Package 'GmAMisc'

March 19, 2020

Title 'Gianmarco Alberti' Miscellaneous

Version 1.1.1

Description Contains many functions useful for univariate outlier detection, permutation-based t-test, permutation-based chi-square test, visualization of residuals, and bootstrap 'Cramer V', plotting of the results of the 'Mann-Whitney' and 'Kruskal-Wallis' test, calculation of 'Brainerd-Robinson' similarity coefficient and subsequent clustering, validation of logistic regression models, optimism-corrected AUC, robust 'Bland-Altman' plot, calculation of posterior probability for different chronological relationships between two Bayesian radiocarbon phases, point pattern analysis, clustering of spatial features.

Depends R (>= 3.4.0), maptools (>= 0.9-2)

Imports caTools (>= 1.17.1), classInt (>= 0.2-3), cluster (>= 2.0.7-1), coin (>= 1.2-2), corrplot (>= 0.84), DescTools (>= 0.99.24), dismo (>= 1.1-4), ggplot2 (>= 3.0.0), ggrepel (>= 0.8.0), graphics (>= 3.4.3), grDevices (>= 3.4.3), gridExtra (>= 2.3), Hmisc (>= 4.1-1), InPosition (>= 0.12.7), kimisc (>= 0.4), lsr (>= 0.5), methods (>= 3.4.3), plyr (>= 1.8.4), pROC (>= 1.12.1), raster (>= 2.6-7), RcmdrMisc (>= 1.0-10), rgdal (>= 1.3-3), rgeos (>= 0.3-28), sp (>= 1.3-1), spatstat (>= 1.56-0), stats (>= 3.4.3), utils (>= 3.4.3)

Suggests archdata (>= 1.2), gdistance (>= 1.2-2), resample (>= 0.4)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Gianmarco Alberti [aut, cre]

Maintainer Gianmarco Alberti <gianmarcoalberti@gmail.com>

Repository CRAN

Date/Publication 2020-03-19 18:10:02 UTC

R topics documented:

Aindex . . . 3
assemblage . . . 4
aucadj . . . 5
BRsim . . . 6
chiperm . . . 9
deaths . . . 10
distCovarModel . . . 10
distDiffTest . . . 13
distRandCum . . . 14
distRandSign . . . 16
events . . . 19
faults . . . 19
featClust . . . 20
impRst . . . 22
impShp . . . 23
kwPlot . . . 23
locations . . . 24
logregr . . . 25
log_regr_data . . . 27
malta_dtm_40 . . . 27
malta_polyg . . . 28
Massachusetts . . . 28
modelvalid . . . 28
mwPlot . . . 31
NNa . . . 32
outlier . . . 34
perm.t.test . . . 35
phases . . . 37
plotJenks . . . 37
points . . . 38
pointsCovarCum . . . 39
pointsCovarDistr . . . 40
pointsCovarModel . . . 42
pointsInPolygons . . . 44
pointsToPointsTess . . . 46
polygons . . . 47
popdensity . . . 48
ppdPlot . . . 48
prob.phases.relat . . . 49
pumps . . . 51
radioc_data . . . 51
refNNa . . . 52
resc.val . . . 53
rndpoints . . . 54
robustBAplot . . . 54
springs . . . 55

Starbucks 56
thiessenpolyg 56
vislim 57

Index **58**

`Aindex` *R function for calculating the Hodder-Okell's A index of spatial association*

Description

The function allows to calculate the Hodder-Okell's A index of spatial association between the features of two point patterns.

Usage

```
Aindex(x, y, studyplot = NULL, B = 199, addmap = FALSE)
```

Arguments

`x`	Point pattern (SpatialPointDataframe class).
`y`	Point pattern (SpatialPointDataframe class).
`studyplot`	Feature (of polygon type; SpatialPolygonsDataFrame class) representing the study area; if not provided, the study area is internally worked out as the bounding polygon based on the union the convex hulls of the x and y patterns. This is only used for visualization purpose, should the user want to plot the two point patterns within the actual study area.
`B`	Number of permutations (199 by default).
`addmap`	FALSE (default) or TRUE if the user does not want or wants a map of the study area and of the two patterns to be displayed.

Details

The functions takes as input two point patterns (SpatialPointDataframe class) and calculate the A index. Details about the latter are provided by:
Orton C. 1980, "Mathematics in Archeology", Glasgow: William Collins Sons & Co Ltd, pp. 154-155
Blankholm P. 1990, "Intrasite spatial Analysis in Theory and Practice", Aarhus: Aarhus University Press, pp. 130-135.

The A index is about equal to 1 when the two patterns are randomly mingled; it is smaller than 1 when the two patterns are segregated; it is larger than 1 when the features of the two point patterns tend to occur together. The computational details are provided by Blankholm's book cited above (page 132).

The significance of the A index is calculated via the randomized approach devised by:
Kintigh K W. 1990, “Intrasite Spatial Analysis: A Commentary of Major Methids”. In Voorrips A, “Mathematics and Information Science in Archaeology: A Flexible Framework”, Studies in Modern Archaeology 3: 165-200

Given two patterns A and B being analysed, the procedure keeps the points location unchanged and randomly assigns the points to either pattern. The random re-assignment is performed B times (199 by default) and each time the A index is calculated. One-tailed and two-tailed p values are calculated following the procedure described by Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, p. 387.

Value

The function produces:
-an histogram showing the frequency distribution of the randomized A index, with vertical reference lines representing the 0.025th and 0.975th quantile of the distribution. A black dot represents the observed A index. At the bottom of the chart the randomized p values are reported;
-optionally (setting the ’addmap’ parameter to TRUE), a map showing the point patterns (and the study area, if supplied).

See Also

distRandSign , distCovarModel , pointsCovarModel

Examples

```
# calculate the Hodder-Okell's A index for the two patterns, and plot the map
Aindex(springs, points, addmap=TRUE)
```

---

assemblage *Dataset: distribution of 7 archaeological objects across 9 assemblages*

Description

A dataset containing the frequencies (counts) of 7 objects across 9 assemblages.

Usage

```
data(assemblage)
```

Format

A data frame with 9 rows and 7 variables

aucadj *R function for optimism-adjusted AUC (internal validation)*

Description

The function allows to calculate the AUC of a (binary) Logistic Regression model, adjusted for optimism.

Usage

```
aucadj(data, fit, B)
```

Arguments

data	Dataframe containing the dataset (note: the Dependent Variable must be stored in the first column to the left).
fit	Object returned from glm() function.
B	Desired number of bootstrap resamples (suggested values: 100 or 200).

Details

The function performs an internal validation of a model via a bootstrap procedure (devised by Harrell and colleagues), which enables to estimate the degree of optimism of a fitted model and the extent to which the model will be able to generalize outside the training dataset. If you want more info, you can refer to this website (http://thestatsgeek.com/2014/10/04/adjusting-for-optimismoverfitting-in-measures-of-predictive-ability-using-bootstrapping/), and/or read the following interesting article (in which the bootstrap procedure is described at page 776): http://thestatsgeek.com/2014/10/04/adjustir for-optimismoverfitting-in-measures-of-predictive-ability-using-bootstrapping/

Value

The returned boxplots represent:
-the distribution of the AUC value in the bootstrap sample (auc.boot), which represents "an estimation of the apparent performance" (according to the aforementioned reference);
-the distribution of the AUC value deriving from the model fitted to the bootstrap samples and evaluated on the original sample (auc.orig), which represents the model performance on independent data.
At the bottom of the chart, the apparent AUC (i.e., the value deriving from the model fitted to the original dataset) and the AUC adjusted for optimism are reported.

See Also

logregr , modelvalid

Examples

```
# load the sample dataset
data(log_regr_data)

# fit a logistic regression model, storing the results into an object called 'model'
model <- glm(admit ~ gre + gpa + rank, data = log_regr_data, family = "binomial")

aucadj(data=log_regr_data, fit=model, B=200)
```

BRsim *R function for Brainerd-Robinson similarity coefficient (and optional clustering)*

Description

The function allows to calculate the Brainerd-Robinson similarity coefficient, taking as input a cross-tabulation (dataframe), and to optionally perform an agglomerative hierarchical clustering.

Usage

```
BRsim(data, which = "rows", correction = FALSE, rescale = TRUE,
  clust = TRUE, part = NULL, aggl.meth = "ward.D2", oneplot = TRUE,
  cex.dndr.lab = 0.85, cex.sil.lab = 0.75, cex.dot.plt.lab = 0.8)
```

Arguments

data	Dataframe containing the dataset (note: assemblages in rows, variables in columns).
which	Takes "rows" (default) if the user wants the coefficients be calculated for the row categories, "cols" if the users wants the coefficients be calculated for the column categories.
correction	Takes FALSE (default) if the user does not want the coefficients to be corrected, while TRUE will provide corrected coefficients.
rescale	Takes FALSE if the user does NOT want the coefficients to be rescaled between 0.0 and 1.0 (i.e., the user will get the original version of the Brainerd-Robinson coefficient (spanning from 0 [maximum dissimilarity] to 200 [maximum similarity]), while TRUE (default) will return rescaled coefficient.
clust	TRUE (default) or FALSE if the user does or does not want a agglomerative hierarchical clustering to be performed.
part	Desired number of clusters; if NULL (default), an optimal partition is calculated (see Details).
aggl.meth	Agglomeration method ("ward.D2" by default).
oneplot	TRUE (default) or FALSE if the user wants or does not want the plots to be visualized in a single window.

`cex.dndr.lab` Set the size of the labels used in the dendrogram.

`cex.sil.lab` Set the size of the labels used in the silhouette plot.

`cex.dot.plt.lab` Set the size of the labels used in the Cleveland's dot charts representing the by-cluster proportions.

Details

The function produces a correlation matrix in tabular form and a heat-map representing, in a graphical form, the aforementioned correlation matrix.

In the heat-map (which is built using the 'corrplot' package), the size and the color of the squares are proportional to the Brainerd-Robinson coefficients, which are also reported by numbers.

In order to "penalize" BR similarity coefficient(s) arising from assemblages with unshared categories, the function does what follows: it divides the BR coefficient(s) by the number of unshared categories plus 0.5. The latter addition is simply a means to be still able to penalize coefficient(s) arising from assemblages having just one unshared category. Also note that joint absences will have no weight on the penalization of the coefficient(s). In case of assemblages sharing all their categories, the corrected coefficient(s) turns out to be equal to the uncorrected one.

By setting the parameter 'clust' to TRUE, the units for which the BR coefficients have been calculated will be clustered. Notice that the clustering is based on a dissimilarity matrix which is internally calculated as the maximum values of the BR coefficient (i.e., 200 for the normal values, 1 for the rescales values) minus the BR coefficient. This allows a simpler reading of the dendrogram which is produced by the function, where the less dissimilar (i.e., more similar) units will be placed at lower levels, while more dissimilar (i.e., less similar) units will be placed at higher levels within the dendrogram.

The latter depicts the hierarchical clustering based (by default) on the Ward's agglomeration method; rectangles identify the selected cluster partition. Besides the dendrogram, a silhouette plot is produced, which allows to measure how 'good' is the selected cluster solution.

As for the latter, if the parameter 'part' is left empty (default), an optimal cluster solution is obtained. The optimal partition is selected via an iterative procedure which locates at which cluster solution the highest average silhouette width is achieved. If a user-defined partition is needed, the user can input the desired number of clusters using the parameter 'part'. In either case, an additional plot is returned besides the cluster dendrogram and the silhouette plot; it displays a scatterplot in which the cluster solution (x-axis) is plotted against the average silhouette width (y-axis). A black dot represent the partition selected either by the iterative procedure or by the user.

