Package: IndTestPP (via r-universe)

October 16, 2026

Type Package

Title Tests of Independence and Analysis of Dependence Between Point Processes in Time

Version 3.0

Date 2020-08-28

Author Ana C. Cebrian

Maintainer Ana C. Cebrian <acebrian@unizar.es>

Imports parallel

Description It provides a general framework to analyse dependence between point processes in time. It includes parametric and non-parametric tests to study independence, and functions for generating and analysing different types of dependence.

License GPL

NeedsCompilation no

Depends R (>= 2.10)

Repository https://anaccebrian.r-universe.dev

Date/Publication 2020-08-28 18:00:03 UTC

RemoteUrl https://github.com/cran/IndTestPP

RemoteRef HEAD

RemoteSha a628d5be9c314513541656d6e2ea28dd9bc91cee

Contents

IndTestPP-package 2
BinPer 3
ComplPos 4
CondTest 5
CountingCor 7
CPSPpoints 8
CPSPPOTevents 9
depchi 12

DepCPSPNHK . . . 13
DepNHCPSP . . . 15
DepNHNeyScot . . . 17
DepNHPPMarked . . . 19
DepNHPPqueue . . . 20
DepqueueNHK . . . 23
DistObs . . . 25
DistShift . . . 26
DistSim . . . 28
DutilleulPlot . . . 30
IndNHNeyScot . . . 32
IndNHPP . . . 34
IntMPP . . . 35
nearestdist . . . 36
NHD . . . 37
NHF . . . 39
NHJ . . . 41
NHK . . . 45
PlotICPSP . . . 48
PlotMargP . . . 49
PlotMCPSP . . . 50
simHPc . . . 51
simNHPc . . . 52
SpecGap . . . 54
TestIndLS . . . 55
TestIndNH . . . 56
TranM . . . 59
TxBHZ . . . 60
uniongentri . . . 61

Index **63**

`IndTestPP-package` *Tests of Independence and Analysis of Dependence between Point Processes in Time*

Description

It provides a general framework to analyse dependence between point processes in time. The package includes several parametric and non-parametric tests to check the independence between two or more homogeneous and nonhomogeneous point processes and some tools to characterize the dependence between them, if it exists. In addition, it includes functions for generating and analysing dependent point processes in time with different types of dependence.

Details

Package:	IndTestPP
Type:	Package
Version:	3.0
Date:	2020-08-28
License:	GPL (>=2)

Author(s)

Ana C. Cebrian Maintainer: Ana C. Cebrian <acebrian@unizar.es>

`BinPer` *Percentage of concordant intervals*

Description

It calculates the proportion of the number of intervals with at least one point in both processes, and the sum of the number of intervals with at least one point in one process, $n_{x,y}/(n_{x,y}+n_{x,0}+n_{0,y})$.

Usage

```
BinPer(posx, posy, ll, T)
```

Arguments

`posx`	Numeric vector. Occurrence points in the first process, N_x.
`posy`	Numeric vector. Occurrence points in the second process, N_y.
`ll`	Numeric value. Lenght of the intervals where the number of points are counted.
`T`	Numeric value. Length of the observed period of the point processes.

Details

In order to be useful, an adequate length of interval ll, depending on the expected dependence, has to be selected.

Value

`binper`	Percentage of concordant intervals.

See Also

`depchi`, `CountingCor`

Examples

```
#generation of two HPP
aux1<-simNHPc(lambda=rep(0.08,200),fixed.seed=123)
aux2<-simNHPc(lambda=rep(0.12,200),fixed.seed=125)
BinPer(aux1$posNH, aux2$posNH,ll=5, T=200)
```

ComplPos *Changes format of the vector of occurrence times in a point process*

Description

It changes the format of the vector of occurrence points in a point process. It builds a vector of length T, the length of the observation period, which takes value 0 at the non occurrence times and the position value (type="Pos"), or 1 (type="Bin"), at the occurence times.

Usage

```
ComplPos(pos, T, type='Pos')
```

Arguments

pos	Numeric vector. Occurrence times.
T	Integer. Length of the observed period.
type	Character string, 'Pos' or 'Bin'. Type of the new format.

Details

This function changes the format of the occurrence points in a point process. The new format is useful when several point processes, in the same observed period, must be specfied; for example, in function NHJ or NHD, where the occurrence times of different point processes must be introduced as a matrix. Since the number of occurrences in each process can be different, in the new format, occurrences in each process are specified as a vector of length T, which takes value 0 at non occurrence times and the time position (if type='Pos') or 1 (if type='Bin') at the occurrence times.

Value

Npos	Numeric vector of lenght T containing the occurrence times in the new format.

See Also

NHD, NHF, NHK, NHJ

Examples

```
pos<-c(4,15,18,34,36,67,98)
Npos<-ComplPos(pos, T=100)
```

CondTest *Conditional test of independence between two Poisson process*

Description

It calculates a test of independence between two Poisson process, based on the analysis of the occurrences in the second process, given that there is an occurrence in the first one. Two different approaches to calculate the p-value are implemented.

It calls the auxiliary function `calcNmu`, not intended for the users.

Usage

```
CondTest(posx, posy, lambday, r, changer = TRUE, type = "All",
 plotRes = FALSE, ...)
```

Arguments

`posx`	Numeric vector. Occurrence points in the first process, N_x.
`posy`	Numeric vector. Occurrence points in the second process, N_y.
`lambday`	Numeric vector. Intensity at each time in N_y.
`r`	Numeric value. The radius of the intervals centered on the occurrence times in N_x.
`changer`	Optional. Logical flag. If it is TRUE, when the defined intervals overlap, their lengths are changed to obtain disjoint intervals. The two overlapping intervals are shortened by half of the overlapped period. In general, the resulting intervals are not centered.
`type`	Optional. Label "Poisson", "Normal" or "All". Approach to be used to calculate test p-values.
`plotRes`	Logical flag. If it is TRUE, the residual differences $(y_i - \mu_i)/\mu_i^{(1/2)}$ are plotted.
`...`	Further arguments to be passed to the function `plot`.

Details

The underlying idea of the tests is to analyze the behaviour of the second process N_y, given that a point has occured in the first one, N_x. Under independence between N_x and N_y, N_y should be a Poisson process with intensity lambday.

Intervals of length 2r centered on each point in N_x are defined. To analyze the behaviour of N_y, two approaces are implemented, both based on the idea that the number of points in each interval should be a Poisson of mean μ_i equal to the integral of lambday in the interval.

"Poisson" option: under the null, and if the intervals are independent (that is if they do not overlap) the number of points in all them should be a Poisson of mean μ, equal to the sum of all the μ_i. The p-values is calculated as $2 * min((P(Y < yo) + P(Y = yo)/2), (P(X > yo) + P(Y = yo)/2))$, where Y is a r.v. with distibution Poisson(μ) and yo is the sum of the observed number of points in

all the intervals. Since the p-values are based on a discrete distribution, they are valid but not exact p-values.

