Package ‘nnspat’

July 22, 2025

Type Package

Title Nearest Neighbor Methods for Spatial Patterns

Version 0.1.2

Author Elvan Ceyhan

Maintainer Elvan Ceyhan <`elvanceyhan@gmail.com`>

Description Contains the functions for testing the spatial patterns (of segregation, spatial symmetry, association, disease clustering, species correspondence, and reflexivity) based on nearest neighbor relations,
especially using contingency tables such as
nearest neighbor contingency tables (Ceyhan (2010) <doi:10.1007/s10651-008-0104-x> and
Ceyhan (2017) <doi:10.1016/j.jkss.2016.10.002> and references therein),
nearest neighbor symmetry contingency tables (Ceyhan (2014) <doi:10.1155/2014/698296>),
species correspondence contingency tables and reflexivity contingency tables (Ceyhan (2018)
<doi:10.2436/20.8080.02.72> for two (or higher) dimensional data.
The package also contains functions for generating patterns of segregation, association,
uniformity in a multi-class setting (Ceyhan (2014) <doi:10.1007/s00477-013-0824-9>),
and various non-random labeling patterns for disease clustering
in two dimensional cases (Ceyhan (2014)
<doi:10.1002/sim.6053>), and for visualization of all these patterns
for the two dimensional data.
The tests are usually (asymptotic) normal z-tests or chi-square tests.

License GPL-2

Encoding UTF-8

LazyData TRUE

Imports MASS, stats, graphics, pcds, Rdpack (>= 0.7)

RdMacros Rdpack

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-12-19 17:40:07 UTC

Contents

`nnspat-package` *nnspat: A package for NN Methods and Their Use in Testing Spatial Patterns*

Description

`nnspat` is a package for computation of spatial pattern tests based on NN relations and generation of various spatial patterns.

Details

The `nnspat` package contains the functions for segregation/association tests based on nearest neighbor contingency tables (NNCTs), and tests for species correspondence, NN symmetry and reflexivity based on the corresponding contingency tables and functions for generating patterns of segregation, association, uniformity, and various non-random labeling (non-RL) patterns for data in two (or more) dimensions. Applications include plant ecology, disease clustering in epidemiology, and so on. #' See (Dixon (1994); Ceyhan (2010, 2017)).

The `nnspat` functions

The `nnspat` functions can be grouped as Auxiliary Functions, NNCT Functions, SCCT Functions, RCT Functions, NN-Symmetry Functions, and the Pattern (Generation) Functions.

Auxiliary Functions

Contains the auxiliary functions used in NN methods, such as indices of NNs, number of shared NNs, Q, R and T values, and so on. In all these functions the data sets are either matrices or data frames.

NNCT Functions

Contains the functions for testing segregation/association using the NNCT. The types of the tests are cell-specific tests, class-specific tests, and overall tests of segregation. See (Ceyhan (2009, 2010)).

SCCT Functions

Contains the functions used for testing species correspondence using the NNCT. The types are NN self and self-sum tests and the overall test of species correspondence. See (Ceyhan (2018)).

RCT Functions

Contains the functions for testing reflexivity using the reflexivity contingency table (RCT). The types are NN self reflexivity and NN mixed-non reflexivity. See (Ceyhan and Bahadir (2017); Bahadir and Ceyhan (2018)).

Symmetry Functions

Contains the functions for testing NN symmetry using the NNCT and Q-symmetry contingency table. The types are NN symmetry and symmetry in shared NN structure. See (Ceyhan (2014)).

Pattern Functions

Contains the functions for generating and visualization of spatial patterns of segregation, association, uniformity clustering and non-RL. See (Ceyhan (2014, 2014)).

References

Bahadir S, Ceyhan E (2018). “On the Number of reflexive and shared nearest neighbor pairs in one-dimensional uniform data.” *Probability and Mathematical Statistics*, **38(1)**, 123-137.

Ceyhan E (2009). “Class-Specific Tests of Segregation Based on Nearest Neighbor Contingency Tables.” *Statistica Neerlandica*, **63(2)**, 149-182.

Ceyhan E (2010). “On the use of nearest neighbor contingency tables for testing spatial segregation.” *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Ceyhan E (2010). “Exact Inference for Testing Spatial Patterns by Nearest Neighbor Contingency Tables.” *Journal of Probability and Statistical Science*, **8(1)**, 45-68.

Ceyhan E (2010). “New Tests of Spatial Segregation Based on Nearest Neighbor Contingency Tables.” *Scandinavian Journal of Statistics*, **37(1)**, 147-165.

Ceyhan E (2010). “Directional clustering tests based on nearest neighbour contingency tables.” *Journal of Nonparametric Statistics*, **22(5)**, 599-616.

Ceyhan E (2014). “Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations.” *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Ceyhan E (2014). “Segregation indices for disease clustering.” *Statistics in Medicine*, **33(10)**, 1662-1684.

Ceyhan E (2014). “Simulation and characterization of multi-class spatial patterns from stochastic point processes of randomness, clustering and regularity.” *Stochastic Environmental Research and Risk Assessment (SERRA)*, **38(5)**, 1277-1306.

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

Ceyhan E (2018). “A contingency table approach based on nearest neighbor relations for testing self and mixed correspondence.” *SORT-Statistics and Operations Research Transactions*, **42(2)**, 125-158.

Ceyhan E, Bahadir S (2017). “Nearest Neighbor Methods for Testing Reflexivity.” *Environmental and Ecological Statistics*, **24(1)**, 69-108.

Dixon PM (1994). “Testing spatial segregation using a nearest-neighbor contingency table.” *Ecology*, **75(7)**, 1940-1948.

`.onAttach` *.onAttach start message*

Description

.onAttach start message

Usage

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

Arguments

`libname`	defunct
`pkgname`	defunct

Value

invisible()

`.onLoad` *.onLoad getOption package settings*

Description

.onLoad getOption package settings

Usage

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

Arguments

`libname`	defunct
`pkgname`	defunct

Value

invisible()

Examples

```
getOption("nnspat.name")
```

`aij.theta` *Closeness or Proximity Matrix for Tango's Spatial Clustering Tests*

Description

This function computes the $A = a_{ij}(\theta)$ matrix useful in calculations for Tango's test $T(\theta)$ for spatial (disease) clustering (see Eqn (2) of Tango (2007). Here, $A = a_{ij}(\theta)$ is any matrix of a measure of the closeness between two points i and j with $aii = 0$ for all $i = 1, \ldots, n$, and $\theta = (\theta_1, \ldots, \theta_p)^t$ denotes the unknown parameter vector related to cluster size and $\delta = (\delta_1, \ldots, \delta_n)^t$, where $\delta_i = 1$ if z_i is a case and 0 otherwise. The test is then

$$T(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{n} \delta_i \delta_j a_{ij}(\theta) = \delta^t A(\theta) \delta$$

where $A = a_{ij}(\theta)$.

$T(\theta)$ becomes Cuzick and Edwards T_k tests statistic (Cuzick and Edwards (1990)), if $a_{ij} = 1$ if z_j is among the kNNs of z_i and 0 otherwise. In this case $\theta = k$ and `aij.theta` becomes `aij.mat` (more specifically, `aij.mat(dat,k)` and `aij.theta(dat,k,model="NN")`.

In Tango's exponential clinal model (Tango (2000)), $a_{ij} = \exp\left(-4\left(\frac{d_{ij}}{\theta}\right)^2\right)$ if $i \neq j$ and 0 otherwise, where θ is a predetermined scale of cluster such that any pair of cases far apart beyond the distance θ cannot be considered as a cluster and d_{ij} denote the Euclidean distance between two points i and j.

In the exponential model (Tango (2007)), $a_{ij} = \exp\left(-\frac{d_{ij}}{\theta}\right)$ if $i \neq j$ and 0 otherwise, where θ and d_{ij} are as above.

In the hot-spot model (Tango (2007)), $a_{ij} = 1$ if $d_{ij} \leq \theta$ and $i \neq j$ and 0 otherwise, where θ and d_{ij} are as above.

The argument `model` has four options, `NN`, `exp.clinal`, `exponential`, and `hot.spot`, with `exp.clinal` being the default. And the `theta` argument specifies the scale of clustering or the clustering parameter in the particular spatial disease clustering model.

See also (Tango (2007)) and the references therein.

Usage

```
aij.theta(dat, theta, model = "exp.clinal", ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`theta`	A predetermined cluster scale so that any pair of cases farther apart then the distance θ is unlikely to be cluster.
`model`	Type of Tango's spatial clustering model with four options: `NN`, `exp.clinal` (default), `exponential`, and `hot.spot`.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

The $A = a_{ij}(\theta)$ matrix useful in calculations for Tango's test $T(\theta)$.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

Tango T (2000). "A test for spatial disease clustering adjusted for multiple testing." *Statistics in Medicine*, **19**, 191-204.

Tango T (2007). "A class of multiplicity adjusted tests for spatial clustering based on case-control point data." *Biometrics*, **63**, 119-127.

See Also

`aij.mat`, `aij.nonzero` and `ceTk`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
k<-3#1 #try also 2,3

#aij for CE's Tk
Aij<-aij.theta(Y,k,model = "NN")
Aij2<-aij.mat(Y,k)
sum(abs(Aij-Aij2)) #check equivalence of aij.theta and aij.mat with model="NN"

Aij<-aij.theta(Y,k,method="max")
Aij2<-aij.mat(Y,k)
range(Aij-Aij2)

theta=.2
aij.theta(Y,theta,model = "exp.clinal")
aij.theta(Y,theta,model = "exponential")
aij.theta(Y,theta,model = "hot.spot")
```

`asycovTkTl` *Asymptotic Covariance between T_k and T_l Values*

Description

This function computes the asymptotic covariance between T_k and T_l values which is used in the computation of the asymptotic variance of Cuzick and Edwards T_{comb} test, which is a linear combination of some T_k tests. The limit is as n_1 goes to infinity.

The argument, n_1, is the number of cases (denoted as `n1` as an argument). The number of cases are denoted as n_1 and number of controls as n_0 in this function to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

The logical argument `nonzero.mat` (default=`TRUE`) is for using the A matrix if `FALSE` or just the matrix of nonzero locations in the A matrix (if `TRUE`) in the computations.

See page 80 of (Cuzick and Edwards (1990)) for more details.

Usage

```
asycovTkTl(dat, n1, k, l, nonzero.mat = TRUE, ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`n1`	Number of cases
`k, l`	Integers specifying the number of NNs (of subjects i and m in $a_{ij}(k)a_{mj}(l)$).

`nonzero.mat`	A logical argument (default is `TRUE`) to determine whether the A matrix or the matrix of nonzero locations of the A matrix will be used in the computation of N_s and N_t. If `TRUE` the nonzero location matrix is used, otherwise the A matrix itself is used.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the asymptotic covariance between T_k and T_l values.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`covTkTl`, `covTcomb`, and `Ntkl`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
n1<-sum(cls==1)

k<-1 #try also 2,3 or sample(1:5,1)
l<-1 #try also 2,3 or sample(1:5,1)
c(k,l)

asycovTkTl(Y,n1,k,l)
asycovTkTl(Y,n1,k,l,nonzero.mat = FALSE)
asycovTkTl(Y,n1,k,l,method="max")
```

`asyvarTk` *Asymptotic Variance of Cuzick and Edwards T_k Test statistic*

Description

This function computes the asymptotic variance of Cuzick and Edwards T_k test statistic based on the number of cases within kNNs of the cases in the data.

The argument, n_1, is the number of cases (denoted as `n1` as an argument). The number of cases are denoted as n_1 and number of controls as n_0 in this function to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

The logical argument `nonzero.mat` (default=`TRUE`) is for using the A matrix if `FALSE` or just the matrix of nonzero locations in the A matrix (if `TRUE`) for computing N_s and N_t, which are required in the computation of the asymptotic variance. N_s and N_t are defined on page 78 of (Cuzick and Edwards (1990)) as follows. $N_s = \sum_i \sum_j a_{ij} a_{ji}$ (i.e., number of ordered pairs for which kNN relation is symmetric) and $N_t = \sum \sum_{i \neq l} \sum a_{ij} a_{lj}$ (i.e, number of triplets (i, j, l) i, j, and l distinct so that j is among kNNs of i and j is among kNNs of l). For the A matrix, see the description of the functions `aij.mat` and `aij.nonzero`.

See (Cuzick and Edwards (1990)) for more details.

Usage

```
asyvarTk(dat, n1, k, nonzero.mat = TRUE, ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`n1`	Number of cases
`k`	Integer specifying the number of NNs (of subject i)
`nonzero.mat`	A logical argument (default is `TRUE`) to determine whether the A matrix or the matrix of nonzero locations of the A matrix will be used in the computation of N_s and N_t. If `TRUE` the nonzero location matrix is used, otherwise the A matrix itself is used.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

A `list` with the elements

`asy.var`	The asymptotic variance of Cuzick and Edwards T_k test statistic for disease clustering
`Ns`	The N_s value standing for the number of ordered pairs for which kNN relation is symmetric, see the description.
`Nt`	The N_t value standing for the number of triplets (i, j, l) i, j, and l distinct so that j is among kNNs of i and j is among kNNs of l see the description.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`ceTk`, `varTk`, and `varTkaij`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
n1<-sum(cls==1)
k<-3 #try also 2,3

asyvarTk(Y,n1,k)
asyvarTk(Y,n1,k,nonzero.mat=FALSE)
asyvarTk(Y,n1,k,method="max")
```

`bvnorm.pdf` *pdf of the Bivariate Normal Distribution*

Description

Computes the value of the probability density function (i.e., density) of the bivariate normal distribution at the specified point `X`, with mean `mu` and standard deviations of the first and second components being s_1 and s_2 (denoted as `s1` and `s2` in the arguments of the function, respectively) and correlation between them being `rho` (i.e., the covariance matrix is $\Sigma = S$ where $S_{11} = s_1^2$, $S_{22} = s_2^2$, $S_{12} = S_{21} = s_1 s_2 rho$).

Usage

```
bvnorm.pdf(X, mu = c(0, 0), s1 = 1, s2 = 1, rho = 0)
```

Arguments

`X`	A set of 2D points of size n (i.e an $n \times 2$ matrix or array) at which the density of the bivariate normal distribution is to be computed.
`mu`	A 1×2 `vector` of real numbers representing the mean of the bivariate normal distribution, default=$(0, 0)$.
`s1, s2`	The standard deviations of the first and second components of the bivariate normal distribution, with default is `1` for both
`rho`	The correlation between the first and second components of the bivariate normal distribution with default=0.

Value

The value of the probability density function (i.e., density) of the bivariate normal distribution at the specified point `X`, with mean `mu` and standard deviations of the first and second components being s_1 and s_2 and correlation between them being `rho`.

Author(s)

Elvan Ceyhan

See Also

mvrnorm

Examples

```
mu<-c(0,0)
s1<-1
s2<-1
rho<-.5

n<-5
Xp<-cbind(runif(n),runif(n))
bvnorm.pdf(Xp,mu,s1,s2,rho)
```

`cellsTij` *Entries for the Types I-IV cell-specific tests*

Description

Returns a `matrix` of same dimension as, `ct`, whose entries are the values of the Types I-IV cell-specific test statistics, $T^I_{ij} - T^{IV}_{ij}$. The row and column names are inherited from `ct`. The type argument specifies the type of the cell-specific test among the types I-IV tests. Equivalent to the function `tct` in this package.

See also (Ceyhan (2017)) and the references therein.

Usage

```
cellsTij(ct, type = "III")
```

Arguments

`ct`	A nearest neighbor contingency table
`type`	The type of the cell-specific test, default=`"III"`. Takes on values `"I"`-`"IV"` (or equivalently `1-4`, respectively.

Value

A `matrix` of the values of Type I-IV cell-specific tests

Author(s)

Elvan Ceyhan

References

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`tct` and `nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

type.lab<-c("I","II","III","IV")
for (i in 1:4)
{ print(paste("T_ij values for cell specific tests for type",type.lab[i]))
  print(cellsTij(ct,i))
}

cellsTij(ct,"II")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)
cellsTij(ct,2)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
cellsTij(ct,2)

ct<-matrix(c(0,10,5,5),ncol=2)
cellsTij(ct,2)
```

`ceTk` *Cuzick and Edwards T_k Test statistic*

Description

This function computes Cuzick and Edwards T_k test statistic based on the number of cases within kNNs of the cases in the data.

For disease clustering, Cuzick and Edwards (1990) suggested a k-NN test based on number of cases among k NNs of the case points. Let z_i be the i^{th} point and d_i^k be the number cases among k NNs of z_i. Then Cuzick-Edwards' k-NN test is $T_k = \sum_{i=1}^n \delta_i d_i^k$, where $\delta_i = 1$ if z_i is a case, and 0 if z_i is a control.

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0's and 1's accordingly. Also, T_1 is identical to the count for cell $(1, 1)$ in the nearest neighbor contingency table (NNCT) (See the function `nnct` for more detail on NNCTs).

See also (Ceyhan (2014); Cuzick and Edwards (1990)) and the references therein.

Usage

```
ceTk(dat, cc.lab, k = 1, case.lab = NULL, ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`cc.lab`	Case-control labels, 1 for case, 0 for control
`k`	Integer specifying the number of NNs (of subject i), default is `1`.
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Cuzick and Edwards T_k test statistic for disease clustering

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

Tcomb, seg.ind, Pseg.coeff and ceTkinv

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))

ceTk(Y,cls)
ceTk(Y,cls,method="max")
ceTk(Y,cls,k=3)
ceTk(Y,cls+1,case.lab = 2)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ceTk(Y,fcls,case.lab="a") #try also ceTk(Y,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  # here ceTk(Y,cls) gives an error message
```

`ceTkinv` *Cuzick and Edwards T_k^inv Test statistic*

Description

This function computes Cuzick and Edwards T_k^{inv} test statistic based on the sum of number of cases closer to each case than the k-th nearest control to the case.

T_k^{inv} test statistic is an extension of the run length test allowing a fixed number of controls in the run sequence.

T_k^{inv} test statistic is defined as $T_k^{inv} = \sum_{i=1}^n \delta_i \nu_i^k$ where $\delta_i = 1$ if z_i is a case, and 0 if z_i is a control and ν_i^k is the number of cases closer to the index case than the k nearest control, i.e., number of cases encountered beginning at z_i until k-th control is encountered.

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0's and 1's accordingly.

Usage

```
ceTkinv(dat, k, cc.lab, case.lab = NULL, ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`k`	Integer specifying the number of the closest controls to subject i.
`cc.lab`	Case-control labels, 1 for case, 0 for control
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

A `list` with two elements

`Tkinv`	Cuzick and Edwards T_k^{inv} test statistic for disease clustering
`run.vec`	The `vector` of number of cases till the `k`-th control for each point in the data set

Author(s)

Elvan Ceyhan

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

`ceTrun`, `ceTk`, and `Tcomb`

Examples

```
n<-20
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
cls
k<-2 #also try 3,4

ceTkinv(Y,k,cls)
ceTkinv(Y,k,cls+1,case.lab = 2)
ceTkinv(Y,k,cls,method="max")

ceTrun(Y,cls)
ceTkinv(Y,k=1,cls)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ceTkinv(Y,k,fcls,case.lab="a") #try also ceTrun(Y,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
```

```
cls<-sample(1:4,n,replace = TRUE)  #here ceTkinv(Y,k,cls) #gives error
```

---

ceTrun *Cuzick and Edwards T_run Test statistic*

Description

This function computes Cuzick and Edwards T_{run} test statistic based on the sum of the number of successive cases from each cases until a control is encountered in the data for detecting rare large clusters.

T_{run} test statistic is defined as $T_{run} = \sum_{i=1}^{n} \delta_i d_i^r$ where $\delta_i = 1$ if z_i is a case, and 0 if z_i is a control and d_i^r is the number successive cases encountered beginning at z_i until a control is encountered.

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0's and 1's accordingly.

See also (Cuzick and Edwards (1990)) and the references therein.

Usage

```
ceTrun(dat, cc.lab, case.lab = NULL, ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`cc.lab`	Case-control labels, 1 for case, 0 for control
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

A `list` with two elements

`Trun`	Cuzick and Edwards T_{run} test statistic for disease clustering
`run.vec`	The `vector` of number of consecutive cases till the first control for each point in the data set

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

ceTk, Tcomb and ceTkinv

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))

ceTrun(Y,cls)
ceTrun(Y,cls,method="max")
ceTrun(Y,cls+1,case.lab = 2)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ceTrun(Y,fcls,case.lab="a") #try also ceTrun(Y,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #here ceTrun(Y,cls) #gives an error message
```

`cov.nnct` *Covariance Matrix of the Cell Counts in an NNCT*

Description

Returns the covariance matrix of cell counts N_{ij} for $i, j = 1, \ldots, k$ in the NNCT, `ct`. The covariance matrix is of dimension $k^2 \times k^2$ and its entries are $cov(N_{ij}, N_{kl})$ when N_{ij} values are by default corresponding to the row-wise vectorization of `ct`. If `byrow=FALSE`, the column-wise vectorization of `ct` is used. These covariances are valid under RL or conditional on Q and R under CSR.

See also (Dixon (1994, 2002); Ceyhan (2010, 2017)).

Usage

```
cov.nnct(ct, varN, Q, R, byrow = TRUE)
```

Arguments

`ct`	A nearest neighbor contingency table
`varN`	The $k \times k$ variance matrix of cell counts of NNCT, `ct`.
`Q`	The number of shared NNs
`R`	The number of reflexive NNs (i.e., twice the number of reflexive NN pairs)
`byrow`	A logical argument (default=`TRUE`). If `TRUE`, rows of `ct` are appended to obtain the vector and if `FALSE` columns of `ct` are appended to obtain the vector.

Value

The $k^2 \times k^2$ covariance matrix of cell counts N_{ij} for $i, j = 1, \ldots, k$ in the NNCT, `ct`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). “On the use of nearest neighbor contingency tables for testing spatial segregation.” *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

Dixon PM (1994). “Testing spatial segregation using a nearest-neighbor contingency table.” *Ecology*, **75(7)**, 1940-1948.

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

See Also

`covNrow2col`, `cov.tct`, and `cov.nnsym`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)

cov.nnct(ct,varN,Qv,Rv)
cov.nnct(ct,varN,Qv,Rv,byrow=FALSE)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
```

```
varN<-var.nnct(ct,Qv,Rv)

cov.nnct(ct,varN,Qv,Rv)
cov.nnct(ct,varN,Qv,Rv,byrow=FALSE)

#1D data points
n<-20  #or try sample(1:20,1)
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
cov.nnct(ct,varN,Qv,Rv)
```

`cov.nnsym` *Covariance Matrix of the Differences of the Off-Diagonal Cell Counts in an NNCT*

Description

Returns the covariance matrix of the differences of the cell counts, $N_{ij} - N_{ji}$ for $i, j = 1, \ldots, k$ and $i \neq j$, in the NNCT, `ct`. The covariance matrix is of dimension $k(k-1)/2 \times k(k-1)/2$ and its entries are $cov(N_{ij} - N_{ji}, N_{kl} - N_{lk})$ where the order of i, j for $N_{ij} - N_{ji}$ is as in the output of `ind.nnsym(k)`. These covariances are valid under RL or conditional on Q and R under CSR.

The argument `covN` is the covariance matrix of N_{ij} (concatenated rowwise).

See also (Dixon (1994); Ceyhan (2014)).

Usage

```
cov.nnsym(covN)
```

Arguments

`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT

Value

The $k(k-1)/2 \times k(k-1)/2$ covariance matrix of the differences of the off-diagonal cell counts $N_{ij} - N_{ji}$ for $i, j = 1, \ldots, k$ and $i \neq j$ in the NNCT, `ct`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Dixon PM (1994). "Testing spatial segregation using a nearest-neighbor contingency table." *Ecology*, **75(7)**, 1940-1948.

See Also

`var.nnsym`, `cov.tct`, `cov.nnct` and `cov.seg.coeff`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv) #default is byrow

cov.nnsym(covN)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

cov.nnsym(covN)
```

`cov.seg.coeff` *Covariance Matrix of Segregation Coefficients in a Multi-class Case*

Description

Returns the covariance matrix of the segregation coefficients in a multi-class case based on the NNCT, `ct`. The covariance matrix is of dimension $k(k+1)/2 \times k(k+1)/2$ and its entry i,j correspond to the entries in the rows i and j of the output of `ind.seg.coeff(k)`. The segregation coefficients in the multi-class case are the extension of Pielou's segregation coefficient for the two-class case. These covariances are valid under RL or conditional on Q and R under CSR.

The argument `covN` is the covariance matrix of N_{ij} (concatenated rowwise).

See also (Ceyhan (2014)).

Usage

```
cov.seg.coeff(ct, covN)
```

Arguments

`ct`	A nearest neighbor contingency table
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT

Value

The $k(k+1)/2$ x $k(k+1)/2$ covariance matrix of the segregation coefficients for the multi-class case based on the NNCT, `ct`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`seg.coeff`, `var.seg.coeff`, `cov.nnct` and `cov.nnsym`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
```

```
W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

cov.seg.coeff(ct,covN)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

cov.seg.coeff(ct,covN)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

cov.seg.coeff(ct,covN)
```

`cov.tct` *Covariance Matrix of the Entries of the Type I-IV TCTs*

Description

Returns the covariance matrix of the entries T_{ij} for $i, j = 1, \ldots, k$ in the TCT for the types I, III, and IV cell-specific tests. The covariance matrix is of dimension $k^2 \times k^2$ and its entries are $cov(T_{ij}, T_{kl})$ when T_{ij} values are by default corresponding to the row-wise vectorization of TCT. The argument covN must be the covariance matrix of N_{ij} values which are obtained from the NNCT by row-wise vectorization. The functions `cov.tctIII` and `cov.tct3` are equivalent. These covariances are valid under RL or conditional on Q and R under CSR.

See also (Ceyhan (2017)).

Usage

```
cov.tct(ct, covN, type = "III")
```

Arguments

`ct`	A nearest neighbor contingency table
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized cell counts of NNCT, `ct`.
`type`	The type of the cell-specific test, default=`"III"`. Takes on values `"I"`-`"IV"` (or equivalently `1-4`, respectively.

Value

The $k^2 \times k^2$ covariance matrix of the entries T_{ij} for $i, j = 1, \ldots, k$ in the Type I-IV TCTs

Author(s)

Elvan Ceyhan

References

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`cov.nnct` and `cov.nnsym`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

cov.tct(ct,covN,type=1)
cov.tct(ct,covN,type="I")
cov.tct(ct,covN,type="II")
cov.tct(ct,covN,type="III")
cov.tct(ct,covN,type="IV")
cov.tctI(ct,covN)

cov.tct(ct,covN)
cov.tctIII(ct,covN)
cov.tct3(ct,covN)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
```

```
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)

covN<-cov.nnct(ct,varN,Qv,Rv)

cov.tct(ct,covN,type=3)
cov.tct(ct,covN,type="III")

cov.tctIII(ct,covN)
cov.tct3(ct,covN)
```

`covNrow2col` *Conversion of the Covariance Matrix of the Row-wise Vectorized Cell Counts to Column-wise Vectorized Cell Counts in an NNCT*

Description

Converts the $k^2 \times k^2$ covariance matrix of row-wise vectorized cell counts N_{ij} for $i, j = 1, \ldots, k$ in the NNCT, `ct` to the covariance matrix of column-wise vectorized cell counts. In the output, the covariance matrix entries are $cov(N_{ij}, N_{kl})$ when N_{ij} values are corresponding to the column-wise vectorization of `ct`. These covariances are valid under RL or conditional on Q and R under CSR.

See also (Dixon (1994, 2002); Ceyhan (2010, 2017)).

Usage

```
covNrow2col(covN)
```

Arguments

`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized cell counts of NNCT, `ct`.

Value

The $k^2 \times k^2$ covariance matrix of column-wise vectorized cell counts N_{ij} for $i, j = 1, \ldots, k$ in the NNCT, `ct`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). “On the use of nearest neighbor contingency tables for testing spatial segregation.” *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

Dixon PM (1994). “Testing spatial segregation using a nearest-neighbor contingency table.” *Ecology*, **75(7)**, 1940-1948.

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

See Also

`cov.nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)

covNrow<-cov.nnct(ct,varN,Qv,Rv)
covNcol1<-cov.nnct(ct,varN,Qv,Rv,byrow=FALSE)
covNcol2<-covNrow2col(covNrow)

covNrow
covNcol1
covNcol2

all.equal(covNcol1,covNcol2)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
```

```
covNrow<-cov.nnct(ct,varN,Qv,Rv)
covNcol1<-cov.nnct(ct,varN,Qv,Rv,byrow=FALSE)
covNcol2<-covNrow2col(covNrow)

covNrow
covNcol1
covNcol2

all.equal(covNcol1,covNcol2)

#1D data points
n<-20  #or try sample(1:20,1)
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
cov.nnct(ct,varN,Qv,Rv)
```

`covTcomb` *Covariance matrix for T_k values in* `Tcomb`

Description

This function computes the covariance matrix for the T_k values used in the T_{comb} test statistics, which is a linear combination of some T_k tests.

The argument, n_1, is the number of cases (denoted as `n1` as an argument). The number of cases is denoted as n_1 to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

The argument `klist` is the `vector` of integers specifying the indices of the T_k values used in obtaining the T_{comb}.

The logical argument `nonzero.mat` (default=`TRUE`) is for using the A matrix if `FALSE` or just the matrix of nonzero locations in the A matrix (if `TRUE`) in the computations.

The logical argument `asy.cov` (default=`FALSE`) is for using the asymptotic covariance or the exact (i.e., finite sample) covariance for the vector of T_k values used in `Tcomb`. If `asy.cov=TRUE`, the asymptotic covariance is used, otherwise the exact covariance is used.

See page 87 of (Cuzick and Edwards (1990)) for more details.

Usage

```
covTcomb(dat, n1, klist, nonzero.mat = TRUE, asy.cov = FALSE, ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`n1`	Number of cases
`klist`	`list` of integers specifying the indices of the T_k values used in obtaining the T_{comb}.
`nonzero.mat`	A logical argument (default is `TRUE`) to determine whether the A matrix or the matrix of nonzero locations of the A matrix will be used in the computation of N_s and N_t. If `TRUE` the nonzero location matrix is used, otherwise the A matrix itself is used.
`asy.cov`	A logical argument (default is `FALSE`) to determine whether asymptotic or exact (i.e., finite sample) covariances between T_k and T_l values are to be used to obtain the entries of the covariance matrix. If `TRUE` the asymptotic covariance values are used, otherwise exact covariance values are used.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the covariance matrix for the T_k values used in `Tcomb`.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`asycovTkTl`, `covTcomb`, and `Ntkl`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
n1<-sum(cls==1)

kl<-sample(1:5,3) #try also sample(1:5,2)
kl
covTcomb(Y,n1,kl)
covTcomb(Y,n1,kl,method="max")
covTcomb(Y,n1,kl,nonzero.mat = FALSE)
```

```
covTcomb(Y,n1,kl,asy=TRUE)
```

`covTkTl` *Finite Sample Covariance between T_k and T_l Values*

Description

This function computes the exact (i.e., finite sample) covariance between T_k and T_l values which is used in the computation of the exact variance of Cuzick and Edwards T_{comb} test, which is a linear combination of some T_k tests.

The logical argument `nonzero.mat` (default=`TRUE`) is for using the A matrix if `FALSE` or just the matrix of nonzero locations in the A matrix (if `TRUE`) in the computations.

See page 80 of (Cuzick and Edwards (1990)) for more details.

Usage

```
covTkTl(dat, n1, k, l, nonzero.mat = TRUE, ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`n1`	Number of cases
`k, l`	Integers specifying the number of NNs (of subjects i and m in $a_{ij}(k)a_{mj}(l)$).
`nonzero.mat`	A logical argument (default is `TRUE`) to determine whether the A matrix or the matrix of nonzero locations of the A matrix will be used in the computation of N_s and N_t. If `TRUE` the nonzero location matrix is used, otherwise the A matrix itself is used.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the exact covariance between T_k and T_l values.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`asycovTkTl`, `covTcomb`, and `Ntkl`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
n1<-sum(cls==1)

k<-1 #try also 2,3 or sample(1:5,1)
l<-1 #try also 2,3 or sample(1:5,1)
c(k,l)

covTkTl(Y,n1,k,l)
covTkTl(Y,n1,k,l,method="max")
asycovTkTl(Y,n1,k,l)

covTkTl(Y,n1,k,l,nonzero.mat = FALSE)
asycovTkTl(Y,n1,k,l,nonzero.mat = FALSE)
```

`dist.std.data` *Interpoint Distance Matrix for Standardized Data*

Description

This function computes and returns the distance matrix computed by using the specified distance measure to compute the distances between the rows of a data matrix which is standardized row or column-wise. That is, the output is the interpoint distance (IPD) matrix of the rows of the given set of points `x` `dist` function in the `stats` package of the standard R distribution. The argument column is the logical argument (default=`TRUE`) to determine row-wise or column-wise standardization. If `TRUE` each column is divided by its standard deviation, else each row is divided by its standard deviation. This function is different from the `dist` function in the `stats` package. `dist` returns the distance matrix in a lower triangular form, and dist.std.data returns in a full matrix of distances of standardized data set. ... are for further arguments, such as `method` and `p`, passed to the `dist` function.

Usage

```
dist.std.data(x, column = TRUE, ...)
```

Arguments

`x`	A set of points in matrix or data frame form where points correspond to the rows.
`column`	A logical argument (default is `TRUE`) to determine whether standardization is row-wise or column-wise. If `TRUE` it is column-wise else row-wise standardization.
`...`	Additional parameters to be passed on the `dist` function.

Value

A distance matrix whose i, j-th entry is the distance between rows i and j of `x`, which is standardized row-wise or column-wise.

Author(s)

Elvan Ceyhan

See Also

`dist`, `ipd.mat`, and `ipd.mat.euc`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
range(ipd)

ipd2<-dist.std.data(Y) #distance of standardized data
range(ipd2)

ipd2<-dist.std.data(Y,method="max") #distance of standardized data
range(ipd2)

#############
Y<-matrix(runif(60,0,100),ncol=3)
ipd<-ipd.mat(Y)
range(ipd)

ipd2<-dist.std.data(Y) #distance of standardized data
range(ipd2)
```

`dist2full` *Converts a lower triangular distance matrix to a full distance matrix*

Description

Converts a lower triangular distance matrix to a full distance matrix with zeroes in the diagonal. The input is usually the result of the `dist` function in the `stats` package. This function is adapted from Everitt's book (Everitt (2004))

Usage

```
dist2full(dis)
```

Arguments

`dis`	A lower triangular matrix, resulting from the `dist` function in the `stats` package

Value

A square (symmetric) distance matrix with zeroes in the diagonal.

Author(s)

Elvan Ceyhan

References

Everitt BS (2004). *An R and S-Plus Companion to Multivariate Analysis.* Springer-Verlag, London, UK.

See Also

`dist`

Examples

```
#3D data points
n<-3
X<-matrix(runif(3*n),ncol=3)
dst<-dist(X)
dist2full(dst)
```

`euc.dist` *The Euclidean distance between two vectors, matrices, or data frames*

Description

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

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

Usage

```
euc.dist(x, y)
```

Arguments

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

Value

Euclidean distance between `x` and `y`

Author(s)

Elvan Ceyhan

References

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

See Also

`dist` from the base package `stats` and `Dist` from the package pcds

Examples

```
B<-c(1,0); C<-c(1/2,sqrt(3)/2);
euc.dist(B,C);
euc.dist(B,B);

x<-runif(10)
y<-runif(10)
euc.dist(x,y)

xm<-matrix(x,ncol=2)
ym<-matrix(y,ncol=2)
euc.dist(xm,ym)

euc.dist(xm,xm)

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

`EV.Nii` *Expected Values of the Self Entries in a Species Correspondence Contingency Table (SCCT)*

Description

Returns a `vector` of length k of expected values of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or the expected values of the diagonal entries N_{ii} in an NNCT. These expected values are valid under RL or CSR.

The argument `ct` can be either the NNCT or SCCT.

See also (Ceyhan (2018)).

Usage

```
EV.Nii(ct)
```

Arguments

`ct`	The NNCT or SCCT

Value

A `vector` of length k whose entries are the expected values of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or of the diagonal entries in an NNCT.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2018). “A contingency table approach based on nearest neighbor relations for testing self and mixed correspondence.” *SORT-Statistics and Operations Research Transactions*, **42(2)**, 125-158.

See Also

`scct` and `EV.nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

EV.Nii(ct)
ct<-scct(ipd,cls)
EV.Nii(ct)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

EV.Nii(ct)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
```

```
ct<-nnct(ipd,cls)

EV.Nii(ct)
ct<-scct(ipd,cls)
EV.Nii(ct)
```

`EV.nnct` *Expected Values of the Cell Counts in NNCT*

Description

Returns a `matrix` of same dimension as, `ct`, whose entries are the expected cell counts of the NNCT under RL or CSR. The class sizes given as the row sums of `ct` and the row and column names are inherited from `ct`.

See also (Dixon (1994); Ceyhan (2010)).

Usage

```
EV.nnct(ct)
```

Arguments

`ct`	A nearest neighbor contingency table

Value

A `matrix` of the expected values of cell counts in the NNCT.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). “On the use of nearest neighbor contingency tables for testing spatial segregation.” *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Dixon PM (1994). “Testing spatial segregation using a nearest-neighbor contingency table.” *Ecology*, **75(7)**, 1940-1948.

See Also

`nnct` and `EV.tct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

EV.nnct(ct)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)
EV.nnct(ct)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

EV.nnct(ct)

ct<-matrix(c(0,10,5,5),ncol=2)
EV.nnct(ct)
```

`EV.rct` *Expected Values of the Cell Counts in RCT*

Description

Returns a `matrix` of same dimension as the RCT, `rfct`, whose entries are the expected cell counts of the RCT under RL or CSR.

See also (Ceyhan and Bahadir (2017)).

Usage

```
EV.rct(rfct, nvec)
```

Arguments

`rfct`	An RCT
`nvec`	The `vector` of class sizes

Value

A `matrix` of the expected values of cell counts in the RCT.

Author(s)

Elvan Ceyhan

References

Ceyhan E, Bahadir S (2017). “Nearest Neighbor Methods for Testing Reflexivity.” *Environmental and Ecological Statistics*, **24(1)**, 69-108.

See Also

`rct`, `EV.nnct` and `EV.tct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)

nvec<-as.numeric(table(cls))
rfct<-rct(ipd,cls)
EV.rct(rfct,nvec)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
nvec<-as.numeric(table(fcls))
rfct<-rct(ipd,fcls)
EV.rct(rfct,nvec)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)

rfct<-rct(ipd,cls)
EV.rct(rfct,nvec)
```

`EV.Tcomb` *Expected Value for Cuzick & Edwards T_comb Test Statistic*

Description

This function computes the expected value of Cuzick & Edwards T_{comb} test statistic in disease clustering, where T_{comb} is a linear combination of some T_k tests.

The argument, n_1, is the number of cases (denoted as `n1` as an argument). The number of cases is denoted as n_1 to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

The argument `klist` is the `vector` of integers specifying the indices of the T_k values used in obtaining the T_{comb}.

The argument `sig` is the covariance matrix of the vector of T_k values used in `Tcomb`, and can be computed via the the `covTcomb` function.

See page 87 of (Cuzick and Edwards (1990)) for more details.

Usage

```
EV.Tcomb(n1, n, klist, sig)
```

Arguments

`n1`	Number of cases
`n`	A positive integer representing the number of points in the data set
`klist`	`list` of integers specifying the indices of the T_k values used in obtaining the T_{comb}.
`sig`	The covariance matrix of the vector of T_k values used in `Tcomb`

Value

Returns the expected value of the T_{comb} test statistic

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`Tcomb`, and `ZTcomb`

Examples

```
n<-20  #or try sample(1:20,1) #try also n<-50, 100
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
n1<-sum(cls==1)

kl<-sample(1:5,3) #try also sample(1:5,2)
kl
sig<-covTcomb(Y,n1,kl)
EV.Tcomb(n1,n,kl,sig)
```

`EV.tct` *Expected Values of the Types I-IV cell-specific tests*

Description

Returns a `matrix` of same dimension as, `ct`, whose entries are the expected values of the T_{ij} values which are the Types I-IV cell-specific test statistics (i.e., $T^I_{ij} - T^{IV}_{ij}$) under RL or CSR. The row and column names are inherited from `ct`. The type argument specifies the type of the cell-specific test among the types I-IV tests.

See also (Ceyhan (2017)) and the references therein.

Usage

```
EV.tct(ct, type = "III")
```

Arguments

`ct`	A nearest neighbor contingency table
`type`	The type of the cell-specific test, default=`"III"`. Takes on values `"I"`-`"IV"` (or equivalently `1-4`, respectively.

Value

A `matrix` of the expected values of Type I-IV cell-specific tests.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2017). "Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables." *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`EV.tctI`, `tct`, and `EV.nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

EV.tct(ct,2)
EV.tct(ct,"II")
```

```
EV.tctI(ct)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)
EV.tct(ct,2)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

EV.tct(ct,2)

ct<-matrix(c(0,10,5,5),ncol=2)
EV.tct(ct,2)
```

`EV.tctI` *Expected Values of the Type I cell-specific tests*

Description

Returns a `matrix` of same dimension as, `ct`, whose entries are the expected values of the Type I cell-specific test statistics, T^I_{ij}. The row and column names are inherited from `ct`. These expected values are valid under RL or CSR.

See also (Ceyhan (2017)) and the references therein.

Usage

```
EV.tctI(ct)
```

Arguments

`ct`	A nearest neighbor contingency table

Value

A `matrix` of the expected values of Type I cell-specific tests.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2017). "Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables." *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`EV.tct`, `tct` and `EV.nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

EV.tctI(ct)
```

`EV.Tkinv` *Expected Value of Cuzick and Edwards T_k^inv Test statistic*

Description

This function computes the expected value of Cuzick and Edwards T_k^{inv} test statistic which is based on the sum of number of cases closer to each case than the k-th nearest control to the case.

The number of cases are denoted as n_1 (denoted as `n1` as an argument) and number of controls as n_0 for both functions (denoted as `n0` as an argument), to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

See the function `ceTkinv` for the details of the T_k^{inv} test.

See (Cuzick and Edwards (1990)) and references therein.

Usage

```
EV.Tkinv(n1, n0, k)
```

Arguments

`n1, n0`	The number of cases and controls
`k`	Integer specifying the number of the closest controls to subject i.

Value

The expected value of Cuzick and Edwards T_k^{inv} test statistic for disease clustering

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`ceTkinv`, `ceTrun`, and `EV.Trun`

Examples

```
n1<-20
n0<-25
k<-2 #try also 2, 3

EV.Tkinv(n1,n0,k)

EV.Tkinv(n1,n0,k=1)
EV.Trun(n1,n0)
```

`exact.nnct` *Exact version of Pearson’s chi-square test on NNCTs*

Description

An object of class `"htest"` performing exact version of Pearson’s chi-square test on nearest neighbor contingency tables (NNCTs) for the RL or CSR independence for 2 classes. Pearson’s χ^2 test is based on the test statistic $\mathcal{X}^2 = \sum_{j=1}^{2}\sum_{i=1}^{2}(N_{ij} - \mu_{ij})^2/\mu_{ij}$, which has χ^2_1 distribution in the limit provided that the contingency table is constructed under the independence null hypothesis. The exact version of Pearson’s test uses the exact distribution of $\mathcal{X}^2$ rather than large sample χ^2 approximation. That is, for the one-sided alternative, we calculate the p-values as in the function `exact.pval1s`; and for the two-sided alternative, we calculate the p-values as in the function `exact.pval2s` with double argument determining the type of the correction.

This test would be equivalent to Fisher’s exact test `fisher.test` if the odds ratio=1 (which can not be specified in the current version), and the odds ratio for the RL or CSR independence null hypothesis is $\theta_0 = (n_1 - 1)(n_2 - 1)/(n_1 n_2)$ which is used in the function and the p-value and confidence interval computations are are adapted from `fisher.test`.

See Ceyhan (2014) for more details.

Usage

```
exact.nnct(
  ct,
  alternative = "two.sided",
  conf.level = 0.95,
  pval.type = "inc",
  double = FALSE
)
```

Arguments

`ct`	A 2×2 NNCT
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the odds ratio
`pval.type`	The type of the p-value correction for the exact test on the NNCT, default=`"inc"`. Takes on values `"inc"`, `"exc"`, `"mid"`, `"tocher"` (or equivalently `1-4`, respectively) for table inclusive, table-exclusive, mid-p-value, and Tocher corrected p-value, respectively.
`double`	A logical argument (default is `FALSE`) to determine whether type I or II correction should be applied to the two-sided p-value. Used only when `alternative="two.sided"`. If `TRUE` type I correction (for doubling the minimum of the one-sided p-value) is applied, otherwise, type II correction (using the probabilities for the more extreme tables) is applied.

Value

A `list` with the elements

`statistic`	The test statistic, it is `NULL` for this function
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the odds ratio in the 2×2 NNCT at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate, i.e., the observed odds ratio the 2×2 NNCT.
`null.value`	Hypothesized null value for the odds ratio in the 2×2 NNCT, which is $\theta_0 = (n_1 - 1)(n_2 - 1)/(n_1 n_2)$ for this function.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the contingency table, `ct`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

See Also

`fisher.test`, `exact.pval1s`, and `exact.pval2s`

Examples

```
n<-20
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

exact.nnct(ct)
fisher.test(ct)

exact.nnct(ct,alt="g")
fisher.test(ct,alt="g")

exact.nnct(ct,alt="l",pval.type = "mid")

#############
ct<-matrix(sample(10:20,9),ncol=3)
fisher.test(ct) #here exact.nnct(ct) gives error message, since number of classes > 2
```

`exact.pval1s` *p-value correction to the one-sided version of exact NNCT test*

Description

In using Fisher's exact test on the 2×2 nearest neighbor contingency tables (NNCTs) a correction may be needed for the p-value. For the one-sided alternatives, the probabilities of more extreme tables are summed up, including or excluding the probability of the table itself (or some middle way). Let the probability of the contingency table itself be $p_t = f(n_{11}|n_1, n_2, c_1; \theta_0)$ where $\theta_0 = (n_1-1)(n_2-1)/(n_1 n_2)$ which is the odds ratio under RL or CSR independence and f is the probability mass function of the hypergeometric distribution. For testing the one-sided alternative $H_o : \theta = \theta_0$ versus $H_a : \theta > \theta_0$, we consider the following four methods in calculating the p-value:

- [(i)] with $S = \{t : t \geq n_{11}\}$, we get the *table-inclusive version* which is denoted as $p^{>}_{inc}$,
- [(ii)] with $S = \{t : t > n_{11}\}$, we get the *table-exclusive version*, denoted as $p^{>}_{exc}$.
- [(iii)] Using $p = p^{>}_{exc} + p_t/2$, we get the *mid-p version*, denoted as $p^{>}_{mid}$.

- [(iv)] We can also use *Tocher corrected version* which is denoted as $p^{>}_{Toc}$ (see `tocher.cor` for details).

See (Ceyhan (2010)) for more details.