Notice that in the silhouette plot, the labels on the left-hand side of the chart show the units' names and the cluster number to which each unit is closer.

The silhouette plot is obtained from the 'silhouette()' function out from the 'cluster' package (https://cran.r-project.org/web/packages/cluster/index.html).

For a detailed description of the silhouette plot, its rationale, and its interpretation, see:
Rousseeuw P J. 1987. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis", Journal of Computational and Applied Mathematics 20, 53-65 (http://www.sciencedirect.com/science/article/pii/03

The function also provides a Cleveland's dot plots that represent by-cluster proportions. The clustered units are grouped according to their cluster membership, the frequencies are summed, and then expressed as percentages. The latter are represented by the dot plots, along with the average percentage. The latter provides a frame of reference to understand which percentage is below, above, or close to the average. The raw data on which the plots are based are stored within the list returned by the function (see below).

Value

The function returns a list storing the following components

- $BR_similarity_matrix: similarity matrix showing the BR coefficients
- $BR_distance_matrix: dissimilarity matrix on which the hierarchical clustering is performed (if selected)
- $avr.silh.width.by.n.of.clusters: average silhouette width by number of clusters (if clustering is selected)
- $partition.silh.data: silhouette data for the selected partition (if clustering is selected)
- $data.w.cluster.membership: copy of the input data table with an additional column storing the cluster membership for each row (if clustering is selected)
- $by.cluster.proportion: data table showing the proportion of column categories across each cluster; rows sum to 100 percent (if clustering is selected)

See Also

corrplot , silhouette

Examples

```
data(assemblage)
coeff <- BRsim(data=assemblage, correction=FALSE, rescale=TRUE, clust=TRUE, oneplot=FALSE)

library(archdata) #load the 'archdata' package

#load the 'Nelson' dataset out of the 'archdata' package
data(Nelson)

#build a table to examine
table <- as.data.frame(as.matrix(Nelson[,3:7]))

# perform the analysis and store the results in the 'res' object
res <- BRsim(table, which="rows", clust=TRUE, oneplot=FALSE)
```

chiperm *R function for permutation-based chi-square test of independence*

Description

The function performs the chi-square test of independence on the basis of permuted tables, whose number is selected by user.

Usage

```
chiperm(data, B = 999, resid = FALSE, filter = FALSE,
  thresh = 1.96, cramer = FALSE)
```

Arguments

data	Dataframe containing the input contingency table.
B	Desired number of permuted tables (999 by default).
resid	TRUE or FALSE (default) if the user does or doesn't want to plot the table of Pearson's standardized residuals.
filter	Takes TRUE or FALSE (default) if the user does or does't want to filter the Pearson's standardized residuals according to the threshold provided by the thresh parameter; by default, the threshold is set at 1.96, which corresponds to an alpha level of 0.05.
thresh	Value of the standardized residuals below which the residuals will be not displayed (by default, the threshold is set at 1.96, which corresponds to an alpha level of 0.05).
cramer	Takes TRUE or FALSE (default) if the user does or doesn't want to calculate and plot the bootstrap confidence interval for Cramer's V.

Details

For the rationale of this approach, see for instance the description provided by:
Beh E.J., Lombardo R. 2014, Correspondence Analysis: Theory, Practice and New Strategies, Chichester, Wiley, pages 62-64.

Value

The function produces:
(1) a chart that displays the permuted distribution of the chi-square statistic based on B permuted tables. The selected number of permuted tables, the observed chi-square, the 95th percentile of the permuted distribution, and the associated p value are reported at the bottom of the chart;

(2) a chart that displays the bootstrap distribution of Cramer's V coefficient, based on a number of bootstrap replicates which is equal to the value of the function's parameter B;

(3) a chart that the Pearson's Standardized Residuals: a colour scale allows to easily understand which residual is smaller (BLUE) or larger (RED) than expected under the hypothesis of independence. Should the user want to only display residuals larger than a given threshold, it suffices to set the filter parameter to TRUE, and to specify the desired threshold by means of the thresh parameter, which is set at 1.96 by default.

Examples

```
data(assemblage)
chiperm(data=assemblage, B=199, resid=TRUE, cramer=TRUE)
```

`deaths` *Dataset: location of cholera deaths in London (after Dr Snow's mid-1800s study of cholera outbreak in Soho).*

Description

A SpatialPointsDataFrame representing the location of cholera deaths in London (after Dr Snow's mid-1800s study of cholera outbreak in Soho).

Usage

```
data(deaths)
```

Format

SpatialPointsDataFrame

`distCovarModel` *R function to model (and test) the dependence of a point pattern on the distance to another pattern*

Description

The function is a wrapper for a number of functions out of the extremely useful 'spatstat' package (specifically, ppm(), cdf.test(), auc(), roc(), effectfun()). It allows to test if there is a significant dependence of the input point pattern on a spatial covariate (first-order effect), the latter being the distance to another feature (of either point or line type).
The function takes as input two datasets: a point patter (SpatialPointsDataFrame class) and a feature (either SpatialPointsDataFrame or SpatialLinesDataFrame class) the distance to which is used as spatial covariate for the input point pattern.

Usage

```
distCovarModel(feature, cov.var, studyplot = NULL, buffer = 0,
  Foxall = FALSE, oneplot = FALSE)
```

Arguments

feature	Feature (of point type; SpatialPointsDataFrame class) representing the spatial point pattern of interest.
cov.var	Feature (of either point or line type; SpatialPointsDataFrame or SpatialLinesDataFrame class) the distance to which represents the spatial covariate.
studyplot	Feature (of polygon type; SpatialPolygonsDataFrame) representing the study area; if not provided, the study area is internally worked out as the bounding polygon based on the union the convex hulls of the feature and of the cov.var data.
buffer	Add a buffer to the convex hull of the study area (0 by default); the unit depends upon the units of the input data.
Foxall	Set to TRUE, will plot the Foxall's J function.
oneplot	Set to TRUE (default), will plot the charts into a single visualization.

Details

The function fits a inhomogeneous Poisson point process (Alternative Model-H1) with the distance to the second feature entered by the user ('cov.var' parameter) used as spatial covariate. In other words, the fitted alternative model is a Poisson point process with intensity of the point pattern as a loglinear function of the distance to the second pattern entered by the user (see Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 309-313). The distance to the second feature is internally calculated via the spatstat's 'distfun()' function.

Also, the function fits a homogeneous Poisson point model (Null Model-H0, equivalent to Complete Spatial Randomness: Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 305-306), that is used as comparison for the inhomogeneous point process model in a Likelihood Ratio test (Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 334-335). A significant result, i.e. a low p-value, suggests rejecting the Null Hypothesis of CSR in favour of the Alternative Hypothesis of a Poisson point process affected by a covariate effect (i.e., inhomogeneous intensity due to the influence of the covariate) (Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 305).

The function returns a 4 plots, which can be arranged in just one visualization setting the parameter oneplot to TRUE:

-plot of the study area along with the point pattern of interest and the second feature entered by the user (whose distance is the spatial covariate);

-plot of the fitted intensity against the spatial covariate (Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 308);

-plot of the cumulative distribution of the covariate at the data points against the cumulative distribution of the covariate at all the spatial location within the study area (rationale: Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 184-185);

-plot of the ROC curve, which help assessing the strength of the dependence on the covariate (Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 187-188).

Setting the parameter Foxall to TRUE, the third plot will be replaced by the chart of the Foxall's J function, which is another "useful statistic" when the covariate is the distance to a spatial pattern (Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 187, 282-284). Values of J are uqual to 1 when the two patterns are independent random patterns; values <1 indicate that the input point pattern tends to be closer to the cov.var pattern than expected for random points; values >1 indicate that the input point pattern avoid the cov.var pattern, i.e. the point pattern is more likely than random points to lie far away from the cov.var pattern (see Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 284).

Value

The function returns a list storing the following components

- $H0-model: info and relevant statistics regarding the Null Model
- $H1-model: info and relevant statistics regarding the Alternative Model
- $Model comparison (LRT) results of the Likelihood Ratio test
- $AIC-H0: AIC of the Null Model
- $AIC-H1: AIC of the Alternative Model
- $KS test: information regarding the cumulative distribution comparison via Kolmogorov-Smirnov test
- $AUC: the AUC statistics

See Also

distRandSign , Aindex , pointsCovarModel , auc , cdf.test , effectfun , ppm , roc

Examples

```
#load a point dataset representing some locations
data(locations)

#load a point dataset representing some locations, the distance to which
#is used as spatial covariate
data(springs)

#perform the analysis, and store the results in the 'results' object
results <- distCovarModel(locations, springs)
```

`distDiffTest` *R function for testing the difference in distance of two point feature datasets to a target feature dataset*

Description

The function allows to perform a permutation-based t-test to test the difference in distance of two point feature datasets to a target feature dataset. The latter can consist of either points, a lines, or polygons.

Usage

```
distDiffTest(feat1, feat2, to.feat, feat1.lab = NULL, feat2.lab = NULL,
  B = 999)
```

Arguments

`feat1`	Point pattern to be tested (of point type; 'SpatialPointsDataFrame' class).
`feat2`	Second point pattern to be tested (of point type; 'SpatialPointsDataFrame' class).
`to.feat`	Target feature (point, polyline, or polygon type; 'SpatialPointsDataFrame', 'SpatialLinesDataFrame', 'SpatialPolygonsDataFrame' class).
`feat1.lab`	Label to be used in the returned chart to indicate the 'feat1' (default: smpl 1).
`feat2.lab`	Label to be used in the returned chart to indicate the 'feat2' (default: smpl 2).
`B`	Desired number of permutations (set at 999 by default).

Details

Under the hood, the function relies on the perm.t.test() function out of this same package. First, for each feature of both patterns, the distance to the nearest target feature is calculated; for each set of features, the distances are eventually averaged; the observed difference between the two averages is stored. Then, the individual observed nearest distances are randomly assigned to either group; the re-assignment is performed B times (999 by default) and each time the difference between the two averages is calculated. The distribution of these permuted average differenes represents the distribution of that statistic under the Null Hypothesis of no difference in distance to the target feature. One-sided and two-sided p-values are reported.

Value

The frequency histogram returned by the function displays the distribution of the permuted mean difference between the two samples; a solid dot indicates the observed mean difference, while an hollow dot represents the mean of the permuted differences. Two dashed blue lines indicates the 0.025 and 0.975 percentile of the permuted distribution. A rug plot at the bottom histgram indicates the individual permuted mean differences. At the bottom of the chart, some information are displayed. In particular, the observed mean difference and the permuted p-values are reported. In the last row, the result of the regular (parametric) t-test (both assuming and not assuming equal variances) is reported to allow users to compare the outcome of these different versions of the test.

See Also

perm.t.test

Examples

```
#test the difference in distance of two sets of points to the nearest geological fault
distDiffTest(feat1=springs, feat2=points, to.feat=faults, B=299)
```

---

distRandCum — *R function to test the significance of the spatial relationship between two features in terms of the cumulative distribution of minimum distances*

Description

The function allows to assess if there is a significant spatial association between a point pattern and the features of another pattern. For instance, users may want to assess if the features of a point pattern tend to lie close to some features represented by polylines.