"Normal" option: under the null, the variables $(N_i - \mu_i)/(\mu_i^{1/2})$ must be zero mean and variance one variables but they are not identically distributed. Under general conditions, the mean of the variables $(N_i - \mu_i)/(\mu_i^{1/2})$ can be approximated by a Normal distribution using the Central limit theorem under the Lindeberg condition for r.v which are independent but not identically distributed. The conditions to have a valid Normal aprroximation are quite weak, even with a complex intensity, mean values of μ_i around 0.6 are valid with $n_x = 50$, and around 0.3 with $n_x = 100$.

Value

A list with elements

`pvP`	P-value obtained with the 'Poisson' approach.
`PvN`	P-value obtained with the 'Normal' approach.
`Ni`	Number of occurrences in each interval.
`mui`	Theoretical mean of the number of occurrences in each interval under the independence assumption.
`Res`	Vector of th residual differences.
`linf`	Lower bound of each interval.
`lsup`	Upper bound of each interval.
`mmu`	Mean of the mui vector. It is used to check the conditions of the approximation of the Normal test.

References

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*.

See Also

`TestIndNH`, `NHK`, `NHJ`, `DutilleulPlot`

Examples

```
#Two dependent Poisson processes from  a NHCPSP
set.seed(30)
lambdao1<-runif(3000)/20
set.seed(31)
lambdao2<-runif(3000)/10
set.seed(32)
lambda12<-runif(3000)/20
lambdaiM<-cbind(lambdao1,lambdao2,lambda12)
aux<-DepNHCPSP(lambdaiM=lambdaiM, d=2,fixed.seed=123, dplot=FALSE)

zz<-CondTest(posx=aux$posNH[[1]],posy=aux$posNH[[2]],lambday=aux$lambdaM[,2], r=2)
zz$pvP
zz$pvN
```

```
# Two independent non homogeneous  Poisson processes
lambdao1<-runif(6000)/20
set.seed(124)
lambdao2<-runif(6000)/10
aux1<-simNHPc(lambda=lambdao1, fixed.seed=123)
aux2<-simNHPc(lambda=lambdao2, fixed.seed=124)

zz<-CondTest(posx=aux1$posNH, posy=aux2$posNH, lambday= aux2$lambda, r=3)
zz$pvP
zz$pvN
```

`CountingCor` *Correlation between the counting variables in two point processes*

Description

This function calculates the correlation coefficient between the number of points in intervals of length ll, in two point processes.

Usage

```
CountingCor(posx, posy, ll, T,  method='spearman', lambdax=NULL,
 lambday=NULL)
```

Arguments

`posx`	Numeric vector. Occurrence times of the points in the first point process.
`posy`	Numeric vector. Occurrence times of the points in the second point process.
`ll`	Numeric value. Lenght of the intervals where the number of points are counted.
`T`	Numeric value. Length of the observed period of the point processes.
`method`	Character string. Correlation coefficient to be calculated. One of "pearson", "kendall", or "spearman"; see `cor.test` for definitions.
`lambdax`	Numeric vector. Intensity vector of the first point process.
`lambday`	Numeric vector. Intensity vector of the second point process.

Details

This function calculates $\rho_{xy,I_l} = Cor(X_{I_l}, Y_{I_l})$, where X_{I_l} and Y_{I_l} are the number of points in an interval I_l in processes N_x and N_y, respectively.

In order to calculate the number of points in each interval in a process, the function `CountP` is used.

Value

`ccor`	Estimated correlation.

See Also

depchi,BinPer

Examples

```
#generation of two HPP
aux1<-simNHPc(lambda=rep(0.08,200),fixed.seed=123)
aux2<-simNHPc(lambda=rep(0.12,200),fixed.seed=125)

CountingCor(aux1$posNH, aux2$posNH,ll=20, method="kendall",T=200)
```

CPSPpoints *Identifying the occurrence points of the indicator processes in a CPSP*

Description

It calculates the occurrence points in the three indicator processes of a bivariate Common Poisson Shock Process (CPSP), using as input the two marginal processes N_1 and N_2.

Usage

```
CPSPpoints(N1,N2,date=NULL, dplot=T, pmfrow=c(2,1), axispoints=NULL,...)
```

Arguments

N1	Binary vector of the first CPSP marginal process; occurrence points must be marked with 1 and the other with 0.
N2	Binary vector of the second CPSP marginal process; occurrence points must be marked with 1 and the other with 0.
date	Optional. A vector or matrix indicating the date of each observation.
dplot	Optional. A logical flag. If it is TRUE, the marginal and indicator processes are plotted.
pmfrow	Optional. A vector of the form (nr, nc) to be supplied as value of the argument mfrow in par.
axispoints	Optional. Numeric vector with the points in the time index where axis ticks and labels (from the first column in date) have to be drawn.
...	Further arguments to be passed to the function plot.

Details

A bivariate CPSP N is usually specified by its two marginal, and possibly dependent, processes N_1 and N_2, which are the observed processes. However, N can be decomposed into three independent indicator processes: $N_{(1)}$, $N_{(2)}$ and $N_{(12)}$, which are the processes of the points occurring only in the first marginal process, only in the second and in both of them (simultaneous points). The union of $N_{(1)}$ and $N_{(12)}$, and $N_{(2)}$ and $N_{(12)}$ gives respectively the two marginal processes.

The points in the marginal N_1, N_2 and indicator $N_{(1)}$, $N_{(2)}$ and $N_{(12)}$ processes can be optionally plotted. If `date` is NULL, default axis are used. Otherwise, the values in `axispoints` are used as the points in the time index where axis ticks and labels, from the first column in `date`, have to be drawn. If `axispoints` is NULL, a default grid of points is built using the function `marca`.

Value

A list with components

`Px1`	Vector of the occurrence points in $N_{(1)}$.
`Px2`	Vector of the occurrence points in $N_{(2)}$.
`Px12`	Vector of the occurrence points in $N_{(12)}$.
`N1`	Input argument.
`N2`	Input argument.
`date`	Input argument.

References

Abaurrea, J. Asin, J. and Cebrian, A.C. (2015). A Bootstrap Test of Independence Between Three Temporal Nonhomogeneous Poisson Processes and its Application to Heat Wave Modeling. *Environmental and Ecological Statistics*, 22(1), 127-144.

See Also

`CPSPPOTevents`, `PlotMCPSP`, `PlotICPSP`

Examples

```
set.seed(123)
X<-as.numeric(runif(100)<0.10)
set.seed(124)
Y<-as.numeric(runif(100)<0.15)

aux<-CPSPpoints(N1=X,N2=Y)
```

CPSPPOTevents — *Identifying the occurrence points of the indicator processes in the CPSP from a POT approach*

Description

This function calculates the occurrence times and other characteristics (length, maximum and mean intensity) of the extreme events of the three indicator processes of a bivariate Common Poisson Shock Process (CPSP) obtained from a Peak Over Threshold (POT) approach.

