Package 'DCluster'

September 3, 2023

Version 0.2-9

Date 2023-09-01

Title Functions for the Detection of Spatial Clusters of Diseases

Encoding UTF-8

Author Virgilio Gómez-Rubio, Juan Ferrándiz-Ferragud and Antonio López-Quílez, with contributions by Roger Bivand.

Maintainer Virgilio Gómez-Rubio <Virgilio.Gomez@uclm.es>

Depends R (>= 1.6.2), boot, spdep, MASS

Description A set of functions for the detection of spatial clusters of disease using count data. Bootstrap is used to estimate sampling distributions of statistics.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-09-03 09:50:02 UTC

R topics documented:

`achisq` *Another Implementation of Pearson's Chi-square Statistic*

Description

Another implementation of Pearson's Chi-square has been written to fit the needs in package *DCLuster*.

achisq.stat is the function that calculates the value of the statistic for the data.

achisq.boot is used when performing a non-parametric bootstrap.

achisq.pboot is used when performing a parametric bootstrap.

Details

This statistic can be used to detect whether observed data depart (over or above) expected number of cases significantly. The test considered stands for relative risks among areas to be equal to an (unknown) constant λ, while the alternative hypotheses is that not all relative risks are equal.

The actual value of the statistic depends on null hypotheses. If we consider that all the relative risks are equal to 1, the value is

T=

$$\sum_i \frac{(O_i - E_i)^2}{E_i}$$

and the degrees of freedom are equal to the number of regions.

On the other hand, if we just consider relative risks to be equal, without specifying their value (i.e., λ is unknown), E_i must be substituted by $E_i \frac{O_+}{E_+}$ and the number of degrees of freedom is the number of regions minus one.

When internal standardization is used, null hypotheses must be all relative risks equal to 1 and the number of degrees of freedom is the number of regions minus one. This is due to the fact that, in this case, $O_+ = E_+$.

References

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: I. The Binomial and Multinomial Distributions. Biometrika 53, 167-182.

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: The Poisson Distribution. Biometrika 53, 183-190.

See Also

DCluster, achiq.stat, achisq.boot, achisq.pboot

`achisq.boot` *Bootstrap Replicates of Pearson's Chi-square Statistic*

Description

Generate bootstrap replicates of the Pearson's Chi-square statistic (function *achisq.stat*), by means of function *boot* from *boot* library. Notice that these functions should not be used separately but as argument *statistic* when calling function *boot*.

achisq.boot is used when performing a non-parametric bootstrap.

achisq.pboot is used when performing a parametric bootstrap.

Usage

```
achisq.boot(data, i, ...)
achisq.pboot(...)
```

Arguments

`data`	A dataframe containing the data, as specified in *DCluster* manpage.
`i`	Permutation generated by the non-parametric bootstrap procedure.
`...`	Additional arguments passed when performing a bootstrap.

Value

Both functions return the value of the statistic.

References

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: I. The Binomial and Multinomial Distributions. Biometrika 53, 167-182.

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: The Poisson Distribution. Biometrika 53, 183-190.

See Also

DCluster, boot, achisq, achisq.stat

Examples

```
library(boot)
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))

niter<-100

#Permutation  model
chq.perboot<-boot(sids, statistic=achisq.boot, R=niter)
plot(chq.perboot)#Display results

#Multinomial model
chq.mboot<-boot(sids, statistic=achisq.pboot, sim="parametric", ran.gen=multinom.sim,  R=niter)
plot(chq.mboot)#Display results

#Poisson model
chq.pboot<-boot(sids, statistic=achisq.pboot, sim="parametric", ran.gen=poisson.sim,  R=niter)
plot(chq.pboot)#Display results

#Poisson-Gamma model
chq.pgboot<-boot(sids, statistic=achisq.pboot, sim="parametric", ran.gen=negbin.sim, R=niter)
plot(chq.pgboot)#Display results
```

---

achisq.stat *Another Implementation of Pearson's Chi-square Statistic*

Description

Compute Pearson's Chi-square statistic. See *achisq* manual page for more details.

achisq.stat computes the test statistic and the test using a hi-square distribution whilst achisq.test performs a bootstrap test.

Usage

```
achisq.stat(data, lambda=NULL)
achisq.test(formula, data, model, R, ...)
```

Arguments

formula	Formula that specifies the underlying model. The observed cases are the response and the expected number of cases must be specified as an offset in the log scale (see example below). Note that now it is not necessary to use Observed and Expected and that any other names can be used to specify the observed and expected cases.
model	Parametric model to be used in the bootstrap test. One of "param", "multinom", "poisson" or "negbin". See the **DCluster** manpage for details.
...	The remaining arguments in 'achisq.stat' not included in 'achisq.test'. This is done so because achisq.test calls achisq.stat in order to perform the test.
R	Number of replicates used in the test to compute the significance of the observed value of the test statistic.
data	A dataframe containing the data, as specified in the **DCluster** manpage.
lambda	The value of the relative risks under the null hypotheses. If its NULL, the second hypotheses commented above is considered and the expected number of cases will automatically be corrected.

Value

A list with three components

T	The value of the statistic.
df	Degrees of freedom of the asinthotic Chi-square distribution.
pvalue	Related pvalue.

References

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: I. The Binomial and Multinomial Distributions. Biometrika 53, 167-182.

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: The Poisson Distribution. Biometrika 53, 183-190.

See Also

DCluster, achisq, achisq.boot, achisq.pboot

Examples

```
library(spdep)

data(nc.sids)
```

```
sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))

#Compute the statistic under the assumption that lambda = 1.
achisq.stat(sids, lambda=1)

#Perform test
achisq.test(Observed~offset(log(Expected)), sids, model="poisson", R=99)
```

---

besagnewell *Besag and Newell's Statistic for Spatial Clustering*

Description

Besag & Newell's statistic looks for clusters of size k, i. e., where the number of observed cases is k. At every area where a case has appeared, the number of neighbouring regions needed to reach k cases is calculated. If this number is too small, that is, too many observed cases in just a few regions with low expected cases, then it is marked as a cluster.

References

Besag, J. and Newell, J.(1991). The detection of clusters in rare diseases. Journal of the Royal Statistical Society A 154, 143-155.

See Also

DCluster, besagnewell.stat, besagnewell.boot, besagnewell.pboot, bn.iscluster

Examples

```
#B&N must use the centroids as grid.
#The size of teh cluster is 20.
#100  bootstrap simulations are performed
#Poisson is the model used in the bootstrap simulations to generate the
#observations.
#Signifiance level is 0'05, even though multiple tests are made.

library(boot)
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

bnresults<-opgam(sids, thegrid=sids[,c("x","y")], alpha=.05,
iscluster=bn.iscluster, set.idxorder=TRUE, k=20, model="poisson",
R=100, mle=calculate.mle(sids) )
```

```
#Plot all the centroids
plot(sids$x, sids$y)

#Plot signifiant centroids in red
points(bnresults$x, bnresults$y, col="red", pch=19)
```

`besagnewell.boot` *Generate Boostrap Replicates of Besag and Newell's Statistic*

Description

Generate boostrap replicates of Besag and Newell's statistic, by means of function *boot* from *boot* library. Notice that these functions should not be used separately but as argument *statistic* when calling function *boot*.

besagnewell.boot is used when performing a non-parametric bootstrap.

When sampling models are *Multinomial* or *Poisson* it is quite straightforwad to obtain the actual p-value as shown in the examples. When *Permutation* or *Negative Binomial* are used, simulation must be used to estimate significance.

Usage

```
besagnewell.boot(data, i, ...)
besagnewell.pboot(...)
```

Arguments

`data`	A dataframe with the data, as explained in *DCluster*.
`i`	Permutation generated by the non-parametric bootstrap.
`...`	Additional arguments needed.

Value

Both functions return the value of the statistic.

References

Besag, J. and Newell, J.(1991). The detection of clusters in rare diseases. Journal of the Royal Statistical Society A 154, 143-155.

See Also

DCluster, boot, besagnewell, besagnewell.stat, bn.iscluster

Examples

```
library(boot)
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

niter<-100

#Permutation  model
besn.perboot<-boot(sids, statistic=besagnewell.boot, R=niter, k=20)
plot(besn.perboot)#Display results
```

---

besagnewell.stat *Besag and Newell's Statistic for Spatial Clustering*

Description

besagnewell.stat computes the statistic around a single location. Data passed must be sorted according to distance to central region, which is supposed to be the first row in the dataframe. Notice that the size of the cluster is *k+1*.

Usage

```
besagnewell.stat(data, k)
```

Arguments

data	A dataframe with the data, as explained in *DCluster*.
k	Cluster size.

Value

A vector of two elements: the value of the statistic and the size of the cluster (which is equal to the value of the statistic).

References

Besag, J. and Newell, J.(1991). The detection of clusters in rare diseases. Journal of the Royal Statistical Society A 154, 143-155.