Usage

```
distRandCum(from.feat, to.feat, studyplot = NULL, buffer = 0,
  B = 200, type = "rand")
```

Arguments

from.feat	Feature (of point type; SpatialPointsDataFrame class) whose spatial association with the to-feature has to be assessed.
to.feat	Feature (point, polyline, or polygon type; SpatialPointsDataFrame, SpatialLinesDataFrame, SpatialPolygonsDataFrame class) in relation to which the spatial association of the from-feature has to be assessed.
studyplot	Feature (of polygon type; SpatialPolygonsDataFrame class) representing the study area; if not provided, the study area is internally worked out as the bounding polygon based on the union the convex hulls of the from- and of the to-feature.
buffer	Add a buffer to the convex hull of the study area (0 by default); the unit depends upon the units of the input data.
B	Number of randomizations to be used (200 by default).
type	By default is set to "rand", which performs the randomization-based analysis; if both the from.feature and the to.feature dataset are of point type, setting the parameter to "perm" allows to opt for the permutation-based approach.

Details

Given a from-feature (event for which we want to estimate the spatial association with the to-feature) and a to-feature (event in relation to which we want to estimate the spatial association for the from-feature), the assessment is performed by means of a randomized procedure:

-keeping fixed the location of the to-feature, random from-features are drawn B times (the number of randomized from-features is equal to the number of observed from-features);
-for each draw, the minimum distance to the to-features is calculated; if the to-feature is made up of polygons, the from-features falling within a polygon will have a distance of 0;
-a cumulative distribution of random minimum distances is thus obtained;
-the cumulative random minimum distances are used to work out an acceptance interval (with significance level equal to 0.05; sensu Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 208) that allows to assess the statistical significance of the cumulative distribution of the observed minimum distances, and that is built using the above-mentioned B realizations of a Complete Spatial Random process.

The from-feature must be a point feature, whilst the to-feature can be a point or a polyline or a polygon feature.

The rationale of the procedure is that, if there indeed is a spatial association between the two features, the from-feature should be closer to the to-feature than randomly generated from-features. If the studyplot shapefile is not provided, the random locations are drawn within a bounding polygon based on the union the convex hulls of the from- and of the to-feature.

If both the from-feature and the to-feature are of point type (SpatialPointsDataFrame class), the user may opt for the randomized procedure described above (parameter 'type' set to 'rand'), or for a permutation-based procedure (parameter 'type' set to 'perm'). Unlike the procedure described above, whereby random points are drawn within the study area, the permutation-based routine builds a cumulative distribution of minimum distances keeping the points location unchanged and randomly assigning the points to either of the two patterns. The re-assignment is performed B times (200 by default) and each time the minimum distance is calculated.

For an example of the use of the analysis, **see** for instance Carrero-Pazos, M. (2018). Density, intensity and clustering patterns in the spatial distribution of Galician megaliths (NW Iberian Peninsula). Archaeological and Anthropological Sciences. https://doi.org/10.1007/s12520-018-0662-2, fig. 6.

Value

The function produces a cumulative distribution chart in which the distribution of the observed minimum distances is represented by a black line, and acceptance interval is represented in grey. The number of iteration used and the type of analysis (whether randomization-based or permutation-based) are reported in the chart's title.

See Also

distRandSign

Examples

```
data(springs)

data(faults)

#perform the analysis using 50 iterations and
#the default randomization-based approach
distRandCum(from.feat=springs, to.feat=faults, B=50)

data("malta_polyg") # load a sample polygon

#perform the analysis; since both patterns are of point type but the 'type' parameter is left
#in its default value ('rand'), the randomization-based approach is used
distRandCum(springs, points, studyplot=malta_polyg, B=50)

#same as above, but using the permutation-based approach
distRandCum(springs, points, studyplot=malta_polyg, type="perm", B=50)
```

---

distRandSign *R function to test for a significant spatial association between two features (points-to-points, points-to-lines, points-to-polygons)*

Description

The function allows to assess if there is a significant spatial association between a point pattern and the features of another pattern. For instance, users may want to assess if some locations tend to lie close to some features represented by polylines. By the same token, users may want to know if there is a spatial association between the location of a given event and the location of another event. See the example provided further below (in the examples section), where the question to address is if there is a spatial association between springs and geological fault-lines; in other words: do springs tend to be located near the geological faults?

Usage

```
distRandSign(from.feat, to.feat, studyplot = NULL, buffer = 0,
  B = 199, oneplot = TRUE, export = FALSE)
```

Arguments

from.feat	Feature (of point type; SpatialPointsDataFrame class) whose spatial association with the to-feature has to be assessed.
to.feat	Feature (point, polyline, or polygon type; SpatialPointsDataFrame, SpatialLinesDataFrame, SpatialPolygonsDataFrame class) in relation to which the spatial association of the from-feature has to be assessed.

studyplot	Feature (of polygon type; SpatialPolygonsDataFrame class) representing the study area; if not provided, the study area is internally worked out as the bounding polygon based on the union the convex hulls of the from- and of the to-feature.
buffer	Add a buffer to the convex hull of the study area (0 by default); the unit depends upon the units of the input data.
B	Number of randomizations to be used (199 by default).
oneplot	TRUE (default) or FALSE if the user wants or does not want the plots displayed in a single window.
export	FALSE (default) or TRUE is the user wants to export the input dataset as a shapefile (see description for further info).

Details

Given a from-feature (event for which we want to estimate the spatial association with the to-feature) and a to-feature (event in relation to which we want to estimate the spatial association for the from-feature), the assessment is performed by means of a randomized procedure:

-keeping fixed the location of the to-feature, random from-features are drawn B times (the number of randomized from-features is equal to the number of observed from-features);
-for each draw, the average minimum distance to the to-features is calculated; if the to-feature is made up of polygons, the from-features falling within a polygon will have a distance of 0;
-a distribution of average minimum distances is thus obtained;
-p values are computed following Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, p. 387.

The from-feature must be a point feature, whilst the to-feature can be a point or a polyline or a polygon feature.

The rationale of the procedure is that, if there indeed is a spatial association between the two features, the from-feature should be on average closer to the to-feature than randomly generated from-features. If the studyplot shapefile is not provided, the random locations are drawn within a bounding polygon based on the union the convex hulls of the from- and of the to-feature.

If both the from-feature and the to-feature are of point type (SpatialPointsDataFrame class), the function also test the spatial association by means of a permuted procedures. Unlike the procedure described above, whereby random points are drawn within the study area, the permutation-based routine builds a distribution of averages minimum distances keeping the points location unchanged and randomly assigning the points to either of the two patterns. The re-assignment is performed B times (199 by default) and each time the average minimum distance is calculated.

The function produces a histogram showing: the distribution of randomized average minimum distances; a black dot indicating the observed average minimum distance; a hollow dot representing the average of the randomized minimum distances; two blue reference lines correspond to the 0.025th and to the 0.975th quantile of the randomized distribution. P-values are reported at the bottom of

the plot.

In case both the from- and the to- feature are of point type, another histogram is produced, which provides the same information of the preceding histogram, but derived from the permutation-based routine that has been detailed above.

The function also produces a map showing the study area, the from- and the to-features. The from-features are given a colour according to whether or not their minimum distance to the nearest to-feature is smaller (GREEN) or larger (RED) than the 0.025 and 0.975 (respectively) of the distribution of the randomized average distances. This information is also reported in a new field that is appended to the input dataset. Optionally, the input dataset (with 2 new columns added) can be exported using the 'export'parameter. The new columns store: the features' minimum distance to the nearest to-feature; a string indicating if the corresponding feature is closer or more distant than expected to the nearest to-feature.

Value

The function returns a list storing the following components

- $from.feat.min.dist: distance of each entity of the from-feature to the nearest entity of the to-feature
- $avrg.obs.min.dist: observed average minimum distance
- $avrg.rnd.min.dist: average randomized minimum distance
- $avrg.perm.min.dist: average permuted minimum distance (returned only when both the from- and to- features are of point type)
- $p.value closer than expected-rnd-
- $p.value closer than expected-perm- (returned only when both the from- and to- features are of point type)
- $p.value more distant than expected-rnd-
- $p.value more distant than expected-perm- (returned only when both the from- and to- features are of point type)
- $p.value different from random-rnd-:
- $p.value different from random-perm- (returned only when both the from- and to- features are of point type)
- $dataset: from.feature dataset with 2 fields added: one ('obs.min.dist') storing the from-features's minimum distance to the nearest to-feature, one ('signif') storing the significance of the distance (see description)

See Also

distRandCum , distCovarModel , Aindex

Examples

```
data(springs)
data(faults)

#calculate the significance of the spatial association between springs and geological
#fault-lines; plots displayed in a single panel
result <- distRandSign(from.feat=springs, to.feat=faults, oneplot=TRUE, B=49)

data(points)
data(polygons)

#calculate the significance of the spatial association between points and polygons
result <- distRandSign(from.feat=points, to.feat=polygons, oneplot=FALSE, B=49)
```

---

events *Dataset: location of fictional events*

Description

A SpatialPointsDataFrame representing fictional events.

Usage

```
data(events)
```

Format

SpatialPointsDataFrame

---

faults *Dataset: geological fault-lines in Malta*

Description

A SpatialLinesDataFrame representing the geological fault-lines in Malta.

Usage

```
data(faults)
```

Format

SpatialLinesDataFrame

`featClust` *R function for features clustering on the basis of distances/area*

Description

The function provides the facility to cluster the features of the input dataset on the basis of either their (projected) coordinates (for points; SpatialPointsDataFrame class) or of their area (for polygons; SpatialPolygonsDataFrame class). If a target feature dataset (to.feat) is provided, the clustering will be based on the distance of the x feature to the nearest to.feature. When a to.feature is specified, the x feature (i.e., the feature that the user wants to cluster) can be either a point (SpatialPointsDataFrame class), or a polyline (SpatialLinesDataFrame class) , or a polygon (SpatialPolygonsDataFrame class) feature. Notice that if all the x features overlap with all the to.feature, all the minimum distances will be 0, and the function will trow an error.

Usage

```
featClust(x, to.feat = NULL, aggl.meth = "ward.D2", part = NULL,
  showID = TRUE, oneplot = TRUE, cex.dndr.lab = 0.85,
  cex.sil.lab = 0.75, cex.feat.lab = 0.65, col.feat.lab = "black",
  export = FALSE)
```

Arguments

`x`	Dataset whose feature are to be clustered; either points (SpatialPointsDataFrame class) or polygons (SpatialPolygonsDataFrame class); if the to.feat is specified, x can also be a polylines feature (SpatialLinesDataFrame class).
`to.feat`	Dataset (NULL by default) representing the feature the distance toward which is used as basis for clustering x; either points (SpatialPointsDataFrame class), polygons (SpatialPolygonsDataFrame class), or polylines (SpatialLinesDataFrame).
`aggl.meth`	Agglomeration method ("ward.D2" by default).
`part`	Desired number of clusters; if NULL (default), an optimal partition is calculated (see Details).
`showID`	TRUE (default) or FALSE if the used wants or does not want the ID of the clustered features to be displayed in the plot where the features are colored by cluster membership.
`oneplot`	TRUE (default) or FALSE if the user wants or does not want the plots to be visualized in a single window.
`cex.dndr.lab`	Set the size of the labels used in the dendrogram.
`cex.sil.lab`	Set the size of the labels used in the silhouette plot.
`cex.feat.lab`	Set the size of the labels used (if 'showID' is set to TRUE) to show the clustered features' IDs.
`col.feat.lab`	Set the color of the clustered features' IDs ('black' by default).
`export`	TRUE or FALSE (default) if the user wants or does not want the clustered input dataset to be exported; if TRUE, the input dataset with a new variable indicating the cluster membership will be exported as a shapefile.