Usage

```
CPSPPOTevents(N1,N2,thres1,thres2, date=NULL, dplot=T, pmfrow=c(2,1),
 axispoints=NULL,...)
```

Arguments

N1	Numeric vector. Series (x_i) whose threshold exceedances define the first CPSP marginal process.
N2	Numeric vector. Series (y_i) whose threshold exceedances define the second CPSP marginal process.
thres1	Numeric value. Threshold used to define the extreme events in (x_i).
thres2	Numeric value. Threshold used to define the extreme events in (y_i).
date	Optional. A vector or matrix indicating the date of each observation.
dplot	Optional. A logical flag. If it is TRUE, the marginal and indicator processes are plotted.
pmfrow	Optional. A vector of the form (nr, nc) to be supplied as value of the argument mfrow in par.
axispoints	Optional. Numeric vector with the points in the time index where axis ticks and labels (from the first column in date) have to be drawn.
...	Further arguments to be passed to the function plot.

Details

A CPSP N can be decomposed into three independent indicator processes: $N_{(1)}$, $N_{(2)}$ and $N_{(12)}$, the processes of the points occurring only in the first marginal process, only in the second and in both of them (simultaneous points). In the CPSP resulting from applying a POT approach, the events in $N_{(1)}$ are a run of consecutive observations where x_i exceeds its extreme threshold but y_i does not exceed its extreme threshold. An extreme event in $N_{(2)}$ is defined analogously. A simultaneous event, or event in $N_{(12)}$, is a run where both series exceed their thresholds.

For the events defined in each indicator process, three magnitudes (length, maximum intensity and mean intensity) are calculated together with the initial point and the point of maximum excess of each event. In $N_{(12)}$, the maximum and the mean intensity in both series (x_i) and (y_i) are calculated.

The occurrence point of each event is the time in the run where the maximum of the sum of the excesses of (x_i) and (y_i) over their threholds occurs; if an observation does not exceed its corresponding threshold, that excess is 0. According to this definition, the occurrence point in $N_{(1)}$ is the point with maximum intensity in (x_i) within the run.

The vectors inddat1, inddat2 and inddat12, elements of the output list, mark the observations that should be used in the estimation of each indicator process. The observations in an extreme event which are not the occurrence point are marked with 0 and treated as non observed in the estimation process. The rest are marked with 1 and must be included in the likelihood function. See function fitPP.fun in package NHPoisson for more details on the use of these indexes in the estiamtion of a point process.

The points in the marginal N_1, N_2 and indicator $N_{(1)}$, $N_{(2)}$ and $N_{(12)}$ processes can be optionally plotted. If date is NULL, default axis are used. Otherwise, the values in axispoints are used as

the points in the time index where axis ticks and labels, from the first column in `date`, have to be drawn. If `axispoints` is NULL, a default grid of points is built using the function `marca`.

Value

A list with components

`Im1`	Vector of mean excesses (over the threshold) of the extreme events in $N_{(1)}$.
`Ix1`	Vector of maximum excesses (over the threshold) of the extreme events in $N_{(1)}$.
`L1`	Vector of lengths of the extreme events in $N_{(1)}$.
`Px1`	Vector of points of maximum excess of the extreme events in $N_{(1)}$.
`Pi1`	Vector of initial points of the extreme events in $N_{(1)}$.
`inddat1`	Index of the observations to be used in the estimation process of $N_{(1)}$.
`Im2`	Vector of mean excesses (over the threshold) of the extreme events in $N_{(2)}$.
`IxY`	Vector of maximum excesses (over the threshold) of the extreme events in $N_{(2)}$.
`L2`	Vector of lengths of the extreme events in $N_{(2)}$.
`Px2`	Vector of points of maximum excess of the extreme events in $N_{(2)}$.
`Pi2`	Vector of initial points of the extreme events in $N_{(2)}$.
`inddat2`	Index of the observations to be used in the estimation process of $N_{(2)}$.
`Im121`	Vector of mean excesses of the series (x_i) in $N_{(12)}$.
`Ix121`	Vector of maximum excesses the series (x_i) in $N_{(12)}$.
`Im122`	Vector of mean excesses of the series (y_i) in $N_{(12)}$.
`Ix122`	Vector of maximum excesses the series (y_i) in $N_{(12)}$.
`L12`	Vector of lengths of the extreme events in $N_{(12)}$.
`Px12`	Vector of points of maximum excess of the extreme events in $N_{(12)}$.
`Pi12`	Vector of initial points of the extreme events in $N_{(12)}$.
`inddat12`	Index of the observations to be used in the estimation process of $N_{(12)}$.
`N1`	Input argument.
`N2`	Input argument.
`thres1`	Input argument.
`thres1`	Input argument.
`date`	Input argument.

References

Abaurrea, J. Asin, J. and Cebrian, A.C. (2015). A Bootstrap Test of Independence Between Three Temporal Nonhomogeneous Poisson Processes and its Application to Heat Wave Modeling. *Environmental and Ecological Statistics*, 22(1), 127-144.

See Also

`CPSPpoints`, `PlotMCPSP`, `PlotICPSP`

Examples

```
data(TxBHZ)
dateT<-cbind(TxBHZ$year,TxBHZ$month,TxBHZ$day)  #year, month and day of the month
marca<- c(1:length(TxBHZ$TxH))[c(1,diff(dateT[,1]))==1] # points at first day of the year
BivEv<-CPSPPOTevents(N1=TxBHZ$TxH,N2=TxBHZ$TxZ,thres1=378,thres2=364, date=dateT,
       axispoints=marca)
```

---

depchi *Estimating extremal dependence coefficientes*

Description

This function estimates and plots the extremal dependence functions $\chi(u)$ and $\bar{\chi}(u)$ against a grid of values in [0,1] to analyse the extremal dependence of two variables.

Usage

```
depchi(X, Y, thresval = c(0:99)/100, tit = "", indgraph = TRUE, bothest = TRUE,
       xlegend = "topleft",mfrow=c(1,2),...)
```

Arguments

X	Numeric vector. Values of the first variable.
Y	Numeric vector. Values of the second variable.
thresval	Numeric vector. Grid values where the functions $\chi(u)$ and $\bar{\chi}(u)$ are evaluated.
tit	Character string. A title for the plots.
indgraph	Logical flag. If it is TRUE, plots are shown in separate windows. If it is FALSE, the layout in mfrow is used.
bothest	Logical flag. If it is TRUE, two estimated coefficientes (for X given Y and for Y given X) are displayed in the same plot. Otherwise, only the coefficient for Y given X is plotted.
xlegend	Optional. Label "topleft","bottomright", etc. Position where the legend on the graph will be located.
mfrow	Optional. A vector of the form c(2, 1) or c(1,2). It gives the layout to draw the two figures in the function.
...	Further arguments to be passed to the function plot.