Usage

```
exact.pval1s(ptable, pval, type = "inc")
```

Arguments

`ptable`	Probability of the observed 2×2 NNCT under the null hypothesis using the hypergeometric distribution for Fisher's exact test.
`pval`	Table inclusive p-value for Fisher's exact test on the NNCT.
`type`	The type of the p-value correction for the one-sided exact test on the NNCT, default=`"inc"`. Takes on values `"inc"`, `"exc"`, `"mid"`, `"tocher"` (or equivalently `1-4`, respectively) for table inclusive, table-exclusive, mid-p-value, and Tocher corrected p-value, respectively.

Value

A modified p-value based on the correction specified in `type`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Exact Inference for Testing Spatial Patterns by Nearest Neighbor Contingency Tables." *Journal of Probability and Statistical Science*, **8(1)**, 45-68.

See Also

`exact.pval2s` and `tocher.cor`

Examples

```
ct<-matrix(sample(20:40,4),ncol=2)
ptab<-prob.nnct(ct)
pv<-.3
exact.pval1s(ptab,pv)
exact.pval1s(ptab,pv,type="exc")
exact.pval1s(ptab,pv,type="mid")
```

`exact.pval2s` *p-value correction to the two-sided version of exact NNCT test*

Description

In using Fisher's exact test on the 2×2 nearest neighbor contingency tables (NNCTs) a correction may be needed for the p-value. For the one-sided alternatives, the probabilities of more extreme tables are summed up, including or excluding the probability of the table itself (or some middle way).

There is additional complexity in p-values for the two-sided alternatives. A recommended method is adding up probabilities of the same size and smaller than the probability associated with the current table. Alternatively, one can double the one-sided p-value (see (Agresti (1992)).

Let the probability of the contingency table itself be $p_t = f(n_{11}|n_1, n_2, c_1; \theta_0)$ where $\theta_0 = (n_1 - 1)(n_2 - 1)/(n_1 n_2)$ which is the odds ratio under RL or CSR independence and f is the probability mass function of the hypergeometric distribution.

Type (I): For double the one-sided p-value, we propose the following four variants:

- [(i)] twice the minimum of p_{inc} for the one-sided tests, which is table-inclusive version for this type of two-sided test, and denoted as p^I_{inc},
- [(ii)] twice the minimum of p_{inc} minus twice the table probability p_t, which is table-exclusive version of this type of two-sided test, and denoted as p^I_{exc},
- [(iii)] table-exclusive version of this type of two-sided test plus p_t, which is mid-p-value for this test, and denoted as p^I_{midd},
- [(iv)]Tocher corrected version (see `tocher.cor` for details).

Type (II): For summing the p-values of more extreme —than that of the table— cases in both directions, the following variants are obtained. The p-value is $p = \sum_S f(t|n_1, n_2, c_1; \theta = 1)$ with

- [(i)] $S = \{t : f(t|n_1, n_2, c_1; \theta = 1) \leq p_t\}$, which is called *table-inclusive version*, p^{II}_{inc},
- [(ii)] the probability of the observed table is included twice, once for each side; that is $p = p^{II}_{inc} + p_t$, which is called *twice-table-inclusive version*, p^{II}_{tinc},
- [(iii)] table-inclusive minus p_t, which is referred as *table-exclusive version*, p^{II}_{exc},
- [(iv)] table-exclusive plus one-half the p_t, which is called *mid-p version*, p^{II}_{mid} and,
- [(v)]*Tocher corrected version*, p^{II}_{Toc}, is obtained as before.

See (Ceyhan (2010)) for more details.

Usage

```
exact.pval2s(ptable, pval, type = "inc", double = FALSE)
```

Arguments

`ptable`	Probability of the observed 2×2 NNCT under the null hypothesis using the hypergeometric distribution for Fisher's exact test.
`pval`	Table inclusive p-value for Fisher's exact test on the NNCT.
`type`	The type of the p-value correction for the two-sided exact test on the NNCT, default=`"inc"`. Takes on values `"inc"`, `"exc"`, `"mid"`, `"tocher"` (or equivalently `1-4`, respectively) for table inclusive, table-exclusive, mid-p-value, and Tocher corrected p-value, respectively.
`double`	A logical argument (default is `FALSE`) to determine whether type I or II correction should be applied to the two-sided p-value. If `TRUE` type I correction (for doubling the minimum of the one-sided p-value) is applied, otherwise, type II correction (using the probabilities for the more extreme tables) is applied.

Value

A modified p-value based on the correction specified in `type`.

Author(s)

Elvan Ceyhan

References

Agresti A (1992). "A Survey of Exact Inference for Contingency Tables." *Statistical Science*, **7(1)**, 131-153.

Ceyhan E (2010). "Exact Inference for Testing Spatial Patterns by Nearest Neighbor Contingency Tables." *Journal of Probability and Statistical Science*, **8(1)**, 45-68.

See Also

`exact.pval1s` and `tocher.cor`

Examples

```
ct<-matrix(sample(20:40,4),ncol=2)
ptab<-prob.nnct(ct)
pv<-.23
exact.pval2s(ptab,pv)
exact.pval2s(ptab,pv,type="exc")
exact.pval2s(ptab,pv,type="mid")
```

`funs.auxcovtct` *Auxiliary Functions for Computing Covariances Between Cell Counts in the TCT*

Description

Five functions: `cov.2cells`, `cov.cell.col`, `covNijCk`, `cov2cols` and `covCiCj`

These are auxiliary functions for computing covariances between entries in the TCT for the types I-IV cell-specific tests. The covariances between T_{ij} values for $i, j = 1, \ldots, k$ in the TCT require covariances between two cells in the NNCT, between a cell and column sum, and between two column sums in the NNCT. `cov.2cells` computes the covariance between two cell counts N_{ij} and N_{kl} in an NNCT, `cov.cell.col` and `covNijCk` are equivalent and they compute the covariance between cell count N_{ij} and sum of column k, C_k, `cov2cols` and `covCiCj` are equivalent and they compute the covariance between sums of two columns, C_i and C_j. The index arguments refer to which entry or column sum is intended in the NNCT. The argument `covN` must be the covariance between N_{ij} values which are obtained from NNCT by row-wise vectorization. These covariances are valid under RL or conditional on Q and R under CSR.

Usage

```
cov.2cells(i, j, k, l, ct, covN)

cov.cell.col(i, j, k, ct, covN)

covNijCk(i, j, k, ct, covN)

cov.2cols(i, j, ct, covN)

covCiCj(i, j, ct, covN)
```

Arguments

`i, j, k, l`	Indices of the cell counts or column sums whose covariance is to be computed. All four are needed for `cov.2cells` referring to cells (i, j) and (k, l); only three indices i, j, k are needed for `cov.cell.col` and `covNijCk` referring to cell (i, j) and column k; only two indices i, j are needed for `cov2cols` and `covCiCj` referring to columns i and j.
`ct`	A nearest neighbor contingency table
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized cell counts of NNCT, `ct`.

Value

`cov.2cells` returns the covariance between two cell counts N_{ij} and N_{kl} in an NNCT, `cov.cell.col` and `covNijCk` return the covariance between cell count N_{ij} and sum of column k, C_k, `cov2cols` and `covCiCj` return the covariance between sums of two columns, C_i and C_j.

Author(s)

Elvan Ceyhan

See Also

`cov.tct` and `cov.nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

cov.2cells(1,1,1,2,ct,covN)

cov.cell.col(2,2,1,ct,covN)
covNijCk(2,2,1,ct,covN)

cov.2cols(2,1,ct,covN)
covCiCj(2,1,ct,covN)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

cov.2cells(2,3,1,2,ct,covN)

cov.cell.col(1,1,2,ct,covN)
covNijCk(1,1,2,ct,covN)

cov.2cols(3,4,ct,covN)
covCiCj(3,4,ct,covN)
```

`funs.base.class.spec` *Base Class-specific Chi-square Tests based on NNCTs*

Description

Two functions: `base.class.spec.ct` and `base.class.spec`.

Both functions are objects of class `"classhtest"` but with different arguments (see the parameter list below). Each one performs class specific segregation tests due to Dixon for $k \geq 2$ classes. That is, each one performs hypothesis tests of deviations of entries in each row of NNCT from the expected values under RL or CSR for each row. Recall that row labels in the NNCT are base class labels. The test for each row i is based on the chi-squared approximation of the corresponding quadratic form and are due to Dixon (2002).

Each function yields the test statistic, p-value and `df` for each base class i, description of the alternative with the corresponding null values (i.e., expected values) for the row i, estimates for the entries in row i for $i = 1, \ldots, k$. The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis for each row is that the corresponding N_{ij} entries in row i are equal to their expected values under RL or CSR.

See also (Dixon (2002); Ceyhan (2009)) and the references therein.

Usage

```
base.class.spec.ct(ct, covN)

base.class.spec(dat, lab, ...)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `base.class.spec.ct` only
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT, `ct`; used in `base.class.spec.ct` only.
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `base.class.spec` only
`lab`	The `vector` of class labels (numerical or categorical), used in `base.class.spec` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `base.class.spec` only

Value

A `list` with the elements

`type`	Type of the class-specific test, which is `"base"` for this function
`statistic`	The `vector` of base class-specific test statistics

`stat.names`	Name of the test statistics
`p.value`	The `vector` of p-values for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is $k - 1$ for this function.
`estimate`	Estimates of the parameters, NNCT, i.e., matrix of the observed N_{ij} values which is the NNCT.
`null.value`	Matrix of hypothesized null values for the parameters which are expected values of the N_{ij} values in the NNCT.
`null.name`	Name of the null values
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `base.class.spec.ct` only
`data.name`	Name of the data set, `dat`, returned by `base.class.spec` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2009). “Class-Specific Tests of Segregation Based on Nearest Neighbor Contingency Tables.” *Statistica Neerlandica*, **63(2)**, 149-182.

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

See Also

`NN.class.spec.ct`, `NN.class.spec`, `class.spec.ct` and `class.spec`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

base.class.spec(Y,cls)
base.class.spec.ct(ct,covN)
base.class.spec(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
```

```
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

base.class.spec(Y,fcls)
base.class.spec.ct(ct,covN)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

base.class.spec(Y,cls)
base.class.spec.ct(ct,covN)
```

`funs.cell.spec.ss` *Pielou's Cell-specific Segregation Test with Normal Approximation (for Sparse Sampling)*

Description

Two functions: `cell.spec.ss.ct` and `cell.spec.ss`.

Both functions are objects of class `"cellhtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of equality of the expected values of the cell counts (i.e., entries) in the NNCT for $k \geq 2$ classes. Each test is appropriate (i.e., have the appropriate asymptotic sampling distribution) when that data is obtained by sparse sampling.

Each cell-specific segregation test is based on the normal approximation of the entries in the NNCT and are due to Pielou (1961).

Each function yields a contingency table of the test statistics, p-values for the corresponding alternative, expected values, lower and upper confidence levels, sample estimates (i.e., observed values) and null value(s) (i.e., expected values) for the N_{ij} values for $i, j = 1, 2, \ldots, k$ and also names of the test statistics, estimates, null values, the description of the test, and the data set used.

The null hypothesis is that all $E(N_{ij}) = n_i c_j / n$ where n_i is the sum of row i (i.e., size of class i) c_j is the sum of column j in the $k \times k$ NNCT for $k \geq 2$. In the output, the test statistic, p-value and the lower and upper confidence limits are valid only for (properly) sparsely sampled data.

See also (Pielou (1961); Ceyhan (2010)) and the references therein.

Usage

```
cell.spec.ss.ct(
  ct,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

cell.spec.ss(
  dat,
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `cell.spec.ss.ct` only
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the entries, N_{ij} in the NNCT
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `cell.spec.ss` only
`lab`	The `vector` of class labels (numerical or categorical), used in `cell.spec.ss` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `cell.spec.ss` only

Value

A `list` with the elements

`statistic`	The `matrix` of Z test statistics for cell-specific tests
`stat.names`	Name of the test statistics
`p.value`	The `matrix` of p-values for the hypothesis test for the corresponding alternative
`LCL`, `UCL`	Matrix of lower and upper confidence levels for the entries N_{ij} in the NNCT at the given confidence level `conf.level` and depends on the type of `alternative`.
`conf.int`	The confidence interval for the estimates, it is `NULL` here, since we provide the `UCL` and `LCL` in `matrix` form.
`cnf.lvl`	Level of the upper and lower confidence limits (i.e., conf.level) of the NNCT entries.
`estimate`	Estimates of the parameters, i.e., matrix of the NNCT entries of the $k \times k$ NNCT, Nij for i,j=1,2,...,k.

`est.name`, `est.name2`	Names of the estimates, former is a shorter description of the estimates than the latter.
`null.value`	Hypothesized null value for the expected values of the NNCT entries, E(Nij) for i,j=1,2,...,k.
`null.name`	Name of the null values
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `cell.spec.ss.ct` only
`data.name`	Name of the data set, `dat`, returned by `cell.spec.ss` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "On the use of nearest neighbor contingency tables for testing spatial segregation." *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Pielou EC (1961). "Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships." *Journal of Ecology*, **49(2)**, 255-269.

See Also

`cell.spec.ct` and `cell.spec`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

cell.spec.ss(Y,cls)
cell.spec.ss.ct(ct)
cell.spec.ss.ct(ct,alt="g")

cell.spec.ss(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

cell.spec.ss(Y,fcls)
cell.spec.ss.ct(ct)
```

```
#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

cell.spec.ss(Y,cls,alt="l")
cell.spec.ss.ct(ct)
cell.spec.ss.ct(ct,alt="l")
```

`funs.class.spec` *Class-specific Chi-square Tests based on NNCTs*

Description

Two functions: `class.spec.ct` and `class.spec`.

Both functions are objects of class `"classhtest"` but with different arguments (see the parameter list below). Each one performs class specific segregation tests for the rows if `type="base"` and columns if `type="NN"` for $k \geq 2$ classes. That is, each one performs hypothesis tests of deviations of entries in each row (column) of NNCT from the expected values under RL or CSR for each row (column) if `type="base"` (`"NN"`). Recall that row labels of the NNCT are base class labels and column labels in the NNCT are NN class labels. The test for each row (column) i is based on the chi-squared approximation of the corresponding quadratic form and are due to Dixon (2002) (Ceyhan (2009)).

The argument `covN` must be covariance of row-wise (column-wise) vectorization of NNCT if `type="base"` (`type="NN"`).

Each function yields the test statistic, p-value and `df` for each base class i, description of the alternative with the corresponding null values (i.e., expected values) for the row (column) i, estimates for the entries in row (column) i for $i = 1, \ldots, k$ if `type="base"` (`type="NN"`). The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis for each row (column) is that the corresponding N_{ij} entries in row (column) i are equal to their expected values under RL or CSR.

See also (Dixon (2002); Ceyhan (2009)) and the references therein.

Usage

```
class.spec.ct(ct, covN, type = "base")

class.spec(dat, lab, type = "base", ...)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `class.spec.ct` only
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT, `ct`; used in `class.spec.ct` only.
`type`	The type of the class-specific tests with default=`"base"`. Takes on values`"base"` for (Dixon's) base class-specific test and`"NN"` for NN class-specific test.
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `class.spec` only
`lab`	The `vector` of class labels (numerical or categorical), used in `class.spec` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `class.spec` only

Value

A `list` with the elements

`type`	Type of the class-specific test, which is `"base"` or `"NN"` for this function
`statistic`	The `vector` of class-specific test statistics
`stat.names`	Name of the test statistics
`p.value`	The `vector` of p-values for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is $k-1$ for base class-specific test and k for NN class-specific test.
`estimate`	Estimates of the parameters, NNCT, i.e., the matrix of the observed N_{ij} values for base class-specific test and transpose of the NNCT for the NN class-specific test.
`null.value`	The `matrix` of hypothesized null values for the parameters which are expected values of the N_{ij} values for the base class-specific test and transpose of this matrix for the NN-class specific test.
`null.name`	Name of the null values
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `class.spec.ct` only
`data.name`	Name of the data set, `dat`, returned by `class.spec` only

References

Ceyhan E (2009). "Class-Specific Tests of Segregation Based on Nearest Neighbor Contingency Tables." *Statistica Neerlandica*, **63(2)**, 149-182.

Dixon PM (2002). "Nearest-neighbor contingency table analysis of spatial segregation for several species." *Ecoscience*, **9(2)**, 142-151.

See Also

`base.class.spec.ct`, `base.class.spec`, `NN.class.spec.ct` and `NN.class.spec`

Examples

```
n<-20
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv) #default is byrow

class.spec(Y,cls)
class.spec(Y,cls,type="NN")

class.spec.ct(ct,covN)
class.spec.ct(ct,covN,type="NN")

class.spec(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

class.spec(Y,fcls)
class.spec(Y,fcls,type="NN")

class.spec.ct(ct,covN)
class.spec.ct(ct,covN,type="NN")

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

class.spec(Y,cls)
class.spec(Y,cls,type="NN")

class.spec.ct(ct,covN)
class.spec.ct(ct,covN,type="NN")
```

`funs.covNii` *Covariance Matrix of the Self Entries in a Species Correspondence Contingency Table (SCCT)*

Description

Two functions: `covNii.ct` and `covNii`.

Both functions return the covariance matrix of the self entries (i.e., first column entries) in a species correspondence contingency table (SCCT) but have different arguments (see the parameter list below). The covariance matrix is of dimension $k \times k$ and its entries are $cov(S_i, S_j)$ where S_i values are the entries in the first column of SCCT (recall that S_i equals diagonal entry N_{ii} in the NNCT). These covariances are valid under RL or conditional on Q and R under CSR.

The argument `ct` which is used in `covNii.ct` only, can be either the NNCT or SCCT. And the argument `Vsq` is the vector of variances of the diagonal entries N_{ii} in the NNCT or the self entries (i.e., the first column) in the SCCT.

See also (Ceyhan (2018)).

Usage

```
covNii.ct(ct, Vsq, Q, R)

covNii(dat, lab, ...)
```

Arguments

`ct`	The NNCT or SCCT, used in `covNii.ct` only
`Vsq`	The `vector` of variances of the diagonal entries N_{ii} in the NNCT or the self entries (i.e., the first column) in the SCCT, used in `covNii.ct` only
`Q`	The number of shared NNs, used in `covNii.ct` only
`R`	The number of reflexive NNs (i.e., twice the number of reflexive NN pairs), used in `covNii.ct` only
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `covNii` only
`lab`	The `vector` of class labels (numerical or categorical), used in `covNii` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function, used in `covNii` only

Value

A `vector` of length k whose entries are the variances of the self entries (i.e., first column) in a species correspondence contingency table (SCCT).

The $k \times k$ covariance matrix of cell counts S_i in the self (i.e., first) column of the SCCT or of the diagonal cell counts N_{ii} for $i = 1, \ldots, k$ in the NNCT.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2018). “A contingency table approach based on nearest neighbor relations for testing self and mixed correspondence.” *SORT-Statistics and Operations Research Transactions*, **42(2)**, 125-158.

See Also

`scct`, `cov.nnct`, `cov.tct` and `cov.nnsym`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)

vsq<-varNii.ct(ct,Qv,Rv)
covNii(Y,cls)
covNii.ct(ct,vsq,Qv,Rv)

covNii(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

covNii(Y,fcls)
covNii.ct(ct,vsq,Qv,Rv)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)

vsq<-varNii.ct(ct,Qv,Rv)
covNii(Y,cls)
```

```
covNii.ct(ct,vsq,Qv,Rv)
```

`funs.covtct` *Functions for Covariances of the Entries of the Types I, III and IV TCTs*

Description

Four functions: `cov.tctI`, `cov.tctIII`, `cov.tct3`, and `cov.tctIV`.

These functions return the covariances between entries in the TCT for the types I, III, and IV cell-specific tests in matrix form which is of dimension $k^2 \times k^2$. The covariance matrix entries are $cov(T_{ij}, T_{kl})$ when T_{ij} values are by default corresponding to the row-wise vectorization of TCT. The argument `CovN` must be the covariance between N_{ij} values which are obtained from the NNCT by row-wise vectorization. The functions `cov.tctIII` and `cov.tct3` are equivalent. These covariances are valid under RL or conditional on Q and R under CSR.

See also (Ceyhan (2017)).

Usage

```
cov.tctI(ct, CovN)

cov.tctIII(ct, CovN)

cov.tct3(ct, CovN)

cov.tctIV(ct, CovN)
```

Arguments

`ct`	A nearest neighbor contingency table
`CovN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized cell counts of NNCT, `ct`.

Value

Each of these functions returns a $k^2 \times k^2$ covariance matrix, whose entries are the covariances of the entries in the TCTs for the corresponding type I-IV cell-specific test. The row and column names are inherited from `ct`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`cov.tct` and `cov.nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

cov.tctI(ct,covN)
cov.tctIII(ct,covN)
cov.tctIV(ct,covN)
```

`funs.kNNdist` *Functions for the $k\hat{}th$ and* k *NN distances*

Description

Two functions: `kthNNdist` and `kNNdist`.

`kthNNdist` returns the distances between subjects and their k^{th} NNs. The output is an $n \times 2$ matrix where n is the data size and first column is the subject index and second column contains the corresponding distances to k^{th} NN subjects.

`kNNdist` returns the distances between subjects and their k NNs. The output is an $n \times (k+1)$ matrix where n is the data size and first column is the subject index and the remaining k columns contain the corresponding distances to k NN subjects.

Usage

```
kthNNdist(x, k, is.ipd = TRUE, ...)

kNNdist(x, k, is.ipd = TRUE, ...)
```

Arguments

x	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
k	Integer specifying the number of NNs (of subjects).

`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

`kthNNdist` returns an $n \times 2$ matrix where n is data size (i.e., number of subjects) and first column is the subject index and second column is the k^{th} NN distances.

`kNNdist` returns an $n \times (k+1)$ matrix where n is data size (i.e., number of subjects) and first column is the subject index and the remaining `k` columns contain the corresponding distances to `k` NN subjects.

Author(s)

Elvan Ceyhan

See Also

`NNdist` and `NNdist2cl`

Examples

```
#Examples for kthNNdist
#3D data points, gives NAs when n<=k
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
kthNNdist(ipd,3)
kthNNdist(Y,3,is.ipd = FALSE)
kthNNdist(ipd,5)
kthNNdist(Y,5,is.ipd = FALSE)
kthNNdist(Y,3,is.ipd = FALSE,method="max")

#1D data points
X<-as.matrix(runif(5)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(5) would not work
ipd<-ipd.mat(X)
kthNNdist(ipd,3)

#Examples for kNNdist
#3D data points, gives NAs if n<=k for n,n+1,...,kNNs
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
kNNdist(ipd,3)
kNNdist(ipd,5)
kNNdist(Y,5,is.ipd = FALSE)

kNNdist(Y,5,is.ipd = FALSE,method="max")
```

```
kNNdist(ipd,1)
kthNNdist(ipd,1)

#1D data points
X<-as.matrix(runif(5)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(5) would not work
ipd<-ipd.mat(X)
kNNdist(ipd,3)
```

`funs.kNNdist2cl` *Functions for the $k\hat{}th$ and* `k` *NN distances*

Description

Two functions: `kthNNdist2cl` and `kNNdist2cl`.

`kthNNdist2cl` returns the distances between subjects from class i and their k^{th} NNs from class j. The output is a `list` with first entry (`kth.nndist`) is an $n_i \times 3$ matrix where n_i is the size of class i and first column is the subject index for class i, second column is the index of the k^{th} NN of class i subjects among class j subjects and third column contains the corresponding k^{th} NN distances. The other entries in the `list` are labels of base class and NN class and the value of `k`, respectively.

`kNNdist2cl` returns the distances between subjects from class i and their `k` NNs from class j. The output is a `list` with first entry (`ind.knndist`) is an $n_i \times (k+1)$ matrix where n_i is the size of class i, first column is the indices of class i subjects, second to $(k+1)$-st columns are the indices of `k` NNs of class i subjects among class j subjects. The second `list` entry (`knndist`) is an $n_i \times k$ matrix where n_i is the size of class i and the columns are the `k`NN distances of class i subjects to class j subjects. The other entries in the `list` are labels of base class and NN class and the value of `k`, respectively.

The argument `within.class.ind` is a logical argument (default=`FALSE`) to determine the indexing of the class i subjects. If `TRUE`, index numbering of subjects is within the class, from 1 to class size (i.e., `1:n_i`), according to their order in the original data; otherwise, index numbering within class is just the indices in the original data.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

Usage

```
kthNNdist2cl(x, k, i, j, lab, within.class.ind = FALSE, is.ipd = TRUE, ...)

kNNdist2cl(x, k, i, j, lab, within.class.ind = FALSE, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).

`k`	Integer specifying the number of NNs (of subjects).
`i, j`	class label of base class and NN classes, respectively.
`lab`	The `vector` of class labels (numerical or categorical)
`within.class.ind`	A logical parameter (default=`FALSE`). If `TRUE`, index numbering of subjects is within the class, from 1 to class size (i.e., `1:n_i`), according to their order in the original data; otherwise, index numbering within class is just the indices in the original data.
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

`kthNNdist2cl` returns the `list` of elements

`kth.nndist`	$n_i \times 3$ matrix where n_i is the size of class i and first column is the subject index for class i, second column is the index of the `k`-th NN of class i subjects among class j subjects and third column contains the corresponding `k`-th NN distances, , returned by `Zseg.ind.ct` only
`base.class`	label of base class
`nn.class`	label of NN class
`k`	value of `k` in kNN

`kNNdist2cl` returns the `list` of elements

`ind.knndist`	$n_i \times (k+1)$ matrix where n_i is the size of class i, first column is the indices of class i subjects, 2nd to $(k+1)$-st columns are the indices of k NNs of class i subjects among class j subjects.
`knndist`	$n_i \times k$ matrix where n_i is the size of class i and the columns are the kNN distances of class i subjects to class j subjects.
`base.class`	label of base class
`nn.class`	label of NN class
`k`	value of `k` in kNN

Author(s)

Elvan Ceyhan

See Also

`NNdist2cl`, `kthNNdist`, and `kNNdist`

Examples

```
#Examples for kthNNdist2cl
#3D data points
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
#two class case
clab<-sample(1:2,n,replace=TRUE) #class labels
table(clab)
kthNNdist2cl(ipd,3,1,2,clab)
kthNNdist2cl(Y,3,1,2,clab,is.ipd = FALSE)
kthNNdist2cl(ipd,3,1,2,clab,within = TRUE)

#three class case
clab<-sample(1:3,n,replace=TRUE) #class labels
table(clab)
kthNNdist2cl(ipd,3,2,3,clab)

#1D data points
n<-15
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
#two class case
clab<-sample(1:2,n,replace=TRUE) #class labels
table(clab)
kthNNdist2cl(ipd,3,1,2,clab) # here kthNNdist2cl(ipd,3,1,12,clab)
#gives an error message

kthNNdist2cl(ipd,3,"1",2,clab)

#Examples for kNNdist2cl
#3D data points
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
#two class case
clab<-sample(1:2,n,replace=TRUE) #class labels
table(clab)
kNNdist2cl(ipd,3,1,2,clab)
kNNdist2cl(Y,3,1,2,clab,is.ipd = FALSE)

kNNdist2cl(ipd,3,1,2,clab,within = TRUE)

#three class case
clab<-sample(1:3,n,replace=TRUE) #class labels
table(clab)
kNNdist2cl(ipd,3,1,2,clab)

#1D data points
n<-15
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
```

```
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
#two class case
clab<-sample(1:2,n,replace=TRUE) #class labels
table(clab)

kNNdist2cl(ipd,3,1,2,clab)
kNNdist2cl(ipd,3,"1",2,clab) #here kNNdist2cl(ipd,3,"a",2,clab)
#gives an error message
```

`funs.overall.nnct` *Dixon's Overall Test of Segregation for NNCT*

Description

Two functions: `overall.nnct.ct` and `overall.nnct`.

Both functions are objects of class `"Chisqtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of deviations of cell counts from the expected values under RL or CSR for all cells (i.e., entries) combined in the NNCT. That is, each test is Dixon's overall test of segregation based on NNCTs for $k \geq 2$ classes. This overall test is based on the chi-squared approximation of the corresponding quadratic form and are due to Dixon (1994, 2002). Both functions exclude the last column of the NNCT (in fact any column will do and last column is chosen without loss of generality), to avoid ill-conditioning of the covariance matrix (for its inversion in the quadratic form).

Each function yields the test statistic, p-value and `df` which is $k(k-1)$, description of the alternative with the corresponding null values (i.e., expected values) of NNCT entries, sample estimates (i.e., observed values) of the entries in NNCT. The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis is that all N_{ij} entries are equal to their expected values under RL or CSR.

See also (Dixon (1994, 2002); Ceyhan (2010, 2017)) and the references therein.

Usage

```
overall.nnct.ct(ct, covN)

overall.nnct(dat, lab, ...)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `overall.nnct.ct` only
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT, `ct`; used in `overall.nnct.ct` only.
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `overall.nnct` only

`lab`	The `vector` of class labels (numerical or categorical), used in `overall.nnct` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `overall.nnct` only

Value

A `list` with the elements

`statistic`	The overall chi-squared statistic
`stat.names`	Name of the test statistic
`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is $k(k-1)$ for this function.
`estimate`	Estimates of the parameters, NNCT, i.e., matrix of the observed N_{ij} values which is the NNCT.
`est.name`, `est.name2`	Names of the estimates, former is a longer description of the estimates than the latter.
`null.value`	Matrix of hypothesized null values for the parameters which are expected values of the the N_{ij} values in the NNCT.
`null.name`	Name of the null values
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `overall.nnct.ct` only
`data.name`	Name of the data set, `dat`, returned by `overall.nnct` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). “On the use of nearest neighbor contingency tables for testing spatial segregation.” *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

Dixon PM (1994). “Testing spatial segregation using a nearest-neighbor contingency table.” *Ecology*, **75(7)**, 1940-1948.

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

See Also

`overall.seg.ct`, `overall.seg`, `overall.tct.ct` and `overall.tct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv) #default is byrow

overall.nnct(Y,cls)
overall.nnct.ct(ct,covN)

overall.nnct(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

overall.nnct(Y,fcls)
overall.nnct.ct(ct,covN)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

overall.nnct(Y,cls)
overall.nnct.ct(ct,covN)
```

`funs.overall.seg` *Overall Segregation Tests for NNCTs*

Description

Two functions: `overall.seg.ct` and `overall.seg`.

All functions are objects of class `"Chisqtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of deviations of cell counts from the expected values under RL or CSR for all cells (i.e., entries) combined in the NNCT or TCT. That is, each test is one of Dixon's or Types I-IV overall test of segregation based on NNCTs or TCTs for $k \geq 2$ classes. Each overall test is based on the chi-squared approximation of the corresponding quadratic form and are due to Dixon (1994, 2002) and to Ceyhan (2010, 2017), respectively. All functions exclude some row and/or column of the TCT, to avoid ill-conditioning of the covariance matrix of the NNCT (for its inversion in the quadratic form), see the relevant functions under See also section below.

The `type="dixon"` or `"nnct"` refers to Dixon's overall test of segregation, and `type="I"-"IV"` refers to types I-IV overall tests, respectively.

Each function yields the test statistic, p-value and `df` which is $k(k-1)$ for type II and Dixon's test and $(k-1)^2$ for the other types, description of the alternative with the corresponding null values (i.e., expected values) of TCT entries, sample estimates (i.e., observed values) of the entries in TCT. The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis is that all N_{ij} or T_{ij} entries for the specified type are equal to their expected values under RL or CSR, respectively.

See also (Dixon (1994, 2002); Ceyhan (2010, 2010)) and the references therein.

Usage

```
overall.seg.ct(ct, covN, type)

overall.seg(dat, lab, type, ...)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `overall.seg.ct` only
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT, `ct`; used in `overall.seg.ct` only.
`type`	The type of the overall test with no default. Takes on values `"dixon"` or `"nnct"` for Dixon's overall test and `"I"-"IV"` for types I-IV cell-specific test (or equivalently `1-6`, respectively).
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `overall.seg` only
`lab`	The `vector` of class labels (numerical or categorical), used in `overall.seg` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `overall.seg` only

Value

A `list` with the elements

`statistic`	The overall chi-squared statistic for the specified type
`stat.names`	Name of the test statistic

`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is $k(k-1)$ for type II and Dixon's tests and $(k-1)^2$ for others.
`estimate`	Estimates of the parameters, NNCT for Dixon's test and type I-IV TCT for others.
`est.name`, `est.name2`	Names of the estimates, former is a longer description of the estimates than the latter.
`null.value`	Matrix of hypothesized null values for the parameters which are expected values of the the N_{ij} values in the NNCT or T_{ij} values in the TCT.
`null.name`	Name of the null values
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `overall.seg.ct` only
`data.name`	Name of the data set, `dat`, returned by `overall.seg` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). “New Tests of Spatial Segregation Based on Nearest Neighbor Contingency Tables.” *Scandinavian Journal of Statistics*, **37(1)**, 147-165.

Ceyhan E (2010). “On the use of nearest neighbor contingency tables for testing spatial segregation.” *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

Dixon PM (1994). “Testing spatial segregation using a nearest-neighbor contingency table.” *Ecology*, **75(7)**, 1940-1948.

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

See Also

`overall.nnct.ct`, `overall.nnct`, `overall.tct.ct` and `overall.tct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
```

```
W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv) #default is byrow

type<-"dixon" #try also "nnct", I", "II", "III", and "IV"
overall.seg(Y,cls,type)
overall.seg(Y,cls,type,method="max")
overall.seg(Y,cls,type="I")

overall.seg.ct(ct,covN,type)
overall.seg.ct(ct,covN,type="I")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

overall.seg(Y,fcls,type="I")
overall.seg.ct(ct,covN,type)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

overall.seg(Y,cls,type="I")
overall.seg.ct(ct,covN,type)
```

`funs.overall.tct` *Types I-IV Overall Tests of Segregation for NNCT*

Description

Two functions: `overall.tct.ct` and `overall.tct`.

All functions are objects of class `"Chisqtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of deviations of cell counts from the expected values under RL or CSR for all cells (i.e., entries) combined in the TCT. That is, each test is one of Types I-IV overall test of segregation based on TCTs for $k \geq 2$ classes. This overall test is based on the chi-squared approximation of the corresponding quadratic form and are due to Ceyhan (2010, 2017). Both functions exclude some row and/or column of the TCT, to avoid ill-conditioning of the covariance matrix of the NNCT (for its inversion in the quadratic form). In particular, type-II removes the last column, and all other types remove the last row and column.

Each function yields the test statistic, p-value and `df` which is $k(k-1)$ for type II test and $(k-1)^2$ for the other types, description of the alternative with the corresponding null values (i.e., expected values) of TCT entries, sample estimates (i.e., observed values) of the entries in TCT. The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis is that all Tij entries for the specified type are equal to their expected values under RL or CSR.

See also (Ceyhan (2010, 2017)) and the references therein.

Usage

```
overall.tct.ct(ct, covN, type = "III")

overall.tct(dat, lab, type = "III", ...)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `overall.tct.ct` only
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT, `ct`; used in `overall.tct.ct` only.
`type`	The type of the overall segregation test, default=`"III"`. Takes on values `"I"`-`"IV"` (or equivalently `1-4`, respectively.
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `overall.tct` only
`lab`	The `vector` of class labels (numerical or categorical), used in `overall.tct` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `overall.tct` only

Value

A `list` with the elements

`statistic`	The overall chi-squared statistic for the specified type
`stat.names`	Name of the test statistic
`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is $k(k-1)$ for type=`"II"` and $(k-1)^2$ for others.
`estimate`	Estimates of the parameters, TCT, i.e., matrix of the observed T_{ij} values which is the TCT.
`est.name`, `est.name2`	Names of the estimates, former is a longer description of the estimates than the latter.

`null.value`	Matrix of hypothesized null values for the parameters which are expected values of the the T_{ij} values in the TCT.
`null.name`	Name of the null values
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `overall.tct.ct` only
`data.name`	Name of the data set, `dat`, returned by `overall.tct` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "New Tests of Spatial Segregation Based on Nearest Neighbor Contingency Tables." *Scandinavian Journal of Statistics*, **37(1)**, 147-165.

Ceyhan E (2017). "Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables." *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`overall.seg.ct`, `overall.seg`, `overall.nnct.ct` and `overall.nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv) #default is byrow

overall.tct(Y,cls)
overall.tct(Y,cls,type="I")
overall.tct(Y,cls,type="II")
overall.tct(Y,cls,type="III")
overall.tct(Y,cls,type="IV")

overall.tct(Y,cls,method="max")

overall.tct.ct(ct,covN)
overall.tct.ct(ct,covN,type="I")

#cls as a factor
na<-floor(n/2); nb<-n-na
```

```
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

overall.tct(Y,fcls)
overall.tct.ct(ct,covN)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

overall.tct(Y,cls)
overall.tct.ct(ct,covN)
```

`funs.pijPij` *The functions for probability of selecting a number of points from respective classes*

Description

The ancillary probability functions used in computation of the variance-covariance matrices of various NN spatial tests such as NNCT tests and tests based on other contingency tables. These functions can be classified as `pij` and `Pij` type functions. The `pij` functions are for individual probabilities and the corresponding `Pij` functions are the summed `pij` values. For example p_{iijk} is the probability of any 4 points with 2 from class i, and others are from classes j and k. These probabilities are for data from RL or CSR.

Usage

```
p11(k, n)

P11(nvec)

p12(k, l, n)

P12(nvec)

p111(k, n)

P111(nvec)
```

```
p1111(k, n)

P1111(nvec)

p112(k, l, n)

P112(nvec)

p122(k, l, n)

p123(k, l, m, n)

P123(nvec)

p1234(k, l, m, p, n)

P1234(nvec)

p1223(k, l, m, n)

p1123(k, l, m, n)

P1123(nvec)

p1122(k, l, n)

P1122(nvec)

p1112(k, l, n)

P1112(nvec)
```

Arguments

`k`, `l`, `m`, `p`	Positive integers, usually representing the class sizes, used in `pij` type functions only. Number of these arguments required depends on the number of distinct indices of p, e.g. p_{ij} requires `k,l,n` and p_{iijk} requires `k,l,m,n` as input.
`n`	A positive integer representing the size of the data set (i.e., number of observations in the data set).
`nvec`	A `vector` of positive integers representing the sizes of classes in the data set, used in `Pij` type functions only.

Value

Probability values for the selected points being from the indicated classes.

See Also

pk

`funs.scct` *Species Correspondence Contingency Table (SCCT)*

Description

Two functions: `scct.ct` and `scct`.

Both functions return the $k \times 2$ species correspondence contingency table (SCCT) but have different arguments (see the parameter list below).

SCCT is constructed by categorizing the NN pairs according to pair type as self or mixed. A base-NN pair is called a self pair, if the elements of the pair are from the same class; a mixed pair, if the elements of the pair are from different classes. Row labels in the RCT are the class labels and the column labels are `"self"` and `"mixed"`. The $k \times 2$ SCCT (whose first column is self column with entries S_i and second column is mixed with entries M_i) is closely related to the $k \times k$ nearest neighbor contingency table (NNCT) whose entries are N_{ij}, where $S_i = N_{ii}$ and $M_i = n_i - N_{ii}$ with n_i is the size of class i.

The function `scct.ct` returns the SCCT given the inter-point distance (IPD) matrix or data set `x`, and the function `scct` returns the SCCT given the IPD matrix. SCCT is a $k \times 2$ matrix where k is number of classes in the data set. (See Ceyhan (2018) for more detail, where SCCT is labeled as CCT for correspondence contingency table).

The argument `ties` is a logical argument (default=`FALSE` for both functions) to take ties into account or not. If `TRUE` a NN contributes $1/m$ to the NN count if it is one of the m tied NNs of a subject.

The argument nnct is a logical argument for `scct.ct` only (default=`FALSE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the $k \times k$ NNCT, and if `FALSE`, `x` is taken to be the IPD matrix.

The argument lab is the `vector` of class labels (default=`NULL` when `nnct=TRUE` in the function `scct.ct` and no default specified for scct).

Usage

```
scct.ct(x, lab = NULL, ties = FALSE, nnct = FALSE)

scct(dat, lab, ties = FALSE, ...)
```

Arguments

`x`	The IPD matrix (if `nnct=FALSE`) or the NNCT (if `nnct=TRUE`), used in `scct.ct` only
`lab`	The `vector` of class labels (numerical or categorical), default=`NULL` when `nnct=FALSE` in the function `scct.ct` and no default specified for scct.
`ties`	A logical argument (default=`FALSE`) to take ties into account or not. If `TRUE` a NN contributes $1/m$ to the NN count if it is one of the m tied NNs of a subject.

`nnct`	A logical parameter (default=`FALSE`). If `TRUE`, `x` is taken to be the $k \times k$ NNCT, and if `FALSE`, `x` is taken to be the IPD matrix, used in `scct.ct` only.
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `scct` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function, used in `scct` only

Value

Returns the $k \times 2$ SCCT where k is the number of classes in the data set.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2018). "A contingency table approach based on nearest neighbor relations for testing self and mixed correspondence." *SORT-Statistics and Operations Research Transactions*, **42(2)**, 125-158.

See Also

`nnct`, `tct`, `rct` and `Qsym.ct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
NNCT<-nnct(ipd,cls)
NNCT

scct(Y,cls)
scct(Y,cls,method="max")

scct.ct(ipd,cls)
scct.ct(ipd,cls,ties = TRUE)
scct.ct(NNCT,nnct=TRUE)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
scct.ct(ipd,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
```

```
NNCT<-nnct(ipd,cls)
NNCT

scct(Y,cls)

scct.ct(ipd,cls)
scct.ct(NNCT,nnct=TRUE)
```

`funs.seg.coeff` *Pielou's Segregation Coefficients for NNCTs*

Description

Two functions: `Pseg.coeff` and `seg.coeff`.

Each function computes segregation coefficients based on NNCTs. The function `Pseg.coeff` computes Pielou's segregation coefficient (Pielou (1961)) for the two-class case (i.e., based on 2×2 NNCTs) and `seg.coeff` is the extension of `Pseg.coeff` to the multi-class case (i.e., for $k \times k$ NNCTs with $k \geq 2$) and provides a $k \times k$ matrix of segregation coefficients (Ceyhan (2014)). Both functions use the same argument, `ct`, for NNCT.

Pielou's segregation coefficient (for two classes) is $S_P = 1 - (N_{12} + N_{21})/(E[N_{12}] + E[N_{21}])$ and the extended segregation coefficents (for $k \geq 2$ classes) are $S_c = 1 - (N_{ii})/(E[N_{ii}])$ for the diagonal cells in the NNCT and $S_c = 1 - (N_{ij} + N_{ji})/(E[N_{ij}] + E[N_{ji}])$ for the off-diagonal cells in the NNCT.

Usage

```
Pseg.coeff(ct)

seg.coeff(ct)
```

Arguments

`ct`	A nearest neighbor contingency table, used in both functions

Value

`Pseg.coeff` returns Pielou's segregation coefficient for 2×2 NNCT `seg.coeff` returns a $k \times k$ matrix of segregation coefficients (which are extended versions of Pielou's segregation coefficient)

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Segregation indices for disease clustering.” *Statistics in Medicine*, **33(10)**, 1662-1684.

Pielou EC (1961). “Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships.” *Journal of Ecology*, **49(2)**, 255-269.

See Also

`seg.ind`, `Zseg.coeff.ct` and `Zseg.coeff`

Examples

```
#Examples for Pseg.coeff
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct
Pseg.coeff(ct)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

Pseg.coeff(ct)

#############
ct<-matrix(sample(1:25,9),ncol=3)
#Pseg.coeff(ct)

#Examples for seg.coeff
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct
seg.coeff(ct)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

seg.coeff(ct)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
```

```
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)
ct<-nnct(ipd,cls)

seg.coeff(ct)
```

`funs.varNii` *Variances of the Self Entries in a Species Correspondence Contingency Table (SCCT)*

Description

Two functions: `varNii.ct` and `varNii`.

Both functions return a `vector` of length k of variances of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or the variances of the diagonal entries N_{ii} in an NNCT, but have different arguments (see the parameter list below). These variances are valid under RL or conditional on Q and R under CSR.

The argument `ct` which is used in `varNii.ct` only, can be either the NNCT or SCCT.

See also (Ceyhan (2018)).

Usage

```
varNii.ct(ct, Q, R)

varNii(dat, lab, ...)
```

Arguments

`ct`	The NNCT or SCCT, used in `varNii.ct` only
`Q`	The number of shared NNs, used in `varNii.ct` only
`R`	The number of reflexive NNs (i.e., twice the number of reflexive NN pairs), used in `varNii.ct` only
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `varNii` only
`lab`	The `vector` of class labels (numerical or categorical), used in `varNii` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function, used in `varNii` only

Value

A `vector` of length k whose entries are the variances of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or of the diagonal entries in an NNCT.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2018). "A contingency table approach based on nearest neighbor relations for testing self and mixed correspondence." *SORT-Statistics and Operations Research Transactions*, **42(2)**, 125-158.

See Also

`scct`, `var.nnct`, `var.tct`, `var.nnsym` and `covNii`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)

varNii(Y,cls)
varNii.ct(ct,Qv,Rv)

varNii(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

varNii(Y,fcls)
varNii.ct(ct,Qv,Rv)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)

varNii(Y,cls)
varNii.ct(ct,Qv,Rv)
```

`funs.vartct` *Functions for Variances of Cell Counts in the Types I, III and IV TCTs*

Description

Three functions: `var.tctI`, `var.tctIII`, and `var.tctIV`.

These functions return the variances of T_{ij} values for $i, j = 1, \ldots, k$ in the TCT in matrix form which is of the same dimension as TCT for types I, III and IV tests. The argument `covN` must be the covariance between N_{ij} values which are obtained from the NNCT by row-wise vectorization. These variances are valid under RL or conditional on Q and R under CSR.

See also (Ceyhan (2017)).

Usage

```
var.tctI(ct, covN)

var.tctIII(ct, covN)

var.tctIV(ct, covN)
```

Arguments

`ct`	A nearest neighbor contingency table
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized cell counts of NNCT, `ct`.

Value

Each of these functions returns a `matrix` of same dimension as, `ct`, whose entries are the variances of the entries in the TCT for the corresponding type of cell-specific test. The row and column names are inherited from `ct`.

References

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`var.tct` and `var.nnct`

`funsAijmat` *Aij matrices for computation of Moments of Cuzick and Edwards T_k Test statistic*

Description

Two functions: `aij.mat` and `aij.nonzero`.

The function `aij.mat` yields the $A = (a_{ij}(k))$ matrix where $a_{ij}(k) = 1$ if z_j is among the kNNs of z_i and 0 otherwise due to Tango (2007). This matrix is useful in calculation of the moments of Cuzick-Edwards T_k tests.

The function `aij.nonzero` keeps only nonzero entries, i.e., row and column entries where in each row, for the entry (r_1, c_1) r_1 is the row entry and c_1 is the column entry. Rows are from 1 to n, which stands for the data point or observation, and column entries are from 1 to `k`, where `k` is specifying the number of kNNs (of each observation) considered. This function saves in storage memory, but needs to be carefully unfolded in the functions to represent the actual the A matrix.

See also (Tango (2007)).

Usage

```
aij.mat(dat, k, ...)

aij.nonzero(dat, k, ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`k`	Integer specifying the number of NNs (of subject i), default is `1`.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

The function `aij.mat` returns the A_{ij} matrix for computation of moments of Cuzick and Edwards T_k Test statistic while the function `aij.nonzero` returns the (locations of the) non-zero entries in the A_{ij} matrix

Author(s)

Elvan Ceyhan

References

Tango T (2007). “A class of multiplicity adjusted tests for spatial clustering based on case-control point data.” *Biometrics*, **63**, 119-127.