Details

If the to.feature is not provided, the function internally calculates a distance matrix (based on the Euclidean Distance) on the basis of the points' coordinates or polygons' area. If the to.feature is provided, the distance matrix will be based on the distance of the x feature to the nearest to.feature. A dendrogram is produced which depicts the hierarchical clustering based (by default) on the Ward's agglomeration method; rectangles identify the selected cluster partition. Besides the dendrogram, a silhouette plot is produced, which allows to measure how 'good' is the selected cluster solution.

As for the latter, if the parameter 'part' is left empty (default), an optimal cluster solution is obtained. The optimal partition is selected via an iterative procedure which locates at which cluster solution the highest average silhouette width is achieved. If a user-defined partition is needed, the user can input the desired number of clusters using the parameter 'part'. In either case, an additional plot is returned besides the cluster dendrogram and the silhouette plot; it displays a scatterplot in which the cluster solution (x-axis) is plotted against the average silhouette width (y-axis). A black dot represent the partition selected either by the iterative procedure or by the user.

Notice that in the silhouette plot, the labels on the left-hand side of the chart show the point ID number and the cluster to which each point is closer.

Also, the function returns a plot showing the input dataset, with features colored by cluster membership. Two new variables are added to the shapefile's dataframe, storing a point ID number and the corresponding cluster membership.

The silhouette plot is obtained from the 'silhouette()' function out from the 'cluster' package (https://cran.r-project.org/web/packages/cluster/index.html).
For a detailed description of the silhouette plot, its rationale, and its interpretation, see:
Rousseeuw P J. 1987. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis", Journal of Computational and Applied Mathematics 20, 53-65 (http://www.sciencedirect.com/science/article/pii/03

For the hierarchical clustering of features, see: Conolly, J., & Lake, M. (2006). Geographic Information Systems in Archaeology. Cambridge: Cambridge University Press, 168-173.

Value

The function returns a list storing the following components

- $dist.matrix: distance matrix
- $avr.silh.width.by.n.of.clusters: average silhouette width by number of clusters
- $partition.silh.data: silhouette data for the selected partition
- $coord.or.area.or.min.dist.by.clust: coordinates, area, or distance to the nearest to.feat coupled with cluster membership
- $dist.stats.by.cluster: by-cluster summary statistics of the x feature distance to the nearest to.feature
- $dataset: the input dataset with two variables added ($feat_ID and $clust, the latter storing the cluster membership)

Examples

```
data(springs)

#perform the analysis and automatically select an optimal partition
res <- featClust(springs)

#as above, but selecting a 3-cluster partition
res <- featClust(springs, part=3)

#cluster springs on the basis of their distance to the nearest geological fault
res <- featClust(springs, faults)

#cluster polygonal areas on the basis of their distance to the nearest spring
res <- featClust(polygons, springs)

#cluster points on the basis of their distance to the nearest polygon
res <- featClust(points, polygons)
```

---

impRst *R function to easily import a raster dataset into R*

Description

The function is a wrapper for the 'raster()' function out of the 'raster' package. It provides the facility to import a raster dataset ('RasterLayer' class) by means of a window that allows the user to navigate through the computer's folders and to select the appropriate file.

Usage

```
impRst()
```

Examples

```
## Not run:
#a window will pop up allowing the user to select the raster dataset
my.raster <- impRst()

## End(Not run)
```

impShp *R function to easily import a vectorial dataset (shapefile) into R*

Description

The function is a wrapper for the 'shapefile()' function out of the 'raster' package. It provides the facility to import a vectorial dataset (of shapefile type) by means of a window that allows the user to navigate through the computer's folders and to select the appropriate file.

Usage

```
impShp()
```

Examples

```
## Not run:
#a window will pop up allowing the user to select the shapefile
my.shapefile <- impShp()

## End(Not run)
```

kwPlot *R function for visually displaying Kruskal-Wallis test's results*

Description

The function allows allows to perform Kruskal-Wallis test, and to display the test's results in a plot along with boxplots.

Usage

```
kwPlot(x, y, strip = FALSE, notch = FALSE, omm = FALSE,
  outl = TRUE, posthoc = FALSE, adjust = "bonferroni")
```

Arguments

x	Object storing the values to be analysed.
y	Object storing a grouping variable with 3 or more levels.
strip	Logical value which takes FALSE (by default) or TRUE if the user wants jittered points to represent individual values.
notch	Logical value which takes FALSE (by default) or TRUE if user does not or do want to have notched boxplots in the final display, respectively; it is worth noting that overlapping of notches indicates a not significant difference at about 95 percent confidence.

omm	It stands for overall mean and median; takes FALSE (by default) or TRUE if user wants the mean and median of the overall sample plotted in the chart (as a dashed RED line and dotted BLUE line respectively).
outl	Logical value which takes FALSE or TRUE (by default) if users want the boxplots to display outlying values.
posthoc	Logical value which takes FALSE (default) or TRUE if user does not or does want to perform a follow-up test (namely, the Dunn's test) in order to locate which group significantly differs from the others.
adjust	Sets the desidered method for p-values adjustment in the context of the Dunn's test; the list of methods is the following: Bonferroni ("bonferroni"; default); Holm ("holm"), Hochberg ("hochberg"), Hommel ("hommel"), Benjamini & Hochberg ("BH" or its alias "fdr"), Benjamini & Yekutieli ("BY"), none ("none"). For more info, see the 'p.adjust' help documentation in R (?p.adjust).

Details

The boxplots display the distribution of the values of the two samples, and jittered points represent the individual observations. At the bottom of the chart, the test statistics (H) is reported, along with the degrees of freedom and the associated p value.
Setting the parameter 'posthoc' to TRUE, the Dunn's test is performed (with Bonferroni adjustment by default): a dot chart is returned, as well as a list of p-values (2-sided). In the dot chart, a RED line indicates the 0.05 threshold. The groups compared on a pairwise basis are indicated on the left-hand side of the chart.

See Also

p.adjust

Examples

```
#create a toy dataset
mydata <- data.frame(values=c(rnorm(30, 100,10),rnorm(30, 80,10),rnorm(30, 98,10)),
group = as.factor(gl(3, 30, labels = c("A", "B", "C"))))

# performs the test, displays the test's result, including jittered points and notches.
# It also performs the Dunn's posthoc test using Bonferroni p-value correction.
kwPlot(x=mydata$values, y=mydata$group, strip=TRUE, notch=TRUE, posthoc=TRUE)
```

---

locations *Dataset: location of fictional locations*

Description

A SpatialPointsDataFrame representing fictional locations.

Usage

```
data(locations)
```

Format

SpatialPointsDataFrame

---

logregr *R function easy binary Logistic Regression and model diagnostics*

Description

The function allows to make it easy to perform binary Logistic Regression, and to graphically display the estimated coefficients and odds ratios. It also allows to visually check model's diagnostics such as outliers, leverage, and Cook's distance.

Usage

```
logregr(data, oneplot = FALSE)
```

Arguments

data	Dataframe containing the dataset (Dependent Variable listed in the first column to the left).
oneplot	Logical value which takes TRUE or FALSE (default) if the user does or doesn't want to group the first set of 8 charts in one panel.

Value

The function may take a while (just matter of few seconds) to completed all the operations, and will eventually return the following charts:

(1) Estimated coefficients, along with each coefficient's confidence interval; a reference line is set to 0. Each bar is given a color according to the associated p-value, and the key to the color scale is reported in the chart's legend.

(2) Odds ratios and their confidence intervals.

(3) A chart that is helpful in visually gauging the discriminatory power of the model: the predicted probability (x axis) are plotted against the dependent variable (y axis). If the model proves to have a high discriminatory power, the two stripes of points will tend to be well separated, i.e. the positive outcome of the dependent variable (points with color corresponding to 1) would tend to cluster around high values of the predicted probability, while the opposite will hold true for the negative outcome of the dependent variable (points with color corresponding to 0). In this case, the AUC

(which is reported at the bottom of the chart) points to a low discriminatory power.

(4) Model's standardized (Pearson's) residuals against the predicted probability; the size of the points is proportional to the Cook's distance, and problematic points are flagged by a label reporting their observation number if the following two conditions happen: residual value larger than 3 (in terms of absolute value) AND Cook's distance larger than 1. Recall that an observation is an outlier if it has a response value that is very different from the predicted value based on the model. But, being an outlier doesn't automatically imply that that observation has a negative effect on the model; for this reason, it is good to also check for the Cook's distance, which quantifies how influential is an observation on the model's estimates. Cook's distance should not be larger than 1.

(5) Predicted probability plotted against the leverage value; dots represent observations, and their size is proportional to their leverage value, and their color is coded according to whether or not the leverage is above (lever. not ok) or below (lever. ok) the critical threshold. The latter is represented by a grey reference line, and is also reported at the bottom of the chart itself. An observation has high leverage if it has a particularly unusual combination of predictor values. Observations with high leverage are flagged with their observation number, making it easy to spot them within the dataset. Remember that values with high leverage and/or with high residual may be potential influencial points and may potentially negatively impact the regression. As for the leverage threshold, it is set at 3*(k+1)/N (following Pituch-Stevens, Applied Multivariate Statistics for the Social Science. Analyses with SAS and IBM's SPSS, Routledge: New York 2016), where k is the number of predictors and N is the sample size.

(6) Predicted probability against the Cook's distance.

(7) Standardized (Pearson's) residuals against the leverage; points representing observations with positive or negative outcome of the dependent variable are given different colors. Further, points' size is proportional to the Cook's distance. Leverage threshold is indicated by a grey reference line, and the threshold value is also reported at the bottom of the chart. Observations are flagged with their observation number if their residual is larger than 3 (in terms of absolute value) OR if leverage is larger than the critical threshold OR if Cook's distance is larger than 1. This allows to easily check which observation turns out to be an outlier or a high-leverage data point or an influential point, or a combination of the three.

(8) Chart that is almost the same as (7) except for the way in which observations are flagged. In fact, they are flagged if the residual is larger than 3 (again, in terms of absolute value) OR if the leverage is higher than the critical threshold AND if a Cook's distance larger than 1 plainly declares them as having a high influence on the model's estimates. Since an observation may be either an outlier or a high-leverage data point, or both, and yet not being influential, the chart allows to spot observations that have an undue influence on our model, regardless of them being either outliers or high-leverage data points, or both.

(9) Observation numbers are plotted against the standardized (Pearson's) residuals, the leverage, and the Cook's distance. Points are labelled according to the rationales explained in the preceding points. By the way, the rationale is also explained at the bottom of each plots.

The function also returns a list storing two components: one is named 'formula' and stores the formula used for the logistic regression; the other contains the model's results.

See Also

modelvalid , aucadj

`log_regr_data` *Dataset: admission to graduate school*

Description

A dataset containing information about the admission to graduate school of 400 individuals; the dataset features a binary dependent variable and 3 predictors. Dataset is after: https://stats.idre.ucla.edu/r/dae/logit-regression/

Usage

```
data(log_regr_data)
```

Format

A data frame with 400 rows and 4 variables

Details

- admit. Binary dependent variable (admission yes=1, admission no=0)
- gre. Graduate Record Exam score; predictor (continuous)
- gpa. Grade Point Average; predictor (continuous)
- rank. Prestige of the undergraduate school (ordinal)

`malta_dtm_40` *Dataset: Malta DTM (40m cell size)*

Description

A RasterLayer representing a Digital Terrain Model for Malta (40m resolution).