Details

The extremal dependence between two variables X and Y is the tendency for one variable to be large, given that the other one is large. The extremal dependence coefficients χ and $\bar{\chi}$ are defined as $\chi = \lim_{u\to 1}\chi(u)$ where $\chi(u) = P(U > u|V > u)$ and (U,V) are the transformed uniform marginals of the variables X and Y.

$\bar{\chi} = \lim_{u\to 1}\bar{\chi}(u)$ where $\bar{\chi}(u) = 2logP(U > u)/logP(U > u, V > u) - 1$.

The function plots $\chi(u)$ and $\bar{\chi}(u)$. These graphs can be used to estimate $\hat{\chi}$ and $\widehat{\bar{\chi}}$, the limits of the functions. In the $\chi(u)$ plot, the expected behaviour under independence of X and Y is also plotted.

χ is on the scale [0, 1], with the set (0, 1] corresponding to asymptotic dependence, and the measure $\bar{\chi}$ falls within the range [-1, 1], with the set [-1, 1) corresponding to asymptotic independence. See Coles et al. (1999) for more details on the definition and interpretation of these indexes.

Value

A list with elements

`chiX`	Estimated $\chi(u)$ function for Y given X evaluated at the threshold grid.
`chiY`	Estimated $\chi(u)$ function for X given Y evaluated at the threshold grid.
`chiBX`	Estimated $\bar{\chi}(u)$ function for Y given X evaluated at the threshold grid.
`chiBY`	Estimated $\bar{\chi}(u)$ function for X given Y evaluated at the threshold grid.
`PX`	Estimation of the probabilities $P(U < thresval)$.
`PY`	Estimation of the probabilities $P(V < thresval)$.
`PXY`	Estimation of the probabilities $P[(U < thresval)\&(V < thresval)]$.
`thresval`	Input argument.

References

Coles, S., Heffernan, J. and Tawn, J. (1999) Dependence measures for extreme value analysis. Extremes, 2, 339-365.

See Also

`CountingCor`, `BinPer`

Examples

```
data(TxBHZ)

aux<-depchi(X=TxBHZ$TxZ,Y=TxBHZ$TxH, thresval = c(0:99)/100,
 tit = "Tx Zaragoza and Tx Huesca", xlegend = "bottom",indgraph="FALSE")
```

`DepCPSPNHK` *Estimating cross K-function and envelopes for marginal processes of a CPSP*

Description

This function estimates the cross K-function between two (homogenous or nonhomogeneous) point processes in time, N_x and N_y. It is evaluated in a grid of distances r and plotted. An envelope built by simulation under the hypothesis that the processes are the marginal processes of a bivariate CPSP is also plotted.

It calls the auxiliary function DepCPSPKenv, not intended for users.

Usage

```
DepCPSPNHK(posx, posy, lambdaix, lambdaiy, lambdaixy, r=NULL, typeEst=1,
                  nsim=1000, conf=0.95,tit=NULL, cores=1,fixed.seed=NULL,...)
```

Arguments

posx	Numeric vector. Occurrence times of the points in the first point process N_x .
posy	Numeric vector. Occurrence times of the points in the second point process N_y .
lambdaix	Numeric vector. Intensity values of $N_{(x)}$.
lambdaiy	Numeric vector. Intensity values of $N_{(y)}$.
lambdaixy	Numeric vector. Intensity values of $N_{(xy)}$.
r	Optional. Numeric vector. Grid values where the K-function must be evaluated. If it is NULL, a default vector is used; see Details.
typeEst	Optional. Two possible values: 1 or 2. They determines which one of the two available estimators of the function K_{ij} has to be used; see Details.
nsim	Optional. Numeric value. Number of simulations to obtain the envelope.
conf	Optional. Numeric value in (0,1). Confidence level of the envelope for the K-function.
tit	Optional. Title to be used in the plot of the K-function.
cores	Optional. Number of cores of the computer to be used in the calculations.
fixed.seed	An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. If it is NULL, a random seed is used.
...	Further arguments to be passed to the function plot.

Details

This function estimates the cross K function between two (homogenous or nonhomogeneous) point processes in time, N_x and N_y. Two different estimators are available, see NHK for more details.

An envelope for the cross K function is built under the hypothesis that the processes are the marginal processes of a bivariate CPSP with intensities of the indicator processes lambdaxi, lambdayi and lambdaxyi. The envelope is based on the simulation of CPSPs, gnerated by function DepNHCPSP.

If argument r is NULL, the following r-grid is used to evaluate the function:

r1<-max(20, floor(T/20))

r<-seq(1,r1,by=2)

if (length(r)>200) r<-seq(1,r1,length.out=200)

where T is the length of the observed period.

Value

A list with elements:

r	Vector of values r where the cross K-function is estimated.
NHKr	Estimated values of $K_{ij}(r)$.
KenvL	Lower bounds of the envelope of $K_{ij}(r)$.
KenvU	Upper bounds of the envelope of $K_{ij}(r)$.
T	Length of the observed period of the processes.

See Also

NHK, DepNHCPSP

Examples

```
set.seed(123)
lambdai1<-runif(500,0.005,0.01)
set.seed(124)
lambdai2<-runif(500,0.005,0.01)
set.seed(125)
lambdai12<-runif(500,0.005,0.02)

#Observed process: independent Poisson processes
posx<-simNHPc(lambda = (lambdai1+lambdai12), fixed.seed = 13)$posNH
posy<-simNHPc(lambda = (lambdai2+lambdai12), fixed.seed = 14)$posNH
aux<-DepCPSPNHK(posx=posx, posy=posy, lambdaix=lambdai1, lambdaiy=lambdai2,
        lambdaixy=lambdai12, fixed.seed=123, r=c(1:10), nsim=500)

#Observed processes: dependent marginal processes of a CPSP
#pos<-DepNHCPSP(lambdaiM=cbind(lambdai1, lambdai2, lambdai12), d=2,
# fixed.seed=123, dplot=F)$posNH
#aux<-DepCPSPNHK(posx=pos$N1, posy=pos$N2, lambdaix=lambdai1,
# lambdaiy=lambdai2, lambdaixy=lambdai12, fixed.seed=125, r=c(1:10))
```

DepNHCPSP *Generating a Common Poisson Shock Process*

Description

This function generates the d marginal processes of a Common Poisson Shock Process, which are d dependent Poisson processes. Both homogeneous and nonhomogeneous processes can be generated. In the case $d = 2$, the processes can be optionally plotted.