See Also

DCluster, besagnewell, besagnewell.boot, besagnewell.pboot

Examples

```
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

besagnewell.stat(sids, k=20)
```

---

bn.iscluster *Clustering Function for Besag and Newell's Method*

Description

This function is used to calculate the significance of the agregation of cases around the current area when scanning the whole area by means of function *opgam*.

When data sampling distribution is *multinomial* or *poisson* the exact p-value is computed. In the other cases (i.e., permutation and negative binomial) it is aproximated by bootstrap.

This function must be passed to function *opgam* as argument *iscluster*.

Usage

```
bn.iscluster(data, idx, idxorder, alpha, k, model="poisson", R=999, mle)
```

Arguments

`data`	A dataframe with the data as explained in *DCluster*.
`idx`	A boolean vector to know the areas in the current circle.
`idxorder`	A permutation of the rows of data to order the regions according to their distance to the current centre.
`alpha`	Test significance.
`k`	Size of the cluster.
`model`	Thge model used to generate random observations. It can be 'permutation', 'multinomial', 'poisson' or 'negbin'.
`R`	Number of bootstrap replicates made to compute pvalue if the local test.
`mle`	Parameters needed to compute the Negative Binomial distribution (if used). See *negbin.sim* manual page for details.

Value

A vector of four elements, as described in *iscluster* manual page.

References

Besag, J. and Newell, J.(1991). The detection of clusters in rare diseases. Journal of the Royal Statistical Society A 154, 143-155.

See Also

DCluster, besagnewell, besagnewell.boot, besagnewell.pboot

Examples

```
library(boot)
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

#B&N's method
bnresults<-opgam(data=sids, thegrid=sids[,c("x","y")], alpha=.05,
iscluster=bn.iscluster, k=20, R=100, model="poisson",
mle=calculate.mle(sids))

#Plot all centroids and significant ones in red
plot(sids$x, sids$y, main="Besag & Newell's method")
points(bnresults$x, bnresults$y, col="red", pch=19)
```

---

calculate.mle *Calculate Parameters Involved in Sampling Procedures*

Description

When boostrap is used to sample values of the statistic under study, it is possible to use argument *mle* to pass the values of the parameters involved in the sampling procedure.

Usage

```
calculate.mle(d, model="poisson")
```

Arguments

d	A dataframe as described in the *DCluster* manual page.
model	Model used to sample data. It can be either "multinomial", "poisson" or "negbin".

Value

A list with the estimates of the parameters involved in the model:

`Multimonial` Total observed cases (*n*) and vector of probabilities (*p*).

`Poisson` Total number of regions (*n*) and vector of means (*lambda*).

`Negative Binomial (Poisson-Gamma)`
Total number of regions (*n*), size and probabilites, calculated after estimating parameters parameters *nu* and *alpha* of the Gamma distribution following equations proposed by Clayton and Kaldor (1989).

See Also

DCluster, observed.sim

Examples

```
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)


#Carry out simulations
datasim<-multinom.sim(sids, mle=calculate.mle(sids, model="multinomal") )

#Estimators for Poisson distribution
datasim<-poisson.sim(sids, mle=calculate.mle(sids, model="poisson") )

#Estimators for Negative Binomial distribution
datasim<-negbin.sim(sids, mle=calculate.mle(sids, model="negbin") )
```

DCluster *A Package for the Detection of Spatial Clusters of Diseases for Count Data*

Description

DCluster is a collection of several methods related to the detection of spatial clusters of diseases. Many widely used methods, such as Openshaw's GAM, Besag and Newell, Kulldorff and Nagarwalla, and others have been implemented.

Besides the calculation of these statistic, bootstrap can be used to test its departure from the null hypotheses, which will be no clustering in the study area. For possible sampling methods can be used to perform the simulations: permutation, Multinomial, Poisson and Poisson-Gamma.

Minor modifications have been made to the methods to use standardized expected number of cases instead of population, since it provides a better approach to the expected number of cases.

Introduction

We'll always suppose that we are working on a study region which is divided into *n* non-overlaping smaller areas where data are measured. Data measured are usually people suffering from a disease or even deaths. This will be refered as *Observed number of cases*. For a given area, its observed number of cases will be denoted by O_i and the sum of these quantities over the whole study region will be O_+.

In the same way can be defined *Population* and *Standardized Expected number of cases*, which will be denoted by P_i and E_i, respectively. The sum of all these quantities are represented by P_+ and E_+.

The basic assumption for the data is that they are independant observations from a Poisson distribution, whose mean is $\theta_i E_i$, where θ_i is the relative risk. That is,

$$O_i \sim Po(\theta_i E_i);\ i = 1, \ldots, n$$

Null hypotheses

Null hypotheses is usually equal relative risks, that is

$$H_0 : \theta_1 = \ldots = \theta_n = \lambda$$

λ may be considered to be known (one, which means standard risk) or unknown. In the last case, E_i must slightly be corrected by multiplying it by the overall relative risk $\frac{O_+}{E_+}$.

Code structure

Function names follow a common format, which is a follows:

- *method name*.statCalculate the statistic itself.
- *method name*.bootPerform a non-parametric bootstrap.
- *method name*.pbootPerform a parametric bootstrap.

Openshaw's G.A.M. has generally been implemented in a function called *gam*, which some methods (Kulldorff & Nagarwalla, Besag & Newell) also use, since they are based on a window scan of the whole region. At every point of the grid, a function is called to determine whether that point is a cluster or not. The name of this function is *shorten method name.iscluster*.

This function calculates the local value of the statistic involved and its signifiance by means of bootstrap. The interface provided, through function *gam*, is quite straightforward to use and it can handle the three methods mentioned and other supplied by the users.

Bootstrap procedures

Four possible bootstrap models have been provided in order to estimate sampling distributions of the statistics provided. The first one is a non-parametric bootstrap, which performs permutations over the observed number of cases, while the three others are parametric bootstrap based on Multinomial, Poisson and Poisson-Gamma distributions.

Permutation method just takes observed number of cases and permute them among all regions, to know whether risk in uniform across the whole study area. It just should be used with care since we'll face the problem of having more observed cases than population in very small populated areas.

Multinomial sampling is based on conditioning the Poisson framework to O_+. THis way $(O_1, \ldots, O_n)$ follows a multinomial distribution of size O_+ and probabilities $(\frac{E_1}{E_+}, \ldots, \frac{E_n}{E_+})$.

Poisson sampling just generates observed number of cases from a Poisson distribution whose mean is E_i.

Poisson-Gamma sampling is based on the Poisson-Gamma model proposed by *Clayton and Kaldor* (1984):

$$O_i|\theta_i \sim Po(\theta_i E_i)$$

$$\theta_i \sim Ga(\nu, \alpha)$$

The distribution of O_i unconditioned to θ_i is Negative Binomial with size ν and probability $\frac{\alpha}{\alpha+E_i}$. The two parameters can be estimated using an Empirical Bayes approach from the Expected and Observed number of cases. Function *empbaysmooth* is provided for this purpose.

Data

One of the parameters, which is usually called *data*, passed to many of the functions in this package is a dataframe which contains the data for each of the regions used in the analysis. Besides, its columns must be labeled:

- **Observed**Observed number of cases.
- **Expected**Standardised expected number of cases.
- **Population**Population at risk.
- **x**Easting coordinate of the region centroid.
- **y**Northing coordinate of the region centroid.

References

Clayton, David and Kaldor, John (1987). Empirical Bayes Estimates of Age-standardized Relative Risks for Use in Disease Mapping. Biometrics 43, 671-681.

Lawson et al (eds.) (1999). Disease Mapping and Risk Assessment for Public Health. John Wiley and Sons, Inc.

Lawson, A. B. (2001). Statistical Methods in Spatial Epidemiology. John Wiley and Sons, Inc.

dcluster *Class for Results from a Test for the Detection of Disease Clusters*

Description

Class 'dcluster' is used to store the main information when a (boostrap) test is performed to detect clusters of disease. Essentially, this class has the same structure and contents as class 'boot' (see in package 'boot') plus some additional information on the test performed.

Additional functions to plot and summarise the results of the test are supplied as well.

Usage

```
## S3 method for class 'dcluster'
plot(x, ...)
## S3 method for class 'dcluster'
print(x, ...)
## S3 method for class 'dcluster'
summary(object, ...)
```

Arguments

x	A 'dcluster' object.
object	A 'dcluster' object.
...	Any other additional arguments needed (for example, to pass additional arguments to the plot function).

Value

These functions do not return anything but produce some plots or print asummary of the test.

empbaysmooth *Empirical Bayes Smoothing*

Description

Smooth relative risks from a set of expected and observed number of cases using a Poisson-Gamma model as proposed by *Clayton and Kaldor* (1987) .

If ν and α are the two parameters of the prior Gamma distribution, smoothed relative risks are $\frac{O_i+\nu}{E_i+\alpha}$.