See Also

`aij.theta` and `EV.Tkaij`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
k<-3 #try also 2,3

Aij<-aij.mat(Y,k)
Aij
Aij2<-aij.mat(Y,k,method="max")
range(Aij,Aij2)

apply(Aij,2,sum) #row sums of Aij

aij.nonzero(Y,k)
aij.nonzero(Y,k,method="max")
```

`funsC_MI_II` *Correction Matrices for the Covariance Matrix of NNCT entries*

Description

Two functions: `correct.cf1` and `correct.cf1`.

Each function yields matrices which are used in obtaining covariance matrices of T_{ij} values for types I and II tests from the usual Chi-Square test of contingency tables (i.e., Pielou's test) applied on NNCTs. The output matrices are to be term-by-term multiplied with the covariance matrix of the entries of NNCT. See Sections 3.1 and 3.2 in (Ceyhan (2010)) or Sections 3.5.1 and 3.5.2 in (Ceyhan (2008)) for more details.

Usage

```
correct.cf1(ct)

correct.cf2(ct)
```

Arguments

`ct`	A nearest neighbor contingency table

Value

Both functions return a correction matrix which is to be multiplied with the covariance matrix of entries of the NNCT so as to obtain types I and II overall tests from Pielou's test of segregation. See the description above for further detail.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2008). “New Tests for Spatial Segregation Based on Nearest Neighbor Contingency Tables.” https://arxiv.org/abs/0808.1409v3 [stat.ME]. Technical Report # KU-EC-08-6, Koç University, Istanbul, Turkey.

Ceyhan E (2010). “New Tests of Spatial Segregation Based on Nearest Neighbor Contingency Tables.” *Scandinavian Journal of Statistics*, **37(1)**, 147-165.

See Also

`nnct.cr1` and `nnct.cr2`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

#correction type 1
CM1<-correct.cf1(ct)
CovN.cf1<-covN*CM1

#correction type 2
CM2<-correct.cf2(ct)
CovN.cf2<-covN*CM2

covN
CovN.cf1
CovN.cf2
```

`funsExpTk` *Expected Value for Cuzick and Edwards T_k Test statistic*

Description

Two functions: `EV.Tk` and `EV.Tkaij`.

Both functions compute the expected value of Cuzick and Edwards T_k test statistic based on the number of cases within kNNs of the cases in the data under RL or CSR independence.

The number of cases are denoted as n_1 (denoted as `n1` as an argument) for both functions and number of controls as n_0 (denoted as `n0` as an argument) in `EV.Tk`, to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

The function `EV.Tkaij` uses Toshiro Tango's moments formulas based on the $A = (a_{ij})$ matrix (and is equivalent to the function `EV.Tk`, see Tango (2007), where $a_{ij}(k) = 1$ if z_j is among the kNNs of z_i and 0 otherwise.

See also (Ceyhan (2014)).

Usage

```
EV.Tk(k, n1, n0)

EV.Tkaij(k, n1, a)
```

Arguments

`k`	Integer specifying the number of NNs (of subject i).
`n1`, `n0`	The number of cases and controls, n_1 used for both functions, and n_0 used in `EV.Tk` only.
`a`	The $A = (a_{ij})$ matrix

Value

The expected value of Cuzick and Edwards T_k test statistic for disease clustering

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

Tango T (2007). "A class of multiplicity adjusted tests for spatial clustering based on case-control point data." *Biometrics*, **63**, 119-127.

See Also

`ceTk` and `EV.Tcomb`

Examples

```
n1<-20
n0<-25
k<-1 #try also 3, 5, sample(1:5,1)

EV.Tk(k,n1,n0)

###
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)
n1<-sum(cls==1)
n0<-sum(cls==0)
a<-aij.mat(Y,k)

EV.Tk(k,n1,n0)
EV.Tkaij(k,n1,a)
```

`funsExpTrun` *Expected Value for Cuzick and Edwards T_run Test statistic*

Description

Two functions: `EV.Trun` and `EV.Trun.alt`.

Both functions compute the expected value of Cuzick and Edwards T_{run} test statistic based on the number of consecutive cases from the cases in the data under RL or CSR independence.

The number of cases are denoted as n_1 (denoted as `n1` as an argument) and number of controls as n_0 for both functions (denoted as `n0` as an argument), to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

The function `EV.Trun.alt` uses a loop and takes slightly longer than the function `EV.Trun`, hence `EV.Trun` is used in other functions.

See also (Cuzick and Edwards (1990)).

Usage

```
EV.Trun(n1, n0)

EV.Trun.alt(n1, n0)
```

Arguments

`n1, n0`	The number of cases and controls used as arguments for both functions.

Value

The expected value of Cuzick and Edwards T_{run} test statistic for disease clustering

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`ceTrun` and `EV.Tk`

Examples

```
n1<-20
n0<-25

EV.Trun(n1,n0)
```

`funsNNclass.spec` *NN Class-specific Chi-square Tests based on NNCTs*

Description

Two functions: `NN.class.spec.ct` and `NN.class.spec`.

Both functions are objects of class `"classhtest"` but with different arguments (see the parameter list below). Each one performs class specific segregation tests for the columns, i.e., NN categories for $k \geq 2$ classes. That is, each one performs hypothesis tests of deviations of entries in each column of NNCT from the expected values under RL or CSR for each column. Recall that column labels in the NNCT are NN class labels. The test for each column i is based on the chi-squared approximation of the corresponding quadratic form and are due to Ceyhan (2009).

The argument `covN` must be covariance of column-wise vectorization of NNCT if the logical argument `byrow=FALSE` otherwise the function converts `covN` (which is done row-wise) to columnwise version with `covNrow2col` function.

Each function yields the test statistic, p-value and `df` for each base class i, description of the alternative with the corresponding null values (i.e., expected values) for the column i, estimates for the entries in column i for $i = 1, \ldots, k$. The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis for each column is that the corresponding N_{ij} entries in column i are equal to their expected values under RL or CSR.

See also (Dixon (2002); Ceyhan (2009)) and the references therein.

Usage

```
NN.class.spec.ct(ct, covN, byrow = TRUE)

NN.class.spec(dat, lab, ...)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `NN.class.spec.ct` only
`covN`	The $k^2 \times k^2$ covariance matrix of column-wise vectorized entries of NNCT, `ct`; used in `NN.class.spec.ct` only.
`byrow`	A logical argument (default=`TRUE`). If `TRUE`, rows of `ct` are appended to obtain the vector of N_{ij} values and `covN` is the covariance matrix for this vector and if `FALSE` columns of `ct` are appended to obtain the N_{ij} vector and `covN` is converted to the row-wise version by covNrow2col function;used in `NN.class.spec.ct` only.
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `NN.class.spec` only
`lab`	The `vector` of class labels (numerical or categorical), used in `NN.class.spec` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `NN.class.spec` only

Value

A `list` with the elements

`type`	Type of the class-specific test, which is `"NN"` for this function
`statistic`	The `vector` of NN class-specific test statistics
`stat.names`	Name of the test statistics
`p.value`	The `vector` of p-values for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is k for this function.
`estimate`	Estimates of the parameters, transpose of the NNCT, i.e., transpose of the matrix of the observed N_{ij} values which is the transpose of NNCT.
`null.value`	Transpose of the matrix of hypothesized null values for the parameters which are expected values of the N_{ij} values in the NNCT.
`null.name`	Name of the null values
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `NN.class.spec.ct` only
`data.name`	Name of the data set, `dat`, returned by `NN.class.spec` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2009). “Class-Specific Tests of Segregation Based on Nearest Neighbor Contingency Tables.” *Statistica Neerlandica*, **63(2)**, 149-182.

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

See Also

`base.class.spec.ct`, `base.class.spec`, `class.spec.ct` and `class.spec`

Examples

```
n<-20
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covNrow<-cov.nnct(ct,varN,Qv,Rv)
covNcol<-covNrow2col(covNrow)

NN.class.spec(Y,cls)
NN.class.spec(Y,cls,method="max")

NN.class.spec.ct(ct,covNrow)
NN.class.spec.ct(ct,covNcol,byrow = FALSE)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

NN.class.spec(Y,fcls)
NN.class.spec.ct(ct,covNrow)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covNrow<-cov.nnct(ct,varN,Qv,Rv)
```

```
covNcol<-covNrow2col(covNrow)

NN.class.spec(Y,cls)

NN.class.spec.ct(ct,covNrow)
NN.class.spec.ct(ct,covNcol,byrow = FALSE)
```

`funsN_I_II` *Correction Matrices for the NNCT entries*

Description

Two functions: `nnct.cr1` and `nnct.cr1`.

Each function yields matrices which are used in obtaining the correction term to be added to the usual Chi-Square test of contingency tables (i.e., Pielou's test) applied on NNCTs to obtain types I and II overall tests. The output contingency tables are to be row-wise vectorized to obtain N_I and N_{II} vectors. See Sections 3.1 and 3.2 in (Ceyhan (2010)) or Sections 3.5.1 and 3.5.2 in (Ceyhan (2008)) for more details.

Usage

```
nnct.cr1(ct)

nnct.cr2(ct)
```

Arguments

`ct`	A nearest neighbor contingency table

Value

Both functions return a $k \times k$ contingency table which is to be row-wise vectorized to obtain N_I and N_{II} vectors which are used in the correction summands to obtain types I and II overall tests from Pielou's test of segregation. See the description above for further detail.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2008). “New Tests for Spatial Segregation Based on Nearest Neighbor Contingency Tables.” `https://arxiv.org/abs/0808.1409v3` [stat.ME]. Technical Report # KU-EC-08-6, Koç University, Istanbul, Turkey.

Ceyhan E (2010). “New Tests of Spatial Segregation Based on Nearest Neighbor Contingency Tables.” *Scandinavian Journal of Statistics*, **37(1)**, 147-165.

See Also

`correct.cf1` and `correct.cf2`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

#correction type 1
ct1<-nnct.cr1(ct)

#correction type 2
ct2<-nnct.cr2(ct)

ct
ct1
ct2
```

`funsOnevsRest` *Functions for one versus rest type labeling*

Description

Two functions: `lab.onevsrest` and `classirest`.

Both functions relabel the points, keeping class i label as is and relabeling the other classes as "rest". Used in the one-vs-rest type comparisons after the overall segregation test is found to be significant.

See also (Ceyhan (2017)).

Usage

```
lab.onevsrest(i, lab)

classirest(i, lab)
```

Arguments

`i`	label of the class that is to be retained in the post-hoc comparison.
`lab`	The `vector` of class labels (numerical or categorical)

Value

Both functions return the data relabeled as class i label is retained and the remaining is relabeled as "rest".

Author(s)

Elvan Ceyhan

References

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`pairwise.lab`

Examples

```
n<-20  #or try sample(1:20,1)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
lab.onevsrest(1,cls)
classirest(2,cls)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
lab.onevsrest("a",fcls)
lab.onevsrest("b",fcls)
classirest("b",fcls)

#cls as a factor
fcls<-rep(letters[1:4],rep(10,4))
lab.onevsrest("b",fcls)
classirest("b",fcls)
```

`funsPseg.ss` *Pielou’s Overall Test of Segregation for NNCT (for Sparse Sampling)*

Description

Two functions: `Pseg.ss.ct` and `Pseg.ss`.

Both functions are objects of class `"Chisqtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of deviations of cell counts from the expected values under independence for all cells (i.e., entries) combined in the NNCT. That is, each test is Pielou’s overall test of segregation based on NNCTs for $k \geq 2$ classes. This overall test is based on the chi-squared approximation, is equivalent to Pearson’s chi-squared test on NNCT and is due to Pielou (1961). Each test is appropriate (i.e., have the appropriate asymptotic sampling distribution) when that data is obtained by sparse sampling.

Each function yields the test statistic, p-value and `df` which is $(k-1)^2$, description of the alternative with the corresponding null values (i.e., expected values) of NNCT entries, sample estimates (i.e., observed values) of the entries in NNCT. The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis is that $E(N_{ij}) = n_i c_j / n$ for all entries in the NNCT where n_i is the sum of row i (i.e., size of class i), c_j is the sum of column j in the $k \times k$ NNCT for $k \geq 2$. In the output, the test statistic and the p-value are valid only for (properly) sparsely sampled data.

See also (Pielou (1961); Ceyhan (2010)) and the references therein.

Usage

```
Pseg.ss.ct(ct, yates = TRUE, sim = FALSE, Nsim = 2000)

Pseg.ss(dat, lab, yates = TRUE, sim = FALSE, Nsim = 2000, ...)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Pseg.ss.ct` only
`yates`	A logical parameter (default=`TRUE`). If `TRUE`, Yates continuity correction is applied, and if `FALSE` the continuity correction is not applied. Equivalent to the `correct` argument in the base function `chisq.test`
`sim`	A logical parameter (default=`FALSE`). If `TRUE`, p-values are computed by Monte Carlo simulation and if `FALSE` the p-value is based on the chi-squared approximation. Equivalent to the `simulate.p.value` argument in the base function `chisq.test`
`Nsim`	A positive integer specifying the number of replicates used in the Monte Carlo test. Equivalent to the `B` argument in the base function `chisq.test`
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Pseg.ss` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Pseg.ss` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Pseg.ss` only

Value

A `list` with the elements

`statistic`	The overall chi-squared statistic
`stat.names`	Name of the test statistic
`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is (k-1)^2 for this function. Yields `NA` if `sim=TRUE` and `NSim` is provided.
`estimate`	Estimates of the parameters, NNCT, i.e., matrix of the observed N_{ij} values which is the NNCT.
`est.name`, `est.name2`	Names of the estimates, they are identical for this function.
`null.value`	Matrix of hypothesized null values for the parameters which are expected values of the the N_{ij} values in the NNCT.

`null.name`	Name of the null values
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Pseg.ss.ct` only
`data.name`	Name of the data set, `dat`, returned by `Pseg.ss` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). “On the use of nearest neighbor contingency tables for testing spatial segregation.” *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Pielou EC (1961). “Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships.” *Journal of Ecology*, **49(2)**, 255-269.

See Also

`overall.nnct.ct`, `overall.nnct`, `overall.seg.ct`, `overall.seg` and `chisq.test`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

Pseg.ss(Y,cls)

Pseg.ss.ct(ct)
Pseg.ss.ct(ct,yates=FALSE)

Pseg.ss.ct(ct,yates=FALSE,sim=TRUE)
Pseg.ss.ct(ct,yates=FALSE,sim=TRUE,Nsim=10000)

Pseg.ss(Y,cls,method="max")
Pseg.ss(Y,cls,yates=FALSE,sim=TRUE,Nsim=10000,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

Pseg.ss(Y,fcls)
Pseg.ss.ct(ct)

#############
n<-40
```

```
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

Pseg.ss(Y,cls)
Pseg.ss.ct(ct,yates=FALSE)

Pseg.ss(Y,cls, sim = TRUE, Nsim = 2000)
Pseg.ss.ct(ct,yates=FALSE)
```

`funsQandR` *Functions for the number of shared NNs, shared NN vector, and the number of reflexive NNs*

Description

Four functions: `Qval`, `Qvec`, `sharedNN` and `Rval`.

`Qval` returns the Q value, the number of points with shared nearest neighbors (NNs), which occurs when two or more points share a NN, for data in any dimension.

`Qvec` returns the Q-value and also yields the Qv vector $Qv = (Q_0, Q_1, \ldots)$ as well for data in any dimension, where Q_j is the number of points shared as a NN by j other points.

`sharedNN` returns the `vector` of number of points with shared NNs, $Q = (Q_0, Q_1, \ldots)$ where Q_i is the number of points that are NN to i points, and if a point is a NN of i points, then there are $i(i-1)$ points that share a NN. So, $Q = \sum_{i>1} i(i-1)Q_i$.

`Rval` returns the number of reflexive NNs, R (i.e., twice the number of reflexive NN pairs).

These quantities are used, e.g., in computing the variances and covariances of the entries of the nearest neighbor contingency tables used for Dixon's tests and other NNCT tests. The input must be the incidence matrix, W, of the NN digraph.

Usage

```
Qval(W)

Qvec(W)

sharedNN(W)

Rval(W)
```

Arguments

`W`	The incidence matrix, W, for the NN digraph

Value

The function `Qval` returns the Q value

The function `Qvec` returns a `list` with two elements

`q`	the Q value, the number of shared NNs
`qvec`	the `vector` of Q_j values

The function `sharedNN` returns a `matrix` with 2 rows, where first row is the j values and second row is the corresponding vector of Q_j values

The function `Rval` returns the R value, the number of reflexive NNs.

See the description above for the details of these quantities.

Author(s)

Elvan Ceyhan

See Also

`Tval`, `QRval`, `sharedNNmc`, and `Ninv`

Examples

```
#Examples for Qval
#3D data points
n<-10
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd)
Qval(W)

#1D data points
X<-as.matrix(runif(10)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(10) would not work
ipd<-ipd.mat(X)
W<-Wmat(ipd)
Qval(W)

#with ties=TRUE in the data
Y<-matrix(round(runif(15)*10),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd,ties=TRUE)
Qval(W)

#with ties=TRUE in the data
Y<-matrix(round(runif(15)*10),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd,ties=TRUE)
Qval(W)

#Examples for Qvec
```

```
#3D data points
n<-10
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd)
Qvec(W)

#2D data points
n<-15
Y<-matrix(runif(2*n),ncol=2)
ipd<-ipd.mat(Y)
W<-Wmat(ipd)
Qvec(W)

#1D data points
X<-as.matrix(runif(15)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(15) would not work
ipd<-ipd.mat(X)
W<-Wmat(ipd)
Qvec(W)

#with ties=TRUE in the data
Y<-matrix(round(runif(15)*10),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd,ties=TRUE)
Qvec(W)

#Examples for sharedNN
#3D data points
n<-10
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd)
sharedNN(W)
Qvec(W)

#1D data points
X<-as.matrix(runif(15)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(5) would not work
ipd<-ipd.mat(X)
W<-Wmat(ipd)
sharedNN(W)
Qvec(W)

#2D data points
n<-15
Y<-matrix(runif(2*n),ncol=2)
ipd<-ipd.mat(Y)
W<-Wmat(ipd)
sharedNN(W)
Qvec(W)

#with ties=TRUE in the data
```

```
Y<-matrix(round(runif(30)*10),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd,ties=TRUE)
sharedNN(W)

#Examples for Rval
#3D data points
n<-10
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd)
Rval(W)

#1D data points
X<-as.matrix(runif(15)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(5) would not work
ipd<-ipd.mat(X)
W<-Wmat(ipd)
Rval(W)

#with ties=TRUE in the data
Y<-matrix(round(runif(30)*10),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd,ties=TRUE)
Rval(W)
```

`funsRowColSums` *Functions for row and column sums of a matrix*

Description

Two functions: `row.sum` and `col.sum`.

`row.sum` returns the row sums of a given matrix (in particular a contingency table) as a vector and `col.sum` returns the column sums of a given matrix as a vector. `row.sum` is equivalent to `rowSums` function and `col.sum` is equivalent to `colSums` function in the base package.

Usage

```
row.sum(ct)

col.sum(ct)
```

Arguments

`ct`	A matrix, in particular a contingency table

Value

`row.sum` returns the row sums of `ct` as a vector `col.sum` returns the column sums of `ct` as a vector

Author(s)

Elvan Ceyhan

See Also

`rowSums` and `colSums`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

row.sum(ct)
rowSums(ct)

col.sum(ct)
colSums(ct)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

row.sum(ct)
rowSums(ct)

col.sum(ct)
colSums(ct)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

row.sum(ct)
rowSums(ct)

col.sum(ct)
colSums(ct)
```

`funsVarTk` *Variance of Cuzick and Edwards T_k Test statistic*

Description

Two functions: `VarTk` and `VarTkaij`.

Both functions compute the (finite sample) variance of Cuzick and Edwards T_k test statistic based on the number of cases within kNNs of the cases in the data under RL or CSR independence.

The common arguments for both functions are `n1`, representing the number of cases and `k`. The number of cases are denoted as n_1 and number of controls as n_0 in this function to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

The logical argument `nonzero.mat` (default=`TRUE`) is for using the A matrix if `FALSE` or just the matrix of nonzero locations in the A matrix (if `TRUE`) for computing N_s and N_t, which are required in the computation of the variance. N_s and N_t are defined on page 78 of (Cuzick and Edwards (1990)) as follows. $N_s = \sum_i \sum_j a_{ij} a_{ji}$ (i.e., number of ordered pairs for which kNN relation is symmetric) and $N_t = \sum \sum_{i \neq l} \sum a_{ij} a_{lj}$ (i.e, number of triplets (i, j, l) i, j, and l distinct so that j is among kNNs of i and j is among kNNs of l).

The function `VarTkaij` uses Toshiro Tango's moments formulas based on the $A = (a_{ij})$ matrix (and is equivalent to the function `VarTk`, see Tango (2007), where $a_{ij}(k) = 1$ if z_j is among the kNNs of z_i and 0 otherwise.

The function `varTkaij` is equivalent to `varTk` (with `$var` extension).

See (Cuzick and Edwards (1990); Tango (2007)).

Usage

```
varTk(dat, n1, k, nonzero.mat = TRUE, ...)

varTkaij(n1, k, a)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `VarTk` only.
`n1`	Number of cases
`k`	Integer specifying the number of NNs (of subject i)
`nonzero.mat`	A logical argument (default is `TRUE`) to determine whether the A matrix or the matrix of nonzero locations of the A matrix will be used in the computation of N_s and N_t. If `TRUE` the nonzero location matrix is used, otherwise the A matrix itself is used. Used in `VarTk` only.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. Used in `VarTk` only.
`a`	The $A = (a_{ij})$ matrix, used in `VarTkaij` only.

Value

The function `VarTk` returns a `list` with the elements

`var.Tk`	The (finite sample) variance of Cuzick and Edwards T_k test statistic for disease clustering

`Ns`	The N_s value standing for the number of ordered pairs for which kNN relation is symmetric, see the description.
`Nt`	The N_t value standing for the number of triplets (i, j, l) i, j, and l distinct so that j is among kNNs of i and j is among kNNs of l see the description.

The function `VarTkaij` returns only `var.Tk` as above.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

Tango T (2007). “A class of multiplicity adjusted tests for spatial clustering based on case-control point data.” *Biometrics*, **63**, 119-127.

See Also

`asyvarTk`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
n1<-sum(cls==1)
k<-2 #try also 2,3

a<-aij.mat(Y,k)

varTk(Y,n1,k)
varTk(Y,n1,k,nonzero.mat=FALSE)
varTk(Y,n1,k,method="max")

n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
n1<-sum(cls==1)
k<-1 #try also 2,3, sample(1:5,1)

a<-aij.mat(Y,k)

varTkaij(n1,k,a)
varTk(Y,n1,k)$var
```

`funsVarTrun` *Variance of Cuzick and Edwards T_run Test statistic*

Description

Two functions: `varTrun` and `varTrun.sim`.

The function `varTrun` computes the (finite sample) variance of Cuzick and Edwards T_{run} test statistic which is based on the number of consecutive cases from the cases in the data under RL or CSR independence. And the function `varTrun.sim` estimates this variance based on simulations under the RL hypothesis.

The only common argument for both functions is `dat`, the data set used in the functions.

n_1 is an argument for `varTrun` and is the number of cases (denoted as `n1` as an argument). The number of cases are denoted as n_1 and number of controls as n_0 in this function to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0's and 1's accordingly. The argument `Nsim` represents the number of resamplings (without replacement) in the RL scheme, with default being `1000`. `cc.lab`, `case.lab` and `Nsim` are arguments for `varTrun.sim` only.

The function `varTrun` might take a very long time when data size is large (even larger than 50), hence the need for the `varTrun.sim` function.

See (Cuzick and Edwards (1990)).

Usage

```
varTrun(dat, n1, ...)

varTrun.sim(dat, cc.lab, Nsim = 1000, case.lab = NULL)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in both functions.
`n1`	Number of cases, used in `varTrun` only.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. Used in `varTrun` only.
`cc.lab`	Case-control labels, 1 for case, 0 for control, used in `varTrun.sim` only.
`Nsim`	The number of simulations, i.e., the number of resamplings under the RL scheme to estimate the variance of T_{run}, used in `varTrun.sim` only.
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`, used in `varTrun.sim` only.

Value

The function `varTrun` returns the variance of Cuzick and Edwards T_{run} test statistic under RL or CSR independence. And the function `varTrun.sim` estimates the same variance based on simulations under the RL hypothesis.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`ceTrun` and `EV.Trun`

Examples

```
n<-20  #or try sample(1:20,1) #try also 40, 50, 60
set.seed(123)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)
n1<-sum(cls==1)
n0<-sum(cls==0)
c(n1,n0)

varTrun(Y,n1)
varTrun(Y,n1,method="max")

n<-15  #or try sample(1:20,1) #try also 40, 50, 60
set.seed(123)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)
n1<-sum(cls==1)
varTrun(Y,n1) #the actual value (might take a long time if n is large)

Nmc<-1000
varTrun.sim(Y,cls,Nsim=Nmc)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
varTrun.sim(Y,fcls,Nsim=Nmc,case.lab="a")
```

`funsW345values` *W_k values for Tango's T test statistic*

Description

Three functions: `W3val`, `W4val` and `W5val`, each of which is needed to compute $E[T^3]$ (i.e., for the skewness of T) where $T = T(\theta)$ which is defined in Equation (2) of Tango (2007) as follows: Let $(z_1, \ldots, z_n)$, $n = n_0 + n_1$, denote the locations of the points in the combined sample when the indices have been randomly permuted so that the z_i contain no information about group membership.

$$T(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{n} \delta_i \delta_j a_{ij}(\theta) = \boldsymbol{\delta}^t \boldsymbol{A}(\theta))\boldsymbol{\delta}$$

where $\delta_i = 1$ if z_i is a case, and 0 if z_i is a control, $\boldsymbol{A}(\theta) = (a_{ij}(\theta))$ could be any matrix of a measure of the closeness between two points i and j with $a_{ii} = 0$ for all $i = 1, \ldots, n$, and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^t$ denotes the unknown parameter vector related to cluster size and $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_n)^t$. Here the number of cases are denoted as n_1 and number of controls as n_0 to match the case-control class labeling, which is just the reverse of the labeling in Tango (2007).

If $\theta = k$ in the nearest neighbors model with $a_{ij}(k) = 1$ if z_j is among the kNNs of z_i and 0 otherwise, then the test statistic $T(\theta) = T_k$ is the Cuzick and Edwards kNN test statistic, T_k Cuzick and Edwards (1990), see also `ceTk`.

W_k values are used for Tango's correction to Cuzick and Edwards kNN test statistic, T_k and W_k here corresponds to W_{k-1} in Tango (2007) (defined for consistency with p_k's and $alpha_r$ having r distinct elements).

The argument of the function is the A_{ij} matrix, `a`, which is the output of the function `aij.mat`. However, inside the function we symmetrize the matrix `a` as `b <- (a+a^t)/2`, to facilitate the formulation.

Usage

```
W3val(a)

W4val(a)

W5val(a)
```

Arguments

`a`	A_{ij} matrix which is the output of the function `aij.mat`.

Value

Each function `Wkval` returns the W_k value for $k = 3, 4, 5$.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

Tango T (2007). “A class of multiplicity adjusted tests for spatial clustering based on case-control point data.” *Biometrics*, **63**, 119-127.

See Also

`ceTk`, `EV.Tk`, `varTk`, `Xsq.ceTk`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
k<-sample(1:5,1) # try also 3, 5, sample(1:5,1)
k
a<-aij.mat(Y,k)
W3val(a)
W4val(a)
W5val(a)

a<-aij.mat(Y,k,method="max")
W3val(a)
W4val(a)
W5val(a)
```

`funsXsq.nnref` *Reflexivity Test with Chi-square Approximation*

Description

Two functions: `Xsq.nnref.ct` and `Xsq.nnref`.

Both functions are objects of class `"Chisqtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of equality of the expected values of the diagonal cell counts (i.e., entries) under RL or CSR in the RCT for $k \geq 2$ classes. That is, each test performs an overall NN reflexivity test (for the vector of entries $(1,1)$ and $(2,2)$, respectively, in the RCT) which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Ceyhan and Bahadir (2017) for more detail).

Each reflexivity test is based on the chi-squared approximation of the corresponding quadratic form for the vector of diagonal entries in the RCT and are due to Ceyhan and Bahadir (2017).

Each function yields the test statistic, p-value and `df` which is 2, description of the alternative with the corresponding null values (i.e., expected values) of the diagonal entries and also the sample estimates (i.e., observed values) of the diagonal entries of RCT (as a vector). The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis is that $E(N_{11}, N_{22}) = (RP_{aa}, RP_{ab})$ in the RCT, where R is the number of reflexive NNs and P_{aa} is the probability of any two points selected are being from the same class and P_{ab} is the probability of any two points selected are being from two different classes.

Usage

```
Xsq.nnref.ct(rfct, nvec, Qv, Tv)

Xsq.nnref(dat, lab, ...)
```

Arguments

`rfct`	An RCT, used in `Xsq.nnref.ct` only
`nvec`	The `vector` of class sizes, used in `Xsq.nnref.ct` only
`Qv`	The number of shared NNs, used in `Xsq.nnref.ct` only
`Tv`	T value, which is the number of triplets (z_i, z_j, z_k) with $NN(z_i) = NN(z_j) = z_k$ and $NN(z_k) = z_j$ where $NN(\cdot)$ is the nearest neighbor function, used in `Xsq.nnref.ct` only.
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Xsq.nnref` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Xsq.nnref` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function, used in `Xsq.nnref` only

Value

A `list` with the elements

`statistic`	The chi-squared test statistic for overall NN reflexivity test
`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is 2 for this function.
`estimate`	Estimates of the parameters, i.e., the observed diagonal entries $(1, 1)$ and $(2, 2)$ in the RCT, `rfct`.
`est.name`, `est.name2`	Names of the estimates, they are identical for this function.
`null.value`	Hypothesized null values for the diagonal entries $(1, 1)$ and $(2, 2)$ in the RCT, which are $E(N_{11}) = RP_{aa}$ and $E(N_{22}) = RP_{ab}$, respectively).
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `rfct`, returned by `Xsq.nnref.ct` only
`data.name`	Name of the data set, `dat`, returned by `Xsq.nnref` only

Author(s)

Elvan Ceyhan

References

Ceyhan E, Bahadir S (2017). “Nearest Neighbor Methods for Testing Reflexivity.” *Environmental and Ecological Statistics*, **24(1)**, 69-108.

See Also

`Znnref.ct`, `Znnref`, `Zself.ref.ct`, `Zself.ref`, `Zmixed.nonref.ct` and `Zmixed.nonref`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

ipd<-ipd.mat(Y)
W<-Wmat(ipd)
Qv<-Qvec(W)$q
R<-Rval(W)
Tv<-Tval(W,R)

nvec<-as.numeric(table(cls))
rfct<-rct(ipd,cls)

Xsq.nnref(Y,cls)
Xsq.nnref.ct(rfct,nvec,Qv,Tv)

Xsq.nnref(Y,cls,method="max")

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

ipd<-ipd.mat(Y)
W<-Wmat(ipd)
Qv<-Qvec(W)$q
R<-Rval(W)
Tv<-Tval(W,R)

nvec<-as.numeric(table(cls))
rfct<-rct(ipd,cls)

Xsq.nnref(Y,cls)
Xsq.nnref.ct(rfct,nvec,Qv,Tv)
```

`funsXsq.nnsym.dx` *Dixon’s NN Symmetry Test with Chi-square Approximation for multiple classes*

Description

Two functions: `Xsq.nnsym.dx.ct` and `Xsq.nnsym.dx`.

Both functions are objects of class `"Chisqtest"` but with different arguments (see the parameter list below). Each one performs the hypothesis test of equality of the expected value of the off-diagonal cell counts (i.e., entries) under RL or CSR in the NNCT for $k \geq 2$ classes. That is, each performs Dixon's overall NN symmetry test. The test is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Ceyhan (2014) for more detail).

Each symmetry test is based on the chi-squared approximation of the corresponding quadratic form and is an extension of Dixon's NN symmetry test, which is extended by Ceyhan (2014).

Each function yields the test statistic, p-value and `df` which is $k(k-1)/2$, description of the alternative with the corresponding null values (i.e., expected values) of differences of the off-diagonal entries,(which is 0 for this function) and also the sample estimates (i.e., observed values) of absolute differences of the off-diagonal entries of NNCT (in the upper-triangular form). The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis is that all $E(N_{ij}) = E(N_{ji})$ entries for all $i \neq j$ (i.e., symmetry in the mixed NN structure).

See also (Ceyhan (2014)) and the references therein.

Usage

```
Xsq.nnsym.dx.ct(ct, covS)

Xsq.nnsym.dx(dat, lab, ...)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Xsq.nnsym.dx.ct` only
`covS`	The $k(k-1)/2 \times k(k-1)/2$ covariance matrix of the differences of the off-diagonal entries in the NNCT, `ct`, usually the output of the function `cov.nnsym`.
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Xsq.nnsym.dx` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Xsq.nnsym.dx` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Xsq.nnsym.dx` only

Value

A `list` with the elements

`statistic`	The chi-squared test statistic for Dixon's overall NN symmetry test
`stat.names`	Name of the test statistic
`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is $k(k-1)/2$ for this function.

`estimate`	Estimates, i.e., absolute differences of the off-diagonal entries of NNCT (in the upper-triangular form).
`est.name`, `est.name2`	Names of the estimates, former is a shorter description of the estimates than the latter.
`null.value`	Hypothesized null values for the differences between the expected values of the off-diagonal entries, which is 0 for this function.
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Xsq.nnsym.dx.ct` only
`data.name`	Name of the data set, `dat`, returned by `Xsq.nnsym.dx` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations.” *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

See Also

`Znnsym.dx.ct`, `Znnsym.dx`, `Znnsym`, `Xsq.nnsym`, `Xsq.nnsym.ss.ct`, `Xsq.nnsym.ss` and `Qsym.test`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv) #default is byrow
covS<-cov.nnsym(covN)

Xsq.nnsym.dx(Y,cls)
Xsq.nnsym.dx.ct(ct,covS)

Xsq.nnsym.dx(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

Xsq.nnsym.dx(Y,fcls)
```

```
Xsq.nnsym.dx.ct(ct,covS)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)
covS<-cov.nnsym(covN)

Xsq.nnsym.dx(Y,cls)
Xsq.nnsym.dx.ct(ct,covS)
```

`funsXsq.nnsym.ss` *Pielou's First Type of NN Symmetry Test with Chi-square Approximation for multiple classes (for Sparse Sampling)*

Description

Two functions: `Xsq.nnsym.ss.ct` and `Xsq.nnsym.ss`.

Both functions are objects of class `"Chisqtest"` but with different arguments (see the parameter list below). Each one performs the hypothesis test of equality of the expected value of the off-diagonal cell counts (i.e., entries) under RL or CSR in the NNCT for $k \geq 2$ classes. That is, each performs Pielou's first type of NN symmetry test which is also equivalent to McNemar's test on the NNCT. The test is appropriate (i.e., have the appropriate asymptotic sampling distribution) provided that data is obtained by sparse sampling. (See Ceyhan (2014) for more detail).

Each symmetry test is based on the chi-squared approximation of the corresponding quadratic form and are due to Pielou (1961).

The argument cont.corr is a logical argument (default=`TRUE`) for continuity correction to this test. If `TRUE` the continuity correction to McNemar's test is implemented, and if `FALSE` such a correction is not implemented.

Each function yields the test statistic, p-value and `df` which is $k(k-1)/2$, description of the alternative with the corresponding null values (i.e., expected values) of differences of the off-diagonal entries,(which is 0 for this function) and also the sample estimates (i.e., observed values) of absolute differences of th off-diagonal entries of NNCT (in the upper-triangular form). The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis is that $E(N_{ij}) = E(N_{ji})$ for all entries for $i \neq j$ (i.e., symmetry in the mixed NN structure). In the output, the test statistic, p-value and `df` are valid only for (properly) sparsely sampled data.

See also (Pielou (1961); Ceyhan (2014)) and the references therein.

Usage

```
Xsq.nnsym.ss.ct(ct, cont.corr = TRUE)

Xsq.nnsym.ss(dat, lab, cont.corr = TRUE, ...)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Xsq.nnsym.ss.ct` only
`cont.corr`	A logical argument (default=`TRUE`). If `TRUE` the continuity correction to McNemar's test is implemented, and if `FALSE` such a correction is not implemented.
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Xsq.nnsym.ss` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Xsq.nnsym.ss` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Xsq.nnsym.ss` only

Value

A `list` with the elements

`statistic`	The chi-squared test statistic for Pielou's first type of NN symmetry test
`stat.names`	Name of the test statistic
`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is $k(k-1)/2$ for this function.
`estimate`	Estimates, i.e., absolute differences of the off-diagonal entries of NNCT (in the upper-triangular form).
`est.name`, `est.name2`	Names of the estimates, former is a shorter description of the estimates than the latter.
`null.value`	Hypothesized null values for the differences between the expected values of the off-diagonal entries, which is 0 for this function.
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Xsq.nnsym.ss.ct` only
`data.name`	Name of the data set, `dat`, returned by `Xsq.nnsym.ss` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations.” *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Pielou EC (1961). “Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships.” *Journal of Ecology*, **49(2)**, 255-269.

See Also

`Znnsym2cl.ss.ct`, `Znnsym2cl.ss`, `Znnsym.ss.ct`, `Znnsym.ss`, `Xsq.nnsym.dx.ct`, `Xsq.nnsym.dx` and `Qsym.test`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

Xsq.nnsym.ss(Y,cls)
Xsq.nnsym.ss.ct(ct)

Xsq.nnsym.ss(Y,cls,method="max")

Xsq.nnsym.ss(Y,cls,cont.corr=FALSE)
Xsq.nnsym.ss.ct(ct,cont.corr=FALSE)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
Xsq.nnsym.ss(Y,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

Xsq.nnsym.ss(Y,cls)
Xsq.nnsym.ss.ct(ct)
Xsq.nnsym.ss.ct(ct,cont.corr = FALSE)
```

`funsXsq.seg.coeff` *Chi-square Test for Segregation Coefficients*

Description

Two functions: `Xsq.seg.coeff.ct` and `Xsq.seg.coeff`.

Each one performs hypothesis tests of (simultaneous) equality of the segregation coefficients in an NNCT to the ones under RL or CSR. That is, each performs the combined Chi-square test for segregation coefficients which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Ceyhan (2014) for more detail).

Each test is based on the Chi-square approximation of the corresponding quadratic form for the segregation coefficients in an NNCT. The segregation coefficients in the multi-class case are the extension of Pielou's segregation coefficient for the two-class case. (See Ceyhan (2014) for more detail).

Each function yields the test statistic, p-value and `df` which is $k(k+1)/2-1$, description of the alternative with the corresponding null values (i.e., expected values) of the segregation coefficients in the NNCT (which are 0 for this function) and also the sample estimates (i.e., observed values) of the segregation coefficients. The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis for all cells (i,j) is that the corresponding segregation coefficients are all equal to the expected value (which is 0) under RL or CSR.

Usage

```
Xsq.seg.coeff.ct(ct, covSC)

Xsq.seg.coeff(dat, lab, ...)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Xsq.seg.coeff.ct` only
`covSC`	The covariance matrix for the segregation coefficients in the NNCT, used in `Xsq.seg.coeff.ct` only. Usually output of the function `cov.seg.coeff`
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Xsq.seg.coeff` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Xsq.seg.coeff` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Xsq.seg.coeff` only

Value

A `list` with the elements

`statistic`	The chi-squared test statistic for the combined segregation coefficients
`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is $k(k+1)/2-1$ for this function.
`estimate`	The `vector` of estimates of the parameters, i.e., observed values of segregation coefficients in the NNCT.

`est.name`, `est.name2`	Names of the estimates, they are identical for this function.
`null.value`	The null value of the parameters, i.e., expected values of segregation coefficients in the NNCT under RL or CSR (which is 0).
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Xsq.seg.coeff.ct` only
`data.name`	Name of the data set, `dat`, returned by `Xsq.seg.coeff` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`seg.coeff`, `Zseg.coeff.ct` and `Zseg.coeff`

Examples

```
n<-20
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

covSC<-cov.seg.coeff(ct,covN)

Xsq.seg.coeff(Y,cls)
Xsq.seg.coeff.ct(ct,covSC)

Xsq.seg.coeff(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

Xsq.seg.coeff.ct(ct,covSC)

#############
n<-40
```

```
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

covSC<-cov.seg.coeff(ct,covN)

Xsq.seg.coeff(Y,cls)
Xsq.seg.coeff.ct(ct,covSC)
```

`funsXsq.spec.cor` *Overall Species Correspondence Test with Chi-square Approximation*

Description

Two functions: `Xsq.spec.cor.ct` and `Xsq.spec.cor`.

Each one performs hypothesis tests of (simultaneous) equality of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or the expected values of the diagonal entries N_{ii} in an NNCT to the ones under RL or CSR. That is, each performs the overall species correspondence test which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Ceyhan (2018) for more detail).

Each test is based on the Chi-square approximation of the corresponding quadratic form for the first column in a species correspondence contingency table (SCCT) or the diagonal entries N_{ii} in an NNCT and are due to (Ceyhan 2018).

Each function yields the test statistic, p-value and `df` which is k, description of the alternative with the corresponding null values (i.e., expected values) of the self entries (i.e., first column) in the SCCT or the diagonal entries in the NNCT and also the sample estimates (i.e., observed values) of these entries. The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis is that all $E[S_1, S_2, \ldots, S_k] = E[N_{11}, N_{22}, \ldots, N_{kk}] = ((n_1(n_1 - 1)/(n - 1), (n_2(n_2 - 1)/(n - 1), \ldots, (n_k(n_k - 1)/(n - 1))$ where n_i is the size of class i and n is the data size.

Usage

```
Xsq.spec.cor.ct(ct, covSC, nnct = FALSE)

Xsq.spec.cor(dat, lab, ...)
```

Arguments

`ct`	The NNCT or SCCT, used in `Xsq.spec.cor.ct` only
`covSC`	The covariance matrix for the self entries (i.e., first column) in the SCCT or the diagonal entries in the NNCT, used in `Xsq.spec.cor.ct` only. Usually output of the functions `covNii.ct` or `covNii`.
`nnct`	A logical parameter (default=`FALSE`). If `TRUE`, `x` is taken to be the $k \times k$ NNCT, and if `FALSE`, `x` is taken to be the IPD matrix, used in `Xsq.spec.cor.ct` only
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Xsq.spec.cor` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Xsq.spec.cor` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Xsq.spec.cor` only

Value

A `list` with the elements

`statistic`	The chi-squared test statistic for overall species correspondence test
`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is k for this function.
`estimate`	The `vector` of estimates of the parameters, i.e., observed values of self entries in the SCCT or diagonal entries in the NNCT.
`est.name`, `est.name2`	Names of the estimates, they are identical for this function.
`null.value`	The `vector` of null values of the parameters, i.e., expected values of self entries in the SCCT or diagonal entries in the NNCT under RL or CSR.
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Xsq.spec.cor.ct` only
`data.name`	Name of the data set, `dat`, returned by `Xsq.spec.cor` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2018). “A contingency table approach based on nearest neighbor relations for testing self and mixed correspondence.” *SORT-Statistics and Operations Research Transactions*, **42(2)**, 125-158.

See Also

`Zself.ref.ct`, `Zself.ref`, `Xsq.nnref.ct` and `Xsq.nnref`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-scct(ipd,cls)
ct

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)

vsq<-varNii.ct(ct,Qv,Rv)
cv<-covNii.ct(ct,vsq,Qv,Rv)
Xsq.spec.cor.ct(ct,cv)
Xsq.spec.cor(Y,cls)
Xsq.spec.cor(Y,cls,method="max")

ct<-nnct(ipd,cls)
Xsq.spec.cor.ct(ct,cv,nnct = TRUE)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-scct(ipd,fcls)
Xsq.spec.cor.ct(ct,cv)
Xsq.spec.cor(Y,fcls)

ct<-nnct(ipd,fcls)
Xsq.spec.cor.ct(ct,cv,nnct=TRUE)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-scct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)

vsq<-varNii.ct(ct,Qv,Rv)
cv<-covNii.ct(ct,vsq,Qv,Rv)
Xsq.spec.cor.ct(ct,cv)

ct<-nnct(ipd,cls)
Xsq.spec.cor.ct(ct,cv,nnct = TRUE)
Xsq.spec.cor(Y,cls)
```

`funsZcell.nnct` *Dixon's Cell-specific Z Tests of Segregation for NNCT*

Description

Two functions: `Zcell.nnct.ct` and `Zcell.nnct`.

Both functions are objects of class `"cellhtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of deviations of cell counts from the expected values under RL or CSR for each cell (i.e., entry) in the NNCT. The test for each cell i, j is based on the normal approximation of the corresponding cell count, N_{ij} and are due to Dixon (1994, 2002).

Each function yields a contingency table of the test statistics, p-values for the corresponding alternative, expected values (i.e., null value(s)), lower and upper confidence levels, sample estimates (i.e., observed values) for the cell counts and also names of the test statistics, estimates, null values, the description of the test, and the data set used.

The null hypothesis for each cell i, j is that the corresponding cell count is equal to the expected value under RL or CSR, that is $E[N_{ii}] = n_i(n_i - 1)/(n - 1)$ and $E[N_{ij}] = n_i n_j/(n - 1)$ where n_i is the size of class i and n is the size of the data set.

See also (Dixon (1994, 2002); Ceyhan (2010)).

