arguments

startingvals

xmatlen

ymatlen

data

Examples

```
## The function is currently defined as
function(startingvals,xmatlen,ymatlen,data){

x<-startingvals[1];
y<-startingvals[2];


#print("here");
#print(data[x:(x+xmatlen-1),y:(y+ymatlen-1)]);

sum(data[x:(x+xmatlen-1),y:(y+ymatlen-1)])^2;




  }
```

belongtoint *belongtoint*

Description

Used by GriegSmith function

Usage

```
belongtoint(bin.vals.vect, int.x, int.y, vect)
```

Arguments

`bin.vals.vect`

`int.x`

`int.y`

`vect`

Examples

```

## The function is currently defined as
function(bin.vals.vect,int.x,int.y,vect){

xval<-bin.vals.vect[1];
yval<-bin.vals.vect[2];



sum(vect[,1] >= xval & vect[,1] < (xval+int.x) & vect[,2] >= yval & vect[,2] < (yval+int.y));

  }
```

envelopes *envelopes*

Description

Used by GriegSmith function

Usage

```
envelopes(iterations = 100, countdata, dimention)
```

Arguments

`iterations`

`countdata`

`dimention`

Examples

```
## The function is currently defined as
function(iterations=100,countdata,dimention){



## Randobly arrange the counts;



GSpermprev<-array();

for (i in 1:iterations){

perm<-matrix(sample(as.vector(countdata),size=2^(dimention*2),replace=FALSE),nrow=2^dimention,byrow=TRUE);
GSperm<-iterate(perm,dimention);


GSpermprev<-cbind(GSpermprev,GSperm[,3])



}


ret.val<-cbind(apply(GSpermprev[,-1],1,quantile,probs=c(.05)),apply(GSpermprev[,-1],1,quantile,probs=c(.95)))

  }
```

`GriegSmith` *Grieg-Smith Calculation*

Description

This function accepts a point process object, a two column matrix of x-y coordinate pairs or a three column matrix containing x-y coordinates and quadrat counts in the third column. If the data contains quadrat counts, then the counts=TRUE option must be selected. The function returns a GriegSmith object which is a matrix with block sizes, sum of squares for each block size as well as mean sums of squares. Simulation envelopes are produced as well through randombly permuting the quadrat counts. The 5th and 95th percentiles of the permutations create the simulation envelope.

Ploting the GriegSmith object produces a plot of the MSr as well as the simulation envelopes.

Usage

```
GriegSmith(datapoints, counts = FALSE, env = 100)
## S3 method for class 'GriegSmith'
plot(x,main, ...)
```

Arguments

`datapoints` datapoints is either a point process object (ppp), a two column matrix of x-y coordinates, or a three column matrix of x-y quadrat coords with a third column of quadrat counts.

`counts` If datapoints is a three column matrix with quadrat counts, then set counts=T

`env` How many permuatations should be used to create the simulation envelopes default=100.

`x` A GriegSmith object created with GriegSmith()

`main` the graph title

`...` other parameters passed to the plot function

Author(s)

Brian McGuire

References

Statistical Methods for Spatial Data Analysis. Oliver Schabenberger and Carol A. Gotway . Boca Raton, FL: Chapman & Hall/CRC, 2005.

Greig-Smith, P. 1952. The use of random and contiguous quadrats in the study of structure in plant communities. Annals of Botany 16:293-316.

Examples

```
data(amacrine,package="spatstat")
GS_ama<-GriegSmith(amacrine);
plot(GS_ama)




## The function is currently defined as
function(datapoints,counts=FALSE,env=100){


if(counts==FALSE){

if(is.ppp(datapoints)){

xmin<-datapoints$window$xrange[1]
xmax<-datapoints$window$xrange[2]
ymin<-datapoints$window$yrange[1]
ymax<-datapoints$window$yrange[2]


datapoints<-cbind(datapoints$x,datapoints$y);



}
else{

xmax<-max(datapoints[,1]);
xmin<-min(datapoints[,1]);

ymax<-max(datapoints[,2]);
ymin<-min(datapoints[,2]);





}





numpts<-length(datapoints[,1]);
startingdim<-ceiling(log(numpts)/(2*log(2)));
counts<-sums(datapoints,2^startingdim,xmin,xmax,ymin,ymax)


}
else {
if (max(datapoints[,1]) != max(datapoints[,2])) stop("Your count data must have equal dimensions")


datapoints<-datapoints[order(datapoints[,2],datapoints[,1]),]
numpts<-sum(datapoints[,3])
startingdim<-ceiling(log(max(datapoints[,1]))/log(2))
counts<-matrix(datapoints[,3],nrow=2^startingdim,byrow=TRUE);


}


actual<-iterate(counts,startingdim);
sims<-envelopes(env,counts,startingdim);
final<-cbind(actual,sims);


colnames(final)<-c("blocksize","SSr","MSr","MSr.05","MSr.95");
```

```
class(final) <- "GriegSmith"


final;




  }
```