ν and α are estimated via Empirical Bayes, by using mean and variance, as described by *Clayton and Kaldor*(1987).

Size and probabilities for a Negative Binomial model are also calculated (see below).

See *Details* for more information.

Usage

```
empbaysmooth(Observed, Expected, maxiter=20, tol=1e-5)
```

Arguments

Observed	Vector of observed cases.
Expected	Vector of expected cases.
maxiter	Maximum number of iterations allowed.
tol	Tolerance used to stop the iterative procedure.

Details

The Poisson-Gamma model, as described by *Clayton and Kaldor*, is a two-layers Bayesian Hierarchical model:

$$O_i|\theta_i \sim Po(\theta_i E_i)$$

$$\theta_i \sim Ga(\nu, \alpha)$$

The posterior distribution of O_i,unconditioned to θ_i, is Negative Binomial with size ν and probability $\alpha/(\alpha + E_i)$.

The estimators of relative risks are $\widehat{\theta}_i = \frac{O_i+\nu}{E_i+\alpha}$. Estimators of ν and α ($\widehat{\nu}$ and $\widehat{\alpha}$,respectively) are calculated by means of an iterative procedure using these two equations (based on mean and variance estimations):

$$\frac{\widehat{\nu}}{\widehat{\alpha}} = \frac{1}{n}\sum_{i=1}^{n}\widehat{\theta}_i$$

$$\frac{\widehat{\nu}}{\widehat{\alpha}^2} = \frac{1}{n-1}\sum_{i=1}^{n}(1+\frac{\widehat{\alpha}}{E_i})(\widehat{\theta}_i - \frac{\widehat{\nu}}{\widehat{\alpha}})^2$$

Value

A list of four elements:

n	Number of regions.
nu	Estimation of parameter ν
alpha	Estimation of parameter α
smthrr	Vector of smoothed relative risks.
size	Size parameter of the Negative Binomial. It is equal to $\widehat{\nu}$

.

prob It is a vector of probabilities of the Negative Binomial, calculated as

$$\frac{\widehat{\alpha}}{\widehat{\alpha} + E_i}$$

.

References

Clayton, David and Kaldor, John (1987). Empirical Bayes Estimates of Age-standardized Relative Risks for Use in Disease Mapping. Biometrics 43, 671-681.

Examples

```
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))

smth<-empbaysmooth(sids$Observed, sids$Expected)
```

---

gearyc *Geary's C Autocorrelation Statistic*

Description

Geary's c statistic is used to measure autocorrelation between areas within a region, as follows:

$$c = \frac{(n-1)\sum_i \sum_j W_{ij}(Z_i - Z_j)^2}{2(\sum_i \sum_j W_{ij}) \sum_k (Z_k - \overline{Z})^2}$$

W is a squared matrix which represents the relationship between each pair of regions. An usual approach is set w_{ij} to 1 if regions i and j have a common boundary and 0 otherwise, or it may represent the inverse distance between the centroids of that two regions.

Small values of this statistic may indicate the presence of highly correlated areas, which may be a cluster.

References

Geary, R. C. (1954). The contiguity ratio and statistical mapping. The Incorporated Statistician 5, 115-145.

See Also

DCluster, gearyc.stat, gearyc.boot, gearyc.pboot

gearyc.boot *Generate Bootstrap Replicates of Geary's C Autocorrelation Statistic*

Description

Generate bootstrap replicates of Geary's C autocorrelation statistic, by means of function *boot* form *boot* library. Notice that these functions should not be used separately but as argument *statistic* when calling function *boot*.

gearyc.boot is used when performing a non-parametric bootstrap.

gearyc.pboot is used when performing a parametric bootstrap.

Usage

```
gearyc.boot(data, i, ...)
gearyc.pboot(...)
```

Arguments

data	A dataframe containing the data, as specified in the **DCluster**manpage.
i	Permutation generated by the bootstrap procedure
...	Aditional arguments passed when performing a bootstrap.

Value

Both functions return the value of the statistic.

References

Geary, R. C. (1954). The contiguity ratio and statistical mapping. The Incorporated Statistician 5, 115-145.

See Also

DCluster, boot, gearyc, gearyc.stat

Examples

```
library(boot)
library(spdep)

data(nc.sids)
col.W <- nb2listw(ncCR85.nb, zero.policy=TRUE)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
```

```
niter<-100

#Permutation model
gc.perboot<-boot(sids, statistic=gearyc.boot, R=niter, listw=col.W,
n=length(ncCR85.nb), n1=length(ncCR85.nb)-1, S0=Szero(col.W) )
plot(gc.perboot)#Display results

#Multinomial model
gc.mboot<-boot(sids, statistic=gearyc.pboot, sim="parametric",
ran.gen=multinom.sim, R=niter, listw=col.W,
        n=length(ncCR85.nb), n1=length(ncCR85.nb)-1, S0=Szero(col.W) )
plot(gc.mboot)#Display results

#Poisson model
gc.pboot<-boot(sids, statistic=gearyc.pboot, sim="parametric",
ran.gen=poisson.sim, R=niter, listw=col.W,
n=length(ncCR85.nb), n1=length(ncCR85.nb)-1, S0=Szero(col.W) )
plot(gc.pboot)#Display results

#Poisson-Gamma model
gc.pgboot<-boot(sids, statistic=gearyc.pboot, sim="parametric",
ran.gen=negbin.sim, R=niter, listw=col.W,
n=length(ncCR85.nb), n1=length(ncCR85.nb)-1, S0=Szero(col.W) )
plot(gc.pgboot)#Display results
```

`gearyc.stat` *Compute Geary's C Autocorrelation Statistic*

Description

Compute Geary's C autocorrelation statistic using either **residuals** or **SMRs** by means of cuntion *geary* from package *spdep*.

gearyc.stat computes the test statistic and the test using a hi-square distribution whilst gearyc.test performs a bootstrap test.

Usage

```
gearyc.stat(data, applyto="SMR", ...)
gearyc.test(formula, data, model, R, ...)
```

Arguments

`formula`	Formula that specifies the underlying model. The observed cases are the response and the expected number of cases must be specified as an offset in the log scale (see example below). Note that now it is not necessary to use Observed and Expected and that any other names can be used to specify the observed and expected cases.

model	Parametric model to be used in the bootstrap test. One of "param", "multinom", "poisson" or "negbin". See the **DCluster** manpage for details.
...	Arguments needed by function *moran* from package *spdep*. In addition, when calling 'gearyc.test' the remaining arguments in 'gearyc.stat' not included in 'gearyc.test'. This is done so because gearyc.test calls gearyc.stat in order to perform the test.
R	Number of replicates used in the test to compute the significance of the observed value of the test statistic.
data	A dataframe containing the data, as specified in the **DCluster** manpage.
applyto	A string with the name of the statistic with which calculate Geary's Index. It may be either *residulas* or *SMR*.

References

Geary, R. C. (1954). The contiguity ratio and statistical mapping. The Incorporated Statistician 5, 115-145.

See Also

DCluster, geary, gearyc, gearyc.boot, gearyc.pboot

Examples

```
library(spdep)
data(nc.sids)
col.W <- nb2listw(ncCR85.nb, zero.policy=TRUE)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))

gearyc.stat(data=sids, listw=col.W, n=length(ncCR85.nb), n1=length(ncCR85.nb)-1,
S0=Szero(col.W) )

gearyc.stat(data=sids, applyto="SMR", listw=col.W, n=length(ncCR85.nb),
n1=length(ncCR85.nb)-1,S0=Szero(col.W) )

gearyc.test(Observed~offset(log(Expected)), data=sids, model="poisson", R=99,
   applyto="SMR", listw=col.W, n=length(ncCR85.nb),
n1=length(ncCR85.nb)-1,S0=Szero(col.W) )
```

---

get.knclusters *Get Areas in a Cluster Detected with Kulldorff's Statistic*

Description

When *kn.iscluster* is called from *opgam* to use Kulldorff's scan statistic for the detection of clusters of disease, *get.knclusters* can be used to get the areas included in each cluster. *opgam* only returns the cluster centres, size and related information but not the areas in the cluster.

Usage

```
get.knclusters(d, knresults)
```

Arguments

d	Data frame with the data, used in the call to *opgam*.
knresults	Data frame returned by a call to *opgam*.

Value

Returns a list with the same length as the number of rows in 'knresults'. Each element in the list is a vector of integers with the row indices of 'd' of the areas in the cluster. The order of the indices reflects the distance to the cluster centre.

References

Kulldorff, Martin and Nagarwalla, Neville (1995). Spatial Disease Clusters: Detection and Inference. Statistics in Medicine 14, 799-810.