Usage

```
Zcell.nnct.ct(
  ct,
  varN,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Zcell.nnct(
  dat,
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Zcell.nnct.ct` only
`varN`	The variance matrix for cell counts in the NNCT, `ct`; used in `Zcell.nnct.ct` only
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the cell counts, i.e. N_{ij} values

`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Zcell.nnct` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Zcell.nnct` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function, used in `Zcell.nnct` only

Value

A `list` with the elements

`statistic`	The `matrix` of Dixon's cell-specific test statistics
`stat.names`	Name of the test statistics
`p.value`	The `matrix` of p-values for the hypothesis test for the corresponding alternative
`LCL`, `UCL`	Matrix of lower and upper confidence levels for the cell counts at the given confidence level `conf.level` and depends on the type of `alternative`.
`conf.int`	The confidence interval for the estimates, it is `NULL` here, since we provide the `UCL` and `LCL` in `matrix` form.
`cnf.lvl`	Level of the upper and lower confidence limits of the cell counts, provided in `conf.level`.
`estimate`	Estimates of the parameters, i.e., matrix of the observed cell counts which is the NNCT
`est.name`, `est.name2`	Names of the estimates, both are same in this function
`null.value`	Matrix of hypothesized null values for the parameters which are expected values of the cell counts.
`null.name`	Name of the null values
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Zcell.nnct.ct` only
`data.name`	Name of the data set, `dat`, returned by `Zcell.nnct` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "On the use of nearest neighbor contingency tables for testing spatial segregation." *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Dixon PM (1994). "Testing spatial segregation using a nearest-neighbor contingency table." *Ecology*, **75(7)**, 1940-1948.

Dixon PM (2002). "Nearest-neighbor contingency table analysis of spatial segregation for several species." *Ecoscience*, **9(2)**, 142-151.

See Also

`Zcell.nnct.2s`, `Zcell.nnct.rs`, `Zcell.nnct.ls`, `Zcell.nnct.pval`, and `Zcell.tct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
varN

Zcell.nnct(Y,cls)
Zcell.nnct(Y,cls,alt="g")

Zcell.nnct.ct(ct,varN)
Zcell.nnct.ct(ct,varN,alt="g")

Zcell.nnct(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
Zcell.nnct(Y,cls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)

Zcell.nnct(Y,cls)
Zcell.nnct.ct(ct,varN)
```

`funsZcell.nnct.pval` *p-values for Cell-specific Z Test Statistics for NNCT*

Description

Four functions: `Zcell.nnct.2s`, `Zcell.nnct.rs`, `Zcell.nnct.ls` and `Zcell.nnct.pval`.

These functions yield a contingency table (i.e., a matrix) of the p-values for the cell-specific Z test statistics for the NNCT and take the cell-specific Z test statistics in matrix form as their argument. `Zcell.nnct.pval` yields an array of size $k \times k \times 3$ where 1st entry of the array is the matrix of p-values for the two-sided alternative, 2nd entry of the array is the matrix of p-values for the left-sided alternative and 3rd entry of the array is the matrix of p-values for the right-sided alternative. And each of `Zcell.nnct.2s`, `Zcell.nnct.rs` and `Zcell.nnct.ls` yield a $k \times k$ matrix of p-values for the two-sided, right-sided and left-sided alternative, respectively.

The functions `Zcell.nnct.2s`, `Zcell.nnct.rs` and `Zcell.nnct.ls` are equivalent to `Zcell.nnct(...,alt)$p.val` where `alt="two-sided"`, `"greater"` and `"less"`, respectively, with the appropriate arguments for the function `Zcell.nnct` (see the examples below).

See also (Dixon (1994, 2002); Ceyhan (2010)).

Usage

```
Zcell.nnct.pval(zt)

Zcell.nnct.2s(zt)

Zcell.nnct.ls(zt)

Zcell.nnct.rs(zt)
```

Arguments

`zt`	A $k \times k$ matrix of the cell-specific Z test statistics

Value

`Zcell.nnct.pval` returns a $k \times k \times 3$ array whose 1st entry is the matrix of p-values for the two-sided alternative, 2nd entry is the matrix of p-values for the left-sided alternative and 3rd entry is the matrix of p-values for the right-sided alternative `Zcell.nnct.2s` returns a $k \times k$ matrix of p-values for the two-sided alternative `Zcell.nnct.rs` returns a $k \times k$ matrix of p-values for the right-sided alternative `Zcell.nnct.ls` returns a $k \times k$ matrix of p-values for the left-sided alternative

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "On the use of nearest neighbor contingency tables for testing spatial segregation." *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Dixon PM (1994). "Testing spatial segregation using a nearest-neighbor contingency table." *Ecology*, **75(7)**, 1940-1948.

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

See Also

`Zcell.nnct` and `Zcell.nnct.ct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
TS<-Zcell.nnct(Y,cls)$statistic
TS
pv<-Zcell.nnct.pval(TS)
pv

Zcell.nnct(Y,cls,alt="t")$p.val
Zcell.nnct(Y,cls,alt="l")$p.val
Zcell.nnct(Y,cls,alt="g")$p.val

Zcell.nnct.2s(TS)

Zcell.nnct.ls(TS)

Zcell.nnct.rs(TS)
```

`funsZcell.spec` *Cell-specific Z Tests of Segregation for NNCTs*

Description

Two functions: `Zcell.spec.ct` and `Zcell.spec`.

All functions are objects of class `"cellhtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of deviations of entries of NNCT or types I-IV TCTs from the expected values under RL or CSR for each entry. The test for each entry i, j is based on the normal approximation of the corresponding T_{ij} value and are due to Dixon (2002) and Ceyhan (2017), respectively.

The `type="dixon"` or `"nnct"` refers to Dixon's cell-specific test of segregation, and `type="I"`-`"IV"` refers to types I-IV cell-specific tests, respectively.

Each function yields a contingency table of the test statistics, p-values for the corresponding alternative, expected values (i.e., null value(s)), lower and upper confidence levels and sample estimates (i.e., observed values) for the N_{ij} or T_{ij} values and also names of the test statistics, estimates, null values, the description of the test, and the data set used.

The null hypothesis for each entry i, j is that the corresponding value N_{ij} or T_{ij} is equal to the expected value under RL or CSR.

See also (Dixon (1994, 2002); Ceyhan (2010, 2017)) and the references therein.

Usage

```
cell.spec.ct(
  ct,
  covN,
  type,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

cell.spec(
  dat,
  lab,
  type,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Zcell.spec.ct` only
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT, `ct`; used in `Zcell.spec.ct` only.
`type`	The type of the cell-specific test with no default. Takes on values `"dixon"` or `"nnct"` for Dixon's cell-specific tests and `"I"`-`"IV"` for types I-IV cell-specific tests (or equivalently `1-6`, respectively).
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the N_{ij} or T_{ij} values
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Zcell.spec` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Zcell.spec` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function, used in `Zcell.spec` only

Value

A `list` with the elements

`statistic`	The `matrix` of cell-specific test statistics
`stat.names`	Name of the test statistics
`p.value`	The `matrix` of p-values for the hypothesis test for the corresponding alternative
`LCL, UCL`	Matrix of lower and upper confidence levels for the N_{ij} or T_{ij} values at the given confidence level `conf.level` and depends on the type of `alternative`.

`conf.int`	The confidence interval for the estimates, it is `NULL` here, since we provide the `UCL` and `LCL` in `matrix` form.
`cnf.lvl`	Level of the upper and lower confidence limits of the entries, provided in `conf.level`.
`estimate`	Estimates of the parameters, NNCT or TCT, i.e., matrix of the observed N_{ij} or T_{ij} values which is NNCT or TCT, respectively.
`est.name`, `est.name2`	Names of the estimates, both are same in this function
`null.value`	Matrix of hypothesized null values for the parameters which are expected values of the the null N_{ij} values in an NNCT or T_{ij} values in an TCT.
`null.name`	Name of the null values
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Zcell.spec.ct` only
`data.name`	Name of the data set, `dat`, returned by `Zcell.spec` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). “On the use of nearest neighbor contingency tables for testing spatial segregation.” *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

Dixon PM (1994). “Testing spatial segregation using a nearest-neighbor contingency table.” *Ecology*, **75(7)**, 1940-1948.

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

See Also

`Zcell.nnct.ct`, `Zcell.nnct`, `Zcell.tct.ct` and `Zcell.tct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
```

```
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

type<-"IV" #"dixon" #try also "nnct", "I", "II", "III", and "IV"
cell.spec(Y,cls,type)
cell.spec(Y,cls,type,alt="g")

cell.spec.ct(ct,covN,type)
cell.spec.ct(ct,covN,type="II",alt="g")

cell.spec(Y,cls,type,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
cell.spec(Y,cls,type="I")

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

cell.spec(Y,cls,type)
cell.spec.ct(ct,covN,type)
```

`funsZcell.tct` *Types I-IV Cell-specific Z Tests of Segregation based on NNCTs*

Description

Two functions: `Zcell.tct.ct` and `Zcell.tct`.

All functions are objects of class `"cellhtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of deviations of entries of types I-IV TCT, T_{ij}, from their expected values under RL or CSR for each entry. The test for each entry i, j is based on the normal approximation of the corresponding T_{ij} value and are due to Ceyhan (2017).

Each function yields a contingency table of the test statistics, p-values for the corresponding alternative, expected values (i.e., null value(s)), lower and upper confidence levels, and sample estimates (i.e., observed values) for the T_{ij} values and also names of the test statistics, estimates, null values, the description of the test, and the data set used.

The null hypothesis for each entry i, j is that the corresponding value T_{ij} is equal to the expected value under RL or CSR, see Ceyhan (2017) for more detail.

See also (Ceyhan (2017)) and references therein.

Usage

```
Zcell.tct.ct(
  ct,
  covN,
  type = "III",
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Zcell.tct(
  dat,
  lab,
  type = "III",
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Zcell.tct.ct` only
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized cell counts of NNCT, `ct`; used in `Zcell.tct.ct` only.
`type`	The type of the cell-specific test, default=`"III"`. Takes on values `"I"`-`"IV"` (or equivalently `1-4`, respectively.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the T_{ij} values
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Zcell.tct` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Zcell.tct` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function, used in `Zcell.tct` only

Value

A `list` with the elements

`statistic`	The `matrix` of Types I-IV cell-specific test statistics

`stat.names`	Name of the test statistics
`p.value`	The `matrix` of p-values for the hypothesis test for the corresponding alternative
`LCL`, `UCL`	Matrix of lower and upper confidence levels for the T_{ij} values at the given confidence level `conf.level` and depends on the type of `alternative`.
`conf.int`	The confidence interval for the estimates, it is `NULL` here, since we provide the `UCL` and `LCL` in `matrix` form.
`cnf.lvl`	Level of the upper and lower confidence limits of the entries, provided in `conf.level`.
`estimate`	Estimates of the parameters, i.e., matrix of the observed T_{ij} values which is the TCT
`est.name`, `est.name2`	Names of the estimates, both are same in this function
`null.value`	Matrix of hypothesized null values for the parameters which are expected values of T_{ij} values in the TCT.
`null.name`	Name of the null values
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Zcell.tct.ct` only
`data.name`	Name of the data set, `dat`, returned by `Zcell.tct` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`Zcell.nnct.ct` and `Zcell.nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)
```

```
type<-"I" #try also "II", "III", and "IV"
Zcell.tct(Y,cls,type)
Zcell.tct(Y,cls,type,alt="g")
Zcell.tct(Y,cls,type,method="max")

Zcell.tct.ct(ct,covN)
Zcell.tct.ct(ct,covN,type)
Zcell.tct.ct(ct,covN,type,alt="g")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
Zcell.tct(Y,cls,type)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

Zcell.tct(Y,cls,type)
Zcell.tct.ct(ct,covN,type)
```

`funsZdir.nnct` *Directional Segregation Test for Two Classes with Normal Approximation*

Description

Two functions: `Zdir.nnct.ct` and `Zdir.nnct`.

Both functions are objects of class `"htest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of equality of the expected value of the the difference between the phat estimates in a 2×2 NNCT to the one under RL or CSR (which is $-1/(n-1)$) where phat estimates are N_{11}/n_1 and N_{21}/n_2. That is, each performs directional (i.e., one-sided) tests based on the 2×2 NNCT (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Ceyhan (2010) for more detail).

The one-sided (or directional) test has two types, specified with the type argument, with default `type="II"`. The second type is $Z_{II} = (T_n - ET_n)/\sqrt{Var(T_n)}$ where $T_n = N_{11}/n_1 - N_{21}/n_2$ (which is the difference between phat values) and the first type is $Z_I = U_n T_n$ where $U_n = \sqrt{n_1 n_2/(C_1 C_2)}$. Each test is based on the normal approximation of the Z_I and Z_{II} based on the 2×2 NNCT and are due to (Ceyhan 2010).

Each function yields the test statistic, p-value for the corresponding alternative, the confidence interval, sample estimate (i.e., observed value) and null (i.e., expected) value for the difference in phat values which is $-1/(n-1)$ for this function and method and name of the data set used.

The null hypothesis is that all $E[Z_{II}] = 0$ and $E[Z_I]$ converges to 0 as class sizes go to infinity (or T_n has mean equal to $-1/(n-1)$ where n is the data size.

Usage

```
Zdir.nnct.ct(
  ct,
  covN,
  type = "II",
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Zdir.nnct(
  dat,
  lab,
  type = "II",
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	The NNCT, used in `Zdir.nnct.ct` only
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT
`type`	The type of the directional (i.e., one-sided) test with default=`"II"`. Takes on values `"I"` and `"II"` for types I and II directional tests (see the description above).
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the difference in phat estimates in the NNCT
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Zdir.nnct` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Zdir.nnct` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Zdir.nnct` only

Value

A `list` with the elements

`statistic`	The Z test statistic for the directional (i.e., one-sided) test of segregation based on the NNCT

`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the difference in phat values in an NNCT at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., the observed difference in phat values in an NNCT.
`null.value`	Hypothesized null value for the difference in phat values in an NNCT which is $-1/(n-1)$ for this function.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Zdir.nnct.ct` only
`data.name`	Name of the data set, `dat`, returned by `Zdir.nnct` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). “Directional clustering tests based on nearest neighbour contingency tables.” *Journal of Nonparametric Statistics*, **22(5)**, 599-616.

See Also

`Zdir.nnct.ss.ct`, `Zdir.nnct.ss`, `overall.nnct.ct` and `overall.nnct`

Examples

```
n<-20
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

Zdir.nnct(Y,cls)
Zdir.nnct.ct(ct,covN)

Zdir.nnct(Y,cls,alt="g")
Zdir.nnct.ct(ct,covN,type="I",alt="l")

Zdir.nnct(Y,cls,method="max")

#cls as a factor
```

```
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

Zdir.nnct(Y,fcls)
Zdir.nnct.ct(ct,covN)

#############
ct<-matrix(1:4,ncol=2)
Zdir.nnct.ct(ct,covN) #gives an error message if ct is defined as ct<-matrix(1:9,ncol=3)
```

`funsZdir.nnct.ss` *Directional Segregation Test for Two Classes with Normal Approximation (for Sparse Sampling)*

Description

Two functions: `Zdir.nnct.ss.ct` and `Zdir.nnct.ss`.

Both functions are objects of class `"htest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of independence in the 2×2 NNCT which implies $Z_P = 0$ or equivalently $N_{11}/n_1 = N_{21}/n_2$. $Z_P = (N_{11}/n_1 - N_{21}/n_2)\sqrt{n_1 n_2 n/(C_1 C_2)}$ where N_{ij} is the cell count in entry i, j, n_i is the sum of row i (i.e., size of class i), c_j is the sum of column j in the 2×2 NNCT; N_{11}/n_1 and N_{21}/n_2 are also referred to as the phat estimates in row-wise binomial framework for 2×2 NNCT (see Ceyhan (2010)).

That is, each performs directional (i.e., one-sided) tests based on the 2×2 NNCT and is appropriate (i.e., have the appropriate asymptotic sampling distribution) when that data is obtained by sparse sampling. (See Ceyhan (2010) for more detail).

Each test is based on the normal approximation of Z_P which is the directional Z-tests for the chi-squared tests of independence for the contingency tables (Bickel and Doksum 1977).

Each function yields the test statistic, p-value for the corresponding alternative, the confidence interval, sample estimate (i.e., observed value) and null (i.e., expected) value for the difference in the phat values (which is 0 for this test) in an NNCT, and method and name of the data set used.

The null hypothesis is that $E[Z_P] = 0$ or equivalently $N_{11}/n_1 = N_{21}/n_2$.

Usage

```
Zdir.nnct.ss.ct(
  ct,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Zdir.nnct.ss(
  dat,
  lab,
```

```
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

ct	The NNCT, used in `Zdir.nnct.ss.ct` only
alternative	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
conf.level	Level of the confidence limits, default is `0.95`, for the difference in phat values in the NNCT
dat	The data set in one or higher dimensions, each row corresponds to a data point, used in `Zdir.nnct.ss` only
lab	The `vector` of class labels (numerical or categorical), used in `Zdir.nnct.ss` only
...	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Zdir.nnct.ss` only

Value

A `list` with the elements

statistic	The Z test statistic for the directional (i.e., one-sided) test of segregation based on the NNCT
p.value	The p-value for the hypothesis test for the corresponding alternative
conf.int	Confidence interval for the difference in phat values in the NNCT at the given confidence level `conf.level` and depends on the type of `alternative`.
estimate	Estimate of the parameter, i.e., the observed difference in phat values in the NNCT.
null.value	Hypothesized null value for the difference in phat values in the NNCT which is 0 for this function.
alternative	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
method	Description of the hypothesis test
ct.name	Name of the contingency table, `ct`, returned by `Zdir.nnct.ss.ct` only
data.name	Name of the data set, `dat`, returned by `Zdir.nnct.ss` only

Author(s)

Elvan Ceyhan

References

Bickel PJ, Doksum AK (1977). *Mathematical Statistics, Basic Ideas and Selected Topics*. Prentice Hall, Englewood Cliffs, NJ.

Ceyhan E (2010). “Directional clustering tests based on nearest neighbour contingency tables.” *Journal of Nonparametric Statistics*, **22(5)**, 599-616.

See Also

`Zdir.nnct.ct`, `Zdir.nnct`, `Pseg.ss.ct` and `Pseg.ss`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

Zdir.nnct.ss(Y,cls)
Zdir.nnct.ss.ct(ct)
Zdir.nnct.ss(Y,cls,alt="g")

Zdir.nnct.ss(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

Zdir.nnct.ss(Y,fcls)
Zdir.nnct.ss.ct(ct)

#############
ct<-matrix(1:4,ncol=2)
Zdir.nnct.ss.ct(ct) #gives an error message if ct<-matrix(1:9,ncol=3)
```

`funsZmixed.nonref` *Mixed-Non-Reflexivity Test with Normal Approximation*

Description

Two functions: `Zmixed.nonref.ct` and `Zmixed.nonref`.

Both functions are objects of class `"htest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of mixed non-reflexivity in the NN structure using the number of mixed-non-reflexive NN pairs (i.e., the second diagonal entry, $(2,2)$) in the RCT for

$k \geq 2$ classes. That is, each test performs a test of mixed non-reflexivity corresponding to entry $(2, 2)$ in the RCT) which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Ceyhan and Bahadir (2017) for more detail).

The mixed non-reflexivity test is based on the normal approximation of the diagonal entry $(2, 2)$ in the RCT and are due to Ceyhan and Bahadir (2017).

Each function yields the test statistic, p-value for the corresponding alternative, the confidence interval, sample estimate (i.e., observed value) and null (i.e., expected) value for the mixed non-reflexivity value (i.e., diagonal entry $(2, 2)$ value, respectively) in the RCT, and method and name of the data set used.

The null hypothesis is that $E(N_{22}) = RP_{ab}$ in the RCT, where R is the number of reflexive NNs and P_{ab} is the probability of any two points selected are being from two different classes.

Usage

```
Zmixed.nonref.ct(
  rfct,
  nvec,
  Qv,
  Tv,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Zmixed.nonref(
  dat,
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`rfct`	An RCT, used in `Zmixed.nonref.ct` only
`nvec`	The `vector` of class sizes, used in `Zmixed.nonref.ct` only
`Qv`	The number of shared NNs, used in `Zmixed.nonref.ct` only
`Tv`	T value, which is the number of triplets (z_i, z_j, z_k) with "$NN(z_i) = NN(z_j) = z_k$ and $NN(z_k) = z_j$ where $NN(\cdot)$ is the nearest neighbor function, used in `Zmixed.nonref.ct` only.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the difference of the off-diagonal entries, $N_{12} - N_{21}$
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Zmixed.nonref` only

`lab`	The `vector` of class labels (numerical or categorical), used in `Zmixed.nonref` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Zmixed.nonref` only

Value

A `list` with the elements

`statistic`	The Z test statistic for mixed non-reflexivity corresponding to entry $(2, 2)$ in the RCT
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the mixed non-reflexivity value (i.e., diagonal entry $(2, 2)$ value) in the RCT at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., the observed diagonal entry $(2, 2)$ in the RCT, `rfct`.
`null.value`	Hypothesized null value for the mixed non-reflexivity value (i.e., expected value of the diagonal entry $(2, 2)$ which is $E(N_{22}) = RP_{ab}$) in the RCT.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `rfct`, returned by `Zmixed.nonref.ct` only
`data.name`	Name of the data set, `dat`, returned by `Zmixed.nonref` only

Author(s)

Elvan Ceyhan

References

Ceyhan E, Bahadir S (2017). "Nearest Neighbor Methods for Testing Reflexivity." *Environmental and Ecological Statistics*, **24(1)**, 69-108.

See Also

`Zself.ref.ct`, `Zself.ref`, `Znnref.ct` and `Znnref`

Examples

```
n<-20
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

ipd<-ipd.mat(Y)
W<-Wmat(ipd)
Qv<-Qvec(W)$q
R<-Rval(W)
```

```
Tv<-Tval(W,R)

nvec<-as.numeric(table(cls))
rfct<-rct(ipd,cls)

Zmixed.nonref(Y,cls)
Zmixed.nonref.ct(rfct,nvec,Qv,Tv)
Zmixed.nonref(Y,cls,alt="g")

Zmixed.nonref(Y,cls,method="max")

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

ipd<-ipd.mat(Y)
W<-Wmat(ipd)
Qv<-Qvec(W)$q
R<-Rval(W)
Tv<-Tval(W,R)

nvec<-as.numeric(table(cls))
rfct<-rct(ipd,cls)

Zmixed.nonref(Y,cls,alt="g")

Zmixed.nonref.ct(rfct,nvec,Qv,Tv)
Zmixed.nonref.ct(rfct,nvec,Qv,Tv,alt="l")
```

`funsZnnref` *Z Tests for NN Reflexivity*

Description

Two functions: `Znnref.ct` and `Znnref`.

Both functions are objects of class `"refhtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of equality of the expected values of the diagonal cell counts (i.e., entries) under RL or CSR in the RCT for $k \geq 2$ classes. That is, each test performs NN reflexivity test (i.e., a test of self reflexivity and a test of mixed non-reflexivity, corresponding to entries $(1,1)$ and $(2,2)$, respectively, in the RCT) which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Ceyhan and Bahadir (2017) for more detail).

The reflexivity test is based on the normal approximation of the diagonal entries in the RCT and are due to Ceyhan and Bahadir (2017).

Each function yields the test statistics, p-values for the corresponding alternative, expected values (i.e., null value(s)), confidence intervals and sample estimates (i.e., observed values)for the self reflexivity and mixed non-reflexivity values (i.e., entries $(1,1)$ and $(2,2)$ values, respectively) in the RCT. Each function also gives names of the test statistics, null values, the description of the test, and the data set used.

The null hypothesis is that $E(N_{11}) = RP_{aa}$ and $E(N_{22}) = RP_{ab}$ in the RCT, where R is the number of reflexive NNs and P_{aa} is the probability of any two points selected are being from the same class and P_{ab} is the probability of any two points selected are being from two different classes.

The `Znnref` functions (i.e., `Znnref.ct` and `Znnref`) are different from the `Znnself` functions (i.e., `Znnself.ct` and `Znnself`) and from `Zself.ref` functions (i.e., `Zself.ref.ct` and `Zself.ref`), and also from `Znnself.sum` functions (i.e., `Znnself.sum.ct` and `Znnself.sum`). `Znnref` functions are for testing the self reflexivity and mixed non-reflexivity using the diagonal entries in the RCT while `Znnself` functions are testing the self reflexivity at a class-specific level (i.e., for each class) using the first column in the SCCT, and `Zself.ref` functions are for testing the self reflexivity for the entire data set using entry $(1,1)$ in RCT, and `Znnself.sum` functions are testing the cumulative species correspondence using the sum of the self column (i.e., the first column) in the SCCT.

Usage

```
Znnref.ct(
  rfct,
  nvec,
  Qv,
  Tv,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Znnref(
  dat,
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`rfct`	An RCT, used in `Znnref.ct` only
`nvec`	The `vector` of class sizes, used in `Znnref.ct` only
`Qv`	The number of shared NNs, used in `Znnref.ct` only
`Tv`	T value, which is the number of triplets (z_i, z_j, z_k) with "$NN(z_i) = NN(z_j) = z_k$ and $NN(z_k) = z_j$ where $NN(\cdot)$ is the nearest neighbor function, used in `Znnref.ct` only.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the difference of the off-diagonal entries, $N_{12} - N_{21}$

`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Znnref` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Znnref` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function, used in `Znnref` only

Value

A `list` with the elements

`statistic`	The Z test statistics for self reflexivity and mixed non-reflexivity, corresponding to entries $(1,1)$ and $(2,2)$ in the RCT
`stat.names`	Name of the test statistics
`p.value`	The p-values for self reflexivity and mixed non-reflexivity tests
`conf.int`	Confidence intervals for the self reflexivity and mixed non-reflexivity values (i.e., diagonal entries $(1,1)$ and $(2,2)$ values, respectively) in the RCT at the given confidence level `conf.level` and depends on the type of `alternative`.
`cnf.lvl`	Level of the onfidence intervals of the diagonal entries, provided in `conf.level`.
`estimate`	Estimates of the parameters, i.e., the observed diagonal entries $(1,1)$ and $(2,2)$ in the RCT, `rfct`.
`null.value`	Hypothesized null values for the self reflexivity and mixed non-reflexivity values (i.e., expected values of the diagonal entries $(1,1)$ and $(2,2)$ values, which are $E(N_{11}) = RP_{aa}$ and $E(N_{22}) = RP_{ab}$, respectively) in the RCT.
`null.name`	Name of the null values
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `rfct`, returned by `Znnref.ct` only
`data.name`	Name of the data set, `dat`, returned by `Znnref` only

Author(s)

Elvan Ceyhan

References

Ceyhan E, Bahadir S (2017). “Nearest Neighbor Methods for Testing Reflexivity.” *Environmental and Ecological Statistics*, **24(1)**, 69-108.

See Also

`Znnself.ct`, `Znnself`, `Zmixed.nonref.ct`, `Zmixed.nonref`, `Xsq.nnref.ct` and `Xsq.nnref`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

ipd<-ipd.mat(Y)
W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
Tv<-Tval(W,Rv)

nvec<-as.numeric(table(cls))
rfct<-rct(ipd,cls)

Znnref(Y,cls)
Znnref(Y,cls,method="max")

Znnref.ct(rfct,nvec,Qv,Tv)
Znnref.ct(rfct,nvec,Qv,Tv,alt="g")

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

ipd<-ipd.mat(Y)
W<-Wmat(ipd)
Qv<-Qvec(W)$q
R<-Rval(W)
Tv<-Tval(W,R)

nvec<-as.numeric(table(cls))
rfct<-rct(ipd,cls)

Znnref(Y,cls,alt="g")

Znnref.ct(rfct,nvec,Qv,Tv)
Znnref.ct(rfct,nvec,Qv,Tv,alt="l")
```

`funsZnnself` *Self-Reflexivity Tests with Normal Approximation*

Description

Two functions: `Znnself.ct` and `Znnself`.

Both functions are objects of class `"cellhtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of equality of the expected values of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or the expected values of the diagonal entries N_{ii} in an NNCT to the ones under RL or CSR. That is, each performs NN self reflexivity for each class test which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. NN self reflexivity is for each class can be viewed as a decomposition of species correspondence for each class. (See Ceyhan (2018) for more detail).

Each test is based on the normal approximation of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or the diagonal entries N_{ii} in an NNCT and are due to (Ceyhan 2018).

Each function yields a `vector` of length k of the test statistics, p-values for the corresponding alternative, null values (i.e., expected values), sample estimates (i.e., observed values) of self entries in the SCCT or diagonal entries in the NNCT, a $k \times 2$ matrix of confidence intervals (where each row is the confidence interval for self entry S_i in the SCCT or diagonal entry N_{ii} in the NNCT) and also names of the test statistics, estimates, null values, the description of the test, and the data set used.

The null hypothesis is that all $E[S_i] = E[N_{ii}] = n_i(n_i - 1)/(n - 1)$ where n_i is the size of class i and n is the data size.

The `Znnself` functions (i.e., `Znnself.ct` and `Znnself`) are different from the `Znnref` functions (i.e., `Znnref.ct` and `Znnref`) and from `Zself.ref` functions (i.e., `Zself.ref.ct` and `Zself.ref`) and also from `Znnself.sum` functions (i.e., `Znnself.sum.ct` and `Znnself.sum`). `Znnself` functions are testing the self reflexivity at a class-specific level (i.e., for each class) using the first column in the SCCT, while `Zself.ref` functions are for testing the self reflexivity for the entire data set using entry $(1,1)$ in RCT, and `Znnref` functions are for testing the self reflexivity and mixed non-reflexivity using the diagonal entries in the RCT, and `Znnself.sum` functions are testing the cumulative species correspondence using the sum of the self column (i.e., the first column) in the SCCT.

Usage

```
Znnself.ct(
  ct,
  VarNii,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Znnself(
  dat,
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	The NNCT or SCCT, used in `Znnself.ct` only
`VarNii`	The variance vector of differences of self entries in the SCCT or diagonal entries in the NNCT, used in `Znnself.ct` only

`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the self entries in the SCCT or diagonal entries in the NNCT
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Znnself` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Znnself` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Znnself` only

Value

A `list` with the elements

`statistic`	The `vector` (of length k) of Z test statistics for NN self reflexivity test
`stat.names`	Name of the test statistics
`p.value`	The `vector` of p-values for the hypothesis test for the corresponding alternative
`LCL`, `UCL`	Lower and Upper Confidence Levels, it is `NULL` here since we provide confidence intervals as a $k \times 2$ matrix.
`conf.int`	The $k \times 2$ matrix of confidence intervals for the estimates, (where each row is the confidence interval for self entry S_i in the SCCT or diagonal entry N_{ii} in the NNCT).
`cnf.lvl`	Level of the confidence intervals (i.e., conf.level) for the self entries in the SCCT or diagonal entries in the NNCT.
`estimate`	The `vector` of estimates of the parameters, i.e., observed values of self entries in the SCCT or diagonal entries in the NNCT.
`est.name`, `est.name2`	Names of the estimates, both are same in this function.
`null.value`	The `vector` of null values of the parameters, i.e., expected values of self entries in the SCCT or diagonal entries in the NNCT under RL or CSR.
`null.name`	Name of the null values
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Znnself.ct` only
`data.name`	Name of the data set, `dat`, returned by `Znnself` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2018). “A contingency table approach based on nearest neighbor relations for testing self and mixed correspondence.” *SORT-Statistics and Operations Research Transactions*, **42(2)**, 125-158.

See Also

Zself.ref.ct, Zself.ref, Znnref.ct, Znnref, Xsq.spec.cor and Xsq.spec.cor.ct

Examples

```
n<-20
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
VarN.diag<-varNii.ct(ct,Qv,Rv)

Znnself(Y,cls)
Znnself(Y,cls,alt="g")

Znnself.ct(ct,VarN.diag)
Znnself.ct(ct,VarN.diag,alt="g")

Znnself(Y,cls,method="max")

ct<-scct(ipd,cls)
Znnself.ct(ct,VarN.diag)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

Znnself(Y,fcls)
Znnself.ct(ct,VarN.diag)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
VarN.diag<-varNii.ct(ct,Qv,Rv)

Znnself(Y,cls,alt="l")
Znnself.ct(ct,VarN.diag)
Znnself.ct(ct,VarN.diag,alt="l")
```

`funsZnnself.sum` *Cumulative Species Correspondence Test with Normal Approximation*

Description

Two functions: `Znnself.sum.ct` and `Znnself.sum`.

Both functions are objects of class `"htest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of equality of the expected value of the sum of the self entries (i.e. first column) in a species correspondence contingency table (SCCT) or the expected values of the sum of the diagonal entries N_{ii} in an NNCT to the one under RL or CSR. That is, each performs a cumulative species correspondence test which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Ceyhan (2018) for more detail).

Each test is based on the normal approximation of the sum of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or the sum of the diagonal entries N_{ii} in an NNCT and are due to (Ceyhan 2018).

Each function yields the test statistic, p-value for the corresponding alternative, the confidence interval, sample estimate (i.e., observed value) and null (i.e., expected) value for the sum of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or the sum of the diagonal entries N_{ii} in an NNCT, and method and name of the data set used.

The null hypothesis is that all $E[S] = \sum_{i=1}^{k} n_i(n_i - 1)/(n - 1)$ where S is the sum of the self column in the SCCT, n_i is the size of class i and n is the data size.

The `Znnself.sum` functions (i.e., `Znnself.sum.ct` and `Znnself.sum`) are different from the Znnself functions (i.e., `Znnself.ct` and `Znnself`), and from the `Znnref` functions (i.e., `Znnref.ct` and `Znnref`) and also from `Zself.ref` functions (i.e., `Zself.ref.ct` and `Zself.ref`). `Znnself.sum` functions are testing the cumulative species correspondence using the sum of the self column (i.e., the first column) in the SCCT, while `Znnself` functions are testing the self reflexivity at a class-specific level (i.e., for each class) using the first column in the SCCT, while `Zself.ref` functions are for testing the self reflexivity for the entire data set using entry $(1, 1)$ in RCT, and `Znnref` functions are for testing the self reflexivity and mixed non-reflexivity using the diagonal entries in the RCT.

Usage

```
Znnself.sum.ct(
  ct,
  covSC,
  nnct = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Znnself.sum(
  dat,
  lab,
```

```
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	The NNCT or SCCT, used in `Znnself.sum.ct` only
`covSC`	The covariance matrix for the self entries (i.e., first column) in the SCCT or the diagonal entries in the NNCT, used in `Znnself.sum.ct` only. Usually output of the functions `covNii.ct` or `covNii`.
`nnct`	A logical parameter (default=`FALSE`). If `TRUE`, `x` is taken to be the $k \times k$ NNCT, and if `FALSE`, `x` is taken to be the IPD matrix, used in `Znnself.sum.ct` only
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the self entries in the SCCT or diagonal entries in the NNCT
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Znnself.sum` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Znnself.sum` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Znnself.sum` only

Value

A `list` with the elements

`statistic`	The Z test statistic for the overall species correspondence test
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the sum of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or the sum of the diagonal entries N_{ii} in an NNCT at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., the observed sum of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or the sum of the diagonal entries N_{ii} in an NNCT.
`null.value`	Hypothesized null value for the sum of the self entries (i.e., first column) in a species correspondence contingency table (SCCT) or the sum of the diagonal entries N_{ii} in an NNCT which is $E[S] = \sum_{i=1}^{k} n_i(n_i - 1)/(n-1)$ where S is the sum of the self column in the SCCT, n_i is the size of class i and n is the data size.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Znnself.sum.ct` only
`data.name`	Name of the data set, `dat`, returned by `Znnself.sum` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2018). “A contingency table approach based on nearest neighbor relations for testing self and mixed correspondence.” *SORT-Statistics and Operations Research Transactions*, **42(2)**, 125-158.

See Also

`Znnself.ct`, `Znnself`, `Znnref.ct`, `Znnref`, `Zself.ref.ct` and `Zself.ref`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-scct(ipd,cls)
ct

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)

vsq<-varNii.ct(ct,Qv,Rv)
cv<-covNii.ct(ct,vsq,Qv,Rv)

Znnself.sum(Y,cls)

Znnself.sum.ct(ct,cv)
Znnself.sum.ct(ct,cv,alt="g")

Znnself.sum(Y,cls,method="max")

ct<-nnct(ipd,cls)
Znnself.sum.ct(ct,cv,nnct = TRUE)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)
ct<-scct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)

vsq<-varNii.ct(ct,Qv,Rv)
cv<-covNii.ct(ct,vsq,Qv,Rv)
```

```
Znnself.sum(Y,cls)

Znnself.sum.ct(ct,cv)
Znnself.sum.ct(ct,cv,alt="g")

ct<-nnct(ipd,cls)
Znnself.sum.ct(ct,cv,nnct = TRUE)

Znnself.sum(Y,cls,alt="g")
```

`funsZnnsym.dx` *Dixon's Pairwise NN Symmetry Test with Normal Approximation*

Description

Two functions: `Znnsym.dx.ct` and `Znnsym.dx`.

Both functions are objects of class `"cellhtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of equality of the expected values of the off-diagonal cell counts (i.e., entries) for each pair i, j of classes under RL or CSR in the NNCT for $k \geq 2$ classes. That is, each performs Dixon's NN symmetry test which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Dixon (1994); Ceyhan (2014) for more detail).

Each symmetry test is based on the normal approximation of the difference of the off-diagonal entries in the NNCT and are due to Dixon (1994).

Each function yields a contingency table of the test statistics, p-values for the corresponding alternative, expected values (i.e., null value(s)), lower and upper confidence levels and sample estimates (i.e., observed values) for the $N_{ij} - N_{ji}$ values for $i \neq j$ (all in the upper-triangular form except for the null value, which is 0 for all pairs) and also names of the test statistics, estimates, null values, the description of the test, and the data set used.

The null hypothesis is that all $E(N_{ij}) = E(N_{ji})$ for $i \neq j$ in the $k \times k$ NNCT (i.e., symmetry in the mixed NN structure) for $k \geq 2$. In the output, the test statistic, p-value and the lower and upper confidence limits are valid for completely mapped data.

See also (Dixon (1994); Ceyhan (2014)) and the references therein.

Usage

```
Znnsym.dx.ct(
  ct,
  varS,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Znnsym.dx(
```

```
  dat,
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Znnsym.dx.ct` only
`varS`	The variance vector of differences of off-diagonal cell counts in NNCT, `ct` , usually output of var.nnsym function.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the difference of the off-diagonal entries, $N_{ij} - N_{ji}$
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Znnsym.dx` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Znnsym.dx` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Znnsym.dx` only

Value

A `list` with the elements

`statistic`	The `matrix` of Z test statistics for Dixon's NN symmetry test (in the upper-triangular form)
`stat.names`	Name of the test statistics
`p.value`	The `matrix` of p-values for the hypothesis test for the corresponding alternative (in the upper-triangular form)
`LCL`, `UCL`	Matrix of Lower and Upper Confidence Levels (in the upper-triangular form) for the $N_{ij} - N_{ji}$ values for $i \neq j$ at the given confidence level `conf.level` and depends on the type of `alternative`.
`conf.int`	The confidence interval for the estimates, it is `NULL` here, since we provide the `UCL` and `LCL` in `matrix` form.
`cnf.lvl`	Level of the upper and lower confidence limits (i.e., conf.level) of the differences of the off-diagonal entries.
`estimate`	Estimates of the parameters, i.e., matrix of the difference of the off-diagonal entries (in the upper-triangular form) of the $k \times k$ NNCT, $N_{ij} - N_{ji}$ for $i \neq j$.
`est.name`, `est.name2`	Names of the estimates, former is a shorter description of the estimates than the latter.
`null.value`	Hypothesized null value for the expected difference between the off-diagonal entries, $E(N_{ij}) - E(N_{ji})$ for $i \neq j$ in the $k \times k$ NNCT, which is 0 for this function.

`null.name`	Name of the null values
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Znnsym.dx.ct` only
`data.name`	Name of the data set, `dat`, returned by `Znnsym.dx` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Dixon PM (1994). "Testing spatial segregation using a nearest-neighbor contingency table." *Ecology*, **75(7)**, 1940-1948.

See Also

`Znnsym2cl.dx.ct`, `Znnsym2cl.dx`, `Znnsym.ss.ct`, `Znnsym.ss`, `Xsq.nnsym.dx.ct` and `Xsq.nnsym.dx`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv) #default is byrow

varS<-var.nnsym(covN)

Znnsym.dx(Y,cls)
Znnsym.dx.ct(ct,varS)

Znnsym.dx(Y,cls,method="max")

Znnsym.dx(Y,cls,alt="g")
Znnsym.dx.ct(ct,varS,alt="g")

#cls as a factor
na<-floor(n/2); nb<-n-na
```

```
fcls<-rep(c("a","b"),c(na,nb))
Znnsym.dx(Y,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv) #default is byrow

varS<-var.nnsym(covN)

Znnsym.dx(Y,cls)
Znnsym.dx.ct(ct,varS)
```

`funsZnnsym.ss` *Pielou's Pairwise NN Symmetry Test with Normal Approximation (for Sparse Sampling)*

Description

Two functions: `Znnsym.ss.ct` and `Znnsym.ss`.

Both functions are objects of class `"cellhtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of equality of the expected values of the off-diagonal cell counts (i.e., entries) for each pair i, j of classes under RL or CSR in the NNCT for $k \geq 2$ classes. That is, each performs Pielou's first type of NN symmetry test which is appropriate (i.e., have the appropriate asymptotic sampling distribution) provided that data is obtained by sparse sampling. (See Ceyhan (2014) for more detail).

Each symmetry test is based on the normal approximation of the differences of the off-diagonal entries in the NNCT and are due to Pielou (1961).

Each function yields a contingency table of the test statistics, p-values for the corresponding alternative, expected values, lower and upper confidence levels, sample estimates (i.e., observed values) and null value(s) (i.e., expected values) for the $N_{ij} - N_{ji}$ values for $i \neq j$ (all in the upper-triangular form except for the null value, which is 0 for all pairs) and also names of the test statistics, estimates, null values, the description of the test, and the data set used.

The null hypothesis is that all $E(N_{ij}) = E(N_{ji})$ for $i \neq j$ in the $k \times k$ NNCT (i.e., symmetry in the mixed NN structure) for $k \geq 2$. In the output, the test statistic, p-value and the lower and upper confidence limits are valid only for (properly) sparsely sampled data.

See also (Pielou (1961); Ceyhan (2014)) and the references therein.

Usage

```
Znnsym.ss.ct(
  ct,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Znnsym.ss(
  dat,
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Znnsym.ss.ct` only
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the difference of the off-diagonal entries, $N_{ij} - N_{ji}$
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Znnsym.ss` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Znnsym.ss` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Znnsym.ss` only

Value

A `list` with the elements

`statistic`	The `matrix` of Z test statistics for Pielou's first type of NN symmetry test (in the upper-triangular form)
`stat.names`	Name of the test statistics
`p.value`	The `matrix` of p-values for the hypothesis test for the corresponding alternative (in the upper-triangular form)
`LCL`, `UCL`	Matrix of Lower and Upper Confidence Levels (in the upper-triangular form) for the $N_{ij} - N_{ji}$ values for $i \neq j$ at the given confidence level `conf.level` and depends on the type of `alternative`.
`conf.int`	The confidence interval for the estimates, it is `NULL` here, since we provide the `UCL` and `LCL` in `matrix` form.
`cnf.lvl`	Level of the upper and lower confidence limits (i.e., conf.level) of the differences of the off-diagonal entries.
`estimate`	Estimates of the parameters, i.e., matrix of the difference of the off-diagonal entries (in the upper-triangular form) of the $k \times k$ NNCT, $N_{ij} - N_{ji}$ for $i \neq j$.

`est.name, est.name2`	Names of the estimates, former is a shorter description of the estimates than the latter.
`null.value`	Hypothesized null value for the expected difference between the off-diagonal entries, $E(N_{ij}) - E(N_{ji})$ for $i \neq j$ in the $k \times k$ NNCT, which is 0 for this function.
`null.name`	Name of the null values
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Znnsym.ss.ct` only
`data.name`	Name of the data set, `dat`, returned by `Znnsym.ss` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Pielou EC (1961). "Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships." *Journal of Ecology*, **49(2)**, 255-269.

See Also

`Znnsym.dx.ct`, `Znnsym.dx`, `Znnsym2cl.ss.ct` and `Znnsym2cl.ss`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

Znnsym.ss(Y,cls)
Znnsym.ss.ct(ct)

Znnsym.ss(Y,cls,method="max")

Znnsym.ss(Y,cls,alt="g")
Znnsym.ss.ct(ct,alt="g")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
```

```
Znnsym.ss(Y,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

Znnsym.ss(Y,cls)
Znnsym.ss.ct(ct)
```

`funsZnnsym2cl.dx` *Dixon's NN Symmetry Test with Normal Approximation for Two Classes*

Description

Two functions: `Znnsym2cl.dx.ct` and `Znnsym2cl.dx`.

Both functions are objects of class `"htest"` but with different arguments (see the parameter list below). Each one performs the hypothesis test of equality of the expected value of the off-diagonal cell counts (i.e., entries) under RL or CSR in the NNCT for $k = 2$ classes. That is, each performs Dixon's NN symmetry test which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Ceyhan (2014) for more detail).

Each symmetry test is based on the normal approximation of the difference of the off-diagonal entries in the NNCT and are due to Dixon (1994).

Each function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is the difference of the off-diagonal entries in the NNCT), and method and name of the data set used.

The null hypothesis is that all $E(N_{12}) = E(N_{21})$ in the 2×2 NNCT (i.e., symmetry in the mixed NN structure).

See also (Dixon (1994); Ceyhan (2014)) and the references therein.

Usage

```
Znnsym2cl.dx.ct(
  ct,
  Q,
  R,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Znnsym2cl.dx(
  dat,
```

```
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Znnsym2cl.dx.ct` only
`Q`	The number of shared NNs, used in `Znnsym2cl.dx.ct` only
`R`	The number of reflexive NNs (i.e., twice the number of reflexive NN pairs), used in `Znnsym2cl.dx.ct` only
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the difference of the off-diagonal entries, $N_{12} - N_{21}$
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Znnsym2cl.dx` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Znnsym2cl.dx` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Znnsym2cl.dx` only

Value

A `list` with the elements

`statistic`	The Z test statistic for Pielou's first type of NN symmetry test
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the difference of the off-diagonal entries, $N_{12} - N_{21}$ in the 2×2 NNCT at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate, i.e., the difference of the off-diagonal entries of the 2×2 NNCT, $N_{12} - N_{21}$.
`null.value`	Hypothesized null value for the expected difference between the off-diagonal entries, $E(N_{12}) - E(N_{21})$ in the 2×2 NNCT, which is 0 for this function.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `dat`, or name of the contingency table, `ct`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Dixon PM (1994). "Testing spatial segregation using a nearest-neighbor contingency table." *Ecology*, **75(7)**, 1940-1948.

See Also

`Znnsym2cl.ss.ct`, `Znnsym2cl.ss`, `Znnsym.dx.ct`, `Znnsym.dx`, `Xsq.nnsym.dx.ct` and `Xsq.nnsym.dx`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)

Znnsym2cl.dx(Y,cls)
Znnsym2cl.dx.ct(ct,Qv,Rv)

Znnsym2cl.dx(Y,cls,method="max")

Znnsym2cl.dx(Y,cls,alt="g")
Znnsym2cl.dx.ct(ct,Qv,Rv,alt="g")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
Znnsym2cl.dx(Y,fcls)

#############
ct<-matrix(sample(1:20,4),ncol=2)
Znnsym2cl.dx.ct(ct,Qv,Rv) #gives an error message if ct<-matrix(sample(1:20,9),ncol=3)
#here, Qv and Rv values are borrowed from above, to highlight a point
```

`funsZnnsym2cl.ss` *Pielou's First Type of NN Symmetry Test with Normal Approximation for Two Classes (for Sparse Sampling)*

Description

Two functions: `Znnsym2cl.ss.ct` and `Znnsym2cl.ss`.

Both functions are objects of class `"htest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of equality of the expected value of the off-diagonal cell counts (i.e., entries) under RL or CSR in the NNCT for $k = 2$ classes. That is, each performs Pielou's first type of NN symmetry test which is appropriate (i.e., have the appropriate asymptotic sampling distribution) provided that data is obtained by sparse sampling. (See Ceyhan (2014) for more detail).