`iterate` *iterate*

Description

Used by GriegSmith function

Usage

```
iterate(counts, startingdim)
```

Arguments

`counts`

`startingdim`

Examples

```


## The function is currently defined as
function(counts,startingdim){


powers<-c(0:startingdim);
square<-2^powers;


x_rects<-sort(c(square,square));
x_rects<-x_rects[c(-1,-length(x_rects))];

y_rects0<-2^(1:startingdim);
y_rects1<-2^(0:(startingdim-1));
y_rects<-c(rbind(y_rects0,y_rects1));

rects<-rbind(cbind(square,square),cbind(x_rects,y_rects));


## rects is a 2 column matrix, the first column is the x length;
## for each iteration of the G-S method, the second column is the y;
## width for each iteration. We have both vertically and horizontally;
## oriented blocks, so we will need to average them.

 rects<-rects[order(rowSums(rects)),]



checkhere<-apply(rects,1,sumofsquares,singlecounts=counts);
mid<-cbind(rects[,1]*rects[,2],rects,checkhere);



ss<-as.matrix(tapply(mid[,4],mid[,1],mean));



ss2<-cbind(ss[-1,1],2*ss[-length(ss),1]);
blocksize<-as.numeric(rownames(ss2))/2
rownames(ss2)<-blocksize;





ssrfinal<-cbind(blocksize,ss2[,2]-ss2[,1],(ss2[,2]-ss2[,1])/(2^(2*startingdim)));


ssrfinal;

  }
```

`sumofsquares` *sumofsquares*

Description

Used by GriegSmith function

Usage

```
sumofsquares(sizematrix, singlecounts)
```

Arguments

`sizematrix`

`singlecounts`

Examples

```
## The function is currently defined as
function(sizematrix,singlecounts){

# print(sizematrix);

xsublength<-sizematrix[1];
ysublength<-sizematrix[2];


xsize<-length(singlecounts[1,]);
ysize<-length(singlecounts[,1]);



xmin<-rep(seq(from=1,to=xsize,by=xsublength),ysize/ysublength);
ymin<-sort(rep(seq(from=1,to=ysize,by=ysublength),xsize/xsublength));


submatricies<-cbind(xmin,ymin);

squaredsums<-sum(apply(submatricies,1,addem,data=singlecounts,xmatlen=xsublength,ymatlen=ysublength));


# print(squaredsums);

## sum up all the numbers in each matrix, square those numbers and add them;



  }
```

sums *sums*

Description

Used by GriegSmith function

Usage

```
sums(coords, dim, xmin = min(coords[, 1]), xmax = max(coords[, 1]), ymin = min(coords[, 2]), ymax = ma
```

Arguments

coords

dim

xmin

xmax

ymin

ymax

Examples

```
## The function is currently defined as
function (coords,dim,xmin=min(coords[,1]),xmax=max(coords[,1]),ymin=min(coords[,2]),ymax=max(coords[,2])){


xints<-((xmax-xmin)/dim);
yints<-((ymax-ymin)/dim);

xbins<-seq(from=xmin, to=xmax-xints, by=xints);
ybins<-seq(from=ymin, to=ymax-yints, by=yints);



bins<-cbind(c(sapply(xbins,rep,dim)), rep(ybins,dim));
cnts<-matrix(apply(bins,1,belongtoint,vect=coords,int.x=xints,int.y=yints),nrow=dim,byrow=TRUE);


  }
```

Index