See Also

DCluster, kullnagar, kullnagar.stat, kullnagar.boot, kullnagar.pboot, opgam

Examples

```
library(boot)
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, Population=nc.sids$BIR74, x=nc.sids$x, y=nc.sids$y)

#K&N's method over the centroids
mle<-calculate.mle(sids, model="poisson")
knresults<-opgam(data=sids, thegrid=sids[,c("x","y")], alpha=.05,
iscluster=kn.iscluster, fractpop=.15, R=99, model="poisson", mle=mle)

#Plot all centroids and significant ones in red
plot(sids$x, sids$y, main="Kulldorff and Nagarwalla's method")
points(knresults$x, knresults$y, col="red", pch=19)

#Plot first cluster with the highest likelihood ratio test in green
clusters<-get.knclusters(sids, knresults)
idx<-which.max(knresults$statistic)
points(sids$x[clusters[[idx]]], sids$y[clusters[[idx]]], col="green", pch=19)
```

`iscluster` *Local Clustering Test Function*

Description

This function is passed to function *gam* as argument *iscluster* to decide whether the current circle must be marked as a cluster or not.

opgam.iscluster.default is the function used by default, based on quantiles of the Poisson distribution.

opgam.iscluster.negbin is similar to the previous one but based on the Negative Binomial distribution. Local significance is estimated using bootstrap since it involves the sum of Negative Binomial variables.

Usage

```
opgam.iscluster.default(data, idx, idxorder, alpha,  ...)
opgam.iscluster.negbin(data, idx, idxorder, alpha, mle, R=999, ...)
```

Arguments

`data`	A dataframe with the data, as explained in *DCluster* manual page.
`idx`	A boolean vector to know the areas in the current circle.
`idxorder`	A permutation of the rows of data to order the regions according to their distance to the current center.
`alpha`	Test signifiance.
`mle`	Estimators of some parameters needed by the Negative Binomial distribution. See *negbin.sim* manual page for details.
`R`	Number of simulations made used to estimate local pvalues.
`...`	Any other arguments required.

Details

These functions take a number of arguments to be able to assess whether the current ball is a cluster or not. We have follow this approarch to create a common framework for all scan methods.

The vector returned by this functions can be of size higher than four, but the first four elements must be those stated in this manual page (and in the same order).

More example can be found in the implementations of other scan methods, such as Besag and Newell's, and Kulldorff and Nagarwalla's.

Value

A vector with four values:

`statistic`	Value of the statistic computed.
`result`	A boolean value, which is *TRUE* for clusters.
`pvalue`	The pvalue obtained for the test performed.
`size`	Size of the cluster in inumber of regions from the centre.

See Also

opgam, besagnewell, bn.iscluster, kullnagar, kn.iscluster, turnbull, tb.iscluster

---

`kn.iscluster` *Clustering Function for Kulldorff and Nagarwalla's Statistic*

Description

kn.iscluster is called from *opgam* when studying the whole area. At every point of the grid, which may be all the centroids, this function is called to determine whether it is a cluster or not by calculating Kulldorff and Nagarwalla's statistic.

See *opgam.iscluster.default* for more details. *kn.gumbel.iscluster* uses a Gumbel distribution to compute the p-values ofr each possible cluster.

Usage

```
kn.iscluster(data, idx, idxorder, alpha, fractpop, use.poisson=TRUE,
 model="poisson", R, mle, ...)
kn.gumbel.iscluster(data, idx, idxorder, alpha, fractpop, use.poisson=TRUE,
 model="poisson", R, mle)
```

Arguments

`data`	A dataframe with the data as explained in *DCluster*.
`idx`	A boolean vector to know the areas in the current circle.
`idxorder`	A permutation of the rows of data to order the regions according to their distance to the current center.
`alpha`	Test signifiance.
`fractpop`	Maximum fraction of the total population used when creating the balls.
`use.poisson`	Use the statistic for Poisson (default) or Bernouilli case.
`model`	Thge model used to generate random observations. It can be 'permutation', 'multinomial', 'poisson' or 'negbin'. See *observed.sim* manual page for details.
`R`	The number of bootstrap replicates to generate.
`mle`	Parameters need by the bootstrap procedure.
`...`	Extra arguments to be passed to kullnagar.stat().

Value

A vector of four elements, as describe in *iscluster* manual page.

References

Kulldorff, Martin and Nagarwalla, Neville (1995). Spatial Disease Clusters: Detection and Inference. Statistics in Medicine 14, 799-810. Abrams A, Kleinman K, Kulldorff M (2010). Gumbel based p-value approximations for spatial scan statistics. International Journal of Health Geographics, 9:61.

See Also

DCluster, kullnagar, kullnagar.stat, kullnagar.boot, kullnagar.pboot

Examples

```
library(boot)
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, Population=nc.sids$BIR74, x=nc.sids$x, y=nc.sids$y)

#K&N's method over the centroids
mle<-calculate.mle(sids, model="poisson")
knresults<-opgam(data=sids, thegrid=sids[,c("x","y")], alpha=.05,
  iscluster=kn.iscluster, fractpop=.5, R=100, model="poisson", mle=mle)

kngumbelres<-opgam(data=sids, thegrid=sids[,c("x","y")], alpha=.05,
  iscluster=kn.gumbel.iscluster, fractpop=.5, R=100, model="poisson",
  mle=mle)

#Plot all centroids and significant ones in red
plot(sids$x, sids$y, main="Kulldorff and Nagarwalla's method")
points(knresults$x, knresults$y, col="red", pch=19)
points(knresults$x, knresults$y, col="blue", pch=20)
```

---

kullnagar *Kulldorff and Nagarwalla's Statistic for Spatial Clustering.*

Description

This method is based on creating a grid over the study area. Each point of the grid is taken to be the centre of all circles that contain up to a fraction of the total population. This is calculated by suming all the population of the regions whose centroids fall inside the circle. For each one of these balls, the likelihood ratio of the next test hypotheses is computed:

$$\begin{aligned} H_0 &: p = q \\ H_1 &: p > q \end{aligned}$$

where *p* is the probability of being a case inside the ball and *q* the probability of being a case outside it. Then, the ball where the maximum of the likelihood ratio is achieved is selected and its value is tested to assess whether it is significant or not.

There are two possible statistics, depending on the model assumed for the data, which can be Bernouilli or Poisson. The value of the likelihood ratio statistic is

$$\max_{z \in Z} \frac{L(z)}{L_0}$$

where *Z* is the set of ball at a given point, z an element of this set, L_0 is the likelihood under the null hypotheses and $L(z)$ is the likelihood under the alternative hypotheses. The actual formulae involved in the calculation can be found in the reference given below.

References

Kulldorff, Martin and Nagarwalla, Neville (1995). Spatial Disease Clusters: Detection and Inference. Statistics in Medicine 14, 799-810.

See Also

DCluster, kullnagar.stat, kullnagar.boot, kullnagar.pboot

`kullnagar.boot` *Generate Bootstrap Replicates of Kulldorff and Nagarwalla's Statistic*

Description

Generate bootstrap replicates of Kulldorff and Nagarwalla's statistic, by calling functions *boot* and *kullnagar.stat*.

kullnagar.boot is used when using non-parametric bootstrap to estimate the distribution of the statistic.

kullnagar.pboot is used when performing parametric bootstrap.

Usage

```
kullnagar.boot(data, i, ...)
kullnagar.pboot(...)
```

Arguments

`data`	A dataframe with the data as explained in *DCluster*.
`i`	Permutation created in non-parametric bootstrap.
`...`	Additional arguments passed to the functions.

Value

Both functions return the value of the statistic.

References

Kulldorff, Martin and Nagarwalla, Neville (1995). Spatial Disease Clusters: Detection and Inference. Statistics in Medicine 14, 799-810.

See Also

DCluster, boot, kullnagar, kullnagar.stat, kn.iscluster

Examples

```
library(boot)
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, Population=nc.sids$BIR74, x=nc.sids$x, y=nc.sids$y)

niter<-100

#Permutation  model
kn.perboot<-boot(sids, statistic=kullnagar.boot, R=niter, fractpop=.2)
plot(kn.perboot)#Display results

#Multinomial model
kn.mboot<-boot(sids, statistic=kullnagar.pboot, sim="parametric",
ran.gen=multinom.sim,  R=niter, fractpop=.2)
plot(kn.mboot)#Display results

#Poisson model
kn.pboot<-boot(sids, statistic=kullnagar.pboot, sim="parametric",
ran.gen=poisson.sim,  R=niter, fractpop=.2)
plot(kn.pboot)#Display results

#Poisson-Gamma model
kn.pgboot<-boot(sids, statistic=kullnagar.pboot, sim="parametric",
ran.gen=negbin.sim, R=niter, fractpop=.2)
plot(kn.pgboot)#Display results
```

`kullnagar.stat` *Kulldorff and Nagarwalla's Statistic for Spatial Clustering.*

Description

Compute Kulldorff and Nagarwalla's spatial statistic for cluster detection around a single region, which is supposed to be the first row of the dataframe. The other regions are supposed to be sorted by distance to the centre in the data frame.