Usage

```
data(malta_dtm_40)
```

Format

RasterLayer

malta_polyg *Dataset: Malta polygon*

Description

A SpatialPolygonsDataFrame representing Malta.

Usage

```
data(malta_polyg)
```

Format

SpatialPolygonsDataFrame

Massachusetts *Dataset: Massachusetts state limit*

Description

A SpatialPolygonsDataFrame representing the limits of Massachusetts.
After https://mgimond.github.io/Spatial/point-pattern-analysis-in-r.html.

Usage

```
data(Massachusetts)
```

Format

SpatialPolygonsDataFrame

modelvalid *R function for binary Logistic Regression internal validation*

Description

The function allows to perform internal validation of a binary Logistic Regression model implementing most of the procedure described in:
Arboretti Giancristofaro R, Salmaso L. "Model performance analysis and model validation in logistic regression". Statistica 2003(63): 375–396.

Usage

```
modelvalid(data, fit, B = 200, g = 10, oneplot = TRUE,
  excludeInterc = FALSE)
```

Arguments

data	Dataframe containing the dataset (Dependent Variable must be stored in the first column to the left).
fit	Object returned from glm() function.
B	Desired number of iterations (200 by default).
g	Number of groups to be used for the Hosmer-Lemeshow test (10 by default).
oneplot	TRUE (default) is the user wants the charts returned in a single visualization.
excludeInterc	If set to TRUE, the chart showing the boxplots of the parameters distribution across the selected iteration will have y-axis limits corresponding to the min and max of the parameters value; this allows better displaying the boxplots of the model parameters when they end up showing up too much squeezed due to comparatively higher/lower values of the intercept. FALSE is default.

Details

The procedure consists of the following steps:

(1) the whole dataset is split into two random parts, a fitting (75 percent) and a validation (25 percent) portion;

(2) the model is fitted on the fitting portion (i.e., its coefficients are computed considering only the observations in that portion) and its performance is evaluated on both the fitting and the validation portion, using AUC as performance measure;

(3) the model's estimated coefficients, p-values, and the p-value of the Hosmer and Lemeshow test are stored;

(4) steps 1-3 are repeated B times, eventually getting a fitting and validation distribution of the AUC values and of the HL test p-values, as well as a fitting distribution of the coefficients and of the associated p-values. The AUC fitting distribution provides an estimate of the performance of the model in the population of all the theoretical fitting samples; the AUC validation distribution represents an estimate of the model's performance on new and independent data.

Value

The function returns:

-a chart with boxplots representing the fitting distribution of the estimated model's coefficients; coefficients' labels are flagged with an asterisk when the proportion of p-values smaller than 0.05

across the selected iterations is at least 95 percent;

-a chart with boxplots representing the fitting and the validation distribution of the AUC value across the selected iterations. for an example of the interpretation of the chart, see the aforementioned article, especially page 390-91;

-a chart of the levels of the dependent variable plotted against the predicted probabilities (if the model has a high discriminatory power, the two stripes of points will tend to be well separated, i.e. the positive outcome of the dependent variable will tend to cluster around high values of the predicted probability, while the opposite will hold true for the negative outcome of the dependent variable);

-a list containing:

- $overall.model.significance: statistics related to the overall model p-value and to its distribution across the selected iterations
- $parameters.stability: statistics related to the stability of the estimated coefficients across the selected iterations
- $p.values.stability: statistics related to the stability of the estimated p-values across the selected iterations
- $AUCstatistics: statistics about the fitting and validation AUC distribution
- $Hosmer-Lemeshow statistics: statistics about the fitting and validation distribution of the HL test p-values

As for the abovementioned statistics:

-full: statistic estimated on the full dataset;

-median: median of the statistic across the selected iterations;

-QRNG: interquartile range across the selected iterations;

-QRNGoverMedian: ratio between the QRNG and the median, expressed as percentage;
-min: minimum of the statistic across the selected iterations;

-max: maximum of the statistic across the selected iterations;

-percent_smaller_0.05: (only for $overall.model.significance, $p.values.stability, and $Hosmer-Lemeshow statistics): proportion of times in which the p-values are smaller than 0.05; please notice that for the overall model significance and for the p-values stability it is desirable that the percentage is at least 95percent, whereas for the HL test p-values it is indeed desirable that the proportion is not larger than 5percent (in line with the interpetation of the test p-value which has to be NOT significant in order to hint at a good fit);

-significant (only for $p.values.stability): asterisk indicating that the p-values of the corresponding coefficient resulted smaller than 0.05 in at least 95percent of the iterations.

See Also

logregr , aucadj

Examples

```
# load the sample dataset
data(log_regr_data)

# fit a logistic regression model, storing the results into an object called 'model'
model <- glm(admit ~ gre + gpa + rank, data = log_regr_data, family = "binomial")

# run the function, using 100 iterations, and store the result in the 'res' object
res <- modelvalid(data=log_regr_data, fit=model, B=100)
```

---

mwPlot *R function for visually displaying Mann-Whitney test's results*

Description

The function allows to perform Mann-Whitney test, and to display the test's results in a plot along with two boxplots. For information about the test, and on what it is actually testing, see for instance the interesting article by R M Conroy, "What hypotheses do "nonparametric" two-group tests actually test?", in The Stata Journal 12 (2012): 1-9.

Usage

```
mwPlot(x, y, xlabl = "x", ylabl = "y", strip = FALSE,
  notch = FALSE, omm = FALSE, outl = TRUE, HL = FALSE)
```

Arguments

x	Object storing the values of the first group being compared.
y	Object storing either the values of the second group being compared or a grouping variable with 2 levels.
xlabl	If y is not a grouping variable, user may want to specify here the name of the x group that will show up in the returned boxplots (default is "x").
ylabl	If y is not a grouping variable, user may want to specify here the name of the y group that will show up in the returned boxplots (default is "y").
strip	Logical value which takes FALSE (by default) or TRUE if the user wants jittered points to represent individual values.
notch	Logical value which takes FALSE (by default) or TRUE if user does not or do want to have notched boxplots in the final display, respectively; it is worth noting that overlapping of notches indicates a not significant difference at about 95 percent confidence.

omm	It stands for overall mean and median; takes FALSE (by default) or TRUE if user wants the mean and median of the overall sample plotted in the chart (as a dashed RED line and dotted BLUE line respectively).
outl	Logical value which takes FALSE or TRUE (by default) if users want the boxplots to display outlying values.
HL	Logical value that takes TRUE or FALSE (default) if the user wants to display the distribution of the pairwise differences between the values of the two samples being compared; the median of that distribution is the Hodges-Lehmann estimator.

Details

The returned boxplots display the distribution of the values of the two samples, and jittered points represent the individual observations.

At the bottom of the chart, a subtitle arranged on three lines reports relevant statistics:
-test statistic (namely, U) and the associated z and p value;
-Probability of Superiority value (which can be interpreted as an effect-size measure, as discussed in: https://nickredfern.wordpress.com/2011/05/12/the-mann-whitney-u-test/);
-another measure of effect size, namely r (see https://stats.stackexchange.com/questions/124501/mann-whitney-u-test-confidence-interval-for-effect-size), whose thresholds are indicated in the last line of the plot's subtitle.

The function may also return a density plot (coupled with a rug plot at the bottom of the same chart) that displays the distribution of the pairwise differences between the values of the two samples being compared. The median of this distribution (which is represented by a blue reference line in the same chart) corresponds to the Hodges-Lehmann estimator.

Examples

```
#create a toy dataset
mydata <- data.frame(values=c(rnorm(30, 100,10),rnorm(30, 80,10)),
group = as.factor(gl(2, 30, labels = c("A", "B"))))

# performs the test, displays the test's result, including jittered points, notches,
#overall median and mean, and the Hodges-Lehmann estimator
mwPlot(x=mydata$values, y=mydata$group, strip=TRUE, omm=TRUE, notch=TRUE, HL=TRUE)
```

NNa *R function for Nearest Neighbor analysis of point patterns*

Description

The function allows to perform the Nearest Neighbor analysis of point patterns to formally test for the presence of a clustered, dispersed, or random spatial arrangement (second-order effect). It also allows to control for a first-order effect (i.e., influence of an underlaying numerical covariate) while performing the analysis. The covariate must be of RasterLayer class. Significance is assessed via a

randomized approach.

Usage

```
NNa(feature, studyplot = NULL, buffer = 0, B = 199, cov.var = NULL,
  addmap = TRUE)
```

Arguments

feature	Feature dataset (of point type; SpatialPointsDataFrame class).
studyplot	Shapefile (of polygon type; SpatialPolygonsDataFrame class) representing the study area; if not provided, the study area is internally worked out as the convex hull enclosing the input feature dataset.
buffer	Add a buffer to the studyplot (0 by default); the unit depends upon the units of the input data.
B	Number of randomizations to be used (199 by default).
cov.var	Numeric covariate (of 'RasterLayer' class).
addmap	TRUE (default) or FALSE if the user wants or does not want a map of the study area and of feature dataset to be also displayed.

Details

The function uses a randomized approach to test the significance of the Clark-Evans R statistic: the observed R value is set against the distribution of R values computed across B iterations (199 by default) in which a set of random points (with a sample size equal to the number of points of the input feature) is drawn and the statistic recomputed.

The function produces a histogram of the randomized R values, with a black dot indicating the observed value and a hollow dot representing the average of the randomized R values. P-values (computed following Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, p. 387), are reported at the bottom of the same chart. Two reference lines represent the two tails of the randomized distribution (left tail, indicating a significant clustered pattern; right tail, indicating a significant dispersed pattern).

Value

The function returns a list storing the following components

- $obs.aver.NN.dist: average of the observed NN distances
- $obs.R: observed R value
- $aver.rand.R: average of the randomized Rs
- $p.value clustered: p-value for a clustered pattern
- $p.value.dispersed: p-value for a dispersed pattern
- $p.value.diff.from.random: p-value for a pattern different from random

See Also

refNNa

Examples

```
data(springs)

#perform the analysis with B set to 99; the result points to a significant clustering
res <- NNa(springs, B=99)

data(Starbucks)
data(popdensity)

#perform the analysis, while controlling for the effect of the population density covariate
res <- NNa(Starbucks, cov.var=popdensity, B=99)
```

---

outlier *R function for univariate outliers detection*

Description

The function allows to perform univariate outliers detection using three different methods. These methods are those described in:
Wilcox R R, "Fundamentals of Modern Statistical Methods: Substantially Improving Power and Accuracy", Springer 2010 (2nd edition), pages 31-35.

Usage

```
outlier(x, method = "mean", addthres = TRUE)
```

Arguments

x	Vector storing the data.
method	Outliers identification method, either "mean" (default), "median", or "boxplot".
addthres	Takes FALSE or TRUE (default) if user does not want or does want some threshold lines be added to the returned chart.

Details

Two of the three methods are robust, and are therefore less prone to the masking effect.
(1) With the mean-based method, an observation is considered outlier if the absolute difference between that observation and the sample mean is more than 2 Standard Deviations away (in either direction) from the mean. In the plot returned by the function, the central reference line is indicating the mean value, while the other two are set at $mean - 2 * SDandmean + 2 * SD$.

(2) The median-based method considers an observation as being outlier if the absolute difference between the observation and the sample median is larger than the Median Absolute Deviation divided by 0.6745. In this case, the central reference line is set at the median, while the other two are set at $median - 2 * MAD/0.6745$ and $median + 2 * MAD/0.6745$.