Each symmetry test is based on the normal approximation of the difference of the off-diagonal entries in the NNCT and are due to Pielou (1961).

Each function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is the difference of the off-diagonal entries in the NNCT), and method and name of the data set used.

The null hypothesis is that $E(N_{12}) = E(N_{21})$ in the 2×2 NNCT (i.e., symmetry in the mixed NN structure). In the output, the test statistic, p-value and the confidence interval are valid only for (properly) sparsely sampled data.

See also (Pielou (1961); Ceyhan (2014)) and the references therein.

Usage

```
Znnsym2cl.ss.ct(
  ct,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Znnsym2cl.ss(
  dat,
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Znnsym2cl.ss.ct` only
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the difference of the off-diagonal entries, $N_{12} - N_{21}$
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Znnsym2cl.ss` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Znnsym2cl.ss` only

`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Znnsym2cl.ss` only

Value

A `list` with the elements

`statistic`	The Z test statistic for Pielou's first type of NN symmetry test
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the difference of the off-diagonal entries, $N_{12} - N_{21}$ in the 2×2 NNCT at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate, i.e., the difference of the off-diagonal entries of the 2×2 NNCT, $N_{12} - N_{21}$.
`null.value`	Hypothesized null value for the expected difference between the off-diagonal entries, $E(N_{12}) - E(N_{21})$ in the 2×2 NNCT, which is 0 for this function.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `dat`, or name of the contingency table, `ct`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Pielou EC (1961). "Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships." *Journal of Ecology*, **49(2)**, 255-269.

See Also

`Xsq.nnsym.ss.ct`, `Xsq.nnsym.ss`, `Znnsym.ss.ct` and `Znnsym.ss`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

Znnsym2cl.ss(Y,cls)
Znnsym2cl.ss.ct(ct)
```

```
Znnsym2cl.ss(Y,cls,method="max")

Znnsym.ss.ct(ct)

Znnsym2cl.ss(Y,cls,alt="g")
Znnsym2cl.ss.ct(ct,alt="g")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
Znnsym2cl.ss(Y,fcls)

#############
ct<-matrix(sample(1:20,4),ncol=2)
Znnsym2cl.ss.ct(ct) #gives an error message if ct<-matrix(sample(1:20,9),ncol=3)
```

`funsZseg.coeff` *Z Tests for Segregation Coefficients*

Description

Two functions: `Zseg.coeff.ct` and `Zseg.coeff`.

Both functions are objects of class `"cellhtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of deviations of segregation coefficients from their expected values under RL or CSR for each segregation coefficient in the NNCT.

The test for each cell i, j is based on the normal approximation of the corresponding segregation coefficient. That is, each performs the segregation coefficient tests which are appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. The segregation coefficients in the multi-class case are the extension of Pielou's segregation coefficient for the two-class case. (See Ceyhan (2014) for more detail).

Each function yields a contingency table of the test statistics, p-values for the corresponding alternative, lower and upper confidence levels, sample estimates (i.e., observed values) and null value (i.e., expected value, which is 0) for the segregation coefficients and also names of the test statistics, estimates, null value, the description of the test, and the data set used.

The null hypothesis for each cell i, j is that the corresponding segregation coefficient equal to the expected value (which is 0) under RL or CSR.

See also (Ceyhan (2014)).

Usage

```
Zseg.coeff.ct(
  ct,
  VarSC,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

```
Zseg.coeff(
  dat,
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Zseg.coeff.ct` only
`VarSC`	The variance matrix for the segregation coefficients in the NNCT, `ct`; used in `Zseg.coeff.ct` only
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the segregation coefficients
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Zseg.coeff` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Zseg.coeff` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. used in `Zseg.coeff` only

Value

A `list` with the elements

`statistic`	The `matrix` of test statistics for the segregation coefficients
`stat.names`	Name of the test statistics
`p.value`	The `matrix` of p-values for the hypothesis test for the corresponding alternative
`LCL`, `UCL`	Matrix of lower and upper confidence levels for the segregation coefficients at the given confidence level `conf.level` and depends on the type of `alternative`.
`conf.int`	Confidence interval for segregation coefficients, it is `NULL` here since we provide the upper and lower confidence limits as $k \times k$ matrices.
`cnf.lvl`	Level of the upper and lower confidence limits of the segregation coefficients, provided in `conf.level`.
`estimate`	Estimate of the parameter, i.e., matrix of the observed segregation coefficients
`est.name`, `est.name2`	Names of the estimates, both are same in this function
`null.value`	Hypothesized null values for the parameters, i.e., expected values of the segregation coefficients, which are all 0 under RL or CSR.
`null.name`	Name of the null value
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`

`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Zseg.coeff.ct` only
`data.name`	Name of the data set, `dat`, returned by `Zseg.coeff` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`seg.coeff` and `Zseg.ind`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

varT<-var.seg.coeff(ct,covN)

Zseg.coeff(Y,cls)
Zseg.coeff.ct(ct,varT)

Zseg.coeff(Y,cls,method="max")

Zseg.coeff(Y,cls,alt="g")
Zseg.coeff.ct(ct,varT,alt="g")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

Zseg.coeff.ct(ct,varT)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
```

```
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

varT<-var.seg.coeff(ct,covN)

Zseg.coeff(Y,cls)
Zseg.coeff.ct(ct,varT)

Zseg.coeff(Y,cls,alt="g")
Zseg.coeff.ct(ct,varT,alt="g")
```

`funsZsegind` *Z Tests for Segregation Indices*

Description

Two functions: `Zseg.ind.ct` and `Zseg.ind`.

Both functions are objects of class `"cellhtest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of deviations of segregation indices from their expected values under RL or CSR for each segregation index in the NNCT. The test for each cell i, j is based on the normal approximation of the corresponding segregation index.

Each function yields a contingency table of the test statistics, p-values for the corresponding alternative, lower and upper confidence levels, sample estimates (i.e., observed values) and null value(s) (i.e., expected values) for the segregation indices and also names of the test statistics, estimates, null value, the description of the test, and the data set used.

The null hypothesis for each cell i, j is that the corresponding segregation index equal to the expected value (which is 0) under RL or CSR.

See also (Ceyhan (2014)).

Usage

```
Zseg.ind.ct(
  ct,
  varN,
  inf.corr = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

```
Zseg.ind(
  dat,
  lab,
  inf.corr = FALSE,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`ct`	A nearest neighbor contingency table, used in `Zseg.ind.ct` only
`varN`	The variance matrix for cell counts in the NNCT, `ct`; used in `Zseg.ind.ct` only
`inf.corr`	A logical argument (default=`FALSE`). If `TRUE`, indices are modified so that they are finite and if `FALSE` the above definition in the description is used.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the segregation indices
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Zseg.ind` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Zseg.ind` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function, used in `Zseg.ind` only

Value

A `list` with the elements

`statistic`	The `matrix` of test statistics for the segregation indices
`stat.names`	Name of the test statistics
`p.value`	The `matrix` of p-values for the hypothesis test for the corresponding alternative
`LCL`, `UCL`	Matrix of lower and upper confidence levels for the segregation indices at the given confidence level `conf.level` and depends on the type of `alternative`.
`cnf.lvl`	Level of the upper and lower confidence limits of the segregation indices, provided in `conf.level`.
`estimate`	Estimate of the parameter, i.e., matrix of the observed segregation indices
`est.name`, `est.name2`	Names of the estimates, both are same in this function
`null.value`	Hypothesized values for the parameters, i.e., the null values of the segregation indices, which are all 0 under RL or CSR.
`null.name`	Name of the null value
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`

`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `ct`, returned by `Zseg.ind.ct` only
`data.name`	Name of the data set, `dat`, returned by `Zseg.ind` only

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Segregation indices for disease clustering.” *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`seg.ind` and `Zseg.coeff`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

seg.ind(ct)
seg.ind(ct,inf.corr=TRUE)
W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
varN

Zseg.ind(Y,cls)
Zseg.ind(Y,cls,inf.corr=TRUE)
Zseg.ind.ct(ct,varN)

Zseg.ind(Y,cls,alt="g")
Zseg.ind.ct(ct,varN,alt="g")

Zseg.ind(Y,cls,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
Zseg.ind(Y,cls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
```

```
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
varN

Zseg.ind(Y,cls)
Zseg.ind(Y,cls,inf.corr = TRUE)

Zseg.ind.ct(ct,varN)
Zseg.ind.ct(ct,varN,inf.corr = TRUE)

#1D data points
n<-20  #or try sample(1:20,1)
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)

Zseg.ind(X,cls)
Zseg.ind.ct(ct,varN)
Zseg.ind.ct(ct,varN,inf.corr=TRUE)
```

`funsZself.ref` *Self-Reflexivity Test with Normal Approximation*

Description

Two functions: `Zself.ref.ct` and `Zself.ref`.

Both functions are objects of class `"htest"` but with different arguments (see the parameter list below). Each one performs hypothesis tests of self reflexivity in the NN structure using the number of self-reflexive NN pairs (i.e., the first diagonal entry, $(1,1)$) in the RCT for $k \geq 2$ classes. That is, each test performs a test of self reflexivity corresponding to entry $(1,1)$ in the RCT) which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data. (See Ceyhan and Bahadir (2017) for more detail).

The self reflexivity test is based on the normal approximation of the diagonal entry $(1,1)$ in the RCT and are due to Ceyhan and Bahadir (2017).

Each function yields the test statistic, p-value for the corresponding alternative, the confidence interval, sample estimate (i.e., observed value) and null (i.e., expected) value for the self reflexivity value (i.e., diagonal entry $(1, 1)$ value, respectively) in the RCT, and method and name of the data set used.

The null hypothesis is that $E(N_{11}) = RP_{aa}$ in the RCT, where R is the number of reflexive NNs and P_{aa} is the probability of any two points selected are being from the same class.

The `Zself.ref` functions (i.e., `Zself.ref.ct` and `Zself.ref`) are different from the `Znnref` functions (i.e., `Znnref.ct` and `Znnref`) and from `Znnself` functions (i.e., `Znnself.ct` and `Znnself`), and also from `Znnself.sum` functions (i.e., `Znnself.sum.ct` and `Znnself.sum`). `Zself.ref` functions are for testing the self reflexivity for the entire data set using entry $(1, 1)$ in RCT while `Znnself` functions are testing the self reflexivity at a class-specific level (i.e., for each class) using the first column in the SCCT, `Znnref` functions are for testing the self reflexivity and mixed non-reflexivity using the diagonal entries in the RCT, and `Znnself.sum` functions are testing the cumulative species correspondence using the sum of the self column (i.e., the first column) in the SCCT.

Usage

```
Zself.ref.ct(
  rfct,
  nvec,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)

Zself.ref(
  dat,
  lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`rfct`	An RCT, used in `Zself.ref.ct` only
`nvec`	The `vector` of class sizes, used in `Zself.ref.ct` only
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the difference of the off-diagonal entries, $N_{12} - N_{21}$
`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in `Zself.ref` only
`lab`	The `vector` of class labels (numerical or categorical), used in `Zself.ref` only
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function, used in `Zself.ref` only

Value

A `list` with the elements

`statistic`	The Z test statistic for self reflexivity corresponding to entry $(1,1)$ in the RCT
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the self reflexivity value (i.e., diagonal entry $(1,1)$ value) in the RCT at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., the observed diagonal entry $(1,1)$ in the RCT, `rfct`.
`null.value`	Hypothesized null value for the self reflexivity value (i.e., expected value of the diagonal entry $(1,1)$ which is $E(N_{11}) = RP_{aa}$) in the RCT.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`ct.name`	Name of the contingency table, `rfct`, returned by `Zself.ref.ct` only
`data.name`	Name of the data set, `dat`, returned by `Zself.ref` only

Author(s)

Elvan Ceyhan

References

Ceyhan E, Bahadir S (2017). “Nearest Neighbor Methods for Testing Reflexivity.” *Environmental and Ecological Statistics*, **24(1)**, 69-108.

See Also

`Znnref.ct`, `Znnref`, `Zmixed.nonref.ct` and `Zmixed.nonref`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)

nvec<-as.numeric(table(cls))
rfct<-rct(ipd,cls)

Zself.ref(Y,cls)
Zself.ref(Y,cls,method="max")

Zself.ref.ct(rfct,nvec)
Zself.ref.ct(rfct,nvec,alt="g")

#############
```

```
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)

nvec<-as.numeric(table(cls))
rfct<-rct(ipd,cls)

Zself.ref(Y,cls,alt="g")

Zself.ref.ct(rfct,nvec)
Zself.ref.ct(rfct,nvec,alt="l")
```

`funsZTkinv` *Z-Test for Cuzick and Edwards T_k^inv statistic*

Description

Two functions: `ZTkinv` and `ZTkinv.sim`, each of which is an object of class `"htest"` performing a z-test for Cuzick and Edwards T_k^{inv} test statistic. See `ceTkinv` for a description of T_k^{inv} test statistic.

The function `ZTkinv` performs a Z-test for T_k^{inv} using asymptotic normality with a simulation estimated variance under RL of cases and controls to the given points. And the function `ZTkinv.sim` performs test forT_k^{inv} based on MC simulations under the RL hypothesis.

Asymptotic normality for the T_k^{inv} is not established yet, but this seems likely according to Cuzick and Edwards (1990). If asymptotic normality holds, it seems a larger sample size would be needed before this becomes an effective approximation. Hence the simulation-based test `ZTkinv.sim` is recommended for use to be safe. When `ZTkinv` is used, this is also highlighted with the warning "asymptotic normality of T_k^{inv} is not yet established, so, simulation-based test is recommended".

All arguments are common for both functions, except for ..., Nvar.sim which are used in `ZTkinv` only, and `Nsim`, which is used in `ZTkinv.sim` only.

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0's and 1's accordingly. The argument `Nvar.sim` represents the number of resamplings (without replacement) in the RL scheme, with default being `1000` for estimating the variance of T_k^{inv} statistic in `ZTkinv`. The argument `Nsim` represents the number of resamplings (without replacement) in the RL scheme, with default being `1000` for estimating the T_k^{inv} values in `ZTkinv.sim`.

Both functions might take a very long time when data size is large or `Nsim` is large.

See also (Cuzick and Edwards (1990)) and the references therein.

Usage

```
ZTkinv(
  dat,
  k,
```

```
  cc.lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  case.lab = NULL,
  Nvar.sim = 1000,
  ...
)

ZTkinv.sim(
  dat,
  k,
  cc.lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  case.lab = NULL,
  Nsim = 1000
)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point, used in both functions.
`k`	Integer specifying the number of the closest controls to subject i, used in both functions.
`cc.lab`	Case-control labels, 1 for case, 0 for control, used in both functions.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`, used in both functions.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for Cuzick and Edwards T_k^{inv} statistic. Used in both functions.
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`, used in both functions.
`Nvar.sim`	The number of simulations, i.e., the number of resamplings under the RL scheme to estimate the variance of Tkinv, used in `ZTkinv` only.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function. Used in `ZTkinv` only.
`Nsim`	The number of simulations, i.e., the number of resamplings under the RL scheme to estimate the T_k^{inv} values, used in `ZTkinv.sim` only.

Value

A `list` with the elements

`statistic`	The Z test statistic for the Cuzick and Edwards T_k^{inv} test
`p.value`	The p-value for the hypothesis test for the corresponding alternative. In `ZTkinv` this is computed using the standard normal distribution, while in `ZTkinv.sim`, it is based on which percentile the observed T_k^{inv} value is among the generated T_k^{inv} values.

`conf.int`	Confidence interval for the Cuzick and Edwards T_k^{inv} value at the given confidence level `conf.level` and depends on the type of `alternative`.

z-critical values are used in the construction of the confidence interval in `ZTkinv`, while the percentile values are used in the generated sample of T_k^{inv} values in `ZTkinv.sim`

`estimate`	Estimate of the parameter, i.e., the Cuzick and Edwards T_k^{inv} value.
`null.value`	Hypothesized null value for the Cuzick and Edwards T_k^{inv} value which is $kn_1(n_1-1)/(n_0+1)$ under RL, where the number of cases are denoted as n_1 and number of controls as n_0.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `dat`

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`ceTkinv` and `EV.Tkinv`

Examples

```
n<-10 #try also 20, 50, 100
set.seed(123)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
k<-2

ZTkinv(Y,k,cls)
ZTkinv(Y,k,cls+1,case.lab = 2,alt="l")
#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ZTkinv(Y,k,fcls,case.lab="a")

n<-10 #try also 20, 50, 100
set.seed(123)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
k<-2 # try also 3,5

ZTkinv.sim(Y,k,cls)
ZTkinv.sim(Y,k,cls,conf=.9,alt="g")
```

```
#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ZTkinv.sim(Y,k,fcls,case.lab="a")

#with k=1
ZTkinv.sim(Y,k=1,cls)
ZTrun(Y,cls)
```

`ind.nnsym` *Index Matrix for Computing the Covariance of Dixon's Overall NN Symmetry Test*

Description

Returns the index matrix for choosing the entries in the covariance matrix for NNCT used for computing the covariance for Dixon's NN symmetry test. The matrix is $k(k-1)/2 \times 2$ with each row is the i, j corresponding to N_{ij} in the NNCT.

Usage

```
ind.nnsym(k)
```

Arguments

`k`	An integer specifying the number of classes in the data set

Value

The $k(k-1)/2 \times 2$ index matrix with each row is the i, j corresponding to N_{ij} in the NNCT

Author(s)

Elvan Ceyhan

See Also

`cov.nnsym` and `ind.seg.coeff`

`ind.seg.coeff` *Index Matrix for Computing the Covariance of Segregation Coefficients*

Description

Returns the index matrix for choosing the entries in the covariance matrix for NNCT used for computing the covariance for the extension of Pielou's segregation coefficient to the multi-class case. The matrix is $k(k+1)/2 \times 2$ with each row is the i, j corresponding to N_{ij} in the NNCT.

Usage

```
ind.seg.coeff(k)
```

Arguments

`k`	An integer specifying the number of classes in the data set

Value

The $k(k+1)/2 \times 2$ index matrix with each row is the i, j corresponding to N_{ij} in the NNCT

Author(s)

Elvan Ceyhan

See Also

`cov.seg.coeff`, `seg.coeff` and `ind.nnsym`

`ipd.mat` *Interpoint Distance Matrix*

Description

This function computes and returns the distance matrix computed by using the specified distance measure to compute the distances between the rows of the set of points `x` and `y` using the `dist` function in the `stats` package of the standard R distribution. If `y` is provided (default=`NULL`) it yields a matrix of distances between the rows of `x` and rows of `y`. Otherwise, it provides a square matrix with i, j-th entry being the distance between row i and row j of `x`. This function is different from the `dist` function in the `stats` package. `dist` returns the distance matrix in a lower triangular form, and `ipd.mat` returns in a full matrix. ...are for further arguments, such as `method` and `p`, passed to the `dist` function.

Usage

```
ipd.mat(x, y = NULL, ...)
```

Arguments

`x`	A set of points in matrix or data frame form where points correspond to the rows.
`y`	A set of points in matrix or data frame form where points correspond to the rows (default=`NULL`).
`...`	Additional parameters to be passed on the `dist` function.

Value

A distance matrix whose i, j-th entry is the distance between row i of `x` and row j of `y` if `y` is provided, otherwise i, j-th entry is the distance between rows i and j of `x`.

Author(s)

Elvan Ceyhan

See Also

`dist`, `ipd.mat.euc`, and `dist.std.data`

Examples

```
#3D data points
n<-3
X<-matrix(runif(3*n),ncol=3)
mtd<-"euclidean" #try also "maximum", "manhattan", "canberra", "binary"
ipd.mat(X,method=mtd)
ipd.mat(X,method="minkowski",p=6)

n<-5
Y<-matrix(runif(3*n),ncol=3)
ipd.mat(X,Y,method=mtd)
ipd.mat(X[1,],Y,method=mtd)
ipd.mat(c(.1,.2,.3),Y,method=mtd)
ipd.mat(X[1,],Y[3,],method=mtd)

#1D data points
X<-as.matrix(runif(3)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(3) would not work
ipd.mat(X)

Y<-as.matrix(runif(5))
ipd.mat(X,Y)
ipd.mat(X[1,],Y)
ipd.mat(X[1,],Y[3,])
```

---

ipd.mat.euc *Euclidean Interpoint Distance Matrix*

Description

Returns the Euclidean interpoint distance (IPD) matrix of a given the set of points `x` and `y` using two for loops with the `euc.dist` function of the current package. If `y` is provided (default=`NULL`) it yields a matrix of Euclidean distances between the rows of `x` and rows of `y`, otherwise it provides a square matrix with i,j-th entry being the Euclidean distance between row i and row j of `x`. This function is different from the `ipd.mat` function in this package. `ipd.mat` returns the full distance matrix for a variety of distance metrics (including the Euclidean metric), while `ipd.mat.euc` uses the Euclidean distance metric only. `ipd.mat.euc(X)` and `ipd.mat(X)` yield the same output for a set of points `X`, as the default metric in `ipd.mat` is also `"euclidean"`.

Usage

```
ipd.mat.euc(x, y = NULL)
```

Arguments

`x`	A set of points in matrix or data frame form where points correspond to the rows.
`y`	A set of points in matrix or data frame form where points correspond to the rows (default=`NULL`).

Value

A distance matrix whose i,j-th entry is the Euclidean distance between row i of `x` and row j of `y` if `y` is provided, otherwise i,j-th entry is the Euclidean distance between rows i and j of `x`.

Author(s)

Elvan Ceyhan

See Also

`dist`, `ipd.mat.euc`, and `dist.std.data`

Examples

```
#3D data points
n<-3
X<-matrix(runif(3*n),ncol=3)
ipd.mat.euc(X)

n<-5
Y<-matrix(runif(3*n),ncol=3)
ipd.mat.euc(X,Y)
```

```
ipd.mat.euc(X[1,],Y)
ipd.mat.euc(c(.1,.2,.3),Y)
ipd.mat.euc(X[1,],Y[3,])

#1D data points
X<-as.matrix(runif(3)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(3) would not work
ipd.mat.euc(X)

Y<-as.matrix(runif(5))
ipd.mat.euc(X,Y)
ipd.mat.euc(X[1,],Y)
ipd.mat.euc(X[1,],Y[3,])
```

`kNN` *Finding the indices of the* `k` *NNs of a given point*

Description

Returns the indices of the `k` nearest neighbors of subject i given data set or IPD matrix `x`. Subject indices correspond to rows (i.e., rows `1:n`) if `x` is the data set and to rows or columns if `x` is the IPD matrix.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

Usage

```
kNN(x, i, k, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`i`	index of (i.e., row number for) the subject whose NN is to be found.
`k`	Integer specifying the number of NNs (of subject i).
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the indices (i.e., row numbers) of the `k` NNs of subject i

Author(s)

Elvan Ceyhan

See Also

NN, NNdist, and NNdist2cl

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
k<-sample(1:5,1)
k
NN(ipd,1)
kNN(ipd,1,k)
kNN(Y,1,k,is.ipd = FALSE)
kNN(Y,1,k,is.ipd = FALSE,method="max")

NN(ipd,5)
kNN(ipd,5,k)
kNN(Y,5,k,is.ipd = FALSE)

#1D data points
X<-as.matrix(runif(15)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(5) would not work
ipd<-ipd.mat(X)
kNN(ipd,3,k)

#with possible ties in the data
Y<-matrix(round(runif(30)*10),ncol=3)
ny<-nrow(Y)
ipd<-ipd.mat(Y)
for (i in 1:ny)
  cat(i,":",kNN(ipd,i,k),"\n")
```

`mat2vec` *Conversion of a Matrix to a Vector*

Description

Converts the contingency table (or any matrix) `ct` to a `vector` by default row-wise (i.e., by appending each row one after the other) or column-wise, and also returns the entry indices (in the original matrix `ct`) in a $k^2 \times 2$ matrix

Usage

```
mat2vec(ct, byrow = TRUE)
```

Arguments

`ct`	A matrix, in particular a contingency table
`byrow`	A logical argument (default=`TRUE`). If `TRUE`, rows of `ct` are appended to obtain the vector and if `FALSE` columns of `ct` are appended to obtain the vector.

Value

A `list` with two elements

`vec`	The `vectorized` form the matrix `ct`, by default appending the rows of `ct`
`ind`	The $k^2 \times 2$ matrix of entry indices (in the original matrix `ct`) whose i-th row corresponds to the i-th entry in `vec`.

Author(s)

Elvan Ceyhan

See Also

`ind.nnsym` and `ind.seg.coeff`,

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct
mat2vec(ct)
mat2vec(ct,byrow=FALSE)

#an arbitrary 3x3 matrix
M<-matrix(sample(10:20,9),ncol=3)
M
mat2vec(M)
mat2vec(M,byrow=FALSE)
```

`matrix.sqrt` *Square root of a matrix*

Description

Computes the square root of the matrix A, where A does not have to be a square matrix, when the square root exists. See https://people.orie.cornell.edu/davidr/SDAFE2/Rscripts/SDAFE2.R

Usage

```
matrix.sqrt(A)
```

Arguments

`A`	A matrix, not necessarily square

Value

Returns the square root of A, if exists, otherwise gives an error message.

Author(s)

Elvan Ceyhan

Examples

```
A<-matrix(sample(20:40,4),ncol=2)
matrix.sqrt(A)

A<-matrix(sample(20:40,16),ncol=4)
matrix.sqrt(A)
#sqrt of inverse of A, or sqrt inverse of A
matrix.sqrt(solve(A))

#non-square matrix
A<-matrix(sample(20:40,20),ncol=4)
matrix.sqrt(A)
```

`Ninv` *Vector of Shared NNs and Number of Reflexive NNs*

Description

Returns the `Qvec` and `R` where $Qvec = (Q_0, Q_1, \ldots)$ with Q_j is the number of points shared as a NN by j other points i.e., number of points that are NN of i points, for $i = 0, 1, 2, \ldots$ and `R` is the number of reflexive pairs where points A and B are reflexive iff they are NN to each other.

Usage

```
Ninv(x, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns a `list` with two elements

`Qvec`	vector of Q_j values
`R`	number of reflexive points

Author(s)

Elvan Ceyhan

See Also

`Qval`, `Qvec`, `sharedNN`, `Rval`, and `QRval`

Examples

```
#3D data points
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd)
sharedNN(W)
Qvec(W)
Ninv(ipd)
Ninv(Y,is.ipd = FALSE)
Ninv(Y,is.ipd = FALSE,method="max")

#1D data points
n<-15
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
W<-Wmat(ipd)
sharedNN(W)
Qvec(W)
Ninv(ipd)

#with possible ties in the data
Y<-matrix(round(runif(30)*10),ncol=3)
ny<-nrow(Y)
ipd<-ipd.mat(Y)
```

```
W<-Wmat(ipd)
sharedNN(W)
Qvec(W)
Ninv(ipd)
```

`NN` *Finding the index of the NN of a given point*

Description

Returns the index (or indices) of the nearest neighbor(s) of subject i given data set or IPD matrix `x`. It will yield a `vector` if there are ties, and subject indices correspond to rows (i.e., rows `1:n`) if `x` is the data set and to rows or columns if `x` is the IPD matrix.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

Usage

```
NN(x, i, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`i`	index of (i.e., row number for) the subject whose NN is to be found.
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the index (indices) i.e., row number(s) of the NN of subject i

Author(s)

Elvan Ceyhan

See Also

`kNN` and `NNsub`

Examples

```
#3D data points
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
NN(ipd,1)
NN(Y,1,is.ipd = FALSE)
NN(ipd,5)
NN(Y,5,is.ipd = FALSE)
NN(Y,5,is.ipd = FALSE,method="max")

#1D data points
X<-as.matrix(runif(15)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(5) would not work
ipd<-ipd.mat(X)
NN(ipd,1)
NN(ipd,5)

#with possible ties in the data
Y<-matrix(round(runif(30)*10),ncol=3)
ny<-nrow(Y)
ipd<-ipd.mat(Y)
for (i in 1:ny)
  cat(i,":",NN(ipd,i),"|",NN(Y,i,is.ipd = FALSE),"\n")
```

`nnct` *Nearest Neighbor Contingency Table (NNCT)*

Description

Returns the $k \times k$ NNCT given the IPD matrix or data set `x` where k is the number of classes in the data set. Rows and columns of the NNCT are labeled with the corresponding class labels.

The argument `ties` is a logical argument (default=`FALSE`) to take ties into account or not. If `TRUE` a NN contributes $1/m$ to the NN count if it is one of the m tied NNs of a subject.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

See also (Dixon (1994, 2002); Ceyhan (2010, 2017)) and the references therein.

Usage

```
nnct(x, lab, ties = FALSE, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`lab`	The `vector` of class labels (numerical or categorical)
`ties`	A logical argument (default=`FALSE`) to take ties into account or not. If `TRUE` a NN contributes $1/m$ to the NN count if it is one of the m tied NNs of a subject.
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the $k \times k$ NNCT where k is the number of classes in the data set.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). “On the use of nearest neighbor contingency tables for testing spatial segregation.” *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

Dixon PM (1994). “Testing spatial segregation using a nearest-neighbor contingency table.” *Ecology*, **75(7)**, 1940-1948.

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

See Also

`nnct.sub`, `scct`, `rct`, and `tct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
nnct(ipd,cls)
nnct(ipd,cls,ties = TRUE)

nnct(Y,cls,is.ipd = FALSE)
nnct(Y,cls,is.ipd = FALSE,method="max")
nnct(Y,cls,is.ipd = FALSE,method="mink",p=6)
```

```
#with one class, it works but really uninformative
cls<-rep(1,n)
nnct(ipd,cls)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
nnct(ipd,fcls)

#cls as an unsorted factor
fcls1<-sample(c("a","b"),n,replace = TRUE)
nnct(ipd,fcls1)

fcls2<-sort(fcls1)
nnct(ipd,fcls2)
#ipd needs to be sorted as well, otherwise this result will not agree with fcls1

nnct(Y,fcls1,ties = TRUE,is.ipd = FALSE)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
nnct(ipd,cls)
nnct(Y,cls,is.ipd = FALSE)

#cls as a factor
fcls<-rep(letters[1:4],rep(10,4))
nnct(ipd,fcls)

#1D data points
n<-20  #or try sample(1:20,1)
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
nnct(ipd,cls)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
nnct(ipd,fcls)

#with possible ties in the data
Y<-matrix(round(runif(3*n)*10),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
nnct(ipd,cls)
nnct(ipd,cls,ties = TRUE)
```

`nnct.boot.dis` *Bootstrap Nearest Neighbor Contingency Table (NNCT)*

Description

Returns the $k \times k$ NNCT with sampling replacement of the points for each base point. That is, for each base point, the rows in the IPD matrix are sampled with replacement and the NN counts are updated accordingly. Row and columns of the NNCT are labeled with the corresponding class labels.

The argument self is a logical argument (default=`TRUE`) for including the base point in the resampling or not. If `TRUE`, for each base point all entries in the row are sampled (with replacement) so, the point itself can also be sampled multiple times and if `FALSE` the point is excluded from the resampling (i.e., other points are sampled with replacement).

The argument `ties` is a logical argument (default=`FALSE`) to take ties into account or not. If `TRUE` a NN contributes $1/m$ to the NN count if it is one of the m tied NNs of a subject.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

Usage

```
nnct.boot.dis(x, lab, self = TRUE, ties = TRUE, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`lab`	The `vector` of class labels (numerical or categorical)
`self`	A logical argument (default=`TRUE`). If `TRUE`, for each base point, all entries in the row are sampled (with replacement) and if `FALSE` the point is excluded from the resampling (i.e., other points are sampled with replacement).
`ties`	A logical argument (default=`FALSE`) to take ties into account or not. If `TRUE` a NN contributes $1/m$ to the NN count if it is one of the m tied NNs of a subject.
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the $k \times k$ NNCT where k is the number of classes in the data set with sampling replacement of the rows of the IPD matrix.

Author(s)

Elvan Ceyhan

See Also

nnct and nnct.sub

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
nnct.boot.dis(ipd,cls)
nnct.boot.dis(Y,cls,is.ipd = FALSE)
#may give different result from above due to random sub-sampling
nnct.boot.dis(ipd,cls,self = FALSE)
nnct.boot.dis(ipd,cls,ties = FALSE)
#differences are due to ties and resampling of distances

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
nnct.boot.dis(ipd,fcls)

#cls as an unsorted factor
fcls<-sample(c("a","b"),n,replace = TRUE)
nnct.boot.dis(ipd,fcls)

fcls<-sort(fcls)
nnct.boot.dis(ipd,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
nnct.boot.dis(ipd,cls)

#cls as a factor
fcls<-rep(letters[1:4],rep(10,4))
nnct.boot.dis(ipd,fcls)

#1D data points
n<-20  #or try sample(1:20,1)
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
nnct.boot.dis(ipd,cls)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
nnct.boot.dis(ipd,fcls)
```

```
#with possible ties in the data
Y<-matrix(round(runif(3*n)*10),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
nnct.boot.dis(ipd,cls)
nnct.boot.dis(ipd,cls,self = FALSE)
nnct.boot.dis(ipd,cls,ties = FALSE)
#differences are due to ties and resampling of distances
```

`nnct.sub` *Nearest Neighbor Contingency Table (NNCT) with (only) base points restricted to a subsample*

Description

Returns the $k \times k$ NNCT with (only) base points are restricted to be in the subset of indices `ss` using the IPD matrix or data set `x` where k is the number of classes in the data set. That is, the base points are the points with indices in `ss` but for the NNs the function checks all the points in the data set (including the points in `ss`). Row and columns of the NNCT are labeled with the corresponding class labels.

The argument `ties` is a logical argument (default=`FALSE`) to take ties into account or not. If `TRUE` a NN contributes $1/m$ to the NN count if it is one of the m tied NNs of a subject.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

Usage

```
nnct.sub(ss, x, lab, ties = FALSE, is.ipd = TRUE, ...)
```

Arguments

`ss`	indices of subjects (i.e., row indices in the data set) chosen to be the base points
`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`lab`	The `vector` of class labels (numerical or categorical)
`ties`	A logical argument (default=`FALSE`) to take ties into account or not. If `TRUE` a NN contributes $1/m$ to the NN count if it is one of the m tied NNs of a subject.
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the $k \times k$ NNCT where k is the number of classes in the data set with (only) base points restricted to a subsample `ss`.

Author(s)

Elvan Ceyhan

See Also

`nnct` and `nnct.boot.dis`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
nnct(ipd,cls)

#subsampling indices
ss<-sample(1:n,floor(n/2))
nnct.sub(ss,ipd,cls)
nnct.sub(ss,Y,cls,is.ipd = FALSE)
nnct.sub(ss,ipd,cls,ties = TRUE)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
nnct.sub(ss,ipd,fcls)

#cls as an unsorted factor
fcls<-sample(c("a","b"),n,replace = TRUE)
nnct(ipd,fcls)
nnct.sub(ss,ipd,fcls)

fcls<-sort(fcls)
nnct.sub(ss,ipd,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ss<-sample(1:40,30)
nnct.sub(ss,ipd,cls)

#cls as a factor
fcls<-rep(letters[1:4],rep(10,4))
nnct.sub(ss,ipd,cls)

#1D data points
```

```
n<-20  #or try sample(1:20,1)
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
nnct(ipd,cls)

#subsampling indices
ss<-sample(1:n,floor(n/2))
nnct.sub(ss,ipd,cls)

#with possible ties in the data
Y<-matrix(round(runif(120)*10),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ss<-sample(1:40,30)
nnct.sub(ss,ipd,cls)
nnct.sub(ss,ipd,cls,ties = TRUE)
```

`NNdist` *Distances between subjects and their NNs*

Description

Returns the distances between subjects and their NNs. The output is an $n \times 2$ matrix where n is the data size and first column is the subject index and second column contains the corresponding distances to NN subjects.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

Usage

```
NNdist(x, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns an $n \times 2$ matrix where n is data size (i.e., number of subjects) and first column is the subject index and second column is the NN distances.

Author(s)

Elvan Ceyhan

See Also

`kthNNdist`, `kNNdist`, and `NNdist2cl`

Examples

```
#3D data points
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
NNdist(ipd)
NNdist(Y,is.ipd = FALSE)
NNdist(Y,is.ipd = FALSE,method="max")

#1D data points
X<-as.matrix(runif(5)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(5) would not work
ipd<-ipd.mat(X)
NNdist(ipd)
NNdist(X,is.ipd = FALSE)
```

`NNdist2cl` *Distances between subjects from class i and their NNs from class j*

Description

Returns the distances between subjects from class i and their nearest neighbors (NNs) from class j. The output is a `list` with first entry (`nndist`) being an $n_i \times 3$ matrix where n_i is the size of class i and first column is the subject index in class i, second column is the subject index in NN class j, and third column contains the corresponding distances of each class i subject to its NN among class j subjects. Class i is labeled as base class and class j is labeled as NN class.

The argument `within.class.ind` is a logical argument (default=`FALSE`) to determine the indexing of the class i subjects. If `TRUE`, index numbering of subjects is within the class, from 1 to class size (i.e., `1:n_i`), according to their order in the original data; otherwise, index numbering within class is just the indices in the original data.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

Usage

```
NNdist2cl(x, i, j, lab, within.class.ind = FALSE, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`i, j`	class label of base class and NN classes, respectively.
`lab`	The `vector` of class labels (numerical or categorical)
`within.class.ind`	A logical parameter (default=`FALSE`). If `TRUE`, index numbering of subjects is within the class, from 1 to class size (i.e., `1:n_i`), according to their order in the original data; otherwise, index numbering within class is just the indices in the original data.
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns a `list` with three elements

`nndist`	$n_i \times 3$ matrix where n_i is the size of class i and first column is the subject index in class i, second column is the subject index in NN class j, and third column contains the corresponding distances of each class i subject to its NN among class j subjects.
`base.class`	label of base class
`nn.class`	label of NN class

Author(s)

Elvan Ceyhan

See Also

`kthNNdist`, `kNNdist`, and `NNdist2cl`

Examples

```
#3D data points
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
#two class case
clab<-sample(1:2,n,replace=TRUE) #class labels
table(clab)
NNdist2cl(ipd,1,2,clab)
NNdist2cl(Y,1,2,clab,is.ipd = FALSE)

NNdist2cl(ipd,1,2,clab,within = TRUE)
```

```
#three class case
clab<-sample(1:3,n,replace=TRUE) #class labels
table(clab)
NNdist2cl(ipd,2,1,clab)

#1D data points
n<-15
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
#two class case
clab<-sample(1:2,n,replace=TRUE) #class labels
table(clab)
NNdist2cl(ipd,1,2,clab)
NNdist2cl(X,1,2,clab,is.ipd = FALSE)
```

`NNsub` *Finding the index of the NN of a given point among a subset of points*

Description

Returns the index (indices) of the nearest neighbor(s) of subject i (other than subject i) among the indices of points provided in the subsample `ss` using the given data set or IPD matrix `x`. The indices in `ss` determine the columns of the IPD matrix to be used in this function. It will yield a `vector` if there are ties, and subject indices correspond to rows (i.e., rows `1:n`) if `x` is the data set and to rows or columns if `x` is the IPD matrix.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

Usage

```
NNsub(ss, x, i, is.ipd = TRUE, ...)
```

Arguments

`ss`	indices of subjects (i.e., row indices in the data set) among with the NN of subject is to be found
`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`i`	index of (i.e., row number for) the subject whose NN is to be found.
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns a `list` with the elements

`base.ind`	index of the base subject
`ss.ind`	the index (indices) i.e., row number(s) of the NN of subject i among the subjects with indices provided in `ss`
`ss.dis`	distance from subject i to its NN among the subjects in `ss`

Author(s)

Elvan Ceyhan

See Also

`NN` and `kNN`

Examples

```
#3D data points
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
#indices of the subsample ss
ss<-sample(1:n,floor(n/2),replace=FALSE)
NNsub(ss,ipd,2)
NNsub(ss,Y,2,is.ipd = FALSE)
NNsub(ss,ipd,5)

#1D data points
n<-15
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
#two class case
clab<-sample(1:2,n,replace=TRUE) #class labels
#indices of the subsample ss
ss<-sample(1:n,floor(n/2),replace=FALSE)
NNsub(ss,ipd,2)
NNsub(ss,ipd,5)

#with possible ties in the data
Y<-matrix(round(runif(60)*10),ncol=3)
ipd<-ipd.mat(Y)
ss<-sample(1:20,10,replace=FALSE) #class labels
NNsub(ss,ipd,2)
NNsub(ss,ipd,5)
```

`Nt.def` *N_t Value (found with the definition formula)*

Description

This function computes the N_t value which is required in the computation of the asymptotic variance of Cuzick and Edwards T_k test. Nt is defined on page 78 of (Cuzick and Edwards (1990)) as follows. $N_t = \sum\sum_{i \neq l}\sum a_{ij}a_{lj}$ (i.e, number of triplets (i, j, l) i, j, and l distinct so that j is among kNNs of i and j is among kNNs of l).

This function yields the same result as the `asyvarTk` and `varTk` functions with `$Nt` inserted at the end.

See (Cuzick and Edwards (1990)) for more details.

Usage

```
Nt.def(a)
```

Arguments

`a`	The $A = (a_{ij})$ matrix. The argument `a` is the A matrix, obtained as output fromm `aij.mat`.

Value

Returns the N_t value standing for the number of triplets (i, j, l) i, j, and l distinct so that j is among kNNs of i and j is among kNNs of l. See the description.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`asyvarTk`, `varTk`, and `varTkaij`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
k<-2 #try also 2,3
a<-aij.mat(Y,k)
Nt.def(a)
```

`Ntkl` *N_tkl Value*

Description

This function computes the N_{tkl} value which is required in the computation of the exact and asymptotic variance of Cuzick and Edwards T_{comb} test, which is a linear combination of some T_k tests. N_{tkl} is defined on page 80 of (Cuzick and Edwards (1990)) as follows. Let $a_{ij}(k)$ be 1 if j is a k NN of i and zero otherwise and $N_t(k,l) = \sum\sum_{i \neq m} \sum a_{ij}(k)a_{mj}(l)$.

The logical argument `nonzero.mat` (default=`TRUE`) is for using the A matrix if `FALSE` or just the matrix of nonzero locations in the A matrix (if `TRUE`) in the computations.

See (Cuzick and Edwards (1990)) for more details.

Usage

```
Ntkl(dat, k, l, nonzero.mat = TRUE, ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`k, l`	Integers specifying the number of NNs (of subjects i and m in $a_{ij}(k)a_{mj}(l)$).
`nonzero.mat`	A logical argument (default is `TRUE`) to determine whether the A matrix or the matrix of nonzero locations of the A matrix will be used in the computation of N_s and N_t (argument is passed on to `asycovTkTl` and `covTkTl`). If `TRUE` the nonzero location matrix is used, otherwise the A matrix itself is used.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the N_{tkl} value. See the description.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`asycovTkTl`, and `covTkTl`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
k<-1 #try also 2,3 or sample(1:5,1)
l<-1 #try also 2,3 or sample(1:5,1)
c(k,l)

Ntkl(Y,k,l)
Ntkl(Y,k,l,nonzero.mat = FALSE)
Ntkl(Y,k,l,method="max")
```

`pairwise.lab` *Keeping the pair of the specified labels in the data*

Description

Keeps only the specified labels i and j and returns the data from classes with these labels and also the corresponding label vector having class labels i and j only.

See also (Ceyhan (2017)).

Usage

```
pairwise.lab(dat, lab, i, j)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`lab`	The `vector` of class labels (numerical or categorical)
`i, j`	Label of the classes that are to be retained in the post-hoc comparison.

Value

A `list` with two elements

`data.pair`	The type of the pattern from which points are to be generated
`lab.pair`	The `"main"` title for the plot of the point pattern

Author(s)

Elvan Ceyhan

References

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`lab.onevsrest` and `classirest`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
pairwise.lab(Y,cls,1,2)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
pairwise.lab(Y,cls,2,3)

#cls as a factor
fcls<-rep(letters[1:4],rep(10,4))
pairwise.lab(Y,fcls,"b","c")
```

`pick.min.max` *Smallest and Largest Distances in a Distance Matrix*

Description

This function finds and returns the `k` smallest and `k` largest distances in a distance matrix or distance object, and also provides pairs of objects these distances correspond to. The code is adapted from `http://people.stat.sc.edu/Hitchcock/chapter1_R_examples.txt`.