Two possible function are provided: *kullnagar.stat.poisson*, for th Poisson case, and *kullnagar.stat.bern*, for the Bernouilli case.

See *kullnagar* manual page for details.

Usage

```
kullnagar.stat(data, fractpop, use.poisson=TRUE, log.v=FALSE)
```

Arguments

`data`	A dataframe with the data as explained in *DCluster*.
`fractpop`	Maximum fraction of the total population used when creating the balls.
`use.poisson`	Use the statistic for Poisson (default) or Bernouilli case.
`log.v`	Whether the logarithm of the statistic is returned or not.

Value

Returns a vector of two elements: the value of the statistic and the size (in number of regions) of the cluster.

References

Kulldorff, Martin and Nagarwalla, Neville (1995). Spatial Disease Clusters: Detection and Inference. Statistics in Medicine 14, 799-810.

See Also

DCluster, kullnagar, kullnagar.stat, kullnagar.boot, kullnagar.pboot

Examples

```
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, Population=nc.sids$BIR74, x=nc.sids$x, y=nc.sids$y)
```

```
dist<-(sids$x-sids$x[1])^2+(sids$y-sids$y[1])^2
index<-order(dist)
#Compute the statistic around the first county
kullnagar.stat(sids[index,], fractpop=.5)
```

---

lognormalEB *Empirical Bayes Smoothing Using a log-Normal Model*

Description

Smooth relative risks from a set of expected and observed number of cases using a log-Normal model as proposed by *Clayton and Kaldor* (1987). There are estimated by $\tilde{\beta}_i = \log((O_i + 1/2)/E_i)$ in order to prevent taking the logarithm of zero.

If this case, the log-relative risks are assumed be independant and to have a normal distribution with mean φ and variance σ^2. Clayton y Kaldor (1987) use the EM algorithm to develop estimates of these two parameters which are used to compute the Empirical Bayes estimate of b_i. The formula is not listed here, but it can be consulted in Clayton and Kaldor (1987).

Usage

```
lognormalEB(Observed, Expected, maxiter = 20, tol = 1e-05)
```

Arguments

`Observed`	Vector of observed cases.
`Expected`	Vector of expected cases.
`maxiter`	Maximum number of iterations allowed.
`tol`	Tolerance used to stop the iterative procedure.

Value

A list of four elements:

`n`	Number of regions.
`phi`	Estimate of φ.
`sigma2`	Estimate of σ^2.
`smthrr`	Vector of smoothed relative risks.

References

Clayton, David and Kaldor, John (1987). Empirical Bayes Estimates of Age-standardized Relative Risks for Use in Disease Mapping. Biometrics 43, 671-681.

Examples

```
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))

smth<-lognormalEB(sids$Observed, sids$Expected)
```

`moranI` *Moran's I Autocorrelation Statistic*

Description

Moran's I statistic measures autocorrelation between areas within a region. It is similar to the correlation coefficient:

$$I = \frac{n \sum_i \sum_j W_{ij}(Z_i - \overline{Z})(Z_j - \overline{Z})}{2(\sum_i \sum_j W_{ij}) \sum_k (Z_k - \overline{Z})^2}$$

W is a squared matrix which represents the relationship between each pair of regions. An usual approach is set w_{ij} to 1 if regions i and j have a common boundary and 0 otherwise, or it may represent the inverse distance between the centroids of these two regions.

High values of this statistic may indicate the presence of groups of zones where values are unusually high. On the other hand, low values of the Moran's statistic will indicate no correlation between neighbouring areas, which may lead to indipendance in the observations.

moranI.stat is the function to calculate the value of the statistic for residuals or SMRs of the data.

moranI.boot is used when performing a non-parametric bootstrap.

moranI.pboot is used when performing a parametric bootstrap.

References

Moran, P. A. P. (1948). The interpretation os statistical maps. Journal of the Royal Statistical Society, Series B 10, 243-251.

See Also

DCluster, moranI.stat, moranI.boot, moranI.pboot

`moranI.boot` *Generate Bootstrap Replicates of Moran's I Autocorrelation Statistic*

Description

Generate bootstrap replicates of Moran's I autocorrelation statistic, by means of function *boot* form *boot* library. Notice that these functions should not be used separately but as argument *statistic* when calling function *boot*.

moranI.boot is used when performing a non-parametric bootstrap.

moranI.pboot is used when performing a parametric bootstrap.

Usage

```
moranI.boot(data, i, ...)
moranI.pboot(...)
```

Arguments

`data`	A dataframe containing the data, as specified in the **DCluster**manpage.
`i`	Permutation generated by the bootstrap procedure
`...`	Aditional arguments passed when performing a bootstrap.

Value

Both functions return the value of the statistic.

References

Moran, P. A. P. (1948). The interpretation os statistical maps. Journal of the Royal Statistical Society, Series B 10, 243-251.

See Also

DCluster, boot, moranI, moranI.stat

Examples

```
library(spdep)
data(nc.sids)
col.W <- nb2listw(ncCR85.nb, zero.policy=TRUE)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))

niter<-100

#Permutation model
```

```
moran.boot<-boot(sids, statistic=moranI.boot, R=niter, listw=col.W,
n=length(ncCR85.nb), S0=Szero(col.W) )
plot(moran.boot)#Display results

#Multinomial model
moran.mboot<-boot(sids, statistic=moranI.pboot, sim="parametric",
ran.gen=multinom.sim,  R=niter, listw=col.W,n=length(ncCR85.nb),
S0=Szero(col.W) )
plot(moran.mboot)#Display results

#Poisson model
moran.pboot<-boot(sids, statistic=moranI.pboot, sim="parametric",
ran.gen=poisson.sim,  R=niter, listw=col.W,n=length(ncCR85.nb),
S0=Szero(col.W) )

plot(moran.pboot)#Display results

#Poisson-Gamma model
moran.pgboot<-boot(sids, statistic=moranI.pboot, sim="parametric",
ran.gen=negbin.sim, R=niter,  listw=col.W,n=length(ncCR85.nb),
S0=Szero(col.W) )

plot(moran.pgboot)#Display results
```

`moranI.stat` *Compute Moran's I Autocorrelation Statistic*

Description

Compute Moran's I autocorrelation statistic using **residuals** or **SMRs** by means of function *moran* from package *spdep*.

moranI.stat computes the test statistic and the test using a hi-square distribution whilst moranI.test performs a bootstrap test.

Usage

```
moranI.stat(data, applyto="SMR", ...)
moranI.test(formula, data, model, R, ...)
```

Arguments

`formula`	Formula that specifies the underlying model. The observed cases are the response and the expected number of cases must be specified as an offset in the log scale (see example below). Note that now it is not necessary to use Observed and Expected and that any other names can be used to specify the observed and expected cases.
`model`	Parametric model to be used in the bootstrap test. One of "param", "multinom", "poisson" or "negbin". See the **DCluster** manpage for details.

...	Arguments needed by function *moran* from package *spdep*. In addition, when calling 'moranI.test' the remaining arguments in 'moranI.stat' not included in 'moranI.test'. This is done so because moranI.test calls moranI.stat in order to perform the test.
R	Number of replicates used in the test to compute the significance of the observed value of the test statistic.
data	A dataframe containing the data, as specified in the **DCluster** manpage.
applyto	A string with the name of the statistic with which calculate Moran's Index. It may be either *residulas* or *SMR*.

Value

The value of the statistic computed.

References

Moran, P. A. P. (1948). The interpretation os statistical maps. Journal of the Royal Statistical Society, Series B 10, 243-251.

See Also

DCluster, moran, moranI, moranI.boot, MoranI.pboot

Examples

```
library(spdep)
data(nc.sids)
col.W <- nb2listw(ncCR85.nb, zero.policy=TRUE)


sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74) )

moranI.stat(data=sids, listw=col.W, n=length(ncCR85.nb), S0=Szero(col.W) )

moranI.stat(data=sids, applyto="residuals", listw=col.W, n=length(ncCR85.nb),
S0=Szero(col.W) )

moranI.test(Observed~offset(log(Expected)), sids, model="poisson", R=99,
   listw=col.W, n=length(ncCR85.nb), S0=Szero(col.W) )
```

---

observed.sim *Randomly Generate Observed Cases from Different Statistical Distributions*

Description

Simulate Observed number of cases according to a Multinomial, Poisson or Negative Binomial distribution.

These functions are used when performing a parametric bootstrap and they must be passed as argument *ran.gen* when calling function *boot*.

multinom.sim generates observations from a Multinomial distribution.

poisson.sim generates observations from a Poisson distribution.

negbin.sim generates observations from a Negative Binomial distribution.