Usage

```
DepNHCPSP(lambdaiM, d,dplot=TRUE, pmfrow=c(2,1), fixed.seed = NULL, ...)
```

Arguments

lambdaiM	Matrix. Each column contains the intensity values of a indicator process .
d	Numeric value. Dimension (number of marginal processes) of the CPSP.
dplot	Optional. A logical flag. If it is TRUE and d=2, the marginal and indicator processes are plotted.
pmfrow	Optional. A vector of the form (nr, nc) to be supplied as argument mfrow in par.
fixed.seed	Optional. An integer or NULL. Value used to set the seed in random generation processes; if it is NULL, a random seed is used.
...	Further arguments to be passed to the function plot.

Details

A CPSP N is usually specified by its marginal, and possibly dependent, processes N_1, N_2..., N_d, which are the observed processes. However, N can be decomposed into m independent indicator processes: $N_{(1)}$, $N_{(2)}$, ..., $N_{(12)}$, ..., $N_{(1...d)}$, which are the processes of the points occurring only in the first marginal process, only in the second,..., simultaneously in the two first marginal processes, ... and in all the marginal processes simultaneously. The number of indicator processes is m, the sum of n choose i for $i = 1, ..., d$. The value m must also be the number of columns of the matrix in argument lambdaiM. The marginal process N_i is obtained as the union of all the indicator processes where the index i appears, $N_{.i.}$. The intensity of N_i is the sum of the intensities of all the indicator processes $N_{.i.}$.

The decomposition into indicator processes can be readily applied for the generation of a CPSP: it reduces to the generation of m independet PPs, see Cebrian et al. (2020) for details. Points are generated in continuous time.

In order to generate d independent Poisson processes, the function IndNHPP has be used.

In the bivariate case $d = 2$, the points in the marginal N_1, N_2 and indicator $N_{(1)}$, $N_{(2)}$ and $N_{(12)}$ processes can be optionally plotted.

Value

A list with elements

posNH	A list of d vectors containing the occurrence points of the d marginal processes. The name of the elements of the list are N1, N2...,Nd.
posNHG	A list of m vectors containing the occurrence points of the m indicator processes.
lambdaM	Matrix. Each column is the intensity vector of a marginal process.

References

Abaurrea, J. Asin, J. and Cebrian, A.C. (2015). Modeling and projecting the occurrence of bivariate extreme heat events using a nonhomogeneous common Poisson shock process. Stochastic and Environmental Research and risk assessment, 29(1), 309-322.

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*.

See Also

DepNHNeyScot, DepNHPPqueue, DepNHPPMarked, IndNHPP

Examples

```
set.seed(123)
lambdai1<-runif(200,0,0.1)
set.seed(124)
lambdai2<-runif(200,0,0.07)
set.seed(125)
lambdai12<-runif(200,0,0.05)
set.seed(126)
lambdai123<-runif(200,0,0.01)
lambdaiM<-cbind(lambdai1, lambdai2,lambdai1, lambdai12, lambdai12, lambdai12, lambdai123)
aux<-DepNHCPSP(lambdaiM=lambdaiM, d=3, fixed.seed=123)

#lambdaiM<-cbind(lambdai1, lambdai2, lambdai12)
#aux<-DepNHCPSP(lambdaiM=lambdaiM, d=2,fixed.seed=123, dplot=TRUE)
```

DepNHNeyScot *Generating a multivariate Neyman-Scott cluster process*

Description

This function generates a multivariate Neyman-Scott cluster process, that is a vector of d dependent (homogeneous or nonhomogeneous) point processes which are Neyman-Scott processes with the same trajectory of cluster centers.

It calls the auxiliary function GenSons (not intended for the users), see Details.

Usage

```
DepNHNeyScot(lambdaParent, d, lambdaNumP = 1, dist = "normal", sigmaC = 1,
                    minC = -1, maxC = 1, dplot=TRUE, fixed.seed=NULL,...)
```

Arguments

lambdaParent	Numeric vector. Intensity values of the Poisson process used to generate the centers of the clusters of the Neyman-Scott process.
d	Integer. Number of dependent processes to be generated.
lambdaNumP	Optional. Numeric vector. Mean values of the number of sons of each dependent process. If its length is equal to 1, the same value is used to generate all the dependent processes.
dist	Optional. Label "normal" or "uniform". Distribution used to generate the points of each cluster.
sigmaC	Optional. Numeric vector. Only used if dist="normal". Standard deviation of the normal distribution. If its length is equal to 1, the same value is used in the d processes.

`minC`	Optional. Numeric vector. Only used if dist="uniform". Lower bounds of the Uniform distribution. If its length is equal to 1, the same value is used in the d processes.
`maxC`	Optional. Numeric vector. Only used if dist="uniform". Upper bounds of the Uniform distribution. If its length is equal to 1, the same value is used in the d processes.
`dplot`	Optional. A logical flag. If it is TRUE, the generated marginal processes are plotted.
`fixed.seed`	Optional. An integer or NULL. Value used to set the seed in random generation processes; if it is NULL, a random seed is used.
`...`	Further arguments to be passed to the function `plot`.

Details

A Neyman-Scott process is a Poisson cluster process where the points in each cluster are randomly distributed around the cluster center, see Neyman and Scott (1958) and Entekhabi et al. (1989).

Homogeneous or NH Neyman-Scott processes in continuous time and with the same trajectory of cluster centers are generated, so that d dependent processe are obtained. First, the Poisson process of the cluster centers is generated. Then, the number of points in each cluster is generated using a Poisson distribution with means which can be different in each process. The distances of each point in the cluster to its centre can be generated using two distributions a N(0, sigmaC) or a Uniform(minC, maxC).

It is noteworthy that high values of `sigmaC` or the range `maxC-minC` lead to a high variability around the centre and to a low dependence between the processes.

The marginal processes of the generated vector can be optionally plotted.

Value

A list with elements:

`posNH`	A list of d vectors, containing the occurrence points of the d dependent processes. The name of the elements of the list are N1, N2,..., Nd.
`sizeCL`	A list of d vectors. Each vector contains the size (number of points) of each cluster in a processes. The name of the elements of the list are size1, size2,..., sized.

References

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*.

Neyman, J., & Scott, E. L. (1958). Statistical approach to problems of cosmology. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1-43.

Entekhabi, D., Rodriguez-Iturbe, I., & Eagleson, P. S. (1989). Probabilistic representation of the temporal rainfall process by a modified Neyman-Scott Rectangular Pulses Model: Parameter estimation and validation. *Water Resources Research*, 25(2), 295-302.

See Also

IndNHNeyScot, DepNHPPqueue, DepNHPPMarked, DepNHCPSP

Examples

```
# Generation of three dependent Neyman-Scott processes with  normal distances
set.seed(123)
lambdaParent<-runif(100,0,0.1)

DepNHNeyScot(lambdaParent=lambdaParent, d=3, lambdaNumP = c(2,3,2),
  dist = "normal", sigmaC = c(3,2,2),fixed.seed=123)
```

DepNHPPMarked *Generating dependent point processes from a marked Poison Process*

Description

This function generates d dependent (homogeneous or nonhomogeneous) point processes using a marked Poisson process, where the marks are generated by a Markov chain process defined by a transition matrix.