(3) The boxplot-based method considers an observation as being an outlier if it is either smaller than the 1st Quartile minus 1.5 times the InterQuartile Range, or larger than the 3rd Quartile minus 1.5 times the InterQuartile Range. In the plot, the central reference line is set at the median, while the other two are set at $1Q - 1.5 * IQR$ and $3Q + 1.5 * IQR$.

Value

The function also returns a list containing information about the chosen method, the mid-point, lower and upper boundaries where non-outlying observations are expected to fall, total number of outlying observations, and a dataframe listing the observations and indicating which is considered outlier.
In the charts, the outlying observations are flagged with their ID number.

Examples

```
# create a toy dataset
mydata <- c(2,3,4,5,6,7,8,9,50,50)

# locate outlier(s) using the median-based method
outlier(mydata, method="median", addthres=TRUE)
```

`perm.t.test` *R function for permutation-based t-test*

Description

The function allows to perform a permutation-based t-test to compare two independent groups. The test's results are graphically displayed within the returned chart.

Usage

```
perm.t.test(data, format, sample1.lab = NULL, sample2.lab = NULL,
  B = 999)
```

Arguments

`data`	Dataframe containing the data.
`format`	It takes "long" if the data are arranged in two columns, with the left-hand one containing the values, and the righ-hand one containing a grouping variable; it takes "short" if the values of the two groups being compared are stored in two different adjacent columns.

sample1.lab	Label for the first sample being tested (default: smpl 1).
sample2.lab	Label for the first sample being tested (default: smpl 2).
B	Desired number of permutations (set at 999 by default).

Details

A permutation t-test proves useful when the assumption of 'regular' t-test are not met. In particular, when the two groups being compared show a very skewed distribution, and when the sample sizes are very unbalanced.

"The permutation test is useful even if we plan to use the two-sample t test. Rather than relying on Normal quantile plots of the two samples and the central limit theorem, we can directly check the Normality of the sampling distribution by looking at the permutation distribution. Permutation tests provide a "gold standard" for assessing two-sample t tests. If the two P-values differ considerably, it usually indicates that the conditions for the two-sample t don't hold for these data. Because permutation tests give accurate P-values even when the sampling distribution is skewed, they are often used when accuracy is very important." (Moore, McCabe, Craig, "Introduction to the Practice of Statistics", New York: W. H. Freeman and Company, 2009).

Value

The frequency histogram returned by the function displays the distribution of the permuted mean difference between the two samples; a solid dot indicates the observed mean difference, while an hollow dot represents the mean of the permuted differences. Two dashed blue lines indicates the 0.025 and 0.975 percentile of the permuted distribution. A rug plot at the bottom histgram indicates the individual permuted mean differences. At the bottom of the chart, some information are displayed. In particular, the observed mean difference and the permuted p-values are reported. In the last row, the result of the regular (parametric) t-test (both assuming and not assuming equal variances) is reported to allow users to compare the outcome of these different versions of the test.

Examples

```
#load the 'resample' package which stores a toy dataset
library(resample)

#load the 'Verizon' dataset
data("Verizon")

#performs the permutation-based t-test using 199 permutations
perm.t.test(Verizon, format="long", B=199)
```

phases — *Dataset: Posterior Probabilities for the chronological relation of the Starting and Ending boundaries of two Bayesian independent 14C phases*

Description

A dataset containing the posterior probability (as calculated by the OxCal program) for the relative chronological relations for the starting and ending boundaries of two Bayesian-defined 14C phases.

Usage

```
data(phases)
```

Format

A data frame with 4 rows and 5 variables

plotJenks — *R function for plotting univariate classification using Jenks' natural break method*

Description

The function allows to break a dataset down into a user-defined number of breaks and to nicely plot the results, adding a number of other relevant information. Implementing the Jenks' natural breaks method, it allows to find the best arrangement of values into different classes.

Usage

```
plotJenks(data, n = 3, brks.cex = 0.7, top.margin = 10, dist = 5)
```

Arguments

data	Vector storing the data.
n	Number of classes in which the dataset must be broken down (3 by default).
brks.cex	Adjusts the size of the labels used in the returned plot to display the classes' break-points.
top.margin	Adjusts the distance of the labels from the top margin of the returned chart.
dist	Adjusts the distance of the labels from the dot used to display the data points.

Details

The function produces a chart in which the values of the input variable are arranged on the x-axis in ascending order, while the index of the individual observations is reported on the y-axis. Vertical dotted red lines correspond to the optimal break-points which best divide the input variable into the selected classes. The break-points (and their values) are reported in the upper part of the chart, onto the corresponding break lines. Also, the chart's subtitle reports the Goodness of Fit value relative to the selected partition, and the partition which correspond to the maximum GoF value.

Value

The function returns a list containing the following components:

- $info: information about whether or not the method created non-unique breaks
- $classif: created classes and number of observations falling in each class
- $classif$brks: classes' break-points
- $breaks$max.GoF: number of classes at which the maximum GoF is achieved
- $class.data: dataframe storing the values and the class in which each value actually falls into

Examples

```
#create a toy dataset
mydata <- rnorm(100, 30, 10)

# performs the analysis, using 6 as number of desired classes,
# and store the results in the 'res' object
res <- plotJenks(mydata, n=6)
```

---

points *Dataset: location of fictional points*

Description

A SpatialPointsDataFrame representing fictional locations.

Usage

```
data(points)
```

Format

SpatialPointsDataFrame

pointsCovarCum *R function to plot the cumulative distribution (and acceptance interval) of the values of a spatial covariate measured at the locations of a point pattern*

Description

The function allows to test if there is a significant dependence of the input point pattern on a underlying spatial numeric covariate (first-order effect).
The function takes as input three datasets: a point patter ('SpatialPointsDataFrame' class), a covariate layer (of 'RasterLayer' class), and (optionally) a polygon feature ('SpatialPolygonsDataFrame' class) representing the study area and exactly matching the extent of the covariate layer. If the latter is not provided, it is internally worked out from the covariate raster and may make the whole function take a while to complete.

Usage

```
pointsCovarCum(feature, cov.var, studyplot = NULL, B = 200,
  cov.var.name = NULL, oneplot = TRUE)
```

Arguments

feature	Feature (of point type; 'SpatialPointsDataFrame' class) representing the spatial point pattern of interest.
cov.var	Numeric covariate (of 'RasterLayer' class).
studyplot	Feature (of polygon type; 'SpatialPolygonsDataFrame' class) representing the study area and exactly matching the extent of the covariate layer. If NULL, it is worked out from the covariate layer (may make the whole function take a while to complete).
B	Number of randomized iterations to be used to calculate the acceptance interval (200 by default).
cov.var.name	Name of the input covariate to be used in the cumulative distribution chart as label for the x axis (NULL by default).
oneplot	Set to TRUE (default), will plot the charts into a single visualization.

Details

The function plots the cumulative distribution of the values of the covariate at the locations of the input point pattern, and adds an acceptance interval (with significance level equal to 0.05; sensu Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 208) that allows to assess the statistical significance of the observed cumulative distribution. The interval is built by calculating the cumulative distribution of B realizations of a Complete Spatial Random process, and keeping the middle 95percent of those B distributions. B is set by default to 200, but can be increased by the user. The number of random points drawn during each of the B

simulations is equal to the number of features of the input point pattern.

For an example of the cumulative distribution plot plus acceptance interval, **see** for instance Carrero-Pazos, M. (2018). Density, intensity and clustering patterns in the spatial distribution of Galician megaliths (NW Iberian Peninsula). Archaeological and Anthropological Sciences. https://doi.org/10.1007/s12520-018-0662-2, figs. 4 and 5.

Value

The function returns a 2 plots, which can be arranged in just one visualization setting the parameter 'oneplot' to TRUE:

-a plot of the point pattern against the underlaying covariate;

-a plot of the cumulative distribution of the values of the covariate at the locations of the point patter along with the above-mentioned acceptance interval.

See Also

pointsCovarModel

Examples

```
#load the point dataset representing the location of Starbucks shops
data(Starbucks)

#load the polygon dataset representing the study area
data(Massachusetts)

#load the raster representing the population density, to be used as covariate
data(popdensity)

results <- pointsCovarCum(feature=Starbucks, cov.var=popdensity, studyplot=Massachusetts,
cov.var.name="population density")
```

---

pointsCovarDistr *R function to plot the frequency distribution of the average value of a spatial covariate measured at randomized locations*

Description

The function allows to test if there is a significant dependence of the input point pattern on a underlying numeric covariate (first-order effect).
The function takes as input three datasets: a point patter ('SpatialPointsDataFrame' class), a covariate layer (of 'RasterLayer' class), and (optionally) a polygon feature ('SpatialPolygonsDataFrame'

class) representing the study area and exactly matching the extent of the covariate layer. If the latter is not provided, it is internally worked out from the covariate raster and may make the whole function take a while to complete.

Usage

```
pointsCovarDistr(feature, cov.var, studyplot = NULL, B = 199,
  oneplot = TRUE)
```

Arguments

feature	Feature (of point type; 'SpatialPointsDataFrame' class) representing the spatial point pattern of interest.
cov.var	Numeric covariate (of 'RasterLayer' class).
studyplot	Feature (of polygon type; 'SpatialPolygonsDataFrame' class) representing the study area and exactly matching the extent of the covariate layer. If NULL, it is worked out from the covariate layer (may make the whole function take a while to complete).
B	Number of randomized iterations to be used to calculate the acceptance interval (199 by default).
oneplot	Set to TRUE (default), will plot the charts into a single visualization.

Details

The function plots a frequency distribution histogram of the average value of a spatial covariate at randomized locations (using B iterations). At each iteration, the number of randomized points is equal to the number of points of the input point pattern. Two blue reference lines correspond to the 0.025th and to the 0.975th quantile of the randomized distribution. A black dot represents the observed mean value of the covariate measured at the locations of the input point pattern. P-values are reported.

Value

The function returns a list storing the following components:

- $obs.cov.values: observed values of the covariate at the point pattern locations
- $obs.average: average of the observed values of the covariate
- $p.value.obs.smaller.than.exp: p.value for the observed average smaller than expected under the Null Hypothesis
- $p.value.obs.larger.than.exp: p.value for the observed average larger than expected under the Null Hypothesis
- $p.value.obs.diff.from.exp: p.value for the observed average different from what expected under the Null Hypothesis

See Also

pointsCovarModel, pointsCovarCum

Examples

```
#load the point dataset representing the location of springs
data(springs)

#load the polygon dataset representing the study area
data(malta_polyg)

#load the raster representing the terrain elevation, to be used as covariate
data(malta_dtm_40)

pointsCovarDistr(feature=springs, cov.var=malta_dtm_40, studyplot=malta_polyg)
```

---

pointsCovarModel *R function to model (and test) the dependence of a point pattern on a spatial numeric covariate*

Description

The function is a wrapper for a number of functions out of the extremely useful 'spatstat' package (specifically, ppm(), cdf.test(), auc(), roc(), effectfun()). It allows to test if there is a significant dependence of the input point pattern on a underlying spatial numeric covariate (first-order effect). The function takes as input three datasets: a point patter ('SpatialPointsDataFrame' class), a covariate layer (of 'RasterLayer' class), and a polygon feature ('SpatialPolygonsDataFrame' class) representing the study area and exactly matching the extent of the covariate layer. If the latter is not provided, it is internally worked out from the covariate raster and may make the whole function take a while to complete.