Usage

```
pick.min.max(ds, k = 1)
```

Arguments

`ds`	A distance matrix or a distance object
`k`	A positive integer representing the number of (min and max) distances to be presented, default is $k = 1$

Value

A `list` with the elements

`min.dis`	The `k` smallest distances in `ds`
`ind.min.dis`	The indices (i.e., row numbers) of the `k` pairs of object which has the `k` smallest distances in `ds`
`max.dis`	The `k` largest distances in `ds`
`ind.max.dis`	The indices (i.e., row numbers) of the `k` pairs of object which has the `k` largest distances in `ds`

Author(s)

Elvan Ceyhan

See Also

`dist`, `ipd.mat`, and `ipd.mat.euc`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
pick.min.max(ipd)
#or
pick.min.max(dist(Y))

pick.min.max(ipd,2)
```

---

pk *Probability of* k *items selected from the class with size* n_1

Description

Returns the ratio $n_1(n_1-1)\cdots(n_1-(k-1))/(n(n-1)\cdots(n-(k-1))$, which is the probability that the k selected objects are from class 1 with size n_1 (denoted as n1 as an argument) and the total data size is n. This probability is valid under RL or CSR.

This function computes the p_k value which is required in the computation of the variance of Cuzick and Edwards T_k test. p_k is defined as the ratio $n_1(n_1-1)\cdots(n_1-(k-1))/(n(n-1)\cdots(n-(k-1))$.

The argument, n_1, is the number of cases (denoted as n1 as an argument). The number of cases are denoted as n_1 and number of controls as n_0 in this function to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

See (Cuzick and Edwards (1990)) for more details.

Usage

```
pk(n, n1, k)

pk(n, n1, k)
```

Arguments

n	A positive integer representing the number of points in the data set
n1	Number of cases
k	Integer specifying the number of NNs (of subject i)

Value

Returns the probability of `k` items selected from `n` items are from the class of interest (i.e., from the class whose size is n_1)

Returns the p_k value. See the description.

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`p11` and `p12` etc.

`asyvarTk`, `varTk`, and `varTkaij`

`plot.Clusters` *Plot a* `Clusters` *object*

Description

Plots the points generated from the pattern (color coded for each class) together with the study window

Usage

```
## S3 method for class 'Clusters'
plot(x, asp = NA, xlab = "x", ylab = "y", ...)
```

Arguments

`x`	Object of class `Clusters`.
`asp`	A numeric value, giving the aspect ratio for y axis to x-axis y/x (default is `NA`), see the official help for `asp` by typing "? asp".
`xlab, ylab`	Titles for the x and y axes, respectively (default is `xlab`="x" and `ylab`="y").
`...`	Additional parameters for `plot`.

Value

None

Examples

```
#TBF
```

`plot.SpatPatterns` *Plot a* `SpatPatterns` *object*

Description

Plots the points generated from the pattern (color coded for each class) together with the study window

Usage

```
## S3 method for class 'SpatPatterns'
plot(x, asp = NA, xlab = "x", ylab = "y", ...)
```

Arguments

`x`	Object of class `SpatPatterns`.
`asp`	A numeric value, giving the aspect ratio for y axis to x-axis y/x (default is `NA`), see the official help for `asp` by typing "? asp".
`xlab`, `ylab`	Titles for the x and y axes, respectively (default is `xlab`="x" and `ylab`="y").
`...`	Additional parameters for `plot`.

Value

None

Examples

```
#TBF
```

`print.cellhtest` *Print a summary of a* `cellhtest` *object*

Description

Printing objects of class "`cellhtest`" by simple `print` methods.

Usage

```
## S3 method for class 'cellhtest'
print(x, digits = getOption("digits"), prefix = "\t", ...)
```

Arguments

`x`	object of class "`summary.cellhtest`"
`digits`	number of significant digits to be used.
`prefix`	string, passed to `strwrap` for displaying the method component of the `classhtest` object.
`...`	Additional parameters for `print`.

Value

None

`print.Chisqtest` *Print a summary of a* `Chisqtest` *object*

Description

Printing objects of class "`Chisqtest`" by simple `print` methods.

Usage

```
## S3 method for class 'Chisqtest'
print(x, digits = getOption("digits"), prefix = "\t", ...)
```

Arguments

`x`	object of class "`summary.Chisqtest`"
`digits`	number of significant digits to be used.
`prefix`	string, passed to `strwrap` for displaying the method component of the `classhtest` object.
`...`	Additional parameters for `print`.

Value

None

`print.classhtest` *Print a summary of a* `classhtest` *object*

Description

Printing objects of class "`classhtest`" by simple `print` methods.

Usage

```
## S3 method for class 'classhtest'
print(x, digits = getOption("digits"), prefix = "\t", ...)
```

Arguments

`x`	object of class "`summary.classhtest`"
`digits`	number of significant digits to be used.
`prefix`	string, passed to `strwrap` for displaying the method component of the `classhtest` object.
`...`	Additional parameters for `print`.

Value

None

`print.Clusters` *Print a* `Clusters` *object*

Description

Prints the `call` of the object of class '`Clusters`' and also the `type` (or description) of the pattern).

Usage

```
## S3 method for class 'Clusters'
print(x, ...)
```

Arguments

`x`	A `Clusters` object.
`...`	Additional arguments for the S3 method '`print`'.

Value

The `call` of the object of class '`Clusters`' and also the `type` (or description) of the pattern).

See Also

`summary.Clusters`, `print.summary.Clusters`, and `plot.Clusters`

Examples

```
#TBF (to be filled)
```

`print.refhtest` *Print a summary of a* `refhtest` *object*

Description

Printing objects of class "`refhtest`" by simple `print` methods.

Usage

```
## S3 method for class 'refhtest'
print(x, digits = getOption("digits"), prefix = "\t", ...)
```

Arguments

`x`	object of class "`summary.refhtest`"
`digits`	number of significant digits to be used.
`prefix`	string, passed to `strwrap` for displaying the method component of the `classhtest` object.
`...`	Additional parameters for `print`.

Value

None

`print.SpatPatterns` *Print a* `SpatPatterns` *object*

Description

Prints the `call` of the object of class '`SpatPatterns`' and also the `type` (or description) of the pattern).

Usage

```
## S3 method for class 'SpatPatterns'
print(x, ...)
```

Arguments

`x`	A `SpatPatterns` object.
`...`	Additional arguments for the S3 method '`print`'.

Value

The `call` of the object of class '`SpatPatterns`' and also the `type` (or description) of the pattern).

See Also

`summary.SpatPatterns`, `print.summary.SpatPatterns`, and `plot.SpatPatterns`

Examples

```
#TBF (to be filled)
```

`print.summary.Clusters`

Print a summary of a `Clusters` *object*

Description

Prints some information about the `object`.

Usage

```
## S3 method for class 'summary.Clusters'
print(x, ...)
```

Arguments

`x`	object of class "`summary.Clusters`", generated by `summary.Clusters`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.Clusters`, `summary.Clusters`, and `plot.Clusters`

`print.summary.SpatPatterns`

Print a summary of a `SpatPatterns` *object*

Description

Prints some information about the `object`.

Usage

```
## S3 method for class 'summary.SpatPatterns'
print(x, ...)
```

Arguments

`x`	object of class "`summary.SpatPatterns`", generated by `summary.SpatPatterns`.
`...`	Additional parameters for `print`.

Value

None

See Also

`print.SpatPatterns`, `summary.SpatPatterns`, and `plot.SpatPatterns`

`prob.nnct` *Probability of the current nearest neighbor contingency table*

Description

Computes the probability of the observed 2×2 nearest neighbor contingency table (NNCT) $p_t = f(n_{11}|n_1, n_2, c_1; \theta)$ where $\theta = (n_1 - 1)(n_2 - 1)/(n_1 n_2)$ which is the odds ratio under RL or CSR independence and f is the probability mass function of the hypergeometric distribution. That is, given the margins of the current NNCT, the probability of obtaining the current table with the odds ratio θ being the value under the null hypothesis. This value is used to compute the table-inclusive and exclusive p-values for the exact inference on NNCTs.

See (Ceyhan (2010)) for more details.

Usage

```
prob.nnct(ct)
```

Arguments

`ct`	A NNCT

Value

The probability of getting the observed NNCT, `ct` , under the null hypothesis.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Exact Inference for Testing Spatial Patterns by Nearest Neighbor Contingency Tables." *Journal of Probability and Statistical Science*, **8(1)**, 45-68.

See Also

`exact.pval1s` and `exact.pval2s`

Examples

```
ct<-matrix(sample(20:40,4),ncol=2)
prob.nnct(ct)

ct<-matrix(sample(20:40,4),ncol=2)
prob.nnct(ct)
```

`QRval` *Number of Shared and Reflexive NNs*

Description

Returns the Q and R values where Q is the number of points shared as a NN by other points i.e., number of points that are NN of other points (which occurs when two or more points share a NN, for data in any dimension) and R is the number of reflexive pairs where A and B are reflexive iff they are NN to each other.

These quantities are used, e.g., in computing the variances and covariances of the entries of the nearest neighbor contingency tables used for Dixon's tests and other NNCT tests.

Usage

```
QRval(njr)
```

Arguments

`njr`	A `list` that is the output of `Ninv` (with first entry in the `list` is `vector` of number of shared NNs and second is the R value, number of reflexive points)

Value

A `list` with two elements

`Q`	the Q value, the number of shared NNs
`R`	the R value, the number of reflexive NNs

Author(s)

Elvan Ceyhan

See Also

`Qval`, `Qvec`, `sharedNN`, `Rval`, and `Ninv`

Examples

```
#3D data points
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
ninv<-Ninv(ipd)
QRval(ninv)
W<-Wmat(ipd)
Qvec(W)$q

#1D data points
n<-15
X<-as.matrix(runif(n))# need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(n) would not work
ipd<-ipd.mat(X)
ninv<-Ninv(ipd)
QRval(ninv)
W<-Wmat(ipd)
Qvec(W)$q

#with possible ties in the data
Y<-matrix(round(runif(30)*10),ncol=3)
ny<-nrow(Y)
ipd<-ipd.mat(Y)
ninv<-Ninv(ipd)
QRval(ninv)
W<-Wmat(ipd)
Qvec(W)$q
```

`Qsym.ct` *Q-symmetry Contingency Table (QCT)*

Description

Returns the $k \times 3$ contingency table for Q-symmetry (i.e., Q-symmetry contingency table (QCT)) given the IPD matrix or data set `x` where k is the number of classes in the data set. Each row in the QCT is the vector of number of points with shared NNs, $Q_i = (Q_{i0}, Q_{i1}, Q_{i2})$ where Q_{ij} is the number of class i points that are NN to class j points for $j = 0, 1$ and Q_{i2} is the number of class i points that are NN to class j or more points. That is, this function pools the cells 3 or larger together for k classes, so, Q_2, Q_3 etc. are pooled, so, the column labels are Q_0, Q_1 and Q_2 with the last one is actually sum of Q_j for $j \geq 2$. Rows the QCT are labeled with the corresponding class labels.

Q-symmetry is also equivalent to Pielou's second type of NN symmetry or the symmetry in the shared NN structure for all classes.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

See also (Pielou (1961); Ceyhan (2014)) and the references therein.

Usage

```
Qsym.ct(x, lab, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`lab`	The `vector` of class labels (numerical or categorical)
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the $k \times 3$ QCT where k is the number of classes in the data set.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations.” *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Pielou EC (1961). “Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships.” *Journal of Ecology*, **49(2)**, 255-269.

See Also

`sharedNNmc`, `Qsym.test` and `scct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(n*3),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)

Qsym.ct(ipd,cls)
Qsym.ct(Y,cls,is.ipd = FALSE)
Qsym.ct(Y,cls,is.ipd = FALSE,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
Qsym.ct(ipd,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)

Qsym.ct(ipd,cls)
```

`Qsym.test` *Pielou’s Second Type of NN Symmetry Test with Chi-square Approximation*

Description

An object of class `"Chisqtest"` performing the hypothesis test of equality of the probabilities for the rows in the Q-symmetry contingency table (QCT). Each row of the QCT is the vector of Qij values where Q_{ij} is the number of class i points that are NN to j points. That is, the test performs Pielou’s second type of NN symmetry test which is also equivalent to Pearson’s test on the QCT (Pielou (1961)). Pielou’s second type of NN symmetry is the symmetry in the shared NN structure for all classes, which is also called Q-symmetry. The test is appropriate (i.e., have the appropriate asymptotic sampling distribution) provided that data is obtained by sparse sampling, although simulations suggest it seems to work for completely mapped data as well. (See Ceyhan (2014) for more detail).

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

The argument combine is a logical argument (default=`TRUE`) to determine whether to combine the 3rd column and the columns to the left. If `TRUE`, this function pools the cells 3 or larger together for k classes in the QCT, so, Q_2, Q_3 etc. are pooled, so, the column labels are Q_0, Q_1 and Q_2 with the last one is actually sum of Q_j for $j \geq 2$ in the QCT. If `FALSE`, the function does not perform the pooling of the cells.

The function yields the test statistic, p-value and `df` which is $(k-1)(n_c-1)$ where n_c is the number of columns in QCT (which reduces to $2(k-1)$, if `combine=TRUE`). It also provides the description of the alternative with the corresponding null values (i.e., expected values) of the entries of the QCT and also the sample estimates of the entries of QCT (i.e., the observed QCT). The function also provides names of the test statistics, the description of the test and the data set used.

The null hypothesis is the symmetry in the shared NN structure for each class, that is, all $E(Q_{ij}) = n_i Q_j / n$ where n_i the size of class i and Q_j is the sum of column j in the QCT (i.e., the total number of points serving as NN to class j other points). (i.e., symmetry in the mixed NN structure).

See also (Pielou (1961); Ceyhan (2014)) and the references therein.

Usage

```
Qsym.test(x, lab, is.ipd = TRUE, combine = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`lab`	The `vector` of class labels (numerical or categorical)
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix (IPD matrix), otherwise, `x` is taken as the data set with rows representing the data points.
`combine`	A logical parameter (default=`TRUE`). If `TRUE`, the cells in column 3 or columns to the left are merged in the QCT, so, Q_2, Q_3 etc. are pooled, so, the column labels are Q_0, Q_1 and Q_2 with the last one is actually sum of Q_j for $j \geq 2$ in the QCT. If `FALSE`, the function does not perform the pooling of the cells.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

A `list` with the elements

`statistic`	The chi-squared test statistic for Pielou's second type of NN symmetry test (i.e., Q-symmetry which is equivalent to symmetry in the shared NN structure)
`p.value`	The p-value for the hypothesis test

`df`	Degrees of freedom for the chi-squared test, which is $(k-1)(n_c-1)$ where n_c is the number of columns in QCT (which reduces to $2(k-1)$ if `combine=TRUE`).
`estimate`	Estimates, i.e., the observed QCT.
`est.name`, `est.name2`	Names of the estimates, they are identical for this function.
`null.value`	Hypothesized null values for the entries of the QCT, i.e., the matrix with entries $E(Q_{ij}) = n_i Q_j / n$ where n_i the size of class i and Q_j is the sum of column j in the QCT (i.e., the total number of points serving as NN to class j other points).
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `x`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Pielou EC (1961). "Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships." *Journal of Ecology*, **49(2)**, 255-269.

See Also

`Znnsym` and `Xsq.nnsym`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)
Qsym.ct(ipd,cls)

Qsym.test(ipd,cls)
Qsym.test(Y,cls,is.ipd = FALSE)
Qsym.test(Y,cls,is.ipd = FALSE,method="max")

Qsym.test(ipd,cls,combine = FALSE)

#cls as a faqctor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
Qsym.test(ipd,fcls)
Qsym.test(Y,fcls,is.ipd = FALSE)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
```

```
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

Qsym.test(ipd,cls)
Qsym.test(Y,cls,is.ipd = FALSE)
```

`rassoc` *Generation of Points Associated with a Given Set of Points*

Description

An object of class `"SpatPatterns"`.

Generates `n_2` 2D points associated with the given set of points (i.e., reference points) X_1 in the `type=type` fashion with the parameter=asc.par which specifies the level of association. The generated points are intended to be from a different class, say class 2 (or X_2 points) than the reference (i.e., X_1 points, say class 1 points, denoted as `X1` as an argument of the function), say class 1 points).

To generate n_2 (denoted as `n2` as an argument of the function)X_2 points, n_2 of X_1 points are randomly selected (possibly with replacement) and for a selected `X1` point, say x_{1ref}, a new point from the class 2, say x_{2new}, is generated from a distribution specified by the type argument.

In type I association, i.e., if `type="I"`, first a $Uniform(0, 1)$ number, U, is generated. If $U \leq p$, x_{2new} is generated (uniform in the polar coordinates) within a circle with radius equal to the distance to the closest X_1 point, else it is generated uniformly within the smallest bounding box containing X_1 points.

In the type C association pattern the new point from the class 2, x_{2new}, is generated (uniform in the polar coordinates) within a circle centered at x_{1ref} with radius equal to r_0, in type U association pattern x_{2new} is generated similarly except it is uniform in the circle.

In type G association, x_{2new} is generated from the bivariate normal distribution centered at x_{1ref} with covariance σI_2 where I_2 is 2×2 identity matrix.

See Ceyhan (2014) for more detail.

Usage

```
rassoc(X1, n2, asc.par, type)
```

Arguments

`X1`	A set of 2D points representing the reference points, also referred as class 1 points. The generated points are associated in a type=type sense with these points.
`n2`	A positive integer representing the number of class 2 points to be generated.
`asc.par`	A positive real number representing the association parameter. For `type="I"`, it is attraction probability, `p`, of class 2 points associated with a randomly selected class 1 point; for `type="C"` or `"U"`, it is the radius of association, `r0`, of class 2 points associated with a randomly selected class 1 point; for `type="G"`, it is the variance of the Gaussian marginals, where the bivariate normal distribution has covariance σI_2 with I_2 being the 2×2 identity matrix.

`type`	The type of the association pattern. Takes on values `"I"`, `"C"`, `"U"` and `"G"` for types I, C, U and G association patterns (see the description above).

Value

A `list` with the elements

`pat.type`	=`"ref.gen"` for the bivariate pattern of association of class 2 points with the reference points (i.e., X_1), indicates reference points are required to be entered as an argument in the function
`type`	The type of the point pattern
`parameters`	The `asc.par` value specifying the level of association
`ref.points`	The input set of reference points X_1, i.e., points with which generated class 2 points are associated.
`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated class 2 points and the number of reference (i.e., X_1) points.
`xlimit, ylimit`	The possible ranges of the x- and y-coordinates of the generated and the reference points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Simulation and characterization of multi-class spatial patterns from stochastic point processes of randomness, clustering and regularity.” *Stochastic Environmental Research and Risk Assessment (SERRA)*, **38(5)**, 1277-1306.

See Also

`rassocI`, `rassocC`, `rassocU`, and `rassocG`

Examples

```
n1<-20; n2<-1000;  #try also n1<-10; n2<-1000;

#with default bounding box (i.e., unit square)
X1<-cbind(runif(n1),runif(n1))

Xdat<-rassoc(X1,n2,asc.par=.05,type="G") #try other types as well
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#with type U association
```

```
Xdat<-rassoc(X1,n2,asc.par=.1,type="U")
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#with type C association
Xdat<-rassoc(X1,n2,asc.par=.1,type=2) #2 is for "C"
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rassocC` *Generation of Points Associated in the Type C Sense with a Given Set of Points*

Description

An object of class `"SpatPatterns"`.

Generates `n_2` 2D points associated with the given set of points (i.e., reference points) X_1 in the type C fashion with a radius of association r_0 (denoted as `r0` as an argument of the function) which is a positive real number. The generated points are intended to be from a different class, say class 2 (or X_2 points) than the reference (i.e., X_1 points, say class 1 points, denoted as `X1` as an argument of the function), say class 1 points). To generate n_2 X_2 points, n_2 of X_1 points are randomly selected (possibly with replacement) and for a selected `X1` point, say x_{1ref}, a new point from the class 2, say x_{2new}, is generated within a circle with radius equal to r_0 (uniform in the polar coordinates). That is, $x_{2new} = x_{1ref} + r_u c(\cos(t_u), \sin(t_u))$ where $r_u \sim U(0, r_0)$ and $t_u \sim U(0, 2\pi)$. Note that, the level of association increases as r_0 decreases, and the association vanishes when r_0 is sufficiently large.

For type C association, it is recommended to take $r_0 \leq 0.25$ times length of the shorter edge of a rectangular study region, or take $r_0 = 1/(k\sqrt{\hat{\rho}})$ with the appropriate choice of k to get an association pattern more robust to differences in relative abundances (i.e., the choice of k implies $r_0 \leq 0.25$ times length of the shorter edge to have alternative patterns more robust to differences in sample sizes). Here $\hat{\rho}$ is the estimated intensity of points in the study region (i.e., # of points divided by the area of the region).

Type C association is closely related to Type U association, see the function `rassocC` and the other association types. In the type U association pattern the new point from the class 2, x_{2new}, is generated uniformly within a circle centered at x_{1ref} with radius equal to r_0. In type G association, x_{2new} is generated from the bivariate normal distribution centered at x_{1ref} with covariance σI_2 where I_2 is 2×2 identity matrix. In type I association, first a $Uniform(0, 1)$ number, U, is generated. If $U \leq p$, x_{2new} is generated (uniform in the polar coordinates) within a circle with radius equal to the distance to the closest X_1 point, else it is generated uniformly within the smallest bounding box containing X_1 points.

See Ceyhan (2014) for more detail.

Usage

```
rassocC(X1, n2, r0)
```

Arguments

`X1`	A set of 2D points representing the reference points, also referred as class 1 points. The generated points are associated in a type C sense (in a circular/radial fashion) with these points.
`n2`	A positive integer representing the number of class 2 points to be generated.
`r0`	A positive real number representing the radius of association of class 2 points associated with a randomly selected class 1 point (see the description below).

Value

A `list` with the elements

`pat.type`	="`ref.gen`" for the bivariate pattern of association of class 2 points with the reference points (i.e., X_1), indicates reference points are required to be entered as an argument in the function
`type`	The type of the point pattern
`parameters`	Radius of association controlling the level of association
`gen.points`	The output set of generated points (i.e., class 2 points) associated with reference (i.e. X_1 points)
`ref.points`	The input set of reference points X_1, i.e., points with which generated class 2 points are associated.
`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated class 2 points and the number of reference (i.e., X_1) points.
`xlimit, ylimit`	The possible ranges of the x- and y-coordinates of the generated and the reference points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Simulation and characterization of multi-class spatial patterns from stochastic point processes of randomness, clustering and regularity." *Stochastic Environmental Research and Risk Assessment (SERRA)*, **38(5)**, 1277-1306.

See Also

`rassocI`, `rassocG`, `rassocU`, and `rassoc`

Examples

```
n1<-20; n2<-1000;  #try also n1<-10; n2<-1000;

r0<-.15 #try also .10 and .20, runif(1)
#with default bounding box (i.e., unit square)
X1<-cbind(runif(n1),runif(n1))  #try also X1<-1+cbind(runif(n1),runif(n1))

Xdat<-rassocC(X1,n2,r0)
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#radius adjusted with the expected NN distance
x<-range(X1[,1]); y<-range(X1[,2])
ar<-(y[2]-y[1])*(x[2]-x[1]) #area of the smallest rectangular window containing X1 points
rho<-n1/ar
r0<-1/(2*sqrt(rho)) #r0=1/(2rho) where \code{rho} is the intensity of X1 points
Xdat<-rassocC(X1,n2,r0)
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rassocG` *Generation of Points Associated in the Type G Sense with a Given Set of Points*

Description

An object of class `"SpatPatterns"`.

Generates `n_2` 2D points associated with the given set of points (i.e., reference points) X_1 in the type G fashion with the parameter sigma which is a positive real number representing the variance of the Gaussian marginals. The generated points are intended to be from a different class, say class 2 (or X_2 points) than the reference (i.e., X_1 points, say class 1 points, denoted as `X1` as an argument of the function), say class 1 points). To generate n_2 (denoted as `n2` as an argument of the function)X_2 points, n_2 of X_1 points are randomly selected (possibly with replacement) and for a selected `X1` point, say x_{1ref}, a new point from the class 2, say x_{2new}, is generated from a bivariate normal distribution centered at x_{1ref} where the covariance matrix of the bivariate normal is a diagonal matrix with sigma in the diagonals. That is, $x_{2new} = x_{1ref} + V$ where $V \sim BVN((0,0), \sigma I_2)$ with I_2 being the 2×2 identity matrix. Note that, the level of association increases as `sigma` decreases, and the association vanishes when `sigma` goes to infinity.

For type G association, it is recommended to take $\sigma \leq 0.10$ times length of the shorter edge of a rectangular study region, or take $r_0 = 1/(k\sqrt{\rho})$ with the appropriate choice of k to get an association pattern more robust to differences in relative abundances (i.e., the choice of k implies $\sigma \leq 0.10$ times length of the shorter edge to have alternative patterns more robust to differences in sample sizes). Here $\hat{\rho}$ is the estimated intensity of points in the study region (i.e., # of points divided by the area of the region).

Type G association is closely related to Types C and U association, see the functions `rassocC` and `rassocU` and the other association types. In the type C association pattern the new point from the class 2, x_{2new}, is generated (uniform in the polar coordinates) within a circle centered at x_{1ref} with radius equal to r_0, in type U association pattern x_{2new} is generated similarly except it is uniform in the circle. In type I association, first a $Uniform(0,1)$ number, U, is generated. If $U \leq p$, x_{2new} is generated (uniform in the polar coordinates) within a circle with radius equal to the distance to the closest X_1 point, else it is generated uniformly within the smallest bounding box containing X_1 points.

See Ceyhan (2014) for more detail.

Usage

```
rassocG(X1, n2, sigma)
```

Arguments

`X1`	A set of 2D points representing the reference points, also referred as class 1 points. The generated points are associated in a type G sense with these points.
`n2`	A positive integer representing the number of class 2 points to be generated.
`sigma`	A positive real number representing the variance of the Gaussian marginals, where the bivariate normal distribution has covariance `BVN((0,0),sigma*I_2)` with I_2 being the 2×2 identity matrix.

Value

A `list` with the elements

`pat.type`	=`"ref.gen"` for the bivariate pattern of association of class 2 points with the reference points (i.e., X_1), indicates reference points are required to be entered as an argument in the function
`type`	The type of the point pattern
`parameters`	The variance of the Gaussian marginals controlling the level of association, where the bivariate normal distribution has covariance σI_2 with I_2 being the 2×2 identity matrix.
`gen.points`	The output set of generated points (i.e., class 2 points) associated with reference (i.e. X_1 points)
`ref.points`	The input set of reference points X_1, i.e., points with which generated class 2 points are associated.
`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated class 2 points and the number of reference (i.e., X_1) points.
`xlimit, ylimit`	The possible ranges of the x- and y-coordinates of the generated and the reference points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Simulation and characterization of multi-class spatial patterns from stochastic point processes of randomness, clustering and regularity.” *Stochastic Environmental Research and Risk Assessment (SERRA)*, **38(5)**, 1277-1306.

See Also

`rassocI`, `rassocG`, `rassocC`, and `rassoc`

Examples

```
n1<-20; n2<-1000;  #try also n1<-10; n2<-1000;

stdev<-.05  #try also .075 and .15
#with default bounding box (i.e., unit square)
X1<-cbind(runif(n1),runif(n1))   #try also X1<-1+cbind(runif(n1),runif(n1))

Xdat<-rassocG(X1,n2,stdev)
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#sigma adjusted with the expected NN distance
x<-range(X1[,1]); y<-range(X1[,2])
ar<-(y[2]-y[1])*(x[2]-x[1]) #area of the smallest rectangular window containing X1 points
rho<-n1/ar
stdev<-1/(4*sqrt(rho)) #r0=1/(2rho) where \code{rho} is the intensity of X1 points
Xdat<-rassocG(X1,n2,stdev)
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rassocI` *Generation of Points Associated in the Type I Sense with a Given Set of Points*

Description

An object of class `"SpatPatterns"`.

Generates `n_2` 2D points associated with the given set of points (i.e., reference points) X_1 in the type I fashion with circular (or radial) between class attraction parameter `p`, which is a probability value between 0 and 1. The generated points are intended to be from a different class, say class 2 (or X_2 points) than the reference (i.e., X_1 points, say class 1 points, denoted as `X1` as an argument of the function). To generate n_2 (denoted as `n2` as an argument of the function) X_2 points, n_2 of X_1 points are randomly selected (possibly with replacement) and for a selected `X1` point, say x_{1ref}, a $Uniform(0,1)$ number, U, is generated. If $U \leq p$, a new point from the class 2, say x_{2new}, is generated within a circle with radius equal to the distance to the closest X_1 point (uniform in the polar coordinates), else the new point is generated uniformly within the smallest bounding box containing X_1 points. That is, if $U \leq p$, $x_{2new} = x_{1ref} + r_u c(\cos(t_u), \sin(t_u))$ where $r_u \sim U(0, rad)$ and $t_u \sim U(0, 2\pi)$ with $rad = \min(d(x_{1ref}, X_1 \setminus \{x_{1ref}\}))$, else $x_{2new} \sim rect(X_1)$ where $rect(X_1)$ is the smallest bounding box containing X_1 points. Note that, the level of association increases as `p` increases, and the association vanishes when `p` approaches to 0.

Type I association is closely related to Type C association in Ceyhan (2014), see the function `rassocC` and also other association types. In the type C association pattern the new point from the class 2, x_{2new}, is generated (uniform in the polar coordinates) within a circle centered at x_{1ref} with radius equal to r_0, in type U association pattern x_{2new} is generated similarly except it is uniform in the circle. In type G association, x_{2new} is generated from the bivariate normal distribution centered at x_{1ref} with covariance σI_2 where I_2 is 2×2 identity matrix.

Usage

```
rassocI(X1, n2, p)
```

Arguments

`X1`	A set of 2D points representing the reference points, also referred as class 1 points. The generated points are associated in a type I sense (in a circular/radial fashion) with these points.
`n2`	A positive integer representing the number of class 2 (i.e., X_2) points to be generated.
`p`	A real number between 0 and 1 representing the attraction probability of class 2 points associated with a randomly selected class 1 point (see the description below).

Value

A `list` with the elements

`pat.type`	equals `"ref.gen"` for the bivariate pattern of association of class 2 (i.e., X_2) points with the reference points (i.e., X_1), indicates reference points are required to be entered as an argument in the function
`type`	The type of the point pattern
`parameters`	Radial (i.e., circular) between class attraction parameter controlling the level of association
`gen.points`	The output set of generated points (i.e., class 2 points) associated with reference (i.e. X_1 points)
`ref.points`	The input set of reference points X_1, i.e., points with which generated class 2 points are associated.

`desc.pat`	Description of the point pattern
`lab`	The class labels of the generated points, it is `NULL` for this function, since only class 2 points are generated in this pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated class 2 points and the number of reference (i.e., X_1) points.
`xlimit, ylimit`	The possible ranges of the x- and y-coordinates of the generated and the reference points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Simulation and characterization of multi-class spatial patterns from stochastic point processes of randomness, clustering and regularity." *Stochastic Environmental Research and Risk Assessment (SERRA)*, **38(5)**, 1277-1306.

See Also

`rassocC`, `rassocG`, `rassocU`, and `rassoc`

Examples

```
n1<-20; n2<-1000;  #try also n1<-10; n2<-1000;

p<- .75 #try also .25, .5, .9, runif(1)
#with default bounding box (i.e., unit square)
X1<-cbind(runif(n1),runif(n1))  #try also X1<-1+cbind(runif(n1),runif(n1))

Xdat<-rassocI(X1,n2,p)
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rassocU` *Generation of Points Associated in the Type U Sense with a Given Set of Points*

Description

An object of class `"SpatPatterns"`.

Generates `n_2` 2D points associated with the given set of points (i.e., reference points) X_1 in the type U fashion with a radius of association r_0 (denoted as `r0` as an argument of the function) which is a positive real number. The generated points are intended to be from a different class, say class 2 (or X_2 points) than the reference (i.e., X_1 points, say class 1 points, denoted as `X1` as an argument of the function), say class 1 points). To generate n_2 (denoted as `n2` as an argument of the function)X_2 points, n_2 of X_1 points are randomly selected (possibly with replacement) and for a selected `X1` point, say x_{1ref}, a new point from the class 2, say x_{2new}, is generated uniformly within a circle with radius equal to r_0. That is, $x_{2new} = x_{1ref} + r_u c(\cos(t_u), \sin(t_u))$ where$r_u = sqrt(U) * r_0$ with $U \sim U(0,1)$ and $t_u \sim U(0,2\pi)$. Note that, the level of association increases as r_0 decreases, and the association vanishes when r_0 is sufficiently large.

For type U association, it is recommended to take $r_0 \leq 0.10$ times length of the shorter edge of a rectangular study region, or take $r_0 = 1/(k\sqrt{\hat{\rho}})$ with the appropriate choice of k to get an association pattern more robust to differences in relative abundances (i.e., the choice of k implies $r_0 \leq 0.10$ times length of the shorter edge to have alternative patterns more robust to differences in sample sizes). Here $\hat{\rho}$ is the estimated intensity of points in the study region (i.e., # of points divided by the area of the region).

Type U association is closely related to Type C association, see the function `rassocC` and the other association types. In the type C association pattern the new point from the class 2, x_{2new}, is generated (uniform in the polar coordinates) within a circle centered at x_{1ref} with radius equal to r_0. In type G association, x_{2new} is generated from the bivariate normal distribution centered at x_{1ref} with covariance σI_2 where I_2 is 2×2 identity matrix. In type I association, first a $Uniform(0,1)$ number, U, is generated. If $U \leq p$, x_{2new} is generated (uniform in the polar coordinates) within a circle with radius equal to the distance to the closest X_1 point, else it is generated uniformly within the smallest bounding box containing X_1 points.

See Ceyhan (2014) for more detail.

Usage

```
rassocU(X1, n2, r0)
```

Arguments

`X1`	A set of 2D points representing the reference points, also referred as class 1 points. The generated points are associated in a type U sense (in a circular/radial fashion) with these points.
`n2`	A positive integer representing the number of class 2 points to be generated.
`r0`	A positive real number representing the radius of association of class 2 points associated with a randomly selected class 1 point (see the description below).

Value

A `list` with the elements

`pat.type`	`="ref.gen"` for the bivariate pattern of association of class 2 points with the reference points (i.e., X_1), indicates reference points are required to be entered as an argument in the function

`type`	The type of the point pattern
`parameters`	Radius of association controlling the level of association
`gen.points`	The output set of generated points (i.e., class 2 points) associated with reference (i.e. X_1 points)
`ref.points`	The input set of reference points X_1, i.e., points with which generated class 2 points are associated.
`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated class 2 points and the number of reference (i.e., X_1) points.
`xlimit`, `ylimit`	The possible ranges of the x- and y-coordinates of the generated and the reference points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Simulation and characterization of multi-class spatial patterns from stochastic point processes of randomness, clustering and regularity." *Stochastic Environmental Research and Risk Assessment (SERRA)*, **38(5)**, 1277-1306.

See Also

`rassocI`, `rassocG`, `rassocC`, and `rassoc`

Examples

```
n1<-20; n2<-1000;  #try also n1<-10; n2<-1000;

r0<-.15 #try also .10 and .20
#with default bounding box (i.e., unit square)
X1<-cbind(runif(n1),runif(n1))  #try also X1<-1+cbind(runif(n1),runif(n1))

Xdat<-rassocU(X1,n2,r0)
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#radius adjusted with the expected NN distance
x<-range(X1[,1]); y<-range(X1[,2])
ar<-(y[2]-y[1])*(x[2]-x[1]) #area of the smallest rectangular window containing X1 points
rho<-n1/ar
r0<-1/(2*sqrt(rho)) #r0=1/(2rho) where \code{rho} is the intensity of X1 points
Xdat<-rassocU(X1,n2,r0)
Xdat
summary(Xdat)
```

```
plot(Xdat,asp=1)
plot(Xdat)
```

`rct` *Reflexivity Contingency Table (RCT)*

Description

Returns the RCT given the IPD matrix or data set `x`, the RCT is 2×2 regardless of the number of classes in the data set.

RCT is constructed by categorizing the NN pairs according to pair type as self or mixed and whether the pair is reflexive or non-reflexive. A base-NN pair is called a reflexive pair, if the elements of the pair are NN to each other; a non-reflexive pair, if the elements of the pair are not NN to each other; a self pair, if the elements of the pair are from the same class; a mixed pair, if the elements of the pair are from different classes. Row labels in the RCT are `"ref"` for reflexive and `"non-ref"` for non-reflexive and column labels are `"self"` and `"mixed"`.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

See also (Ceyhan and Bahadir (2017); Bahadir and Ceyhan (2018)) and the references therein.

Usage

```
rct(x, lab, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`lab`	The `vector` of class labels (numerical or categorical)
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the 2×2 RCT, see the description above for more detail.

Author(s)

Elvan Ceyhan

References

Bahadir S, Ceyhan E (2018). "On the Number of reflexive and shared nearest neighbor pairs in one-dimensional uniform data." *Probability and Mathematical Statistics*, **38(1)**, 123-137.

Ceyhan E, Bahadir S (2017). "Nearest Neighbor Methods for Testing Reflexivity." *Environmental and Ecological Statistics*, **24(1)**, 69-108.

See Also

`nnct`, `tct` and `scct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)

rct(ipd,cls)
rct(Y,cls,is.ipd = FALSE)
rct(Y,cls,is.ipd = FALSE,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
rct(ipd,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)

rct(ipd,cls)
```

`rdiag.clust` *Generation of Points with Clusters along the First Diagonal*

Description

An object of class `"Clusters"`.

Generates `n` 2D points with `k` ($k \geq 2$) clusters along the first diagonal where about n/k points belongs to each cluster.

If `distribution="uniform"`, the points are uniformly generated in their square supports where one square is the unit square (i.e., with vertices $(0,0),(1,0),(1,1),(0,1)$), and the others are unit squares translated $j\sqrt{2}d$ units along the first diagonal for $j = 1,2,\ldots,k-1$ (i.e., with vertices $(jd,jd),(1+jd,jd),(1+jd,1+jd),(jd,1+jd)$).

If `distribution="bvnormal"`, the points are generated from the bivariate normal distribution with means equal to the centers of the above squares (i.e., for each cluster with `mean=`$((1 + jd)/2, (1 + jd)/2)$ for $j = 0, 1, \ldots, k-1$ and the covariance matrix sdI_2, where I_2 is the 2×2 identity matrix.

Notice that the clusters are more separated, i.e., generated data indicates more clear clusters as d increases in either positive or negative direction with $d = 0$ indicating one cluster in the data. For a fixed d, when `distribution="bvnormal"`, the clustering gets stronger if the variance of each component, sd^2, gets smaller, and clustering gets weaker as the variance of each component gets larger where default is $sd = 1/6$.

Usage

```
rdiag.clust(n, k, d, sd = 1/6, distribution = c("uniform", "bvnormal"))
```

Arguments

`n`	A positive integer representing the number of points to be generated from the two clusters
`k`	A positive integer representing the number of clusters to be generated
`d`	Shift in the first diagonal indicating the level of clustering in the data. Larger absolute values in either direction (i.e., positive or negative) would yield stronger clustering.
`sd`	The standard deviation of the components of the bivariate normal distribution with default $sd = 1/6$, used only when `distribution="bvnormal"`.
`distribution`	The argument determining the distribution of each cluster. Takes on values `"uniform"` and `"bvnormal"` whose centers are d units apart along the first diagonal direction.

Value

A `list` with the elements

`type`	The type of the clustering pattern
`parameters`	The number of clusters, `k`, the diagonal shift `d` representing the level of clustering (for both distribution types) and standard deviation, `sd`, for the bivariate normal distribution only
`gen.points`	The output set of generated points from the clusters.
`desc.pat`	Description of the clustering pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The number of generated points.
`xlimit`, `ylimit`	The possible ranges of the x- and y-coordinates of the generated points

Author(s)

Elvan Ceyhan

See Also

rhor.clust and rrot.clust

Examples

```
n<-20  #or try sample(1:20,1);  #try also n<-50; n<-1000;
d<-.5 #try also -75,.75, 1
k<-3 #try also 5

#data generation
Xdat<-rdiag.clust(n,k,d)
Xdat

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#data generation (bvnormal)
n<-20  #or try sample(1:20,1);  #try also n<-50; n<-1000;
d<-.5 #try also -.75,.75, 1
k<-3 #try also 5
Xdat<-rdiag.clust(n,k,d,distr="bvnormal") #try also Xdat<-rdiag.clust(n,k,d,sd=.09,distr="bvnormal")
Xdat

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rhor.clust` *Generation of Points with Clusters along the Horizontal Axis*

Description

An object of class `"Clusters"`.

Generates `n` 2D points with `k` ($k \geq 2$) clusters along the horizontal axis where about n/k points belongs to each cluster.

If `distribution="uniform"`, the points are uniformly generated in their square supports where one square is the unit square (i.e., with vertices $(0,0),(1,0),(1,1),(0,1)$), and the others are d units shifted horizontally from each other so that their lower end vertices are $(j-1)+(j-1)d$ for $j=1,2,\ldots,k$.

If `distribution="bvnormal"`, the points are generated from the bivariate normal distribution with means equal to the centers of the above squares (i.e., for each cluster with mean=(j+(j-1)d-1/2,1/2) for $j=1,2,\ldots,k$ and the covariance matrix sdI_2, where I_2 is the 2×2 identity matrix.

Notice that the clusters are more separated, i.e., generated data indicates more clear clusters as d increases in either direction with $d=0$ indicating one cluster in the data. For a fixed d, when `distribution="bvnormal"`, the clustering gets stronger if the variance of each component, sd^2, gets smaller, and clustering gets weaker as the variance of each component gets larger where default is $sd=1/6$.

Usage

```
rhor.clust(n, k, d, sd = 1/6, distribution = c("uniform", "bvnormal"))
```

Arguments

`n`	A positive integer representing the number of points to be generated from all the clusters
`k`	A positive integer representing the number of clusters to be generated
`d`	Horizontal shift indicating the level of clustering in the data. Larger absolute values in either direction (i.e., positive or negative) would yield stronger clustering.
`sd`	The standard deviation of the components of the bivariate normal distribution with default $sd = 1/6$, used only when `distribution="bvnormal"`.
`distribution`	The argument determining the distribution of each cluster. Takes on values `"uniform"` and `"bvnormal"` whose centers are d units apart along the horizontal direction.

Value

A `list` with the elements

`type`	The type of the clustering pattern
`parameters`	The number of clusters, `k`, and the horizontal shift, `d`, representing the level of clustering (for both distribution types) and standard deviation, `sd`, for the bivariate normal distribution only.
`gen.points`	The output set of generated points from the `k` clusters.
`desc.pat`	Description of the clustering pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The number of generated points.
`xlimit, ylimit`	The possible ranges of the x- and y-coordinates of the generated points

Author(s)

Elvan Ceyhan

See Also

`rdiag.clust` and `rrot.clust`

Examples

```
n<-100;  #try also n<-50; or n<-1000;
d<-.5 #try also -.5,.75, 1
k<-3 #try also 5

#data generation
Xdat<-rhor.clust(n,k,d)
```

```
Xdat

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#data generation (bvnormal)
n<-100;  #try also n<-50; n<-1000;
d<-.1 #try also -.1, .75, 1
k<-3 #try also 5
Xdat<-rhor.clust(n,k,d,distr="bvnormal") #try also Xdat<-rhor.clust(n,k,d,sd=.15,distr="bvnormal")
Xdat

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rnonRL` *Non-Random Labeling of a Given Set of Points*

Description

An object of class `"SpatPatterns"`.

Given the set of n points, `dat`, in a region, this function assigns some of them as cases, and the rest as controls in a non-RL `type=type` fashion.

Type I nonRL pattern assigns n_1 =`round(n*prop,0)` of the data points as cases, and the rest as controls with first selecting a point, Z_i, as a case and assigning the label case to the remaining points with infection probabilities `prob=c(prop+((1-prop)*rho)/(1:k))` where `rho` is a parameter adjusting the NN dependence of infection probabilities.

Type II nonRL pattern assigns n_1 =`round(n*ult.prop,0)` of them as cases, and the rest as controls with first selecting k_0 =`round(n*init.prop,0)` as cases initially, then selecting a contagious case and then assigning the label case to the remaining points with infection probabilities inversely proportional to their position in the kNNs.

Type III nonRL pattern assigns n_1 =`round(n*prop,0)` of them as cases, and the rest as controls with first selecting a point, Z_i, as a case and assigning the label case to the remaining points with infection probabilities $prob = rho(1 - d_{ij}/d_{\max})^{pow}$ where d_{ij} is the distance from Z_j to Z_i for $j \neq i$, $d_{\max}$ is the maximum of d_{ij} values, `rho` is a scaling parameter for the infection probabilities and `pow` is a parameter in the power adjusting the distance dependence.

Type IV nonRL pattern assigns n_1 =`round(n*ult.prop,0)` of them as cases, and the rest as controls with first selecting k_0 =`round(n*init.prop,0)` as cases initially and assigning the label case to the remaining points with infection probabilities equal to the scaled bivariate normal density values at those points.

The number of cases in Types I and III will be n_1 on the average if the argument `poisson=TRUE` (i.e., n_1 =`rpois(1,round(n*prop,0))`), otherwise n_1 =`round(n*prop,0)`. The initial and ultimate number of cases in Types II and IV will be k_0 and n_1 on the average if the argument `poisson=TRUE` (i.e., k_0 =`rpois(1,round(n*init.prop,0))` and n_1 =`rpois(1,round(n*ult.prop,0)))`, otherwise they will be exactly equal to n_1 =`round(n*ult.prop,0)` and k_0 =`round(n*init.prop,0)`.

At each type, we stop when we first exceed n_1 cases. That is, the procedure ends when number of cases n_c exceed n_1, and $n_c - n_1$ of the cases (other than the initial case(s)) are randomly selected and relabeled as controls, i.e., 0s, so that the number of cases is exactly n_1.

In the output cases are labeled as 1 and controls as 0, and initial contagious case is marked with a red cross in the plot of the pattern.

See Ceyhan (2014) and the functions `rnonRLI`, `rnonRLII`, `rnonRLIII`, and `rnonRLIV` for more detail on each type of non-RL pattern.

Although the non-RL pattern is described for the case-control setting, it can be adapted for any two-class setting when it is appropriate to treat one of the classes as cases or one of the classes behave like cases and other class as controls.

The parameters of the non-RL patterns are specified in the argument `par.vec`, and the logical arguments `rand.init` and poisson pass on to the types where required. `rand.init` is not used in type I but used in all other types, poisson is used in all types, and init.from.cases is used in type I non-RL only.

Usage

```
rnonRL(
  dat,
  par.vec,
  type,
  rand.init = TRUE,
  poisson = FALSE,
  init.from.cases = TRUE
)
```

Arguments

`dat`	A set of points the non-RL procedure is applied to obtain cases and controls randomly in the `type=type` fashion (see the description).
`par.vec`	The parameter vector. It is `c(prop,k,rho)` for type I, `c(k,rho,pow,init.prop,ult.prop)` for type II, `c(prop,rho,pow)` for type III, and `c(init.prop,ult.prop,s1,s2,rho)` for type IV non-RL patterns. The parameters must be entered in this order in `par.vec` as a vector. See the respective functions for more detail on the parameters.
`type`	The type of the non-RL pattern. Takes on values `"I"`-`"IV"` for types I-IV non-RL patterns (see the description above).
`rand.init`	A logical argument (default is `TRUE`) to determine the choice of the initial case(s) in the data set, `dat` for types II-IV non-RL pattern. If `rand.init=TRUE` then the initial case(s) is (are) selected randomly from the data points, and if `rand.init=FALSE`, the first one is labeled as a case for type III and the first `init.prop*n` entries in the data set, `dat`, are labeled as the cases types II and IV.
`poisson`	A logical argument (default is `FALSE`) to determine whether the number of cases is random or fixed. In types II and IV initial and ultimate number of cases, k_0

	and n_1, will be random if `poisson=TRUE` and fixed otherwise. In types I and III the number of cases, n_1, will be random if poisson=TRUEURE and fixed otherwise. See the description.
`init.from.cases`	A logical argument (default is `TRUE`) to determine whether the initial cases at each round will be take from cases or controls in type I non-RL pattern. The initial cases are taken from cases if `init.from.cases=TRUE`, and from controls otherwise. See the function `rnonRLI`.

Value

A `list` with the elements

`pat.type`	`="cc"` for the case-control patterns for RL or non-RL of the given data points, `dat`
`type`	The type of the point pattern
`parameters`	`par.vec`, the parameters required for each type of non-RL pattern. See the description in the parameter list.
`lab`	The labels of the points as 1 for cases and 0 for controls after the nonRL procedure is applied to the data set, `dat`. Cases are denoted as red dots and controls as black circles in the plot.
`init.cases`	The initial cases in the data set, `dat`. Marked with red crosses in the plot of the points.
`cont.cases`	The contagious cases in the data set, `dat` in type II non-RL pattern. Denoted as blue points in the plot of the points.
`gen.points`, `ref.points`	Both are `NULL` for this function, as initial set of points, `dat`, are provided for all of the non-RL procedures.
`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of cases and controls.
`xlimit`, `ylimit`	The possible ranges of the x- and y-coordinates of the generated and the reference points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`rnonRLI`, `rnonRLII`, `rnonRLIII`, and `rnonRLIV`

Examples

```
#data generation
n<-40;  #try also n<-20; n<-100;
dat<-cbind(runif(n,0,1),runif(n,0,1))

#Type I non-RL pattern
#c(prop,k,rho) for type I
prop<-.5; knn<-3; rho<- .3
prv<-c(prop,knn,rho)

Xdat<-rnonRL(dat,type="I",prv) #labeled data
# or try Xdat<-rnonRL(dat,type="I",prv) for type I non-RL
Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#Type II non-RL pattern
#c(k,rho,pow,init.prop,ult.prop) for type II
rho<-.8; pow<-2; knn<-5; ip<-.3; up<-.5
prv<-c(knn,rho,pow,ip,up)

Xdat<-rnonRL(dat,type="II",prv) #labeled data
# or try Xdat<-rnonRL(dat,type="I",prv) for type I non-RL
Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#Type III non-RL pattern
#c(prop,rho,pow) for type III
prop<- .5; rho<-.8; pow<-2
prv<-c(prop,rho,pow)

Xdat<-rnonRL(dat,type="III",prv) #labeled data
# or try Xdat<-rnonRL(dat,type="I",prv) for type I non-RL
Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#Type IV non-RL pattern
#c(init.prop,ult.prop,s1,s2,rho) for type IV
```

```
ult<-.5; int<- .1; s1<-s2<-.4; rho<- .1
prv<-c(int,ult,s1,s2,rho)

Xdat<-rnonRL(dat,type="IV",prv) #labeled data
# or try Xdat<-rnonRL(dat,type="I",prv) for type I non-RL
Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rnonRLI` *Type I Non-Random Labeling of a Given Set of Points*

Description

An object of class `"SpatPatterns"`.