Usage

```
DepNHPPMarked(lambdaTot, MarkovM, inival = 1, dplot=TRUE, fixed.seed=NULL,...)
```

Arguments

lambdaTot	Numeric vector. Intensity values of the Poisson process used to generate the dependent processes.
MarkovM	Matrix. Trasition probabilities of the d-state Markov chain used to generate the marks of the process.
inival	Optional. Initial mark value used to generate the series of marks.
dplot	Optional. A logical flag. If it is TRUE, the marginal processes are plotted.
fixed.seed	Optional. An integer or NULL. Value used to set the seed in random generation processes; if it is NULL, a random seed is used.
...	Further arguments to be passed to the function plot.

Details

Points of the marked Poisson process are generated in continuous time, using the following procedure: First, a trajectory of the underlying Poisson process is generated. Then, the mark series is generated using a d-state Markov chain. The mark series takes values in 1,2,...,d and determines in which of the d processes the points occur.

The marginal processes defined by the marks are not Poisson unless the generated marks are independent observations, see Isham (1980).

A transition matrix $P = (p_{ij})$ with equal rows leads to d independent point processes, and the more similar the rows of P, the less dependent the resulting processes. The spectral gap (SpecGap) measures the dependence between the generated processes, see Abaurrea et al. (2014).

Tha marginal processes of the marked process can be optionally plotted using `dplot=TRUE`.

Value

A list with elements

`posNH`	A list of d vectors, containing the occurrence points in each marginal point process. The name of the elements of the list are N1, N2,..., Nd.
`posNHG`	Numeric vector of the occurrences times of the generated Poisson process.
`mark`	Vector of the generated marks.
`lambdaTot`	Input argument.
`MarkovM`	Input argument.

References

Abaurrea, J. Asin, J. and Cebrian, A.C. (2015). A Bootstrap Test of Independence Between Three Temporal Nonhomogeneous Poisson Processes and its Application to Heat Wave Modeling. *Environmental and Ecological Statistics*, 22(1), 127-144.

Isham, V. (1980). Dependent thinning of point processes. *J. Appl. Probab.*, 17(4), 987-95.

See Also

DepNHPPqueue, DepNHNeyScot, DepNHCPSP, IndNHPP, SpecGap

Examples

```
# Generation of three dependent  point processes using a marked PP
set.seed(123)
lambdaTot<-runif(1000)/10
aux<-DepNHPPMarked(lambdaTot=lambdaTot,
 MarkovM=cbind(c(0.3,0.1,0.6), c(0.1, 0.6, 0.3), c(0.6, 0.3,0.1)),fixed.seed=123)
print(cbind(aux$posNH, aux$mark))
print(aux$posNHs)
```

DepNHPPqueue *Generating dependent point processes by a tandem queueing network*

Description

It generates d dependent (homogeneous or nonhomogeneous) Poisson processes using $d-1$ queues in tandem.

Usage

```
DepNHPPqueue(lambda, d,   T=NULL,  nEv=NULL,  nserv="infty", Clambda=TRUE,
             ddist='exp',argd=1,dplot=TRUE, fixed.seed=NULL, ...)
```

Arguments

lambda	Numeric value or vector. Intensity of the first Poisson process. If its length is 1, homogeneous processes are generated.
d	Integer. Number of dependent processes to be generated.
T	Optional. Positive integer. Length of the period where the point are going to be generated. Only used in homogeneous processes (if lambda is a constant).
nEv	Optional. Positive integer. Number of points to be generated in the processs. Only used in homogeneous processes.
Clambda	Optional logical flag. Only used in nonhomogeneous processes. If it is TRUE, the intensity vectors of the generated point processes is calculated.
nserv	Optional. Number of servers in the queue system; only two values are possible: 1 or "infty" if the number of servers is infinity.
ddist	Character string. Identification of the probability distribution of the serving time. It must be one of the names of the probability distributions available in the stats package, see Distributions; for example "gamma", "geom", etc.
.	
argd	Optional. A value, vector or a matrix (with d-1 rows) containing the arguments to be used in the probability distribution in ddist; see Details.
dplot	Optional. A logical flag. If it is TRUE, the marginal processes are plotted.
fixed.seed	Optional. An integer or NULL. Value used to set the seed in random generation processes; if it is NULL, a random seed is used.
...	Further arguments to be passed to the function plot.

Details

The generation algorithm is described in Cebrian et al. (2020). The generation of dependent homogeneous Poisson processes can be based on a M\ M\ 1 or a M\ M \ ∞ queue, and the intensity of each output process is equal to the intensity of the input process, see Burke(1956). The generation of dependent nonhomogeneous Poisson processes is based on a M(t)\ G \ ∞ queue and the intensity of each output process is equal to the convolution $\lambda_{out}(t) = \lambda_{inp}(t) * f(t)$, where $\lambda_{inp}(t)$ is the input intensity and $f(t)$ the density function of the service time, see Keilson and Servi (1994).

In the homogeneous processes, the argument λ can be an integer or a vector with equal values. In the first case, the argument nEv or T must be specified; in the second, the length of the vector determines the length of period where the points are generated (as in the nonhomogeneous case).

The serving times in all the queues must have a probability distribution in the same family (ddist) but the parameters of that distribution can be different in each queue. If the parameters are the same in all the queues, argument argd is a numeric value (if there is only one parameter) or a vector; if the parameters are different in each queue, argd must be a matrix with (d-1) rows, even if there is only one parameter it must be a (d-1) $\times$ 1 matrix.

The occurrence times in each output process are sorted, so that the output time in row i does not always correspond to the input time in the same row. The number of points in the output process may be lower than in the input (if some outputs have not been observed).

Value

A list with elements

posNH	A list of d vectors, containing the occurrence points in each point process. The name of the elements in the list are N1, N2,..., Nd.
lambdaM	A d-column matrix containing the intensity vectors of the d dependent processes, in nonhomogeneous processes and if Clambda=TRUE. Otherwise, the input argument lambda.

References

Burke, P. J. (1956). The Output of a Queuing System. *Operations Research*. 4(6), 699-704.

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*

Keilson, J. Servi, L.D. (1994). Networks of nonhomogeneous M G ∞. *J. Appl. Probab.*, 31, 157-68.