Usage

```
multinom.sim(data, mle=NULL)

poisson.sim(data, mle=NULL)

negbin.sim(data, mle=NULL)
```

Arguments

`data`	A dataframe as described in the *DCluster* manual page.
`mle`	List containing the parameters of the distributions to be used. If they are not provided, then they are calculated from the data. Its value argument *mle* in function *boot*. The elements in the list depend on the distribution to be used: • Multimonial Total observed cases (n) and vector of probabilities (p). • Poisson Total number of regions (n) and vector of means (*lambda*). • Negative Binomial Total number of regions (n) and parameters *nu* and *alpha* of the Gamma distribution.

Value

A dataframe equal to the argument *data*, but in which the Observed column has been substituted by sampled observations. See *DCluster* manual page for more details.

See Also

DCluster

Examples

```
library(spdep)

data(nc.sids)
```

```
sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

#Carry out simulations
datasim<-multinom.sim(sids, mle=calculate.mle(sids, model="multinomal") )

#Estimators for Poisson distribution
datasim<-poisson.sim(sids, mle=calculate.mle(sids, model="poisson") )

#Estimators for Negative Binomial distribution
datasim<-negbin.sim(sids, mle=calculate.mle(sids, model="negbin") )
```

---

opgam *Openshaw's GAM*

Description

Scan an area with Openshaw's Geographical Analysis Machine to look for clusters.

opgam is the main function, while *gam.intern* is called from there.

Usage

```
opgam(data, thegrid=NULL, radius=Inf, step=NULL, alpha,
 iscluster=opgam.iscluster.default, set.idxorder=TRUE, ...)
opgam.intern(point, data, rr, set.idxorder, iscluster, alpha, ...)
```

Arguments

data	A dataframe with the data, as described in *DCluster* manual page.
thegrid	A two-columns matrix containing the points of the grid to be used. If it is null, a rectangular grid of step *step* is built.
radius	The radius of the circles used in the computations.
step	The step of the grid.
alpha	Significance level of the tests performed.
iscluster	Function used to decide whether the current circle is a possible cluster or not. It must have the same arguments and return the same object than *gam.iscluster.default*
.	
set.idxorder	Whether an index for the ordering by distance to the center of the current ball is calculated or not.
point	Point where the curent ball is centred.
rr	rr=radius*radius .
...	Aditional arguments to be passed to *iscluster*.

Details

The *Geographical Analysis Machine* was developed by Openshaw et al. to perform geographical studies of the relationship between different types of cancer and their proximity to nuclear plants.

In this method, a grid of a fixed step is built along the study region, and small balls of a given radius are created at each point of the grid. Local observed and expected number of cases and population are calculated and a function is used to assess whether the current ball is a cluster or not. For more information about this function see *opgam.iscluster.default*, which is the default function used.

If the obverved number of cases excess a critical value, which is calculated by a function passed as an argument, then that circle is marked as a possible cluster. At the end, all possible clusters are drawn on a map. Clusters may be easily identified then.

Notice that we have follow a pretty flexible approach, since user-implemented functions can be used to detect clusters, such as those related to ovedispersion (Pearson's Chi square statistic, Potthoff-Whittinghill's statistic) or autocorrelation (Moran's I statistic and Geary's c statistic), or a bootstrap procedure, although it is not recommended because it can be VERY slow.

Value

A dataframe with five columns:

`x`	Easting coordinate of the center of the cluster.
`y`	Northing coordinate of the center of the cluster.
`statistic`	Value of the statistic computed.
`cluster`	Is it a cluster (according to the criteria used)? It should be always TRUE.
`pvalue`	Significance of the cluster.

References

Openshaw, S. and Charlton, M. and Wymer, C. and Craft, A. W. (1987). A mark I geographical analysis machine for the automated analysis of point data sets. International Journal of Geographical Information Systems 1, 335-358.

Waller, Lance A. and Turnbull, Bruce W. and Clarck, Larry C. and Nasca, Philip (1994). Spatial Pattern Analyses to Detect Rare Disease Clusters. In 'Case Studies in Biometry'. Chapter 1, 3-23.

See Also

DCluster, opgam.iscluster.default

Examples

```
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)
```

```
#GAM using the centroids of the areas in data
sidsgam<-opgam(data=sids,  radius=30, step=10, alpha=.002)

#Plot centroids
plot(sids$x, sids$y, xlab="Easting", ylab="Northing")
#Plot points marked as clusters
points(sidsgam$x, sidsgam$y, col="red", pch="*")
```

---

pottwhitt *Potthoff-Whittinghill's Statistic for Overdispersion*

Description

This statistic can be used to test for homogeinity among all the relative risks. The test statistic is:

$$E_+ \sum_{i=1}^{n} \frac{O_i(O_i - 1)}{E_i}$$

If we supposse that the data are generated from a multinomial model, this is the locally U.M.P. when considering the next hypotheses:

$$\begin{aligned} H_0 &: \theta_1 = \ldots = \theta_n = \lambda \\ H_1 &: \theta_i \sim Ga(\lambda^2/\sigma^2, \lambda/\sigma^2) \end{aligned}$$

Notice that in this case, λ is supposed to be unknown. The alternative hypotheses means that relative risks come all from a Gamma distribution with mean λ and variance σ^2.

pottwhitt.stat is the function to calculates the value of the statistic for the data.

pottwhitt.boot is used when performing a non-parametric bootstrap.

pottwhitt.pboot is used when performing a parametric bootstrap.

References

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: I. The Binomial and Multinomial Distributions. Biometrika 53, 167-182.

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: The Poisson Distribution. Biometrika 53, 183-190.

See Also

DCluster, pottwhitt.stat, pottwhitt.boot, pottwhitt.pboot

`pottwhitt.boot` *Bootstrap Replicates of Potthoff-Whittinghill's Statistic*

Description

Generate bootstrap replicates of Potthoff-Whittinghill's statistic (function *pottwhitt.stat*), by means of function *boot* from the *boot* library. Notice that these functions should not be used separately but as argument *statistic* when calling function *boot*.

pottwhitt.boot is used when performing a non-parametric bootstrap.

pottwhitt.pboot is used when performing a parametric bootstrap.

Usage

```
pottwhitt.boot(data, i)
pottwhitt.pboot(...)
```

Arguments

`data`	A dataframe containing the data, as specified in the **DCluster** manual page.
`i`	Permutation generated by the bootstrap procedure
`...`	Additional arguments passed when performing a bootstrap.

Value

Both functions return the value of the statistic.

References

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: I. The Binomial and Multinomial Distributions. Biometrika 53, 167-182.

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: The Poisson Distribution. Biometrika 53, 183-190.

See Also

DCluster, pottwhitt, pottwhitt.stat

Examples

```
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)
```

```
niter<-100


#Permutation model
pw.boot<-boot(sids, statistic=pottwhitt.boot, R=niter)
plot(pw.boot)#Plot results


#Multinomial model
pw.mboot<-boot(sids, statistic=pottwhitt.pboot, sim="parametric", ran.gen=multinom.sim, R=niter)
plot(pw.mboot)#Plot results

#Poisson model
pw.pboot<-boot(sids, statistic=pottwhitt.pboot, sim="parametric", ran.gen=poisson.sim, R=niter)
plot(pw.pboot)#Plot results

#Poisson-Gamma model
pw.pgboot<-boot(sids, statistic=pottwhitt.pboot, sim="parametric", ran.gen=negbin.sim, R=niter)
plot(pw.pgboot)#Plot results
```

`pottwhitt.stat` *Compute Potthoff-Whittinghill's Statistic*

Description

Compute Pottwhoff-Whittinghill's statistic.

pottwhitt.stat computes the test statistic and the test using a hi-square distribution whilst pottwhitt.test performs a bootstrap test.

Usage

```
pottwhitt.stat(data)
pottwhitt.test(formula, data, model, R, ...)
```

Arguments

`formula`	Formula that specifies the underlying model. The observed cases are the response and the expected number of cases must be specified as an offset in the log scale (see example below). Note that now it is not necessary to use Observed and Expected and that any other names can be used to specify the observed and expected cases.
`model`	Parametric model to be used in the bootstrap test. One of "param", "multinom", "poisson" or "negbin". See the **DCluster** manpage for details.
`...`	The remaining arguments in 'achisq.stat' not included in 'achisq.test'. This is done so because achisq.test calls achisq.stat in order to perform the test.

R	Number of replicates used in the test to compute the significance of the observed value of the test statistic.
data	A dataframe containing the data, as specified in the **DCluster** manpage.

Value

A list with the following elements:

T	The value of the statistic.
asintmean	Mean of the asymptotical Normal distribution.
asintvar	Variance of the asymptotical Normal distribution.
pvalue	Significance of the statistic.

References

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: I. The Binomial and Multinomial Distributions. Biometrika 53, 167-182.

Potthoff, R. F. and Whittinghill, M.(1966). Testing for Homogeneity: The Poisson Distribution. Biometrika 53, 183-190.

See Also

DCluster

Examples

```
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

pottwhitt.stat(sids)

pottwhitt.test(Observed~offset(log(Expected)),sids, model="poisson", R=99)
```

---

rmultin *Generate Random Observations from a Multinomial Distribution*

Description

This function generates a random observation from a multinomial distribution.