Given the set of n points, `dat`, in a region, this function assigns n_1 =`round(n*prop,0)` of them as cases, and the rest as controls with first selecting a point, Z_i, as a case and assigning the label case to the remaining points with infection probabilities `prob=c(prop+((1-prop)*rho)/(1:k))` where `rho` is a parameter adjusting the NN dependence of infection probabilities. The number of cases will be n_1 on the average if the argument `poisson=TRUE` (i.e., n_1 =`rpois(1,round(n*prop,0))`), otherwise n_1 =`round(n*prop,0)`. We stop when we first exceed n_1 cases. `rho` must be between `-prop/(1-prop)` and 1 for the infection probabilities to be valid. The `init.from.cases` is a logical argument (with default=`TRUE`) to determine the initial cases are from the cases or controls (the first initial case is always from controls), so, if `TRUE`, initial cases (other than the first initial case) are selected randomly among the cases (as if they are contagious), otherwise, they are selected from controls as new cases infecting their kNNs. otherwise first entry is chosen as the case (or case is recorded as the first entry) in the data set, `dat`.

Algorithmically, first all dat points are treated as non-cases (i.e., controls or healthy subjects). Then the function follows the following steps for labeling of the points:

step 0: n_1 is generated randomly from a Poisson distribution with `mean = n*prop`, so that the average number of cases is `n*prop`.

step 0: n_1 is generated randomly from a Poisson distribution with `mean = round(n*prop,0)`, so that the average number of cases will be `round(n*prop,0)` if the argument `poisson=TRUE`, else n_1 =`round(n*prop,0)`.

step 1: Initially, one point from dat is selected randomly as a case. In the first round this point is selected from the controls, and the subsequent rounds, it is selected from cases if the argument `init.from.cases=TRUE`, and from controls otherwise. Then it assigns the label case to the kNNs among controls of the initial case selected in step 1 with infection probabilities `prob=c(prop+((1-prop)*rho)/(1:k))`, see the description for the details of the parameters in the `prob`.

step 2: Then this initial case and cases among its kNNs (possibly all $k + 1$ points) in step 2 are removed from the data, and for the remaining control points step 1 is applied where initial point is from cases or control based on the argument init.from.cases.

step 3: The procedure ends when number of cases n_c exceeds n_1, and $n_c - n_1$ of the cases (other than the initial cases) are randomly selected and relabeled as controls, i.e., 0s, so that the number of cases is exactly n_1.

In the output cases are labeled as 1 and controls as 0. Note that the infection probabilities of the kNNs of each initial case increase with increasing rho, and infection probability decreases for increasing k in the kNNs.

See Ceyhan (2014) for more detail where type I non-RL pattern is the case 1 of non-RL pattern considered in Section 6 with n_1 is fixed as a parameter rather than being generated from a Poisson distribution and `init=FALSE`.

Although the non-RL pattern is described for the case-control setting, it can be adapted for any two-class setting when it is appropriate to treat one of the classes as cases or one of the classes behave like cases and other class as controls.

Usage

```
rnonRLI(dat, prop = 0.5, k, rho, poisson = FALSE, init.from.cases = TRUE)
```

Arguments

`dat`	A set of points the non-RL procedure is applied to obtain cases and controls randomly in the type I fashion (see the description).
`prop`	A real number between 0 and 1 (inclusive) representing the proportion of new cases (on the average) infected by the initial cases, i.e., number of newly infected cases (in addition to the initial cases) is Poisson with `mean=round(n*prop)` where n is the number of points in `dat`, if the argument `poisson=TRUE`, else it is `round(n*prop)`.
`k`	An integer representing the number of NNs considered for each initial case, i.e., kNNs of each initial case are candidates to be infected to become cases.
`rho`	A parameter for labeling the kNNs of each initial case as cases such that kNNs of each initial case is infected with decreasing probabilities `prob=c(prop+((1-prop)*rho)/(1:k))` where `rho` has to be between `-prop/(1-prop)` and 1 for `prob` to be a `vector` of probabilities.
`poisson`	A logical argument (default is `FALSE`) to determine whether the number of cases n_1, will be random or fixed. If `poisson=TRUE` then the n_1 is from a Poisson distribution, n_1 =`rpois(1,round(n*prop,0))` otherwise it is fixed, n_1 =`round(n*prop,0)`.
`init.from.cases`	A logical argument (default is `TRUE`) to determine whether the initial cases at each round will be take from cases or controls. At first round, the initial cases are taken from controls. And in the subsequent rounds, the initial cases are taken from cases if `init.from.cases=TRUE`, and from controls otherwise.

Value

A `list` with the elements

`pat.type`	`="cc"` for the case-control patterns for RL or non-RL of the given data points, `dat`

`type`	The type of the point pattern
`parameters`	`prop`, `rho`, and `k` values for this non-RL pattern, see the description for these parameters.
`dat.points`	The set of points non-RL procedure is applied to obtain cases and controls randomly in the type I fashion
`lab`	The labels of the points as 1 for cases and 0 for controls after the type I nonRL procedure is applied to the data set, `dat`. Cases are denoted as red dots and controls as black circles in the plot.
`init.cases`	The initial cases in the data set, `dat`. Marked with red crosses in the plot of the points.
`gen.points`, `ref.points`	Both are `NULL` for this function, as initial set of points, `dat`, are provided for the non-RL procedure.
`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of cases and controls.
`xlimit`, `ylimit`	The possible ranges of the x- and y-coordinates of the generated and the reference points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Segregation indices for disease clustering.” *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`rnonRLII`, `rnonRLIII`, `rnonRLIV`, and `rnonRL`

Examples

```
n<-40;  #try also n<-20; n<-100;
#data generation
dat<-cbind(runif(n,0,1),runif(n,0,1))

prop<-.5; #try also .25, .75
rho<- .3
knn<-3 #try 2 or 5

Xdat<-rnonRLI(dat,prop,knn,rho,poisson=FALSE,init=FALSE)
#labeled data try also poisson=TRUE or init=FALSE
Xdat

table(Xdat$lab)
```

```
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#normal original data
n<-40;  #try also n<-20; n<-100;
#data generation
dat<-cbind(rnorm(n,0,1),rnorm(n,0,1))

prop<-.50; #try also .25, .75
rho<- .3
knn<-5 #try 2 or 3

Xdat<-rnonRLI(dat,prop,knn,rho,poisson=FALSE) #labeled data try also poisson=TRUE
Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rnonRLII` *Type II Non-Random Labeling of a Given Set of Points*

Description

An object of class `"SpatPatterns"`.

Given the set of n points, `dat`, in a region, this function assigns n_1 =`round(n*ult.prop,0)` of them as cases, and the rest as controls with first selecting k_0 =`round(n*init.prop,0)` as cases initially, then selecting a contagious case and then assigning the label case to the remaining points with infection probabilities inversely proportional to their position among the kNNs.

The initial and ultimate number of cases will be k_0 and n_1 on the average if the argument `poisson=TRUE` (i.e., k_0 =`rpois(1,round(n*init.prop,0))` and n_1 =`rpois(1,round(n*ult.prop,0))`), otherwise they will be exactly equal to n_1 =`round(n*ult.prop,0)` and k_0 =`round(n*init.prop,0)`. More specifically, let $z_1, \ldots, z_{k_0}$ be the initial cases. Then one of the cases is selected as a contagious case, say z_j and then its kNNs (among the non-cases) are found. Then label these kNN non-case points as cases with infection probabilities `prob` equal to the value of the `rho*(1/(1:k))^pow` values at these points, where `rho` is a scaling parameter for the infection probabilities and `pow` is a parameter in the power adjusting the kNN dependence. We stop when we first exceed n_1 cases. `rho` has to be in $(0, 1)$ for `prob` to be a `vector` of probabilities, and for a given `rho`, `pow` must be $> \ln(rho)/\ln(k)$. If `rand.init=TRUE`, first k_0 entries are chosen as the initial cases in the data set, `dat`, otherwise, k_0 initial cases are selected randomly among the data points.

Algorithmically, first all dat points are treated as non-cases (i.e., controls or healthy subjects). Then the function follows the following steps for labeling of the points:

step 0: n_1 is generated randomly from a Poisson distribution with `mean` = `round(n*ult.prop,0)`, so that the average number of ultimate cases will be `round(n*ult.prop,0)` if the argument `poisson=TRUE`, else n_1 =`round(n*ult.prop,0)`. And k_0 is generated randomly from a Poisson distribution with `mean = round(n*init.prop,0)`, so that the average number of initial cases will be `round(n*init.prop,0)` if the argument `poisson=TRUE`, else k_0 =`round(n*init.prop,0)`.

step 1: Initially, k_0 many points from dat are selected as cases. The selection of initial cases are determined based on the argument `rand.init` (with default=`TRUE`) where if `rand.init=TRUE` then the initial cases are selected randomly from the data points, and if `rand.init= FALSE`, the first k_0 entries in the data set, `dat`, are selected as the cases.

step 2: Then it selects a contagious case among the cases, and randomly labels its `k` control NNs as cases with decreasing infection probabilities `prob=rho*(1/(1:k))^pow`. See the description for the details of the parameters in the `prob`.

step 3: The procedure ends when number of cases n_c exceeds n_1, and $n_c - n_1$ of the cases (other than the initial cases) are randomly selected and relabeled as controls, i.e., 0s, so that the number of cases is exactly n_1.

Note that the infection probabilities of the `k`NNs of each initial case increase with increasing rho; and probability of infection decreases as further NNs are considered from a contagious case (i.e., as `k` increases in the `k`NNs).

See Ceyhan (2014) for more detail where type II non-RL pattern is the case 2 of non-RL pattern considered in Section 6 with n_1 is fixed as a parameter rather than being generated from a Poisson distribution and `pow=1`.

Although the non-RL pattern is described for the case-control setting, it can be adapted for any two-class setting when it is appropriate to treat one of the classes as cases or one of the classes behave like cases and other class as controls.

Usage

```
rnonRLII(
  dat,
  k,
  rho,
  pow,
  init.prop,
  ult.prop,
  rand.init = TRUE,
  poisson = FALSE
)
```

Arguments

`dat`	A set of points the non-RL procedure is applied to obtain cases and controls randomly in the type II fashion (see the description).
`k`	An integer representing the number of NNs considered for each contagious case, i.e., `k`NNs of each contagious case are candidates to be infected to become cases.
`rho`	A scaling parameter for the probabilities of labeling the points as cases (see the description).
`pow`	A parameter in the power adjusting the `k`NN dependence in the probabilities of labeling the points as cases (see the description).

`init.prop`	A real number between 0 and 1 representing the initial proportion of cases in the data set, `dat`. The selection of the initial cases depends on the parameter `rand.init` (see the description).
`ult.prop`	A real number between 0 and 1 representing the ultimate proportion of cases in the data set, `dat` after the non-RL assignment.
`rand.init`	A logical argument (default is `TRUE`) to determine the choice of the initial cases in the data set, `dat`. If `rand.init=TRUE` then the initial cases are selected randomly from the data points, and if `rand.init= FALSE`, the first `init.prop*n` entries in the data set, `dat`, are labeled as the cases.
`poisson`	A logical argument (default is `FALSE`) to determine whether the number of initial and ultimate cases, k_0 and n_1, will be random or fixed. If `poisson=TRUE` then the k_0 and n_1 are from a Poisson distribution, $k_0 =$`rpois(1,round(n*init.prop,0))` and $n_1 =$`rpois(1,round(n*ult.prop,0))` otherwise they are fixed, $k_0 =$`round(n*init.prop,0)` and $n_1 =$`round(n*ult.prop,0)`.

Value

A `list` with the elements

`pat.type`	`="cc"` for the case-control patterns for RL or non-RL of the given data points, `dat`
`type`	The type of the point pattern
`parameters`	Number of NNs, `k`, a scaling parameter for the infection probabilities of kNNs, rho, a parameter in the power adjusting the kNN dependence of the infection probabilities, initial proportion of cases, `init.prop`, and the ultimate proportion of cases, `ult.prop`.
`dat.points`	The set of points non-RL procedure is applied to obtain cases and controls randomly in the type II fashion
`lab`	The labels of the points as 1 for cases and 0 for controls after the type II nonRL procedure is applied to the data set, `dat`. Cases are denoted as red dots and controls as black circles in the plot.
`init.cases`	The initial cases in the data set, `dat`. Denoted as red crosses in the plot of the points.
`cont.cases`	The contagious cases in the data set, `dat`. Denoted as blue points in the plot of the points.
`gen.points`, `ref.points`	Both are `NULL` for this function, as initial set of points, `dat`, are provided for the non-RL procedure.
`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of cases and controls.
`xlimit`, `ylimit`	The possible ranges of the x- and y-coordinates of the generated and the reference points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`rnonRLI`, `rnonRLIII`, `rnonRLIV`, and `rnonRL`

Examples

```
n<-40;  #try also n<-20; n<-100;
#data generation
dat<-cbind(runif(n,0,1),runif(n,0,1))

rho<-.8
pow<-2
knn<-5 #try 2 or 3
ip<-.3 #initial proportion
up<-.5 #ultimate proportion

Xdat<-rnonRLII(dat,knn,rho,pow,ip,up,poisson=FALSE) #labeled data, try poisson=TRUE
Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#normal original data
n<-40;  #try also n<-20; n<-100;
#data generation
dat<-cbind(rnorm(n,0,1),rnorm(n,0,1))

rho<-0.8
pow<-2
knn<-5 #try 2 or 3
ip<-.3 #initial proportion
up<-.5 #ultimate proportion

Xdat<-rnonRLII(dat,knn,rho,pow,ip,up,poisson=FALSE) #labeled data, try poisson=TRUE
Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rnonRLIII` *Type III Non-Random Labeling of a Given Set of Points*

Description

An object of class `"SpatPatterns"`.

Given the set of n points, `dat`, in a region, this function assigns n_1 =`round(n*prop,0)` of them as cases, and the rest as controls with first selecting a point, Z_i, as a case and assigning the label case to the remaining points with infection probabilities $prob = rho(1 - d_{ij}/d_{\max})^{pow}$ where d_{ij} is the distance from Z_j to Z_i for $j \neq i$, $d_{\max}$ is the maximum of d_{ij} values, `rho` is a scaling parameter for the infection probabilities and `pow` is a parameter in the power adjusting the distance dependence. The number of cases will be n_1 on the average if the argument `poisson=TRUE` (i.e., n_1 =`rpois(1,round(n*prop,0))`), otherwise n_1 =`round(n*prop,0)`. We stop when we first exceed n_1 cases. `rho` has to be positive for `prob` to be a `vector` of probabilities, and for a given `rho`, `pow` must be $> -\ln(rho)/\ln(1 - d_{ij}/d_{\max})$, also when `pow` is given, `rho` must be $< (1 - d_{ij}/d_{\max})^{-pow}$. If `rand.init=TRUE`, initial case is selected randomly among the data points, otherwise first entry is chosen as the case (or case is recorded as the first entry) in the data set, `dat`.

Algorithmically, first all dat points are treated as non-cases (i.e., controls or healthy subjects). Then the function follows the following steps for labeling of the points:

step 0: n_1 is generated randomly from a Poisson distribution with `mean = round(n*prop,0)`, so that the average number of cases will be round(n*prop,0) if the argument `poisson=TRUE`, else n_1 =`round(n*prop,0)`.

step 1: Initially, one point from dat is selected as a case. The selection of initial case is determined based on the argument `rand.init` (with default=`TRUE`) where if `rand.init=TRUE` then the initial case is selected randomly from the data points, and if `rand.init= FALSE`, the first entry in the data set, `dat`, is selected as the case.

step 2: Then it assigns the label case to the remaining points with infection probabilities $prob = rho(1 - d_{ij}/d_{\max})^{pow}$, see the description for the details of the parameters in the `prob`.

step 3: The procedure ends when number of cases n_c exceeds n_1, and $n_c - n_1$ of the cases (other than the initial contagious case) are randomly selected and relabeled as controls, i.e., 0s, so that the number of cases is exactly n_1.

In the output cases are labeled as 1 and controls as 0, and initial contagious case is marked with a red cross in the plot of the pattern. Note that the infection probabilities of the points is inversely proportional to their distances to the initial case and increase with increasing `rho`. This function might take a long time for certain choices of the arguments. For example, if `pow` is taken to be too large, the infection probabilities would be too small, and case assignment will take a rather long time.

See Ceyhan (2014) for more detail where type III non-RL pattern is the case 3 of non-RL pattern considered in Section 6 with n_1 is fixed as a parameter rather than being generated from a Poisson distribution and $k_{den} = 1$ and pow is represented as k_{pow}.

Although the non-RL pattern is described for the case-control setting, it can be adapted for any two-class setting when it is appropriate to treat one of the classes as cases or one of the classes behave like cases and other class as controls.

Usage

```
rnonRLIII(dat, prop, rho, pow, rand.init = TRUE, poisson = FALSE)
```

Arguments

`dat`	A set of points the non-RL procedure is applied to obtain cases and controls randomly in the type III fashion (see the description).
`prop`	A real number between 0 and 1 (inclusive) representing the proportion of new cases (on the average) infected by the initial case, i.e., number of newly infected cases (in addition to the first case) is Poisson with `mean=round(n*prop)` where n is the number of points in `dat`, if the argument `poisson=TRUE`, else it is `round(n*prop)`.
`rho`	A scaling parameter for the probabilities of labeling the points as cases (see the description).
`pow`	A parameter in the power adjusting the distance dependence in the probabilities of labeling the points as cases (see the description).
`rand.init`	A logical argument (default is `TRUE`) to determine the choice of the initial case in the data set, `dat`. If `rand.init=TRUE` then the initial case is selected randomly from the data points, and if `rand.init= FALSE`, the first entry in the data set, `dat`, is labeled as the initial case.
`poisson`	A logical argument (default is `FALSE`) to determine whether the number of cases n_1, will be random or fixed. If `poisson=TRUE` then the n_1 is from a Poisson distribution, n_1 =`rpois(1,round(n*prop,0))` otherwise it is fixed, n_1 =`round(n*prop,0)`.

Value

A `list` with the elements

`pat.type`	`="cc"` for the case-control patterns for RL or non-RL of the given data points, `dat`
`type`	The type of the point pattern
`parameters`	rho and pow, where `rho` is the scalign parameter and `pow` is the parameter in the power adjusting the distance dependence in probabilities of labeling the points as cases.
`dat.points`	The set of points non-RL procedure is applied to obtain cases and controls randomly in the type III fashion
`lab`	The labels of the points as 1 for cases and 0 for controls after the type III nonRL procedure is applied to the data set, `dat`. Cases are denoted as red dots and controls as black circles in the plot.
`init.cases`	The initial case in the data set, `dat`. Marked with a red cross in the plot of the points.
`cont.cases`	The contagious cases in the data set, `dat`. Denoted as blue points in the plot of the points.
`gen.points`, `ref.points`	Both are `NULL` for this function, as initial set of points, `dat`, are provided for the non-RL procedure.

`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of cases and controls.
`xlimit, ylimit`	The possible ranges of the x- and y-coordinates of the generated and the reference points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`rnonRLI`, `rnonRLII`, `rnonRLIV`, and `rnonRL`

Examples

```
n<-40;  #try also n<-20; n<-100;
prop<- .5; #try also .25, .75
#data generation
dat<-cbind(runif(n,0,1),runif(n,0,1))

rho<-.8
pow<-2

Xdat<-rnonRLIII(dat,prop,rho,pow,poisson=FALSE) #labeled data, try also poisson=TRUE

Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#normal original data
n<-40;  #try also n<-20; n<-100;
dat<-cbind(rnorm(n,0,1),rnorm(n,0,1))

prop<- .5; #try also .25, .75
rho<-.8
pow<-2

Xdat<-rnonRLIII(dat,prop,rho,pow,poisson=FALSE) #labeled data, try also poisson=TRUE
Xdat

table(Xdat$lab)
```

```
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rnonRLIV` *Type IV Non-Random Labeling of a Given Set of Points*

Description

An object of class `"SpatPatterns"`.

Given the set of n points, `dat`, in a region, this function assigns n_1 =`round(n*ult.prop,0)` of them as cases, and the rest as controls with first selecting k_0 =`round(n*init.prop,0)` as cases initially and assigning the label case to the remaining points with infection probabilities equal to the scaled bivariate normal density values at those points. The initial and ultimate number of cases will be k_0 and n_1 on the average if the argument `poisson=TRUE` (i.e., k_0 =`rpois(1,round(n*init.prop,0))` and n_1 =`rpois(1,round(n*ult.prop,0))`), otherwise they will be exactly equal to n_1 =`round(n*ult.prop,0)` and k_0 =`round(n*init.prop,0)`. More specifically, let $z_1, \ldots, z_{k_0}$ be the initial cases and for $j = 1, 2, \ldots, k_0$ let $\phi_{G,j}(z_i)$ be the value of the pdf of the $BVN(z_j, s_1, s_2, rho)$, which is the bivariate normal distribution mean=z_j and standard deviations of the first and second components being s_1 and s_2 (denoted as `s1` and `s2` as arguments of the function) and correlation between them being ρ (denoted as `rho` as an argument of the function) (i.e., the covariance matrix is $\Sigma = S$ where $S_{11} = s_1^2$, $S_{22} = s_2^2$, $S_{12} = S_{21} = s_1 s_2 \rho$). Add these pdf values as $p_j = \sum_{j=1}^{k_0} \phi_{G,j}(z_i)$ for each $i = 1, 2, \ldots, n$ and find $p_{\max} = \max p_j$. Then label the points (other than the initial cases) as cases with infection probabilities `prob` equal to the value of the $p_j/p_{\max}$ values at these points. We stop when we first exceed n_1 cases. ρ has to be in (-1,1) for `prob` to be a valid probability and s_1 and s_2 must be positive (actually these are required for the BVN density to be nondegenerately defined). If `rand.init=TRUE`, first k_0 entries are chosen as the initial cases in the data set, `dat`, otherwise, k_0 initial cases are selected randomly among the data points.

Algorithmically, first all dat points are treated as non-cases (i.e., controls or healthy subjects). Then the function follows the following steps for labeling of the points:

step 0: n_1 is generated randomly from a Poisson distribution with `mean = round(n*ult.prop,0)`, so that the average number of ultimate cases will be `round(n*ult.prop,0)` if the argument `poisson=TRUE`, else n_1 =`round(n*ult.prop,0)`. And k_0 is generated randomly from a Poisson distribution with `mean = round(n*init.prop,0)`, so that the average number of initial cases will be round(n*init.prop,0) if the argument `poisson=TRUE`, else k_0 =`round(n*init.prop,0)`.

step 1: Initially, k_0 many points from dat are selected as cases. The selection of initial cases are determined based on the argument `rand.init` (with default=`TRUE`) where if `rand.init=TRUE` then the initial cases are selected randomly from the data points, and if `rand.init= FALSE`, the first k_0 entries in the data set, `dat`, are selected as the cases.

step 2: Then it assigns the label case to the remaining points with infection probabilities $prob = \sum_{j=1}^{k_0} \phi_{G,j}(z_i)/p_{\max}$, which is the sum of the BVN densities scaled by the maximum of such sums. See the description for the details of the parameters in the `prob`.

step 3: The procedure ends when number of cases n_c exceed n_1, and $n_c - n_1$ of the cases (other than the initial cases) are randomly selected and relabeled as controls, i.e., 0s, so that the number of cases is exactly n_1.

In the output cases are labeled as 1 and controls as 0, and initial contagious case is marked with a red cross in the plot of the pattern.

See Ceyhan (2014) for more detail where type IV non-RL pattern is the case 4 of non-RL pattern considered in Section 6 with n_1 and k_0 are fixed as parameters and `rho` is represented as k_{pow} and $rho/k_{den} = 1$ in the article.

Although the non-RL pattern is described for the case-control setting, it can be adapted for any two-class setting when it is appropriate to treat one of the classes as cases or one of the classes behave like cases and other class as controls.

Usage

```
rnonRLIV(
  dat,
  init.prop,
  ult.prop,
  s1,
  s2,
  rho,
  rand.init = TRUE,
  poisson = FALSE
)
```

Arguments

`dat`	A set of points the non-RL procedure is applied to obtain cases and controls randomly in the type IV fashion (see the description).
`init.prop`	A real number between 0 and 1 representing the initial proportion of cases in the data set, `dat`. The selection of the initial cases depends on the parameter `rand.init` and the number of initial cases depends on the parameter poisson (see the description).
`ult.prop`	A real number between 0 and 1 representing the ultimate proportion of cases in the data set, `dat` after the non-RL assignment. The number of ultimate cases depends on the parameter poisson (see the description).
`s1, s2`	Positive real numbers representing the standard deviations of the first and second components of the bivariate normal distribution.
`rho`	A real number between -1 and 1 representing the correlation between the first and second components of the bivariate normal distribution.
`rand.init`	A logical argument (default is `TRUE`) to determine the choice of the initial case in the data set, `dat`. If `rand.init=TRUE` then the initial case is selected randomly from the data points, and if `rand.init= FALSE`, the first k_0 entries in the data set, `dat`, is labeled as the initial case.
`poisson`	A logical argument (default is `FALSE`) to determine whether the number of initial and ultimate cases, k_0 and n_1, will be random or fixed. If `poisson=TRUE` then

the k_0 and n_1 are from a Poisson distribution, k_0 =`rpois(1,round(n*init.prop,0))` and n_1 =`rpois(1,round(n*ult.prop,0))` otherwise they are fixed, k_0 =`round(n*init.prop,0)` and n_1 =`round(n*ult.prop,0)`.

Value

A `list` with the elements

`pat.type`	="cc" for the case-control patterns for RL or non-RL of the given data points, `dat`
`type`	The type of the point pattern
`parameters`	initial and ultimate proportion of cases after the non-RL procedure is applied to the data, `s1`, `s2` and `rho` which are standard deviations and the correlation for the components of the bivariate normal distribution.
`dat.points`	The set of points non-RL procedure is applied to obtain cases and controls randomly in the type IV fashion
`lab`	The labels of the points as 1 for cases and 0 for controls after the type IV nonRL procedure is applied to the data set, `dat`. Cases are denoted as red dots and controls as black circles in the plot.
`init.cases`	The initial cases in the data set, `dat`. Marked with red crosses in the plot of the points.
`gen.points`, `ref.points`	Both are `NULL` for this function, as initial set of points, `dat`, are provided for the non-RL procedure.
`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of cases and controls.
`xlimit`, `ylimit`	The possible ranges of the x- and y-coordinates of the generated and the reference points

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Segregation indices for disease clustering.” *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`rnonRLI`, `rnonRLII`, `rnonRLIII`, and `rnonRL`

Examples

```
n<-40;  #try also n<-20; n<-100;
ult<-.5; #try also .25, .75
#data generation
dat<-cbind(runif(n,0,1),runif(n,0,1))

int<-.1
s1<-s2<-.4
rho<- .1

Xdat<-rnonRLIV(dat,int,ult,s1,s2,rho,poisson=FALSE) #labeled data, try also with poisson=TRUE
Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#normal original data
n<-40;  #try also n<-20; n<-100;
dat<-cbind(rnorm(n,0,1),rnorm(n,0,1))
ult<-.5; #try also .25, .75

int<-.1
s1<-s2<-.4
rho<-0.1

Xdat<-rnonRLIV(dat,int,ult,s1,s2,rho,poisson=FALSE) #labeled data, try also with poisson=TRUE
Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rrot.clust` *Generation of Points with Rotational Clusters*

Description

An object of class `"Clusters"`.

Generates `n` 2D points with `k` ($k \geq 2$) clusters with centers `d` unit away from origin and angles between the rays joining successive centers and origin is $2\pi/k$ where about n/k points belongs to each cluster.

If `distribution="uniform"`, the points are uniformly generated in their square supports with unit edge lengths and centers at $(d\cos(j2\pi/k), d\cos(j2\pi/k))$ for $j = 1, 2, \ldots, k$.

If `distribution="bvnormal"`, the points are generated from the bivariate normal distribution with means equal to the centers of the above squares (i.e., for each cluster with `mean=`$(d\cos(j2\pi/k), d\cos(j2\pi/k))$ for $j = 1, 2, \ldots, k$ and the covariance matrix sdI_2, where $sd = d\sqrt{2(1 - cos(2\pi/k))}/3$ and I_2 is the 2×2 identity matrix.

Notice that the clusters are more separated, i.e., generated data indicates more clear clusters as d increases in either direction with $d = 0$ indicating one cluster in the data. For a fixed d, when `distribution="bvnormal"`, the clustering gets stronger if the variance of each component, sd^2, gets smaller, and clustering gets weaker as the variance of each component gets larger where default is $sd = d\sqrt{2(1 - cos(2\pi/k))}/3$.

Usage

```
rrot.clust(
  n,
  k,
  d,
  sd = d * sqrt(2 * (1 - cos(2 * pi/k)))/3,
  distribution = c("uniform", "bvnormal")
)
```

Arguments

`n`	A positive integer representing the number of points to be generated from all the clusters
`k`	A positive integer representing the number of clusters to be generated
`d`	Radial shift indicating the level of clustering in the data. Larger absolute values in either direction (i.e., positive or negative) would yield stronger clustering.
`sd`	The standard deviation of the components of the bivariate normal distribution with default $sd = d\sqrt{2(1 - cos(2\pi/k))}/3$, used only when `distribution="bvnormal"`.
`distribution`	The argument determining the distribution of each cluster. Takes on values `"uniform"` and `"bvnormal"` whose centers are d units apart along the horizontal direction.

Value

A `list` with the elements

`type`	The type of the clustering pattern
`parameters`	The number of clusters, `k`, and the radial shift, `d`, representing the level of clustering (for both distribution types) and standard deviation, `sd`, for the bivariate normal distribution only.
`gen.points`	The output set of generated points from the `k` clusters.
`desc.pat`	Description of the clustering pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The number of generated points.
`xlimit, ylimit`	The possible ranges of the x- and y-coordinates of the generated points

Author(s)

Elvan Ceyhan

See Also

`rdiag.clust` and `rhor.clust`

Examples

```
n<-100;  #try also n<-50; n<-1000;
d<- 1.5 #try also -1, 1, 1.5, 2
k<-3 #try also 5
#data generation
Xdat<-rrot.clust(n,k,d)
Xdat

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#data generation (bvnormal)
n<-100;  #try also n<-50; n<-1000;
d<- 1.5 #try also -1, 1, 1.5, 2
k<-3 #try also 5
Xdat<-rrot.clust(n,k,d,distr="bvnormal") #also try Xdat<-rrot.clust(n,k,d,sd=.5,distr="bvnormal")
Xdat

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rseg` *Generation of Points under Segregation of Two Classes*

Description

An object of class `"SpatPatterns"`.

Generates `n_i` 2D points from class j with parameters r_j for $j = 1, 2$. The generated points are from two different classes which are segregated from each other. The pattern generation starts with the initial points `X1.init` and `X2.init` (with default=`NULL` for both). If both `X1.init=NULL` and `X2.init=NULL`, both `X1.init` and `X2.init` are generated uniformly in the unit square. If only `X1.init=NULL`, `X1.init` is the sum of a point uniformly generated in the unit square and `X2.init` and if only `X2.init=NULL`, `X2.init` is the sum of a point uniformly generated in the unit square and `X1.init`. After the initial points from each class are available, n_j points from class j are generated as `Xj[i,]<-Xj[(i-1),]+ru*c(cos(tu),sin(tu))` where `ru<-runif(1,0,rj)` and `tu<-runif(1,0,2*pi)` for $i = 2, \ldots, n_j$ with `Xj[1,]=Xj.init` for $j = 1, 2$. That is, at each step the new point in class j is generated within a circle with radius equal to r_j (uniform in the polar coordinates). Note that, the level of segregation is stronger if the initial points are further apart, and the level of segregation increases as the radius values gets smaller.

Usage

```
rseg(n1, n2, r1, r2, X1.init = NULL, X2.init = NULL)
```

Arguments

`n1`, `n2`	Positive integers representing the number of class 1 and class 2 (i.e., X_1 and X_2) points to be generated under the segregation pattern.
`r1`, `r2`	Positive real numbers representing the radius of attraction within class, i.e., radius of the circle center and generated points are from the same class.
`X1.init`, `X2.init`	2D points representing the initial points for the segregated classes, default=`NULL` for both. If both `X1.init=NULL` and `X2.init=NULL`, both `X1.init` and `X2.init` are generated uniformly in the unit square. If only `X1.init=NULL`, `X1.init` is the sum of a point uniformly generated in the unit square and `X2.init` and if only `X2.init=NULL`, `X2.init` is the sum of a point uniformly generated in the unit square and `X1.init`. The initial points are marked with crosses in the plot of the points.

Value

A `list` with the elements

`pat.type`	`"2c"` for the 2-class pattern of segregation of the two classes
`type`	The type of the point pattern
`parameters`	Radial (i.e., circular) within class radii of segregation, `r1` and `r2`, controlling the level of segregation
`lab`	The class labels of the generated points, it is 1 class 1 or X_1 points and 2 for class 2 or X_2 points
`init.cases`	The initial points for class 1 and class 2, one initial point for each class.
`gen.points`	The output set of generated points (i.e., class 1 and class 2 points) segregated from each other.
`ref.points`	The input set of reference points, it is `NULL` for this function.
`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The `vector` of two numbers, which are the number of generated class 1 and class 2 points.
`xlimit`, `ylimit`	The possible ranges of the x- and y-coordinates of the generated and the initial points

Author(s)

Elvan Ceyhan

See Also

`rassoc`

Examples

```
n1<-20;  #try also n1<-10; n1<-100;
n2<-20; #try also n1<-40; n2<-50

r1<-.3; r2<-.2

#data generation
Xdat<-rseg(n1,n2,r1,r2) #labeled data
Xdat

table(Xdat$lab)

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#with one initial point
X1init<-c(3,2)

Xdat<-rseg(n1,n2,r1,r2,X1.init=X1init)
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#with two initial points
X1init<-c(3,2)
X2init<-c(4,2)

Xdat<-rseg(n1,n2,r1,r2,X1init,X2init)
Xdat
summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`rself.ref` *Generation of Points from Self Correspondence Pattern*

Description

An object of class `"SpatPatterns"`.

Generates n_1 2D points from class 1 and n_2 (denoted as `n2` as an argument) 2D points from class 2 in such a way that self-reflexive pairs are more frequent than expected under CSR independence.

If `distribution="uniform"`, the points from class 1, say X_i are generated as follows: $X_i \overset{iid}{\sim} Uniform(S_1)$ for $S_1 = (c1r[1], c1r[2])^2$ for $i = 1, 2, \ldots, n_{1h}$ where $n_{1h} = \lfloor n_1/2 \rfloor$, and for $k = n_{1h}, +1, \ldots, n_1$, $X_k = X_{k-n_{1h}} + r(\cos(T_k), \sin(T_k))$ where $r \sim Uniform(0, r_0)$ and T_k are iid $\sim Uniform(0, 2\pi)$. Similarly, the points from class 2, say Y_j are generated as follows: $Y_j \overset{iid}{\sim} Uniform(S_2)$ for $S_2 = (c2r[1], c2r[2])^2$ for $j = 1, 2, \ldots, n_{2h}$ where $n_{2h} = \lfloor n_2/2 \rfloor$), and for $l = n_{2h}, +1, \ldots, n_2$, $Y_l = Y_{l-n_{2h}} + r(\cos(T_l), \sin(T_l))$ where $r \sim Uniform(0, r_0)$ and $T_l \overset{iid}{\sim} Uniform(0, 2\pi)$. This version is the case IV in the article (Ceyhan (2018)).

If `distribution="bvnormal"`, the points from class 1, say X_i are generated as follows: $X_i \overset{iid}{\sim} BVN(CM(S_1), I_{2x})$ where $CM(S_1)$ is the center of mass of S_1 and I_{2x} is a 2×2 matrix with diagonals equal to s_1^2 with $s_1 = (c1r[2] - c1r[1])/3$ and off-diagonals are 0 for $i = 1, 2, \ldots, n_{1h}$ where $n_{1h} = \lfloor n_1/2 \rfloor$, and for $k = n_{1h} + 1, \ldots, n_1$, $X_k = Z_k + r(\cos(T_k), \sin(T_k))$ where $Z_k \sim BVN(X_{k-n_{1h}}, I_2(r_0))$ with $I_2(r_0)$ being the 2×2 matrix with diagonals $r_0/3$ and 0 off-diagonals, $r \sim Uniform(0, r_0)$ and T_k are iid $\sim Uniform(0, 2\pi)$. Similarly, the points from class 2, say Y_j are generated as follows: $Y_j \overset{iid}{\sim} BVN(CM(S_2), I_{2y})$ where $CM(S_1)$ is the center of mass of S_1 and I_{2y} is a 2×2 matrix with diagonals equal to s_2^2 with $s_2 = (c2r[2] - c2r[1])/3$ and off-diagonals are 0 for $j = 1, 2, \ldots, n_{2h}$ where $n_{2h} = \lfloor n_2/2 \rfloor$), and for $l = n_{2h}, +1, \ldots, n_2$, $Y_l = W_k + r(\cos(T_l), \sin(T_l))$ where $W_l \sim BVN(Y_{l-n_{2h}}, I_2(r_0))$ with $I_2(r_0)$ being the 2×2 matrix with diagonals $r_0/3$ and 0 off-diagonals, $r \sim Uniform(0, r_0)$ and $T_l \overset{iid}{\sim} Uniform(0, 2\pi)$.

Notice that the classes will be segregated if the supports S_1 and S_2 are separated, with more separation implying stronger segregation. Furthermore, r_0 (denoted as `r0` as an argument) determines the level of self-reflexivity or self correspondence, i.e., smaller r_0 implies a higher level of self correspondence and vice versa for higher r_0 .

See also (Ceyhan (2018)) and the references therein.

Usage

```
rself.ref(n1, n2, c1r, c2r, r0, distribution = c("uniform", "bvnormal"))
```

Arguments

`n1, n2`	Positive integers representing the numbers of points to be generated from the two classes
`c1r, c2r`	Ranges of the squares which constitute the supports of the two classes
`r0`	The radius of attraction which determines the level of self-reflexivity (or self correspondence) in both the uniform and bvnormal distributions for the two classes
`distribution`	The argument determining the distribution of each class. Takes on values `"uniform"` and `"bvnormal"` (see the description for the details).

Value

A `list` with the elements

`pat.type`	`"2c"` for the 2-class pattern of self-correspondence of the two classes
`type`	The type of the spatial pattern
`parameters`	The radius of attraction r_0 which determines the level of self-correspondence.

`lab`	The class labels of the generated points, it is 1 class 1 or X1 points and 2 for class 2 or X_2 points
`init.cases`	The initial points for class 1 and class 2, one initial point for each class, marked with a cross in the plot.
`gen.points`	The output set of generated points from the self-correspondence pattern.
`ref.points`	The input set of reference points, it is `NULL` for this function.
`desc.pat`	Description of the species correspondence pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The number of generated points.
`xlimit`, `ylimit`	The possible ranges of the x- and y-coordinates of the generated points

Author(s)

Elvan Ceyhan

See Also

`Zself.ref` and `Xsq.spec.cor`

Examples

```
n1<-50;  #try also n1<-50; n1<-1000;
n2<-50;  #try also n2<-50; n2<-1000;

c1r<-c(0,1) #try also c(0,5/6), C(0,3/4), c(0,2/3)
c2r<-c(0,1) #try also c(1/6,1), c(1/4,1), c(1/3,1)
r0<-1/9 #try also 1/7, 1/8

#data generation
Xdat<-rself.ref(n1,n2,c1r,c2r,r0)
Xdat

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)

#data generation (bvnormal)
Xdat<-rself.ref(n1,n2,c1r,c2r,r0,distr="bvnormal")
Xdat

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`runif.circ` *Generation of Uniform Points in a Circle*

Description

An object of class `"SpatPatterns"`.

Generates `n` 2D points uniformly in the circle with center=`cent` and radius=`rad` using the rejection sampling approach (i.e., the function generates points in the smallest square containing the circle, keeping only the points inside the circle until n points are generated). The defaults for `cent=c(0,0)` and `rad=1`.

Usage

```
runif.circ(n, cent = c(0, 0), rad = 1)
```

Arguments

`n`	A positive integer representing the number of points to be generated uniformly in the circle
`cent`	A 2D point representing the center of the circle, with default=`c(0,0)`
`rad`	A positive real number representing the radius of the circle.

Value

A `list` with the elements

`pat.type`	`"1c"` for the 1-class pattern of the uniform data in the circle
`type`	The type of the point pattern
`parameters`	center of the circle, `cent`, and the radius of the circle, `rad`
`lab`	The class labels of the generated points, `NULL` for this function, since points belong to the same class
`init.cases`	The initial points, `NULL` for this function
`gen.points`	The output set of generated points uniform in the circle.
`ref.points`	The input set of reference points, it is `NULL` for this function.
`desc.pat`	Description of the point pattern
`mtitle`	The `"main"` title for the plot of the point pattern
`num.points`	The number of generated points.
`xlimit, ylimit`	The possible ranges of the x- and y-coordinates of the generated points

Author(s)

Elvan Ceyhan

See Also

runif

Examples

```
n<-20  #or try sample(1:20,1);  #try also 10, 100, or 1000;
r<-.1; #try also r<-.3 or .5
cent<-c(1,2)

#data generation
Xdat<-runif.circ(n,cent,r) #generated data
Xdat

summary(Xdat)
plot(Xdat,asp=1)
plot(Xdat)
```

`seg.ind` *Dixon's Segregation Indices for NNCTs*

Description

Returns Dixon's segregation indices in matrix form based on entries of the NNCT, `ct`. Segregation index for cell i, j is defined as $log(N_{ii}(n-n_i)/((n_i-N_{ii})(n_i-1))$ if $i=j$ and as $log(N_{ij}(n-n_j-1)/((n_i-N_{ij})(n_j))$ if $i \neq j$. See (Dixon (2002); Ceyhan (2014)).

The argument `inf.corr` is a logical argument (default=`FALSE`) to avoid $\pm\infty$ for the segregation indices. If `TRUE` indices are modified so that they are finite and if `FALSE` the above definition is used. (See Ceyhan (2014) for more detail).

Usage

```
seg.ind(ct, inf.corr = FALSE)
```

Arguments

`ct`	A contingency table, in particular an NNCT
`inf.corr`	A logical argument (default=`FALSE`). If `TRUE`, indices are modified so that they are finite and if `FALSE` the above definition in the description is used.

Value

Returns a `matrix` of segregation indices which is of the same dimension as `ct`.

Author(s)

Elvan Ceyhan

See Also

`Pseg.coeff`, `seg.coeff`, `Zseg.ind`, and `Zseg.ind.ct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct
seg.ind(ct)
seg.ind(ct,inf.corr = TRUE)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

seg.ind(ct)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

seg.ind(ct)
seg.ind(ct,inf.corr = TRUE)

ct<-matrix(c(0,10,5,5),ncol=2)
seg.ind(ct)

seg.ind(ct,inf.corr = TRUE)
```

`sharedNNmc` *The Shared NN Vectors for Multiple Classes*

Description

Returns a `matrix` with k rows where each row is the vector of number of points with shared NNs, $Q_i = (Q_{i0}, Q_{i1}, \ldots)$ where Q_{ij} is the number of class i points that are NN to class j points. The function also returns the indices of columns with nonzero sums as a vector.

The output matrix of shared NNs is used in testing symmetry in shared NN structure (i.e., Q-symmetry or Pielou's second type of symmetry), e.g., in functions `Qsym.ct` and `Qsym.test`.

See also (Pielou (1961); Ceyhan (2014)) and the references therein.

Usage

```
sharedNNmc(x, lab, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`lab`	The `vector` of class labels (numerical or categorical)
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix (IPD matrix), otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

The function `Qval` returns the Q value

The function `Qvec` returns a `list` with two elements

`q`	the Q value, the number of shared NNs
`qvec`	the `vector` of Q_j values

The function `sharedNN` returns a `matrix` with 2 rows, where first row is the j values and second row is the corresponding vector of Q_j values

The function `Rval` returns the R value, the number of reflexive NNs.

Returns a `list` with two elements

`Nv`	A k-row matrix of shared NNs by class where each row of the matrix is the vector of number of points with shared NNs $Q_i = (Q_{i0}, Q_{i1}, \ldots)$ where Q_{ij} is the number of class i points that are NN to j points.
`col.ind`	The `vector` of indices of columns with nonzero sums

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Pielou EC (1961). "Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships." *Journal of Ecology*, **49(2)**, 255-269.

See Also

`Qval`, `Qvec` and `sharedNN`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)

sharedNNmc(ipd,cls)
sharedNNmc(Y,cls,is.ipd = FALSE)
sharedNNmc(Y,cls,is.ipd = FALSE,method="max")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
sharedNNmc(ipd,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)

sharedNNmc(ipd,cls)
```

`SkewTk` *Skewness of Cuzick and Edwards T_k Test statistic*

Description

This function estimates the skewness of Cuzick and Edwards T_k test statistic under the RL hypothesis. Skewness of a random variable T is defined as $E(T-\mu)^3/(E(T-\mu)^2)^{1.5}$ where $\mu = ET$.

Skewness is used for Tango's correction to Cuzick and Edwards kNN test statistic, T_k. Tango's correction is a chi-square approximation, and its degrees of freedom is estimated using the skewness estimate (see page 121 of Tango (2007)).

The argument, n_1, is the number of cases (denoted as `n1` as an argument) and `k` is the number of NNs considered in T_k test statistic. The argument of the function is the A_{ij} matrix, `a`, which is the output of the function `aij.mat`. However, inside the function we symmetrize the matrix `a` as `b <- (a+a^t)/2`, to facilitate the formulation.

The number of cases are denoted as n_1 and number of controls as n_0 in this function to match the case-control class labeling, which is just the reverse of the labeling in Cuzick and Edwards (1990).

Usage

```
SkewTk(n1, k, a)
```

Arguments

`n1`	Number of cases
`k`	Integer specifying the number of NNs (of subject i)
`a`	A_{ij} matrix which is the output of the function `aij.mat`.