See Also

IndNHPP, DepNHNeyScot, DepNHPPMarked, DepNHCPSP

Examples

```
#Generation of 3 dependent HPPs, with  one server  and exponential service time equal to 10
aux<-DepNHPPqueue(lambda=0.05, d=3,  nEv=25,fixed.seed=123,nserv=1, argd=0.1)
aux$posNH

#Generation of 4 dependent NHPPs, with infinity servers and different mean service times
#at each queue
lambda<-runif(200,0,0.1)
aux<-DepNHPPqueue(lambda=lambda, d=4, fixed.seed=123, argd=cbind(c(0.1, 0.3, 0.1)))
aux$posNH


#Generation of 3 dependent NHPPs, with infinity servers and  Gamma service times
#with different parameters at each queue
lambda<-runif(200,0,0.1)
aux<-DepNHPPqueue(lambda=lambda, d=3, ddist='gamma',fixed.seed=123,
                  argd=cbind(c(0.1, 0.3), c(1,2)))
aux$posNH
```

DepqueueNHK *Estimating cross K-function and envelopes for the marginal processes of a queue*

Description

This function estimates the cross K-function between two (homogenous or nonhomogeneous) point processes in time, N_x and N_y. It is evaluated in a grid of distances r and plotted. An envelope built by simulation under the hypothesis that the processes are the marginal processes (the input and the output) of a queue is also plotted.

It calls the auxiliary functions NHKaux, NHKaux2, NHKaux3 and DepqueueKenv, not intended for users.

Usage

```
DepqueueNHK(posx, posy, lambda, T=NULL, nserv='infty', ddist='exp',argd=1,r=NULL,
 typeEst=1, nsim=1000, conf=0.95,tit=NULL, cores=1,fixed.seed=NULL,...)
```

Arguments

posx	Numeric vector. Occurrence times of the points in the first point processes N_x.
posy	Numeric vector. Occurrence times of the points in the second point processes N_y.
lambda	Numeric vector. Intensity values of the input process N_x.
T	Optional. Positive integer. Length of the period where the point are going to be generated. Only used in homogeneous processes (if lambda is a constant).
nserv	Optional. Number of servers in the queue system; only two values are possible: 1 or "infty" if the number of servers is infinity.
ddist	Character string. Identification of the probability distribution of the serving time. It must be one of the names of the probability distributions available in the stats package, see Distributions; for example "gamma", "geom", etc.
.	
argd	Optional. Numeric value or vector containing the arguments to be used in the probability distribution in ddist.
r	Optional. Numeric vector. Grid values where the K-function must be evaluated. If it is NULL, a default vector is used; see Details.
typeEst	Optional. Two possible values: 1 or 2. They determine which one of the two available estimators of the function K_{ij} has to be used; see Details.
nsim	Optional. Numeric value. Number of simulations to obtain the envelope.
conf	Optional. Numeric value in (0,1). Confidence level of the envelope for the K-function.
tit	Optional. Title to be used in the plot of the K-function.

cores	Optional. Number of cores of the computer to be used in the calculations.
fixed.seed	An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.
...	Further arguments to be passed to the function plot.

Details

This function estimates the cross K function between two (homogenous or nonhomogeneous) point processes in time, N_x and N_y. Two different estimators are available, see NHK for details.

An envelope for the cross K function is built under the hypothesis that the processes are the input and the output processes of a queue. The envelope is based on simulations, where processes generated by DepNHPPqueue are used. The distribution of the serving time is specified in arguments ddist and argd. The queue can have one or infinity servers.

If argument r is NULL, the following r-grid is used to evaluate the function

r1<-max(20, floor(T/20))

r<-seq(1,r1,by=2)

if (length(r)>200) r<-seq(1,r1,length.out=200)

where T is the length of the observed period.

Value

A list with elements:

r	Vector of values r where the cross K-function is estimated.
NHKr	Estimated values of $K_{ij}(r)$.
KenvL	Lower bounds of the envelope of$K_{ij}(r)$.
KenvU	Upper bounds of the envelope of $K_{ij}(r)$.
T	Length of the observed period of the processes.

See Also

NHK, DepNHPPqueue

Examples

```
#Observed process: independent Poisson processes
set.seed(123)
T<-1000
lambda1<-runif(T,0,0.05)
dendist<- dexp(c(1:T), 1)
lambda2<-convolve(lambda1,rev(dendist),type='o')[1:T]
posx<-simNHPc(lambda = lambda1, fixed.seed = 134)$posNH
posy<-simNHPc(lambda = lambda2, fixed.seed = 135)$posNH

DepqueueNHK(posx=posx, posy=posy, lambda=lambda1,  ddist='exp',argd=1,
 r=seq(1,40, by=5), fixed.seed=123,nsim=500)
```

```
#Observed process: input and output processes  of a queue
#aux<-DepNHPPqueue(lambda=lambda1, d=2, fixed.seed=123, argd=1, dplot=FALSE)$posNH
#DepqueueNHK(posx=aux[[1]], posy=aux[[2]], lambda=lambda1,  ddist='exp',argd=1,
# r=seq(1,40, by=5), fixed.seed=123)
```

`DistObs` *Calculates the set of close points and the mean distance in a vector of processes, for each point in the first process*

Description

Given a set of two or three processes, this function calculates the set of close points and the mean distance for each point in the first process.

It calls the functions `calcdist`, not intended for the users, and `uniongentri`.

Usage

```
DistObs(posx, posy, posz=NULL, info = FALSE, PA = FALSE, procName=c('X','Y','Z'),...)
```

Arguments

`posx`	Numeric vector. Position of the occurrence points in the first process.
`posy`	Numeric vector. Position of the occurrence points in the second process.
`posz`	Optional. Numeric vector. Position of the occurrence points in the third process.
`info`	Optional logical flag. If it is TRUE, information about the generated points is showed on the screen and dotcharts and bivariate charts of the occurrence points of the processes are displayed.
`PA`	Optional logical flag. If it is TRUE, the close point relation is broadened by including the previous and the following points to the overlapping intervals.
`procName`	Vector of character strings. Labels for the first, second and third processes.
`...`	Further arguments to be passed to the function `plot`, if argument *info="TRUE"*.

Details

Given a set of two or three point proccesses, for each point t_{x_i} in the first process of the set, this function calcultes its set of close points and the mean distance to its close points. The definition of set of close points can be found in Abaurrea et al. (2015), and the distances are defined as $|t_{y_j} - t_{x_i}|$ if there are two processes, and as $|t_{y_j} - t_{x_i}| + |t_{z_k} - t_{x_i}|$ if there are three.

Value

`DistTri`	The vector of the mean distances of points t_{x_i}.

References

Abaurrea, J. Asin, J. and Cebrian, A.C. (2015). A Bootstrap Test of Independence Between Three Temporal Nonhomogeneous Poisson Processes and its Application to Heat Wave Modeling. *Environmental and Ecological Statistics*, 22(1), 127-144.

See Also

TestIndNH, DistSim, uniongentri

Examples

```
data(TxBHZ)
dateT<-cbind(TxBHZ$year,TxBHZ$month,TxBHZ$day)  #year, month and day of the month
BivEv<-CPSPPOTevents(N1=TxBHZ$TxH,N2=TxBHZ$TxZ,thres1=36.4, thres2=37.8, date=dateT)
aux<-DistObs(BivEv$Px1, BivEv$Px2, BivEv$Px12,info = TRUE)
```

`DistShift` *Generates by translation a vector of independent processes, and calculates the set of close points and the mean distance for each point in the first process*

Description

This function generates a vector of two (or three) independent processes, conditionally on the first one, by shifting the second (and the third) process.