Usage

```
rmultin(n, p)
```

Arguments

n	Total size (and NOT the number of variables involved in the multinomial distribution).
p	Vector of probabilities. The sum of all its elements must be one.

Value

A vector with the sample which has been generated.

Examples

```
for(i in 1:10)
print(rmultin(10, c(1/3, 1/3, 1/3) ))
```

---

stone *Stone's Test*

Description

Stone's Test is used to assess risk around given locations (i. e., a putative pollution source). The null hypotheses is that relative risks are constant across areas, while the alternative is that there is descending trend in relative risks as distance to the focus increases. That is

$$\begin{aligned} H_0 &: \theta_1 = \ldots = \theta_n = \lambda \\ H_1 &: \theta_1 \geq \ldots \geq \theta_n \end{aligned}$$

Supposing data sorted by distance to the putative pollution source, Stone's statistic is as follows:

$$\max_j \left(\frac{\sum_{i=1}^{j} O_i}{\sum_{i=1}^{j} E_i} \right)$$

Depending on whether λ is known (usually 1) or not, E_i may need a minor correction, which are not done automatically. See *achisq* manual page for details.

References

Stone, R. A. (1988). Investigating of excess environmental risks around putative sources: Statistical problems and a proposed test. Statistics in Medicine 7,649-660.

See Also

DCluster, stone.stat, stone.boot, stone.pboot

`stone.boot` *Generate Boostrap Replicates of Stone's Statistic*

Description

Generate bootstrap replicates of Stone's statictic, by means of function *boot* from *boot* package. Notice that these functions should not be used separately but as argument *statistic* when calling function *boot*.

stone.boot is used when performing a non-parametric bootstrap.

stone.pboot is used when performing a parametric bootstrap.

Usage

```
stone.boot(data, i, ...)
stone.pboot(...)
```

Arguments

`data`	A dataframe with all the data, as explained in the *DCluster* manual page.
`i`	Permutation created in non-parametric bootstrap.
`...`	Additional arguments passed to the functions.

Value

Both functions return the value of the statistic.

References

Stone, R. A. (1988). Investigating of excess environmental risks around putative sources: Statistical problems and a proposed test. Statistics in Medicine 7,649-660.

See Also

DCluster, boot, stone.stat

Examples

```
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

niter<-100
```

```
#All Tests are performed around county 78.


#Permutation  model
st.perboot<-boot(sids, statistic=stone.boot, R=niter, region=78)
plot(st.perboot)#Display results

#Multinomial model
st.mboot<-boot(sids, statistic=stone.pboot, sim="parametric",
  ran.gen=multinom.sim,  R=niter, region=78)
plot(st.mboot)#Display results

#Poisson model
st.pboot<-boot(sids, statistic=stone.pboot, sim="parametric",
  ran.gen=poisson.sim,  R=niter, region=78)
plot(st.pboot)#Display results

#Poisson-Gamma model
st.pgboot<-boot(sids, statistic=stone.pboot, sim="parametric",
  ran.gen=negbin.sim, R=niter, region=78)
plot(st.pgboot)#Display results
```

`stone.stat` *Compute Stone's Statistic*

Description

Calculate Stone's statistic. See *stone* manual page for details.

stone.stat computes the test statistic and the test using a hi-square distribution whilst stone.test performs a bootstrap test.

Usage

```
stone.stat(data, region, sorted=FALSE, lambda)
stone.test(formula, data, model, R, ...)
```

Arguments

`formula`	Formula that specifies the underlying model. The observed cases are the response and the expected number of cases must be specified as an offset in the log scale (see example below). Note that now it is not necessary to use Observed and Expected and that any other names can be used to specify the observed and expected cases.
`model`	Parametric model to be used in the bootstrap test. One of "param", "multinom", "poisson" or "negbin". See the **DCluster** manpage for details.

...	The remaining arguments in 'stone.stat' not included in 'stone.test'. This is done so because stone.test calls stone.stat in order to perform the test.
R	Number of replicates used in the test to compute the significance of the observed value of the test statistic.
data	A dataframe containing the data, as specified in the **DCluster** manpage.
region	Region where around which we want to test for a cluster. It must a row number of *data*.
sorted	Whether the data are already sorted by distance to *region*.
lambda	Value of the null hypotheses. It may NULL (i. e., not known) or a number.

Value

A vector of two elements with the value of the statistic and the region (counting from the centre) where it was achieved.

References

Stone, R. A. (1988). Investigating of excess environmental risks around putative sources: Statistical problems and a proposed test. Statistics in Medicine 7,649-660.

See Also

DCluster

Examples

```
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

#Compute Stone's statistic around Anson county
region<-which(row.names(nc.sids)=="Robeson")
stone.stat(sids, region=region, lambda=1)


stone.test(Observed~offset(log(Expected)), sids, model="poisson", R=99,
   region=region, lambda=1)
```

tango *Tango's Statistic for General Clustering*

Description

Tango's statistic to perform a general clustering test is expressed as follows:

$$T = (r - p)' A (r - p)$$

where $r' = [O_1/O_+, \ldots, O_n/O_+]$, $p' = [E_1/E_+, \ldots, E_n/E_+]$ and A is a matrix of closeness which measures the cloneness between two zones (the higher the closer).

Tango proposes to take $A_{ij} = \exp\{-D_{ij}/\phi\}$, where D_{ij} is the distance between centroids of regions *i* and *j*, and ϕ is a constant that measures how strong is the relationship between regions in a general way.

References

Tango, Toshiro (1995). A Class of Tests for Detecting 'General' and 'Focused' Clustering of Rare Diseases. Statistics in Medicine 14, 2323-2334.

See Also

DCluster, tango.stat, tango.boot, tango.pboot

tango.boot *Generate Bootstrap Replicated of Tango's Statistic*

Description

Generate bootstrap replicated of Tango's statistic for general clustering, by means of function *boot* from *boot* library. Notice that these functions should not be used separately but as argument *statistic* when calling function *boot*.

tango.boot is used when performing non-parametric bootstrap.

tango.pboot must be used for parametric bootstrap.

Usage

```
tango.boot(data, i, ...)
tango.pboot(...)
```

Arguments

`data`	Dataframe with the data as described in *DCluster*.
`i`	Permutation generated by the non-parametric boostrap procedure.
`...`	Additional arguments passed when performing a bootstrap.

Value

Both functions return the value of the statistic.

References

Tango, Toshiro (1995). A Class of Tests for Detecting 'General' and 'Focused' Clustering of Rare Diseases. Statistics in Medicine 14, 2323-2334.

See Also

DCluster, boot, tango, tango.stat

Examples

```
library(boot)
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74) )
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)


#Calculate neighbours based on distance
coords<-as.matrix(sids[,c("x", "y")])

dlist<-dnearneigh(coords, 0, Inf)
dlist<-include.self(dlist)
dlist.d<-nbdists(dlist, coords)

#Calculate weights. They are globally standardised but it doesn't
#change significance.
col.W.tango<-nb2listw(dlist, glist=lapply(dlist.d, function(x) {exp(-x)}),
        style="C")

niter<-100


#Permutation model
tn.boot<-boot(sids, statistic=tango.boot, R=niter, listw=col.W.tango,
zero.policy=TRUE)
plot(tn.boot)#Display results

#Multinomial model
tn.mboot<-boot(sids, statistic=tango.pboot, sim="parametric",
ran.gen=multinom.sim,  R=niter, listw=col.W.tango, zero.policy=TRUE)

plot(tn.mboot)#Display results

#Poisson model
tn.pboot<-boot(sids, statistic=tango.pboot, sim="parametric",
```

```
ran.gen=poisson.sim,  R=niter, listw=col.W.tango, zero.policy=TRUE)

plot(tn.pboot)#Display results

#Poisson-Gamma model
tn.pgboot<-boot(sids, statistic=tango.pboot, sim="parametric",
ran.gen=negbin.sim, R=niter, listw=col.W.tango, zero.policy=TRUE)
plot(tn.pgboot)#Display results
```

`tango.stat` *Compute Tango's Statistic for General Clustering*

Description

Compute Tango's statistic for general clustering. See *tango* manual page for details.

tango.stat computes the test statistic and the test using a hi-square distribution whilst tango.test performs a bootstrap test.

Usage

```
tango.stat(data, listw, zero.policy)
tango.test(formula, data, model, R, ...)
```

Arguments

`formula`	Formula that specifies the underlying model. The observed cases are the response and the expected number of cases must be specified as an offset in the log scale (see example below). Note that now it is not necessary to use Observed and Expected and that any other names can be used to specify the observed and expected cases.
`model`	Parametric model to be used in the bootstrap test. One of "param", "multinom", "poisson" or "negbin". See the **DCluster** manpage for details.
`...`	The remaining arguments in 'tango.stat' not included in 'tango.test'. This is done so because tango.test calls tango.stat in order to perform the test.
`R`	Number of replicates used in the test to compute the significance of the observed value of the test statistic.
`data`	A dataframe containing the data, as specified in the **DCluster** manpage.
`listw`	Neighbours list with spatial weights created, for example, by 'nb2listw' (package *spdep*).
`zero.policy`	See *nb2listw* in package *spdep*.