Value

The skewness of Cuzick and Edwards T_k test statistic for disease clustering

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

Tango T (2007). "A class of multiplicity adjusted tests for spatial clustering based on case-control point data." *Biometrics*, **63**, 119-127.

See Also

`ceTk`, `EV.Tk`, and `varTk`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)
n1<-sum(cls==1)

k<-sample(1:5,1) # try also 3, 5, sample(1:5,1)
k
a<-aij.mat(Y,k)

SkewTk(n1,k,a)
```

`summary.Clusters` *Return a summary of a* `Clusters` *object*

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the `type` of the pattern, `parameters` of the pattern, study window, some sample points from the generated pattern, reference points (if any for the bivariate pattern), and number of points for each class

Usage

```
## S3 method for class 'Clusters'
summary(object, ...)
```

Arguments

`object`	Object of class `Clusters`.
`...`	Additional parameters for `summary`.

Value

The `call` of the object of class '`Clusters`', the `type` of the pattern, `parameters` of the pattern, study window, some sample points from the generated pattern, reference points (if any for the bivariate pattern), and number of points for each class

Examples

```
#TBF
```

`summary.SpatPatterns` *Return a summary of a* `SpatPatterns` *object*

Description

Returns the below information about the `object`:

`call` of the function defining the `object`, the `type` of the pattern, `parameters` of the pattern, study window, some sample points from the generated pattern, reference points (if any for the bivariate pattern), and number of points for each class

Usage

```
## S3 method for class 'SpatPatterns'
summary(object, ...)
```

Arguments

`object`	Object of class `SpatPatterns`.
`...`	Additional parameters for `summary`.

Value

The `call` of the object of class '`SpatPatterns`', the `type` of the pattern, `parameters` of the pattern, study window, some sample points from the generated pattern, reference points (if any for the bivariate pattern), and number of points for each class

Examples

```
#TBF
```

`swamptrees` *Tree Species in a Swamp Forest*

Description

Locations and species classification of trees in a plot in the Savannah River, SC, USA. Locations are given in meters, rounded to the nearest 0.1 decimal. The data come from a one-hectare (200-by-50m) plot in the Savannah River Site. The 734 mapped stems included 156 Carolina ashes (Fraxinus caroliniana), 215 water tupelos (Nyssa aquatica), 205 swamp tupelos (Nyssa sylvatica), 98 bald cypresses (Taxodium distichum) and 60 stems from 8 additional three species (labeled as Others (OT)). The plots were set up by Bill Good and their spatial patterns described in (Good and Whipple (1982)), the plots have been maintained and resampled by Rebecca Sharitz and her colleagues of the Savannah River Ecology Laboratory. The data and some of its description are borrowed from the swamp data entry in the `dixon` package in the CRAN repository.

See also (Good and Whipple (1982); Jones et al. (1994); Dixon (2002)).

Usage

```
data(swamptrees)
```

Format

A data frame with 734 rows and 4 variables

Details

Text describing the variable (i.e., column) names in the data set.

- x,y: x and y (i.e., Cartesian) coordinates of the trees
- live: a categorical variable that indicates the tree is alive (labeled as 1) or dead (labeled as 0)
- sp: species label of the trees:

  FX: Carolina ash (Fraxinus caroliniana)
  NS: Swamp tupelo (Nyssa sylvatica)
  NX: Water tupelo (Nyssa aquatica)
  TD: Bald cypress (Taxodium distichum)
  OT: Other species

Source

Prof. Philip Dixon's website

References

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

Good BJ, Whipple SA (1982). “Tree spatial patterns: South Carolina bottomland and swamp forests.” *Bulletin of the Torrey Botanical Club*, **109(4)**, 529-536.

Jones RH, Sharitz RR, James SM, Dixon PM (1994). “Tree population dynamics in seven South Carolina mixed-species forests.” *Bulletin of the Torrey Botanical Club*, **121(4)**, 360-368.

Examples

```
data(swamptrees)
plot(swamptrees$x,swamptrees$y, col=as.numeric(swamptrees$sp),pch=19,
xlab='',ylab='',main='Swamp Trees')
```

`Tcomb` *Cuzick & Edwards Tcomb Test Statistic*

Description

This function computes the value of Cuzick & Edwards T_{comb} test statistic in disease clustering, where T_{comb} is a linear combination of some T_k tests.

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0’s and 1’s accordingly.

The argument `klist` is the `vector` of integers specifying the indices of the T_k values used in obtaining the T_{comb}.

The logical argument `nonzero.mat` (default=`TRUE`) is for using the A matrix if `FALSE` or just the matrix of nonzero locations in the A matrix (if `TRUE`) in the computations.

The logical argument `asy.cov` (default=`FALSE`) is for using the asymptotic covariance or the exact (i.e., finite sample) covariance for the vector of T_k values used in `Tcomb` in the standardization of T_{comb}. If `asy.cov=TRUE`, the asymptotic covariance is used, otherwise the exact covariance is used.

See page 87 of (Cuzick and Edwards (1990)) for more details.

Usage

```
Tcomb(
  dat,
  cc.lab,
  klist,
  case.lab = NULL,
  nonzero.mat = TRUE,
  asy.cov = FALSE,
  ...
)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`cc.lab`	Case-control labels, 1 for case, 0 for control
`klist`	`list` of integers specifying the indices of the T_k values used in obtaining the T_{comb}.
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`.
`nonzero.mat`	A logical argument (default is `TRUE`) to determine whether the A matrix or the matrix of nonzero locations of the A matrix will be used in the computation of N_s and N_t. If `TRUE` the nonzero location matrix is used, otherwise the A matrix itself is used.
`asy.cov`	A logical argument (default is `FALSE`) to determine whether asymptotic or exact (i.e., finite sample) covariances between T_k and T_l values are to be used to obtain the entries of the covariance matrix. If `TRUE` the asymptotic covariance values are used, otherwise exact covariance values are used.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

Returns the value of the T_{comb} test statistic

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`ceTk`, `EV.Tcomb`, and `ZTcomb`

Examples

```
n<-20  #or try sample(1:20,1) #try also n<-50, 100
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
n1<-sum(cls==1)

kl<-sample(1:5,3) #try also sample(1:5,2)
kl
Tcomb(Y,cls,kl)
Tcomb(Y,cls,kl,method="max")
Tcomb(Y,cls+1,kl,case.lab=2)
Tcomb(Y,cls,kl,nonzero.mat = FALSE)
Tcomb(Y,cls,kl,asy.cov = TRUE)
```

```
#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
Tcomb(Y,fcls,kl,case.lab="a")
```

`tct` *T Contingency Table (TCT)*

Description

Returns the `T` contingency table (TCT), which is a matrix of same dimension as, `ct`, whose entries are the values of the Types I-IV cell-specific test statistics, $T^I_{ij} - T^{IV}_{ij}$. The row and column names are inherited from `ct`. The type argument specifies the type of the cell-specific test among the types I-IV tests.

See also (Ceyhan (2017)) and the references therein.

Usage

```
tct(ct, type = "III")
```

Arguments

`ct`	A nearest neighbor contingency table
`type`	The type of the cell-specific test, default=`"III"`. Takes on values `"I"`-`"IV"` (or equivalently `1-4`, respectively.

Value

A `matrix` of the values of Type I-IV cell-specific tests

Author(s)

Elvan Ceyhan

References

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`cellsTij` and `nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

type.lab<-c("I","II","III","IV")
for (i in 1:4)
{ print(paste("T_ij values for cell specific tests for type",type.lab[i]))
  print(tct(ct,i))
}

tct(ct,"II")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)
tct(ct,2)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
tct(ct,2)

ct<-matrix(c(0,10,5,5),ncol=2)
tct(ct,2)
```

`tocher.cor` *Tocher's randomized correction to the exact p-value*

Description

Tocher's modification is used for the Fisher's exact test on the contingency tables making it less conservative, by including the probability for the current table based on a randomized test (Tocher (1950)). It is applied When table-inclusive version of the p-value, $p^{>}_{inc}$, is larger, but table-exclusive version, $p^{>}_{exc}$, is less than the level of the test α, a random number, U, is generated from uniform distribution in $(0, 1)$, and if $U \leq (\alpha - p^{>}_{exc})/p_t$, $p^{>}_{exc}$ is used, otherwise p_{inc} is used as the p-value.

Table-inclusive and exclusive p-values are defined as follows. Let the probability of the contingency table itself be $p_t = f(n_{11}|n_1, n_2, c_1; \theta)$ where θ is the odds ratio under the null hypothesis (e.g. $\theta = 1$ under independence) and f is the probability mass function of the hypergeometric distribution. In testing the one-sided alternative $H_o : \theta = 1$ versus $H_a : \theta > 1$, let $p = \sum_S f(t|n_1, n_2, c_1; \theta = 1)$, then with $S = \{t : t \geq n_{11}\}$, we get the *table-inclusive version* which is denoted as $p^{>}_{inc}$ and with $S = \{t : t > n_{11}\}$, we get the *table-exclusive version*, denoted as $p^{>}_{exc}$.

See (Ceyhan (2010)) for more details.

Usage

```
tocher.cor(ptable, pval)
```

Arguments

`ptable`	Probability of the contingency table under the null hypothesis using the hypergeometric distribution for Fisher's exact test.
`pval`	Table inclusive p-value for Fisher's exact test on the contingency table.

Value

A modified p-value based on the Tocher's randomized correction.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "Exact Inference for Testing Spatial Patterns by Nearest Neighbor Contingency Tables." *Journal of Probability and Statistical Science*, **8(1)**, 45-68.

Tocher KD (1950). "Extension of the Neyman-Pearson theory of tests to discontinuous variates." *Biometrika*, **37**, 130-144.

See Also

`prob.nnct`, `exact.pval1s`, and `exact.pval2s`

Examples

```
ptab<-.03
pval<-.06
tocher.cor(ptab,pval)
```

`Tval` *T value in NN structure*

Description

Returns the T value, which is the number of triplets (z_i, z_j, z_k) with "$NN(z_i) = NN(z_j) = z_k$ and $NN(z_k) = z_j$" where $NN(\cdot)$ is the nearest neighbor function. Note that in the NN digraph, $T + R$ is the sum of the indegrees of the points in the reflexive pairs.

This quantity (together with Q and R) is used in computing the variances and covariances of the entries of the reflexivity contingency table. See (Ceyhan and Bahadir (2017)) for further details.

Usage

```
Tval(W, R)
```

Arguments

`W`	The incidence matrix, W, for the NN digraph
`R`	The number of reflexive NNs (i.e., twice the number of reflexive NN pairs)

Value

Returns the T value. See the description above for the details of this quantity.

Author(s)

Elvan Ceyhan

See Also

`Qval`, `Qvec`, `sharedNN`, and `Rval`

Examples

```
#3D data points
n<-10
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd)
R<-Rval(W)
Tval(W,R)

#1D data points
X<-as.matrix(runif(15)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(5) would not work
ipd<-ipd.mat(X)
W<-Wmat(ipd)
R<-Rval(W)
```

```
Tval(W,R)

#with ties=TRUE in the data
Y<-matrix(round(runif(30)*10),ncol=3)
ipd<-ipd.mat(Y)
W<-Wmat(ipd,ties=TRUE)
R<-Rval(W)
Tval(W,R)
```

`var.nnct` *Variances of Cell Counts in an NNCT*

Description

Returns the variances of cell counts N_{ij} for $i, j = 1, \ldots, k$ in the NNCT, `ct` in matrix form which is of the same dimension as `ct`. These variances are valid under RL or conditional on Q and R under CSR.

See also (Dixon (1994, 2002); Ceyhan (2010, 2017)).

Usage

```
var.nnct(ct, Q, R)
```

Arguments

`ct`	A nearest neighbor contingency table
`Q`	The number of shared NNs
`R`	The number of reflexive NNs (i.e., twice the number of reflexive NN pairs)

Value

A `matrix` of same dimension as, `ct`, whose entries are the variances of the cell counts in the NNCT with class sizes given as the row sums of `ct`. The row and column names are inherited from `ct`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). "On the use of nearest neighbor contingency tables for testing spatial segregation." *Environmental and Ecological Statistics*, **17(3)**, 247-282.

Ceyhan E (2017). "Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables." *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

Dixon PM (1994). “Testing spatial segregation using a nearest-neighbor contingency table.” *Ecology*, **75(7)**, 1940-1948.

Dixon PM (2002). “Nearest-neighbor contingency table analysis of spatial segregation for several species.” *Ecoscience*, **9(2)**, 142-151.

See Also

`var.tct`, `var.nnsym`, and `cov.nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)
ct

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
var.nnct(ct,Qv,Rv)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)
var.nnct(ct,Qv,Rv)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
var.nnct(ct,Qv,Rv)
```

`var.nnsym` *Variances of Differences of Off-Diagonal Entries in an NNCT*

Description

Returns the variances of differences of off-diagonal cell counts $N_{ij} - N_{ji}$ for $i, j = 1, \ldots, k$ and $i \neq j$ in the NNCT, `ct` in a vector of length $k(k-1)/2$, the order of i, j for $N_{ij} - N_{ji}$ is as in the output of `ind.nnsym(k)`. These variances are valid under RL or conditional on Q and R under CSR.

See also (Dixon (1994); Ceyhan (2014)).

Usage

```
var.nnsym(covN)
```

Arguments

`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT

Value

A `vector` of length $k(k-1)/2$, whose entries are the variances of differences of off-diagonal cell counts $N_{ij} - N_{ji}$ for $i, j = 1, \ldots, k$ and $i \neq j$ in the NNCT.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Dixon PM (1994). "Testing spatial segregation using a nearest-neighbor contingency table." *Ecology*, **75(7)**, 1940-1948.

See Also

`var.nnct`, `var.tct` and `cov.nnct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv) #default is byrow
```

```
var.nnsym(covN)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

var.nnsym(covN)
```

`var.seg.coeff` *Variances of Segregation Coefficients in a Multi-class Case*

Description

Returns the variances of segregation coefficients in a multi-class case based on the NNCT, `ct` in a `vector` of length $k(k+1)/2$, the order of the variances are as in the order of rows output of `ind.seg.coeff(k)`. These variances are valid under RL or conditional on Q and R under CSR.

See also (Ceyhan (2014)).

The argument covN is the covariance matrix of N_{ij} (concatenated rowwise).

Usage

```
var.seg.coeff(ct, covN)
```

Arguments

`ct`	A nearest neighbor contingency table
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT

Value

A `vector` of length $k(k+1)/2$, whose entries are the variances of segregation coefficients for the entry i,j in the NNCT, where the order of the variances are as in the order of rows output of `ind.seg.coeff(k)`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`seg.coeff`, `cov.seg.coeff`, `var.nnsym` and `var.nnct` and

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

var.seg.coeff(ct,covN)
varPseg.coeff(ct,covN)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

var.seg.coeff(ct,covN)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ipd<-ipd.mat(Y)
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

var.seg.coeff(ct,covN)
```

`var.tct` *Variances of Entries in a TCT*

Description

Returns the variances of T_{ij} values for $i, j = 1, \ldots, k$ in the TCT in matrix form which is of the same dimension as TCT for types I-IV tests. The argument `covN` must be the covariance between N_{ij} values which are obtained from the NNCT by row-wise vectorization. type determines the type of the test for which variances are to be computed, with default=`"III"`. These variances are valid under RL or conditional on Q and R under CSR.

See also (Ceyhan (2010, 2017)).

Usage

```
var.tct(ct, covN, type = "III")
```

Arguments

`ct`	A nearest neighbor contingency table
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized cell counts of NNCT, `ct`.
`type`	The type of the cell-specific test, default=`"III"`. Takes on values `"I"`-`"IV"` (or equivalently `1-4`, respectively.

Value

A `matrix` of same dimension as, `ct`, whose entries are the variances of the entries in the TCT for the corresponding type of cell-specific test. The row and column names are inherited from `ct`.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2010). “New Tests of Spatial Segregation Based on Nearest Neighbor Contingency Tables.” *Scandinavian Journal of Statistics*, **37(1)**, 147-165.

Ceyhan E (2017). “Cell-Specific and Post-hoc Spatial Clustering Tests Based on Nearest Neighbor Contingency Tables.” *Journal of the Korean Statistical Society*, **46(2)**, 219-245.

See Also

`var.nnct`, `var.tctI`, `var.tctIII`, `var.tctIV` , and `cov.tct`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

var.tct(ct,covN,"I")
var.tct(ct,covN,2)
var.tct(ct,covN,"III")
var.tct(ct,covN,"IV")

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)

covN<-cov.nnct(ct,varN,Qv,Rv)

var.tct(ct,covN,"I")
var.tct(ct,covN,2)
```

`varPseg.coeff` *Variance of Pielou's Segregation Coefficient for 2 Classes*

Description

Returns the variance of Pielou's coefficient of segregation for the two-class case (i.e., based on 2×2 NNCTs)in a 2×2 NNCT. This variance is valid under RL or conditional on Q and R under CSR.

See also (Ceyhan (2014)) for more detail.

Usage

```
varPseg.coeff(ct, covN)
```

Arguments

`ct`	A nearest neighbor contingency table
`covN`	The $k^2 \times k^2$ covariance matrix of row-wise vectorized entries of NNCT

Value

The variance of Pielou's coefficient of segregation for the two-class case.

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

See Also

`Pseg.coeff`, `seg.coeff` and `var.seg.coeff`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))
ct<-nnct(ipd,cls)

W<-Wmat(ipd)
Qv<-Qvec(W)$q
Rv<-Rval(W)
varN<-var.nnct(ct,Qv,Rv)
covN<-cov.nnct(ct,varN,Qv,Rv)

varPseg.coeff(ct,covN)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ct<-nnct(ipd,fcls)

varPseg.coeff(ct,covN)

#############
ct<-matrix(sample(1:25,9),ncol=3)
#varPseg.coeff(ct,covN)
```

`varTkinv.sim` *Simulated Variance of Cuzick and Edwards T_k^inv Test statistic*

Description

This function estimates the variance of Cuzick and Edwards T_k^{inv} test statistic by Monte Carlo simulations under the RL hypothesis.

The exact variance of T_k^{inv} is currently not available and (Cuzick and Edwards (1990)) say that "The permutational variance of T_k^{inv} becomes unwieldy for $k > 1$ and is more easily simulated", hence we estimate the variance of T_k^{inv} by RL of cases and controls to the given point data.

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0's and 1's accordingly. The argument `Nsim` represents the number of resamplings (without replacement) in the RL scheme, with default being `1000`.

See (Cuzick and Edwards (1990)).

See the function `ceTkinv` for the details of the T_k^{inv} test.

Usage

```
varTkinv.sim(dat, k, cc.lab, Nsim = 1000, case.lab = NULL)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point,
`k`	Integer specifying the number of the closest controls to subject i.
`cc.lab`	Case-control labels, 1 for case, 0 for control
`Nsim`	The number of simulations, i.e., the number of resamplings under the RL scheme to estimate the variance of T_k^{inv}
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`.

Value

The simulation estimated variance of Cuzick and Edwards T_k^{inv} test statistic for disease clustering

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`ceTkinv` and `EV.Tkinv`

Examples

```
set.seed(123)
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)
n1<-sum(cls==1)
k<-2

Nmc<-1000
varTkinv.sim(Y,k,cls,Nsim=Nmc)

set.seed(1)
varTrun.sim(Y,cls,Nsim=Nmc)
set.seed(1)
varTkinv.sim(Y,k=1,cls,Nsim=Nmc)

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
varTkinv.sim(Y,k,fcls,Nsim=Nmc,case.lab="a")
```

`Wmat` *The incidence matrix* `W` *for the NN digraph*

Description

Returns the $W = (w_i j)$ matrix which is used to compute Q, R and T values in the NN structure. $w_{ij} = I($ point j is a NN of point $i))$ i.e., $w_{ij} = 1$ if point j is a NN of point i and 0 otherwise.

The argument `ties` is a logical argument (default=`FALSE`) to take ties into account or not. If `TRUE` the function takes ties into account by making $w_{ij} = 1/m$ if point j is a NN of point i and there are m tied NNs and 0 otherwise. If `FALSE`, $w_{ij} = 1$ if point j is a NN of point i and 0 otherwise. The matrix W is equivalent to $A = (a_{ij})$ matrix with $k = 1$, i.e., `Wmat(X)=aij.mat(X,k=1)`.

The argument `is.ipd` is a logical argument (default=`TRUE`) to determine the structure of the argument `x`. If `TRUE`, `x` is taken to be the inter-point distance (IPD) matrix, and if `FALSE`, `x` is taken to be the data set with rows representing the data points.

Usage

```
Wmat(x, ties = FALSE, is.ipd = TRUE, ...)
```

Arguments

`x`	The IPD matrix (if `is.ipd=TRUE`) or a data set of points in matrix or data frame form where points correspond to the rows (if `is.ipd = FALSE`).
`ties`	A logical parameter (default=`FALSE`) to take ties into account in computing the W matrix, so, if it is `TRUE`, $w_{ij} = 1/m$ if point j is a NN of point i and there are m tied NNs and 0 otherwise and if `FALSE`, $w_{ij} = 1$ if point j is a NN of point i and 0 otherwise.
`is.ipd`	A logical parameter (default=`TRUE`). If `TRUE`, `x` is taken as the inter-point distance matrix, otherwise, `x` is taken as the data set with rows representing the data points.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

The incidence matrix $W = (w_ij)$ where $w_{ij} = I($ point j is a NN of point $i))$, i.e., $w_{ij} = 1$ if point j is a NN of point i and 0 otherwise.

Author(s)

Elvan Ceyhan

See Also

`aij.mat`, `aij.nonzero`, and `aij.theta`

Examples

```
n<-3
X<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(X)
Wmat(ipd)
Wmat(X,is.ipd = FALSE)

n<-5
Y<-matrix(runif(3*n),ncol=3)
ipd<-ipd.mat(Y)
Wmat(ipd)
Wmat(Y,is.ipd = FALSE)
Wmat(Y,is.ipd = FALSE,method="max")

Wmat(Y,is.ipd = FALSE)
aij.mat(Y,k=1)

#1D data points
X<-as.matrix(runif(5)) # need to be entered as a matrix with one column
#(i.e., a column vector), hence X<-runif(5) would not work
ipd<-ipd.mat(X)
Wmat(ipd)
Wmat(X,is.ipd = FALSE)
```

```
#with ties=TRUE in the data
Y<-matrix(round(runif(15)*10),ncol=3)
ipd<-ipd.mat(Y)
Wmat(ipd,ties=TRUE)
Wmat(Y,ties=TRUE,is.ipd = FALSE)
```

`Xsq.ceTk` *Chi-square Approximation to Cuzick and Edwards T_k Test statistic*

Description

An object of class `"Chisqtest"` performing a chi-square approximation for Cuzick and Edwards T_k test statistic based on the number of cases within kNNs of the cases in the data.

This approximation is suggested by Tango (2007) since T_k statistic had high skewness rendering the normal approximation less efficient. The chi-square approximation is as follows: $\frac{T_k - ET_k}{\sqrt{VarT_k}} \approx \frac{\chi^2_\nu - \nu}{\sqrt{2\nu}}$ where χ^2_ν is a chi-square random variable with ν df, and $\nu = 8/skewnees(T_k)$ (see `SkewTk` for the skewness).

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0's and 1's accordingly.

The logical argument `nonzero.mat` (default=`FALSE`) is for using the A matrix if `FALSE` or just the matrix of nonzero locations in the A matrix (if `TRUE`).

The logical argument `asy.var` (default=`FALSE`) is for using the asymptotic variance or the exact (i.e., finite sample) variance for the variance of T_k in its standardization. If `asy.var=TRUE`, the asymptotic variance is used for $Var[T_k]$ (see `asyvarTk`), otherwise the exact variance (see `varTk`) is used.

See also (Tango (2007)) and the references therein.

Usage

```
Xsq.ceTk(
  dat,
  cc.lab,
  k,
  case.lab = NULL,
  nonzero.mat = TRUE,
  asy.var = FALSE,
  ...
)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`cc.lab`	Case-control labels, 1 for case, 0 for control

`k`	Integer specifying the number of NNs (of subject i).
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`.
`nonzero.mat`	A logical argument (default is `TRUE`) to determine whether the A matrix or the matrix of nonzero locations of the A matrix will be used in the computations. If `TRUE` the nonzero location matrix is used, otherwise the A matrix itself is used.
`asy.var`	A logical argument (default is `FALSE`) to determine whether the asymptotic variance or the exact (i.e., finite sample) variance for the variance of T_k in its standardization. If `TRUE`, the asymptotic variance is used for $Var[T_k]$, otherwise the exact variance is used.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

A `list` with the elements

`statistic`	The chi-squared test statistic for Tango's chi-square approximation to Cuzick & Edwards' T_k test for disease clustering.
`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is $8/$skewness where skewness is the output of `SkewTk` function.
`estimate`	Estimates, i.e., the observed T_k value.
`est.name`, `est.name2`	Names of the estimates, they are almost identical for this function.
`null.value`	Hypothesized null value for Cuzick & Edwards' T_k, which is ET_k.
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `dat`

Author(s)

Elvan Ceyhan

References

Tango T (2007). "A class of multiplicity adjusted tests for spatial clustering based on case-control point data." *Biometrics*, **63**, 119-127.

See Also

`ceTk`, `ZceTk` and `SkewTk`

Examples

```
set.seed(123)
n<-20
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)

k<-sample(1:5,1) # try also 1, 3, 5,
k

Xsq.ceTk(Y,cls,k)
Xsq.ceTk(Y,cls,k,nonzero.mat=FALSE)
Xsq.ceTk(Y,cls+1,k,case.lab = 2)
Xsq.ceTk(Y,cls,k,method="max")

Xsq.ceTk(Y,cls,k,asy.var=TRUE)
```

`Xsq.nnsym` *Overall NN Symmetry Test with Chi-square Approximation*

Description

An object of class `"Chisqtest"` performing the hypothesis test of equality of the expected values of the off-diagonal cell counts (i.e., entries) under RL or CSR in the NNCT for $k \geq 2$ classes. That is, the test performs Dixon's or Pielou's (first type of) overall NN symmetry test which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data or for sparsely sample data, respectively. (See Pielou (1961); Dixon (1994); Ceyhan (2014) for more detail).

The `type="dixon"` refers to Dixon's overall NN symmetry test and `type="pielou"` refers to Pielou's first type of overall NN symmetry test. The symmetry test is based on the chi-squared approximation of the corresponding quadratic form and `type="dixon"` yields an extension of Dixon's NN symmetry test, which is extended by Ceyhan (2014) and `type="pielou"` yields Pielou's overall NN symmetry test.

The function yields the test statistic, p-value and `df` which is $k(k-1)/2$, description of the alternative with the corresponding null values (i.e., expected values) of differences of the off-diagonal entries,(which is 0 for this function) and also the sample estimates (i.e., observed values) of absolute differences of the off-diagonal entries of NNCT (in the upper-triangular form). The functions also provide names of the test statistics, the description of the test and the data set used.

The null hypothesis is that all $E(N_{ij}) = E(N_{ji})$ for $i \neq j$ in the $k \times k$ NNCT (i.e., symmetry in the mixed NN structure) for $k \geq 2$. In the output, if if `type="pielou"`, the test statistic, p-value and the df are valid only for (properly) sparsely sampled data.

See also (Pielou (1961); Dixon (1994); Ceyhan (2014)) and the references therein.

Usage

```
Xsq.nnsym(dat, lab, type = "dixon", ...)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`lab`	The `vector` of class labels (numerical or categorical)
`type`	The type of the overall NN symmetry test with default=`"dixon"`. Takes on values `"dixon"` and `"pielou"` for Dixon's and Pielou's (first type) overall NN symmetry test
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function

Value

A `list` with the elements

`statistic`	The chi-squared test statistic for Dixon's or Pielou's (first type of) overall NN symmetry test
`stat.names`	Name of the test statistic
`p.value`	The p-value for the hypothesis test
`df`	Degrees of freedom for the chi-squared test, which is $k(k-1)/2$ for this function.
`estimate`	Estimates, i.e., absolute differences of the off-diagonal entries of NNCT (in the upper-triangular form).
`est.name`, `est.name2`	Names of the estimates, former is a shorter description of the estimates than the latter.
`null.value`	Hypothesized null values for the differences between the expected values of the off-diagonal entries, which is 0 for this function.
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `dat`, or name of the contingency table, `ct`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Dixon PM (1994). "Testing spatial segregation using a nearest-neighbor contingency table." *Ecology*, **75(7)**, 1940-1948.

Pielou EC (1961). "Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships." *Journal of Ecology*, **49(2)**, 255-269.

See Also

`Znnsym.ss`, `Znnsym.dx` and `Znnsym2cl`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

Xsq.nnsym(Y,cls)
Xsq.nnsym(Y,cls,method="max")
Xsq.nnsym(Y,cls,type="pielou")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))

Xsq.nnsym(Y,fcls)
Xsq.nnsym(Y,fcls,type="pielou")

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

Xsq.nnsym(Y,cls)
Xsq.nnsym(Y,cls,type="pielou")
```

`ZceTk` *Z-test for Cuzick and Edwards T_k statistic*

Description

An object of class `"htest"` performing a z-test for Cuzick and Edwards T_k test statistic based on the number of cases within kNNs of the cases in the data.

For disease clustering, Cuzick and Edwards (1990) suggested a k-NN test T_k based on number of cases among k NNs of the case points. Under RL of n_1 cases and n_0 controls to the given locations in the study region, T_k approximately has $N(E[T_k], Var[T_k]/n_1)$ distribution for large n_1.

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0's and 1's accordingly. Also, T_1 is identical to the count for cell $(1, 1)$ in the nearest neighbor contingency table (NNCT) (See the function `nnct` for more detail on NNCTs). Thus, the z-test for T_k is same as the cell-specific z-test for cell $(1, 1)$ in the NNCT (see `cell.spec`).

The logical argument `nonzero.mat` (default=`TRUE`) is for using the A matrix if `FALSE` or just the matrix of nonzero locations in the A matrix (if `TRUE`) in the computations.

The logical argument `asy.var` (default=`FALSE`) is for using the asymptotic variance or the exact (i.e., finite sample) variance for the variance of T_k in its standardization. If `asy.var=TRUE`, the asymptotic variance is used for $Var[T_k]$ (see `asyvarTk`), otherwise the exact variance (see `varTk`) is used.

See also (Ceyhan (2014); Cuzick and Edwards (1990)) and the references therein.

Usage

```
ZceTk(
  dat,
  cc.lab,
  k,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  case.lab = NULL,
  nonzero.mat = TRUE,
  asy.var = FALSE,
  ...
)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`cc.lab`	Case-control labels, 1 for case, 0 for control
`k`	Integer specifying the number of NNs (of subject i).
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for Cuzick and Edwards T_k statistic
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`
`nonzero.mat`	A logical argument (default is `TRUE`) to determine whether the A matrix or the matrix of nonzero locations of the A matrix will be used in the computation of N_s and N_t (argument is passed on to `asyvarTk`). If `TRUE` the nonzero location matrix is used, otherwise the A matrix itself is used.
`asy.var`	A logical argument (default is `FALSE`) to determine whether the asymptotic variance or the exact (i.e., finite sample) variance for the variance of T_k in its standardization. If `TRUE`, the asymptotic variance is used for $Var[T_k]$, otherwise the exact variance is used.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

A `list` with the elements

`statistic`	The Z test statistic for the Cuzick and Edwards T_k test
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the Cuzick and Edwards T_k value at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., the Cuzick and Edwards T_k value.
`null.value`	Hypothesized null value for the Cuzick and Edwards T_k value which is $kn_1(n_1-1)/(n-1)$ for this function.

`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `dat`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Segregation indices for disease clustering." *Statistics in Medicine*, **33(10)**, 1662-1684.

Cuzick J, Edwards R (1990). "Spatial clustering for inhomogeneous populations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`ceTk`, `cell.spec`, and `Xsq.ceTk`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))
k<-1 #try also 2,3, sample(1:5,1)

ZceTk(Y,cls,k)
ZceTk(Y,cls,k,nonzero.mat=FALSE)
ZceTk(Y,cls,k,method="max")

ZceTk(Y,cls+1,k,case.lab = 2,alt="l")
ZceTk(Y,cls,k,asy.var=TRUE,alt="g")
```

`Znnsym` *NN Symmetry Test with Normal Approximation*

Description

An object of class `"cellhtest"` performing hypothesis test of equality of the expected values of the off-diagonal cell counts (i.e., entries) for each pair i, j of classes under RL or CSR in the NNCT for $k \geq 2$ classes. That is, the test performs Dixon's or Pielou's (first type of) NN symmetry test which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data or for sparsely sample data, respectively. (See Pielou (1961); Dixon (1994); Ceyhan (2014) for more detail).

The `type="dixon"` refers to Dixon's NN symmetry test and `type="pielou"` refers to Pielou's first type of NN symmetry test. The symmetry test is based on the normal approximation of the difference of the off-diagonal entries in the NNCT and are due to Pielou (1961); Dixon (1994).

The function yields a contingency table of the test statistics, p-values for the corresponding alternative, expected values (i.e., null value(s)), lower and upper confidence levels and sample estimate for the $N_{ij} - N_{ji}$ values for $i \neq j$ (all in the upper-triangular form except for the null value, which is 0 for all pairs) and also names of the test statistics, estimates, null values, the description of the test, and the data set used.

The null hypothesis is that all $E(N_{ij}) = E(N_{ji})$ for $i \neq j$ in the $k \times k$ NNCT (i.e., symmetry in the mixed NN structure) for $k \geq 2$. In the output, if if `type="pielou"`, the test statistic, p-value and the lower and upper confidence limits are valid only for (properly) sparsely sampled data.

See also (Pielou (1961); Dixon (1994); Ceyhan (2014)) and the references therein.

Usage

```
Znnsym(
  dat,
  lab,
  type = "dixon",
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  ...
)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`lab`	The `vector` of class labels (numerical or categorical)
`type`	The type of the NN symmetry test with default=`"dixon"`. Takes on values `"dixon"` and `"pielou"` for Dixon's and Pielou's (first type) NN symmetry test
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the difference of the off-diagonal entries, $N_{12} - N_{21}$
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function

Value

A `list` with the elements

`statistic`	The `matrix` of Z test statistics for the NN symmetry test (in the upper-triangular form)
`stat.names`	Name of the test statistics
`p.value`	The `matrix` of p-values for the hypothesis test for the corresponding alternative (in the upper-triangular form)

`LCL`, `UCL`	Matrix of Lower and Upper Confidence Levels (in the upper-triangular form) for the $N_{ij} - N_{ji}$ values for $i \neq j$ at the given confidence level `conf.level` and depends on the type of `alternative`.
`conf.int`	The confidence interval for the estimates, it is `NULL` here, since we provide the `UCL` and `LCL` in `matrix` form.
`cnf.lvl`	Level of the upper and lower confidence limits (i.e., conf.level) of the differences of the off-diagonal entries.
`estimate`	Estimates of the parameters, i.e., matrix of the difference of the off-diagonal entries (in the upper-triangular form) of the $k \times k$ NNCT, $N_{ij} - N_{ji}$ for $i \neq j$.
`est.name`, `est.name2`	Names of the estimates, former is a shorter description of the estimates than the latter.
`null.value`	Hypothesized null value for the expected difference between the off-diagonal entries, $E(N_{ij}) - E(N_{ji})$ for $i \neq j$ in the $k \times k$ NNCT, which is 0 for this function.
`null.name`	Name of the null values
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `dat`, or name of the contingency table, `ct`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations.” *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Dixon PM (1994). “Testing spatial segregation using a nearest-neighbor contingency table.” *Ecology*, **75(7)**, 1940-1948.

Pielou EC (1961). “Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships.” *Journal of Ecology*, **49(2)**, 255-269.

See Also

`Znnsym.ss.ct`, `Znnsym.ss`, `Znnsym.dx.ct`, `Znnsym.dx` and `Znnsym2cl`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

Znnsym(Y,cls)
```

```
Znnsym(Y,cls,method="max")
Znnsym(Y,cls,type="pielou")
Znnsym(Y,cls,type="pielou",method="max")

Znnsym(Y,cls,alt="g")
Znnsym(Y,cls,type="pielou",alt="g")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
Znnsym(Y,fcls)

#############
n<-40
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:4,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

Znnsym(Y,cls)
Znnsym(Y,cls,type="pielou")
```

`Znnsym2cl` *NN Symmetry Test with Normal Approximation for Two Classes*

Description

An object of class `"htest"` performing hypothesis test of equality of the expected value of the off-diagonal cell counts (i.e., entries) under RL or CSR in the NNCT for $k = 2$ classes. That is, the test performs Dixon's or Pielou's (first type of) NN symmetry test which is appropriate (i.e., have the appropriate asymptotic sampling distribution) for completely mapped data and for sparsely sample data, respectively. (See Ceyhan (2014) for more detail).

The symmetry test is based on the normal approximation of the difference of the off-diagonal entries in the NNCT and are due to Pielou (1961); Dixon (1994).

The `type="dixon"` refers to Dixon's NN symmetry test and `type="pielou"` refers to Pielou's first type of NN symmetry test.

The function yields the test statistic, p-value for the corresponding alternative, the confidence interval, estimate and null value for the parameter of interest (which is the difference of the off-diagonal entries in the NNCT), and method and name of the data set used.

The null hypothesis is that all $E(N_{12}) = E(N_{21})$ in the 2×2 NNCT (i.e., symmetry in the mixed NN structure).

See also (Pielou (1961); Dixon (1994); Ceyhan (2014)) and the references therein.

Usage

```
Znnsym2cl(
  dat,
  lab,
```

```
  type = "dixon",
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`lab`	The `vector` of class labels (numerical or categorical)
`type`	The type of the NN symmetry test with default=`"dixon"`. Takes on values `"dixon"` and `"pielou"` for Dixon's and Pielou's (first type) NN symmetry test
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for the difference of the off-diagonal entries, $N_{12} - N_{21}$

Value

A `list` with the elements

`statistic`	The Z test statistic for Pielou's first type of NN symmetry test
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the difference of the off-diagonal entries, $N_{12} - N_{21}$ in the 2×2 NNCT at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate, i.e., the difference of the off-diagonal entries of the 2×2 NNCT, $N_{12} - N_{21}$.
`null.value`	Hypothesized null value for the expected difference between the off-diagonal entries, $E(N_{12}) - E(N_{21})$ in the 2×2 NNCT, which is 0 for this function.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `dat`, or name of the contingency table, `ct`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). "Testing Spatial Symmetry Using Contingency Tables Based on Nearest Neighbor Relations." *The Scientific World Journal*, **Volume 2014**, Article ID 698296.

Dixon PM (1994). "Testing spatial segregation using a nearest-neighbor contingency table." *Ecology*, **75(7)**, 1940-1948.

Pielou EC (1961). "Segregation and symmetry in two-species populations as studied by nearest-neighbor relationships." *Journal of Ecology*, **49(2)**, 255-269.

See Also

`Znnsym2cl.ss.ct`, `Znnsym2cl.ss`, `Znnsym2cl.dx.ct`, `Znnsym2cl.dx`, `Znnsym.ss.ct`, `Znnsym.ss`, `Znnsym.dx.ct`, `Znnsym.dx`, `Znnsym.dx.ct`, `Znnsym.dx` and `Znnsym`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(1:2,n,replace = TRUE)  #or try cls<-rep(1:2,c(10,10))

Znnsym2cl(Y,cls)
Znnsym2cl(Y,cls,type="pielou")

Znnsym2cl(Y,cls,alt="g")
Znnsym2cl(Y,cls,type="pielou",alt="g")
```

`ZTcomb` *Z-test for Cuzick and Edwards T_comb statistic*

Description

An object of class `"htest"` performing a z-test for Cuzick and Edwards T_{comb} test statisticin disease clustering, where T_{comb} is a linear combination of some T_k tests.

For disease clustering, Cuzick and Edwards (1990) developed a k-NN test T_k based on number of cases among k NNs of the case points, and also proposed a test combining various T_k tests, denoted as T_{comb}.

See page 87 of (Cuzick and Edwards (1990)) for more details.

Under RL of n_1 cases and n_0 controls to the given locations in the study region, T_{comb} approximately has $N(E[T_{comb}], Var[T_{comb}])$ distribution for large n_1.

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0's and 1's accordingly.

The argument `klist` is the `vector` of integers specifying the indices of the T_k values used in obtaining the T_{comb}.

The logical argument `nonzero.mat` (default=`TRUE`) is for using the A matrix if `FALSE` or just the matrix of nonzero locations in the A matrix (if `TRUE`) in the computations.

The logical argument `asy.cov` (default=`FALSE`) is for using the asymptotic covariance or the exact (i.e., finite sample) covariance for the vector of T_k values used in `Tcomb` in the standardization of T_{comb}. If `asy.cov=TRUE`, the asymptotic covariance is used, otherwise the exact covariance is used.

See also (Ceyhan (2014); Cuzick and Edwards (1990)) and the references therein.

Usage

```
ZTcomb(
  dat,
  cc.lab,
  klist,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  case.lab = NULL,
  nonzero.mat = TRUE,
  asy.cov = FALSE,
  ...
)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`cc.lab`	Case-control labels, 1 for case, 0 for control
`klist`	`list` of integers specifying the indices of the T_k values used in obtaining the T_{comb}.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for Cuzick and Edwards T_{comb} statistic
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`.
`nonzero.mat`	A logical argument (default is `TRUE`) to determine whether the A matrix or the matrix of nonzero locations of the A matrix will be used in the computation of covariance of T_k values forming the `T_{comb}` statistic (argument is passed on to `covTcomb`). If `TRUE` the nonzero location matrix is used, otherwise the A matrix itself is used.
`asy.cov`	A logical argument (default is `FALSE`) to determine whether asymptotic or exact (i.e., finite sample) covariances between T_k and T_l values are to be used to obtain the entries of the covariance matrix.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

A `list` with the elements

`statistic`	The Z test statistic for the Cuzick and Edwards T_{comb} test
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the Cuzick and Edwards T_{comb} value at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., the Cuzick and Edwards T_{comb} value.

`null.value`	Hypothesized null value for the Cuzick and Edwards T_{comb} value which is $E[T_{comb}]$ for this function, which is the output of `EV.Tcomb` function.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `dat`

Author(s)

Elvan Ceyhan

References

Ceyhan E (2014). “Segregation indices for disease clustering.” *Statistics in Medicine*, **33(10)**, 1662-1684.

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`Tcomb`, `EV.Tcomb`, and `covTcomb`

Examples

```
n<-20  #or try sample(1:20,1)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))

kl<-sample(1:5,3) #try also sample(1:5,2)
ZTcomb(Y,cls,kl)
ZTcomb(Y,cls,kl,method="max")

ZTcomb(Y,cls,kl,nonzero.mat=FALSE)
ZTcomb(Y,cls+1,kl,case.lab = 2,alt="l")
ZTcomb(Y,cls,kl,conf=.9,alt="g")
ZTcomb(Y,cls,kl,asy=TRUE,alt="g")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ZTcomb(Y,fcls,kl,case.lab="a")
```

ZTrun *Z-test for Cuzick and Edwards T_run statistic*

Description

An object of class `"htest"` performing a z-test for Cuzick and Edwards T_{run} test statistic which is based on the number of consecutive cases from the cases in the data under RL or CSR independence.

Under RL of n_1 cases and n_0 controls to the given locations in the study region, T_{run} approximately has $N(E[T_{run}], Var[T_{run}])$ distribution for large n.

The argument `cc.lab` is case-control label, 1 for case, 0 for control, if the argument `case.lab` is `NULL`, then `cc.lab` should be provided in this fashion, if `case.lab` is provided, the labels are converted to 0's and 1's accordingly.

The logical argument var.sim (default=`FALSE`) is for using the simulation estimated variance or the exact variance for the variance of T_{run} in its standardization. If `var.sim=TRUE`, the simulation estimated variance is used for $Var[T_{run}]$ (see `varTrun.sim`), otherwise the exact variance (see `varTrun`) is used. Moreover, when `var.sim=TRUE`, the argument `Nvar.sim` represents the number of resamplings (without replacement) in the RL scheme, with default being `1000`.

The function `varTrun` might take a very long time when data size is large (even larger than 50); in this case, it is recommended to use `var.sim=TRUE` in this function.

See also (Cuzick and Edwards (1990)) and the references therein.

Usage

```
ZTrun(
  dat,
  cc.lab,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  case.lab = NULL,
  var.sim = FALSE,
  Nvar.sim = 1000,
  ...
)
```

Arguments

`dat`	The data set in one or higher dimensions, each row corresponds to a data point.
`cc.lab`	Case-control labels, 1 for case, 0 for control
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"` or `"greater"`.
`conf.level`	Level of the upper and lower confidence limits, default is `0.95`, for Cuzick and Edwards T_{run} statistic
`case.lab`	The label used for cases in the `cc.lab` (if `cc.lab` is not provided then the labels are converted such that cases are 1 and controls are 0), default is `NULL`.

`var.sim`	A logical argument (default is `FALSE`) to determine whether the simulation estimated variance or the exact variance be used for the variance of T_{run} in its standardization. If `var.sim=TRUE`, the simulation estimated variance is used for $Var[T_{run}]$ (see `varTrun.sim`), otherwise the exact variance (see `varTrun`) is used.
`Nvar.sim`	The number of simulations, i.e., the number of resamplings under the RL scheme to estimate the variance of T_{run}, used only when `var.sim=TRUE`.
`...`	are for further arguments, such as `method` and `p`, passed to the `dist` function.

Value

A `list` with the elements

`statistic`	The Z test statistic for the Cuzick and Edwards T_{run} test
`p.value`	The p-value for the hypothesis test for the corresponding alternative
`conf.int`	Confidence interval for the Cuzick and Edwards T_{run} value at the given confidence level `conf.level` and depends on the type of `alternative`.
`estimate`	Estimate of the parameter, i.e., the Cuzick and Edwards T_{run} value.
`null.value`	Hypothesized null value for the Cuzick and Edwards T_{run} value which is $n_1(n_1-1)/(n_0+1)$ for this function.
`alternative`	Type of the alternative hypothesis in the test, one of `"two.sided"`, `"less"`, `"greater"`
`method`	Description of the hypothesis test
`data.name`	Name of the data set, `dat`

Author(s)

Elvan Ceyhan

References

Cuzick J, Edwards R (1990). “Spatial clustering for inhomogeneous populations (with discussion).” *Journal of the Royal Statistical Society, Series B*, **52**, 73-104.

See Also

`ceTrun`, `ZceTk`, and `ZTcomb`

Examples

```
n<-20  #or try sample(1:20,1) #try also 40, 50, 60
set.seed(123)
Y<-matrix(runif(3*n),ncol=3)
cls<-sample(0:1,n,replace = TRUE)  #or try cls<-rep(0:1,c(10,10))

ZTrun(Y,cls)
ZTrun(Y,cls,method="max")
ZTrun(Y,cls,var.sim=TRUE)
```

```
ZTrun(Y,cls+1,case.lab = 2,alt="l") #try also ZTrun(Y,cls,conf=.9,alt="g")

#cls as a factor
na<-floor(n/2); nb<-n-na
fcls<-rep(c("a","b"),c(na,nb))
ZTrun(Y,fcls,case.lab="a")
```

Index