Usage

```
pointsCovarModel(feature, cov.var, studyplot = NULL, oneplot = FALSE)
```

Arguments

feature	Feature (of point type; SpatialPointsDataFrame class) representing the spatial point pattern of interest.
cov.var	Numeric covariate (of RasterLayer class).
studyplot	Feature (of polygon type; SpatialPolygonsDataFrame) representing the study area and exactly matching the extent of the covariate layer. If NULL, it is worked out from the covariate layer (may make the whole function take a while to complete).
oneplot	Set to TRUE (default), will plot the charts into a single visualization.

Details

The function fits a inhomogeneous Poisson point process (Alternative Model-H1) with intensity of the point pattern as a loglinear function of the underlaying numerical covariate (see Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 307-309). Also, the function fits a homogeneous Poisson point model (Null Model-H0, equivalent to Complete Spatial Randomness: Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 305-306), that is used as comparison for the inhomogeneous point process model in a Likelihood Ratio test (Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 334-335). A significant result, i.e. a low p-value, suggests rejecting the Null Hypothesis of CSR in favour of the Alternative Hypothesis of a Poisson point process affected by a covariate effect (i.e., inhomogeneous intensity due to the influence of the covariate) (Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 305).

Value

The function returns a 4 plots, which can be arranged in just one visualization setting the parameter 'oneplot' to TRUE:

-plot of the point pattern along with the underlaying covariate raster;

-plot of the fitted intensity against the spatial covariate (Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 308);

-plot of the cumulative distribution of the covariate at the data points against the cumulative distribution of the covariate at all the spatial location within the study area (rationale: Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 184-185);

-plot of the ROC curve, which help assessing the strength of the dependence on the covariate (Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 187-188).

-a list containing:

- $H0-model: info and relevant statistics regarding the Null Model
- $H1-model: info and relevant statistics regarding the Alternative Model
- $Model comparison (LRT): results of the Likelihood Ratio test
- $AIC-H0: AIC of the Null Model
- $AIC-H1: AIC of the Alternative Model
- $KS test: information regarding the cumulative distribution comparison via Kolmogorov-Smirnov test
- $AUC: AUC statistic

See Also

pointsCovarCum , distCovarModel , distRandSign

Examples

```
#load the point dataset representing the location of Starbucks shops
data(Starbucks)

#load the polygon dataset representing the study area
data(Massachusetts)

#load the raster representing the population density, to be used as covariate
data(popdensity)

#note: a warning message reporting that 4 out of 699 points
# have values of the covariate undefined is expected
results <- pointsCovarModel(Starbucks, popdensity, Massachusetts)
```

`pointsInPolygons` *R function to test points-in-polygons relationship*

Description

The function allows to test:
-scenario a:
if there is a significant spatial association between a set of points and a set of polygons, in terms of points falling within the polygons. In other words, it aims at testing whether a set of points falls inside a set of polygons more often than would be expected by chance. The basic assumption is that the polygons are completely contained within the study plot. If the shapefile (of polygon type) representing the study plot is not provided, the calculations use the bounding polygon based on the union the convex hulls of the point and of the polygon feature.
-scenario b:
if the distribution of points within a set of polygons totally covering the study area can be considered random, or if the observed points count for each polygon is larger or smaller than expected. P values are also reported.

Usage

```
pointsInPolygons(point.feat, polyg.feat, studyplot = NULL, scenario,
  buffer = 0, cex.text = 0.7)
```

Arguments

`point.feat`	Feature (of point type; SpatialPointsDataFrame class) whose spatial association with the polygons has to be assessed.
`polyg.feat`	Feature (polygon type; SpatialPolygonsDataFrame) in relation to which the spatial association of the points has to be assessed.

studyplot	Shapefile (of polygon type; SpatialPolygonsDataFrame) representing the study area; if not provided, the study area is internally worked out as the bounding polygon based on the union the convex hulls of the point and of the polygon feature.
scenario	Select one of the two types of analysis available ("a" or "b").
buffer	Add a buffer to the convex hull of the study area (0 by default); the unit depends upon the units of the input data.
cex.text	Modify the size of the labels in the plot produced by the 'scenario b' option.

Details

The computations relative to scenario "a" are based on the 'dbinom() and 'pbinom()' functions. The probability of observed count within polygons is $dbinom(x, size = n.of.points, prob = p)$, where 'x' is the observed number of points within polygons, 'n.of.points' is the total number of points, and 'p' is the probability that a single point will be found within a polygon, which is equal to the ratio between the area of the polygons and the total area of the study plot. The probability that x or fewer points will be found within the polygons is $pbinom(x, size = n.of.points, prob = p)$.

The calculations relative to the scenario "b" are again based on the binomial distribution: the probability of the observed counts is $dbinom(x, size = n.of.points, prob = p)$, where 'x' is the observed number of points within a given polygon, 'n.of.points' is the total number of points, and 'p' is equal to the size of each polygon relative to sum of the polygons' area. The probability that x or fewer points will be found within a given polygon is $pbinom(x, size = n.of.points, prob = p)$.

Value

For scenario "a" the function produces a plot of the points and polygons (plus the study area), and relevant information are reported at the bottom of the chart itself.

A list is also returned, containing what follows:
-$Polygons' area;
-$Study area's area;
-$Total # of points;
-$Observed # of points in polygons;
-$Expected # of points in polygons;
-$Exact probability of observed count within polygons;
-$Probability of <= observed count within polygons;
-$Probability of >= observed count within polygons.

For scenario "b" the function returns a plot showing the polygons plus the dots; in each polygon the observed and expected counts are reported, and the p-value of the observed count is indicated.

A matrix is also returned, containing what follows:
-polygons' area;
-percentage area (size of each polygon relative to sum of the polygons' area; it corresponds to the probability (p) fed into the binomial distribution function);
-observed number of points;
-expected number of points;

-probability of observed counts;
-probability of observed counts <= than expected;
-probability of observed counts >= than expected.

See Also

pointsToPointsTess , dbinom , pbinom

Examples

```
data(points)
data(polygons)

result <- pointsInPolygons(points, polygons, scenario="a")

data(events)
data(thiessenpolyg)

result <- pointsInPolygons(events, thiessenpolyg, scenario="b")
```

---

pointsToPointsTess *R function to test the relationship of a set of points with the Thiessen tessellation built around points belonging to another feature dataset*

Description

The function can be considered as a special case of the scenario "b" tested by the 'pointsInPolygons()' function provided by this same package, with the exception that in this case the polygons are not entered by the user but are internally created by the function around the to-feature.

Usage

```
pointsToPointsTess(from.feat, to.feat, cex.text = 0.7)
```

Arguments

from.feat	Feature (of point type; SpatialPointsDataFrame) whose spatial association with to-feature has to be assessed.
to.feat	Feature (of point type; SpatialPointsDataFrame) in relation to which the spatial association of the from-feature has to be assessed.
cex.text	Modify the size of the labels in the returned plot.

Details

The question this function may allow to address is: do the points belonging to a feature dataset tend to occur close to any of the points in another feature dataset than expected if the points would be randomly scattered across the study area? To help addressing this question, the function creates Thiessen polygons around the input 'to.feature' and then runs the 'pointsInPolygons()' function using its 'scenario b'.
For further details, see the help documentation of the 'pointsInPolygons()' function.

See Also

pointsInPolygons

Examples

```
data(locations)
data(events)

result <- pointsToPointsTess(events, locations)

data(deaths)
data(pumps)

result <- pointsToPointsTess(deaths, pumps)
```

---

polygons *Dataset: location of fictional polygons*

Description

A SpatialPolygonsDataFrame representing fictional polygons.

Usage

```
data(polygons)
```

Format

SpatialPolygonsDataFrame

popdensity *Dataset: Massachusetts population density*

Description

A RasterLayer representing the population density in Massachusetts.
After https://mgimond.github.io/Spatial/point-pattern-analysis-in-r.html.

Usage

```
data(popdensity)
```

Format

RasterLayer

ppdPlot *R function for plotting Posterior Probability Densities for Bayesian modeled 14C dates/parameters*

Description

The function allows plot Posterior Probability Densities with a nice outlook thanks to 'ggplot2'.
It takes as input a dataframe that must be organized as follows (it is rather easy to do that once the data have been exported from OxCal):
-calendar dates (first column to the left);
-posterior probabilities (second column);
-grouping variables (third column), which could contain the names of the events of interest (e.g., phase 1 start, phase 1 end, phase 2 start, phase 2 end, etc).

Usage

```
ppdPlot(data, lower = min(data[, 1]), upper = max(data[, 1]), type)
```

Arguments

data	Dataframe containing the data as returned by the OxCal program.
lower	Lower limit of the calendar date axis.
upper	Upper limit of the calendar date axis; if the lower and upper parameters are not provided, the default values will be the earliest and latest calendar dates.
type	Type of plot the user wishes to plot (a: curves outlined by a line; b: curves plotted as solid areas; c: combination of a and b).

Examples

```
#load a toy dataset
data(radioc_data)

#plot the Posterior Probability Densities for the phases' parameters
ppdPlot(radioc_data, type="a")

#plot the Posterior Probability Densities for the phases' parameters,
# setting different boundaries for the x-axis and using filled curves instead of simple outlines
ppdPlot(radioc_data, -1000, 100, type="b")
```

`prob.phases.relat` *R function to calculate the Posterior Probability for different chronological relations between two Bayesian radiocarbon phases*

Description

The function allows to calculate the posterior probability for different chronological relations between two phases defined via Bayesian radiocarbon modeling. For the results to make sense, the phases have to be defined as independent if one wishes to assess what is the posterior probability for different relative chronological relations between them.

Usage

```
prob.phases.relat(data = NULL, sAoldersB = NULL, sAoldereB = NULL,
  eAoldersB = NULL, eAoldereB = NULL, sBoldersA = NULL,
  sBoldereA = NULL, eBoldersA = NULL, eBoldereA = NULL,
  sort = FALSE)
```

Arguments

`data`	Matrix containing the posterior probability of the chronological relation between the Starting and Ending boundaries of two independent phases, as returned by the OxCal's 'Order' query (see Details).
`sAoldersB`	Probability of startA being older than startB.
`sAoldereB`	Probability of startA being older than endB.
`eAoldersB`	Probability of endA being older than startB.
`eAoldereB`	Probability of endA being older than endB.
`sBoldersA`	Probability of startB being older than startA.
`sBoldereA`	Probability of startB being older than endA.
`eBoldersA`	Probability of endB being older than startA.
`eBoldereA`	Probability of endB being older than endA.
`sort`	logical which takes TRUE or FALSE (default) if the user does or does not want the returned posterior probabilities sorted in descending order.

Details

The rationale for this approach is made clear in an article by Buck et al 1992 (https://doi.org/10.1016/0305-4403(92)90025-X), and it runs as follows: "if we do not make any assumption about the relationship between the phases, can we test how likely they are to be in any given order"?

Data can be fed into the function in two ways:

-the function takes as input the table provided by the 'OxCal' program as result of the 'Order' query.
Once the table as been saved from 'OxCal' in .csv format, you have to feed it in R. A .csv file can be imported into R using (for instance): $mydata < -read.table(file.choose(), header = TRUE, sep = ",", dec = ".", as.is = T)$;

be sure to insert the phases' parameters (i.e., the starting and ending boundaries of the two phases) in the OxCal's Order query in the following order: StartA, EndA, StartB, EndB; that is, first the start and end of your first phase, then the start and end of the second one; you can give any name to your phases, as long as the order is like the one described.

-alternatively, 8 relevant parameters (which can be read off from the Oxcal's Order query output) can be manually fed into the function (see the Arguments section above).