References

Tango, Toshiro (1995). A Class of Tests for Detecting 'General' and 'Focused' Clustering of Rare Diseases. Statistics in Medicine 14, 2323-2334.

See Also

DCluster, tango, tango.boot, tango.pboot

Examples

```
library(spdep)
data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74) )
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

#Calculate neighbours based on distance
coords<-as.matrix(sids[,c("x", "y")])

dlist<-dnearneigh(coords, 0, Inf)
dlist<-include.self(dlist)
dlist.d<-nbdists(dlist, coords)

#Calculate weights. They are globally standardised but it doesn't
#change significance.
col.W.tango<-nb2listw(dlist, glist=lapply(dlist.d, function(x) {exp(-x)}),
        style="C")

#use exp(-D) as closeness matrix
tango.stat(sids, col.W.tango, zero.policy=TRUE)

tango.test(Observed~offset(log(Expected)), sids, model="poisson", R=99,
   list=col.W.tango, zero.policy=TRUE)
```

`Tests for Overdispertion`

Likelihood Ratio Test and Dean's Tests for Overdispertion

Description

When working with count data, the assumption of a Poisson model is common. However, sometimes the variance of the data is significantly higher that their mean which means that the assumption of that data have been drawn from a Poisson distribution is wrong.

In that case is is usually said that data are overdispersed and a better model must be proposed. A good choice is a Negative Binomial distribution (see, for example, `negative.binomial`.

Tests for overdispersion available in this package are the Likelihood Ratio Test (LRT) and Dean's P_B and P'_B tests.

Usage

```
test.nb.pois(x.nb, x.glm)
DeanB(x.glm, alternative="greater")
DeanB2(x.glm, alternative="greater")
```

Arguments

`x.nb`	Fitted Negative Binomial.
`x.glm`	Fitted Poisson model.
`alternative`	Alternative hipothesis to be tested. It can be "less", "greater" or "two.sided", although the usual choice will often be "greater".

Details

The LRT is computed to compare a fitted Poisson model against a fitted Negative Binomial model.

Dean's P_B and P'_B tests are score tests. These two tests were proposed for the case in which we look for overdispersion of the form $var(Y_i) = \mu_i(1 + \tau\mu_i)$, where $E(Y_i) = \mu_i$. See Dean (1992) for more details.

Value

An object of type *htest* with the results (p-value, etc.).

References

Dean, C.B. (1992), Testing for overdispersion in Poisson and binomial regression models, *J. Amer. Statist. Assoc.* 87, 451-457.

See Also

DCluster, achisq.stat, pottwhit.stat, negative.binomial (MASS), glm.nb (MASS)

Examples

```
library(spdep)
library(MASS)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74))
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

x.glm<-glm(Observed~1+offset(log(sids$Expected)), data=sids, family=poisson())
x.nb<-glm.nb(Observed~1+offset(log(Expected)), data=sids)

print(test.nb.pois(x.nb, x.glm))
print(DeanB(x.glm))
print(DeanB2(x.glm))
```

whittermore *Whittermore's Statistic*

Description

Whittermore's statistic is defined as follows:

$$W = \frac{n-1}{n} r^{'} Dr$$

where $r^{'} = [O_1/O_+, \ldots, O_n/O_+]$ and D is a distance matrix between the centroids of the areas.

It can be used to assess whether the region under study tends to cluster or not.

References

Whittermore, A. S. and Friend, N. and Byron, W. and Brown, J. R. and Holly, E. A. (1987). A test to detect clusters of disease. Biometrika 74, 631-635.

See Also

DCluster, whittermore.stat, whittermore.boot, whittermore.pboot

whittermore.boot *Generate Bootstrap Replicates of Whittermore's Statistic*

Description

Generate bootstrap replicates of Whittermore's statistic by means of function *boot* from *boot* library. Notice that these functions should not be used separately but as argument *statistic* when calling function *boot*.

whittermore.boot is used to perform a non-parametric bootstrap

whittermore.pboot is used when using parametric bootstrap.

Usage

```
whittermore.boot(data, i, ...)
whittermore.pboot(...)
```

Arguments

data	A dataframe with the data as explained in *DCluster*.
i	Permutation generated by the non-parametric bootstrap procedure.
...	Additional arguments passed when performing a bootstrap.

Value

Both functions return the value of the statistic.

References

Whittermore, A. S. and Friend, N. and Byron, W. and Brown, J. R. and Holly, E. A. (1987). A test to detect clusters of disease. Biometrika 74, 631-635.

See Also

DCluster, boot, whittermore, whittermore.stat

Examples

```
library(boot)
library(spdep)

data(nc.sids)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74) )
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

#Calculate neighbours based on distance
coords<-as.matrix(sids[,c("x", "y")])

dlist<-dnearneigh(coords, 0, Inf)
dlist<-include.self(dlist)
dlist.d<-nbdists(dlist, coords)

#Calculate weights. They are globally standardised but it doesn't
#change significance.
col.W.whitt<-nb2listw(dlist, glist=dlist.d, style="C")

niter<-100

#Permutation model
wt.boot<-boot(sids, statistic=whittermore.boot, R=niter, listw=col.W.whitt,
zero.policy=TRUE)
plot(wt.boot)#Display results

#Multinomial model
wt.mboot<-boot(sids, statistic=whittermore.pboot, sim="parametric",
ran.gen=multinom.sim,  R=niter,  listw=col.W.whitt, zero.policy=TRUE)

plot(wt.mboot)#Display results

#Poisson model
wt.pboot<-boot(sids, statistic=whittermore.pboot, sim="parametric",
ran.gen=poisson.sim,  R=niter,  listw=col.W.whitt, zero.policy=TRUE)

plot(wt.pboot)#Display results
```

```
#Poisson-Gamma model
wt.pgboot<-boot(sids, statistic=whittermore.pboot, sim="parametric",
ran.gen=negbin.sim, R=niter, listw=col.W.whitt, zero.policy=TRUE)
plot(wt.pgboot)#Display results
```

`whittermore.stat` *Compute Whittermore's Statistic*

Description

Compute Whittermore's statistic. See *whittermore* manual page for more details.

whittermore.stat computes the test statistic and the test using a hi-square distribution whilst whittermore.test performs a bootstrap test.

Usage

```
whittermore.stat(data, listw, zero.policy=FALSE)
whittermore.test(formula, data, model, R, ...)
```

Arguments

`formula`	Formula that specifies the underlying model. The observed cases are the response and the expected number of cases must be specified as an offset in the log scale (see example below). Note that now it is not necessary to use Observed and Expected and that any other names can be used to specify the observed and expected cases.
`model`	Parametric model to be used in the bootstrap test. One of "param", "multinom", "poisson" or "negbin". See the **DCluster** manpage for details.
`...`	The remaining arguments in 'whittermore.stat' not included in 'whittermore.test'. This is done so because whittermore.test calls whittermore.stat in order to perform the test.
`R`	Number of replicates used in the test to compute the significance of the observed value of the test statistic.
`data`	A dataframe containing the data, as specified in the **DCluster** manpage.
`listw`	Neighbours list with spatial weights created, for example, by 'nb2listw' (package *spdep*).
`zero.policy`	See *nb2listw* in package *spdep*.

Value

The value of the statistic.

References

Whittermore, A. S. and Friend, N. and Byron, W. and Brown, J. R. and Holly, E. A. (1987). A test to detect clusters of disease. Biometrika 74, 631-635.

See Also

DCluster, whittermore, whittermore.boot, whittermore.pboot

Examples

```
library(spdep)
data(nc.sids)
col.W <- nb2listw(ncCR85.nb, zero.policy=TRUE)

sids<-data.frame(Observed=nc.sids$SID74)
sids<-cbind(sids, Expected=nc.sids$BIR74*sum(nc.sids$SID74)/sum(nc.sids$BIR74) )
sids<-cbind(sids, x=nc.sids$x, y=nc.sids$y)

#Calculate neighbours based on distance
coords<-as.matrix(sids[,c("x", "y")])

dlist<-dnearneigh(coords, 0, Inf)
dlist<-include.self(dlist)
dlist.d<-nbdists(dlist, coords)

#Calculate weights. They are globally standardised but it doesn't
#change significance.
col.W.whitt<-nb2listw(dlist, glist=dlist.d, style="C")


whittermore.stat(sids, col.W.whitt, zero.policy=TRUE)

whittermore.test(Observed~offset(log(Expected)), sids, model="poisson", R=99,
   listw=col.W.whitt, zero.policy=TRUE)
```

Index