Given two phases A and B, the function allows to calculate the posterior probability for:

-A being within B
-B being within A
-A starting earlier but overlapping with B
-B starting earlier but overlapping with A
-A being entirely before B
-B being entirely before A
-sA being within B
-eA being within B
-sB being within A
-eB being within A
where 's' and 'e' refer to the starting and ending boundaries of a phase.

The function will return a table and a dot plot.

Thanks are due to Dr. Andrew Millard (Durham University) for the help provided in working out the operations on probabilities.

Examples

```
#load a toy dataset
data(phases)

 #calculate the Posterior Probability for the chronological relation between two phases,
```

```
 # stores the results in the 'res' object, and produce a dot chart.
res <- prob.phases.relat(phases)

# same as above, but manually feeding relevant parameters
res <- prob.phases.relat(data=NULL, sAoldersB=0.613, sAoldereB=1, eAoldersB=0.0010,
eAoldereB=0.666, sBoldersA= 0.386, sBoldereA=0.999, eBoldersA=0.000039, eBoldereA=0.3334)
```

`pumps` *Dataset: location of public water pumps in London (after Dr Snow's mid-1800s study of cholera outbreak in Soho).*

Description

A SpatialPointsDataFrame representing SpatialPointsDataFrame representing the location of public water pumps in London (after Dr Snow's mid-1800s study of cholera outbreak in Soho).

Usage

```
data(pumps)
```

Format

SpatialPointsDataFrame

`radioc_data` *Dataset: Posterior Probabilities for the Starting and Ending boundaries of two 14C phases*

Description

A dataset containing the calendar dates and the associated posterior probability for the starting and ending boundaries of two 14C phases.

Usage

```
data(radioc_data)
```

Format

A data frame with 680 rows and 3 variables

Details

- dates. Calendar dates
- prob. Posterior probability
- group. Grouping variable containing strings referring to the start and ending boundaries of the two 14C phases

refNNa *R function for refined Nearest Neighbor analysis of point patterns (G function)*

Description

The function allows to perform the refined Nearest Neighbor analysis of point patterns.

Usage

```
refNNa(feature, studyplot = NULL, buffer = 0, B = 200,
  cov.var = NULL, order = 1)
```

Arguments

feature	Feature dataset (of point type).
studyplot	Shapefile (of polygon type) representing the study area; if not provided, the study area is internally worked out as the convex hull enclosing the input feature dataset.
buffer	Add a buffer to the studyplot (0 by default); the unit depends upon the units of the input data.
B	Number of randomizations to be used (200 by default).
cov.var	Numeric covariate (of RasterLayer class) (NULL by default).
order	Integer indicating the kth nearest neighbour (1 by default).

Details

The function plots the cumulative Nearest Neighbour distance, along with an acceptance interval (with significance level equal to 0.05; sensu Baddeley et al., "Spatial Point Patterns. Methodology and Applications with R", CRC Press 2016, 208) based on B (set to 200 by default) realizations of a Complete Spatial Random process. The function also allows to control for a first-order effect (i.e., influence of an underlaying numerical covariate) while performing the analysis. The covariate must be of 'RasterLayer class'.

The function uses a randomized approach to build the mentioned acceptance interval whereby cumulative distributions of average NN distances of random points are computed across B iterations. In each iteration, a set of random points (with sample size equal to the number of points of the input feature) is drawn.

Thanks are due to Dason Kurkiewicz for the help provided in writing the code to calculate the acceptance interval.

See Also

NNa

Examples

```
data(springs)

#produces a plot representing the cumulative nearest neighbour distance distribution;
#an acceptance interval based on 99 randomized simulations is also shown.
refNNa(springs, B=99)

#load the Startbucks datset
data(Starbucks)

#load the raster representing the numerical covariate
data(popdensity)

#perform the analysis, controlling for the 1st order effect
refNNa(Starbucks, cov.var=popdensity, B=99)
```

`resc.val` *R function to rescale the values of a dataset between a minimum and a maximum set by the user*

Description

The function allows to rescale the values of a dataset between a minimum and a maximum that are specified by the user. In doing that, it allows to preserve the shape of the distribution of the original data.

Usage

```
resc.val(x, min.v = 0, max.v = 100)
```

Arguments

`x`	Vector containing the values to be rescaled.
`min.v`	Minimum value of the rescaled dataset (0 by default).
`max.v`	Maximum value of the rescaled dataset (100 by default).

Details

The function produces two density charts representing the distribution of the original and of the rescaled dataset. It also returns a dataframe storing the original and rescaled values.

Examples

```
#generate a random dataset of size 30, normally distributed with mean 1000 and
#standard deviation 10
dataset <- rnorm(30, 1000,10)

#rescale the dataset to be constrained between 10 and 100
resc.val(dataset, min.v=10, max.v=100)
```

---

rndpoints *Dataset: random points*

Description

A SpatialPointsDataFrame representing random locations.

Usage

```
data(rndpoints)
```

Format

SpatialPointsDataFrame

---

robustBAplot *R function to plot a robust version of the Bland-Altman plot*

Description

The function allows to plot a robust version of the Bland-Altman plot.

Usage

```
robustBAplot(a, b, z = 1.96, methAlab = "Method A",
  methBlab = "Method B", cex = 0.6)
```

Arguments

a	Vector storing the first set of measurements to be compared.
b	Vector storing the second set of measurements to be compared.
z	Value for the confidence interval for the median difference; set by default to 1.96 (corresponding to 95 percent CI).
methAlab	Label to be used in the returned chart to refer to the method yielding the first set of measurements.
methBlab	Label to be used in the returned chart to refer to the method yielding the second set of measurements.
cex	Size of the data points.

Details

The function returns a chart based on robust (i.e. resistant to outlying values) measures of central tendency and variability: median and Median Absolute Deviation (MAD) (Wilcox R R. 2001. "Fundamentals of modern statistical methods: Substantially improving power and accuracy". New York: Springer) instead of mean and standard deviation.

The x-axis displays the median of the two variables being compared, while the y-axis represents their difference. A solid horizontal line represents the bias, i.e. the median of the differences reported on the y-axis. Two dashed horizontal lines represent the region in which 95percent of the observations are expected to lie; they are set at the median plus or minus z*(MAD/0.6745).

Examples

```
#create a first toy vector
a <- rnorm(30,10,1)

#create a second toy vector
b <- a*runif(30,1,1.5)

robustBAplot(a,b)
```

---

springs *Dataset: location of springs in Malta*

Description

A SpatialPointsDataFrame representing the location of springs in Malta.

Usage

```
data(springs)
```

Format

SpatialPointsDataFrame

Starbucks — *Dataset: location of Starbucks in Massachusetts*

Description

A SpatialPointsDataFrame representing the location of Starbucks in Massachusetts.
After https://mgimond.github.io/Spatial/point-pattern-analysis-in-r.html.

Usage

```
data(Starbucks)
```

Format

SpatialPointsDataFrame

thiessenpolyg — *Dataset: Thiessen polygons around the points represented in the 'locations' dataset*

Description

A SpatialPolygonsDataFrame representing Thiessen polygons around the points represented in the 'locations' dataset.

Usage

```
data(thiessenpolyg)
```

Format

SpatialPolygonsDataFrame

vislim *R function for computing the limit of visibility of an object given its height*

Description

The function allows to plot the angular size of an object (in degrees) against the distance from the observer, and to compute at which distance from the observer the angular size of the object hits the limit of human visual acuity (0.01667 degrees).

Usage

```
vislim(vis.degree = 0.01667, targ.h)
```

Arguments

vis.degree	Limit of human visual acuity (0.01667 by default).
targ.h	Target size (=height in meters).

Details

The function returns:
-a plot displaying the decay in angular size as function of the object's distance from the observer; a black dot represents the distance at which the angular size hits the limit of human visual acuity;
-the value (in km) of the visibility limit.

Examples

```
# calculate the visibility limit of an object of size 6m, and store the result (20.62 km)
#in the 'limit' object
limit <- vislim(targ.h=6)
```

Index

∗Topic **Aindex**
Aindex, 3
∗Topic **NNa**
NNa, 32
∗Topic **aucadj**
aucadj, 5
∗Topic **chiperm**
chiperm, 9
∗Topic **covariate**
pointsCovarModel, 42
∗Topic **datasets**
assemblage, 4
deaths, 10
events, 19
faults, 19
locations, 24
log_regr_data, 27
malta_dtm_40, 27
malta_polyg, 28
Massachusetts, 28
phases, 37
points, 38
polygons, 47
popdensity, 48
pumps, 51
radioc_data, 51
rndpoints, 54
springs, 55
Starbucks, 56
thiessenpolyg, 56
∗Topic **distCovarModel**
distCovarModel, 10
∗Topic **distDiffTest**
distDiffTest, 13
∗Topic **distRandCum**
distRandCum, 14
∗Topic **distRandSign**
distRandSign, 16
∗Topic **featClust**
featClust, 20
∗Topic **impRst**
impRst, 22
∗Topic **impShp**
impShp, 23
∗Topic **kwPlot**
kwPlot, 23
∗Topic **logregr**
logregr, 25
∗Topic **modelvalid**
modelvalid, 28
∗Topic **mwPlot**
mwPlot, 31
∗Topic **outlier**
outlier, 34
∗Topic **perm.t.test**
perm.t.test, 35
∗Topic **plotJenks**
plotJenks, 37
∗Topic **pointsCovarCum**
pointsCovarCum, 39
∗Topic **pointsCovarDistr**
pointsCovarDistr, 40
∗Topic **pointsInPolygons**
pointsInPolygons, 44
∗Topic **pointsToPointsTess**
pointsToPointsTess, 46
∗Topic **ppdPlot**
ppdPlot, 48
∗Topic **prob.phases.relat**
prob.phases.relat, 49
∗Topic **refNNA**
refNNa, 52
∗Topic **resc.val**
resc.val, 53
∗Topic **robustBAplot**
robustBAplot, 54
∗Topic **similarity**
BRsim, 6

∗Topic **vislim**
vislim, 57

Aindex, 3, *12*, *18*
assemblage, 4
auc, *12*
aucadj, 5, *27*, *31*

BRsim, 6

cdf.test, *12*
chiperm, 9
corrplot, *8*

dbinom, *46*
deaths, 10
distCovarModel, *4*, 10, *18*, *43*
distDiffTest, 13
distRandCum, 14, *18*
distRandSign, *4*, *12*, *15*, 16, *43*

effectfun, *12*
events, 19

faults, 19
featClust, 20

impRst, 22
impShp, 23

kwPlot, 23

locations, 24
log_regr_data, 27
logregr, *5*, 25, *31*

malta_dtm_40, 27
malta_polyg, 28
Massachusetts, 28
modelvalid, *5*, *27*, 28
mwPlot, 31

NNa, 32, *52*

outlier, 34

p.adjust, *24*
pbinom, *46*
perm.t.test, *14*, 35
phases, 37
plotJenks, 37

points, 38
pointsCovarCum, 39, *42, 43*
pointsCovarDistr, 40
pointsCovarModel, *4*, *12*, *40*, *42*, 42
pointsInPolygons, 44, *47*
pointsToPointsTess, *46*, 46
polygons, 47
popdensity, 48
ppdPlot, 48
ppm, *12*
prob.phases.relat, 49
pumps, 51

radioc_data, 51
refNNa, *34*, 52
resc.val, 53
rndpoints, 54
robustBAplot, 54
roc, *12*

silhouette, *8*
springs, 55
Starbucks, 56

thiessenpolyg, 56

vislim, 57