It also calculates the set of close points and the mean distance in the generated vector, for each point t_{x_i} in the first process.

Usage

```
DistShift(posx,posy,posz=NULL, T,  shii1, shii2=NULL, PA = FALSE, info=FALSE,...)
```

Arguments

`posx`	Numeric vector. Position of the occurrence points in the first process.
`posy`	Numeric vector. Position of the occurrence points in the second process.
`posz`	Optional. Numeric vector. Position of the occurrence points in the third process.
`T`	Numeric value. Length of the observed period of the processes.
`shii1`	Numeric value. Distance used to shift the points in the second process. It must be a positive value lower than T.
`shii2`	Optional. Numeric value. Distance used to shift the points in the third process. It must be NULL, if there are two processes, and a positive value lower than T, if there are three.
`PA`	Optional. Logical flag. If it is TRUE, the close point relation is broadened by including the previous and the following points to the overlapping intervals.

info	Optional. Logical flag. If it is TRUE, information about the generated points is shown on the screen and dotcharts and bivariate charts of the occurrence points of the three processes are displayed.
...	Further arguments to be passed to the functions plot and dotchart if the argument info=T

Details

This function is mainly used in the application of the Lotwicck- Silverman approach, see Lotwick and Silverman (1982), to generate a pair of independent processes with the same marginal distributions than the observed ones. These processes are used for example to build a test to assess independence between two or three processes, see TestIndLS.

The key idea is to wrap the processes onto a circumference by identifying the opposite sides of the time interval where they are observed. The first process is fixed, while the others are shifted over the circumference a given amount. The idea of this translation is to keep the marginal distribution of the processes but to break any dependence between them, without the need of parametric models to describe the marginal patterns.

The function also calculates the set of close points and the mean distance for each point t_{x_i} in the first process, in the new shifted vector of processes.

Value

DistTri	The vector of the mean distances of points t_{x_i} in the shifted processes.

References

Lotwick, H.W. and Silverman, B.W. (1982). Methods for analysing Spatial processes of several types of points. *J.R. Statist. Soc. B*, 44(3), pp. 406-13

See Also

TestIndLS, DistSim

Examples

```
set.seed(123)
lambdax<-runif(200, 0.01,0.17)
set.seed(124)
lambday<-runif(200, 0.015,0.15)
set.seed(125)
lambdaz<-runif(200, 0.005,0.1)
posx<-simNHPc(lambda=lambdax, fixed.seed=123)$posNH
posy<-simNHPc(lambda=lambday, fixed.seed=123)$posNH
posz<-simNHPc(lambda=lambdaz, fixed.seed=123)$posNH

aux<-DistShift(posx=posx, posy=posy, posz=posz, T=200, shii1=59, shii2=125 )
```

DistSim — *Generates a vector of independent processes, and calculates the set of close points and the mean distance for each point in the first process*

Description

This function generates a vector of two (or three) independent homogeneous or nonhomogeneous processes conditionally on the first one, by simulating the second (and the third) process using a parametric model (Poisson processes or Neyman-Scott cluster processes).

It also calculates the set of close points and the mean distance in the generated vector, for each point t_{x_i} in the first process.

`DistSimfix` allows to fix a seed in the generation process.

Usage

```
DistSim(posx, NumProcess=2, type = "Poisson", lambdaMarg = NULL,
 lambdaParent = NULL, lambdaNumP=NULL, dist = "normal", sigmaC = 1,
 minC = -1, maxC = 1, PA = FALSE,info=FALSE,...)

DistSimfix(posx, NumProcess=2, type = "Poisson", lambdaMarg = NULL,
 lambdaParent = NULL,lambdaNumP=NULL, dist = "normal", sigmaC = 1,
 minC = -1, maxC = 1, PA = FALSE,info=FALSE,fixed.seed=1,...)
```

Arguments

`posx`	Numeric vector. Position of the occurrence points in the first process.
`NumProcess`	Optional. Integer equal to 2 or 3, the number of processes in the vector.
`type`	Optional. Label "Poisson" or "PoissonCluster". Type of point processes to be generated. Up to now, only two types are available: Poisson processes ("Poisson") and Neyman-Scott cluster processes ("PoissonCluster").
`lambdaMarg`	Two-column matrix. Only used when *type="Poisson"*. Each column is the intensity $\lambda(t)$ used to generate the processes.
`lambdaParent`	Numeric vector. Only used when *type="PoissonCluster"*. Intensity values of the Poisson process used to generate the centres of the clusters of the Neyman-Scott process.
`lambdaNumP`	Numeric vector (length ≤ 2). Only used when *type="PoissonCluster"*. Mean values of the number of sons in each process. If its length is 1 and `NumProcess=2`, the same value is used for both processes.
`dist`	Optional. Label "normal" or "uniform". Only used when *type="PoissonCluster"*. Distribution used to generate the point distances in each cluster.
`sigmaC`	Optional. Numeric vector. Only used when *type="PoissonCluster"* and *dist="normal"*. Standard deviation of the normal distribution. If its length is 1 and `NumProcess=2`, the same value is used for both processes.

minC	Optional. Numeric vector. Only used when *type="PoissonCluster"* and *dist="uniform"*. Lower bounds of the Uniform distribution. If its length is 1 and NumProcess=2, the same value is used for both processes.
maxC	Optional. Numeric vector. Only used when *type="PoissonCluster"* and *dist="uniform"*. Upper bounds of the Uniform distribution. If its length is 1 and NumProcess=2, the same value is used for both processes.
PA	Optional. Logical flag. If it is TRUE, the close point relation is broadened by including the previous and the following points to the overlapping intervals.
info	Optional. Logical flag. If it is TRUE, information about the generated points is shown on the screen and dotcharts and bivariate charts of the occurrence points of the three processes are displayed.
fixed.seed	Optional. Only available in DistSimfix. Integer value used to set the seed in random generation procedures.
...	Further arguments to be passes to the functions plot and dotchart if argument info=T.

Details

This function is mainly used in the application of a parametric bootstrap approach to generate a pair of independent processes with the same marginal distributions than the observed ones. To that aim, the first process is fixed and the others are generated using a parametric model. These processes are used for example to build a test to assess the independecne between two or three processes, see TestIndNH.

Two types of processes (Poisson, "Poisson", and Neyman-Scott cluster processes,"PoissonCluster") can be generated. Generation of nonhomogeneous Poisson processes is done using the inversion algorithm, see simNHPc. For generation of Neyman-Scott processes, see IndNHNeyScot.

The function also calculates the set of close points and the mean distance for each point t_{x_i} in the first process, in the new generated vector of processes.

The lenght of the period where the processes are generated is determined by the length of the argument lambdaParent or the number of rows of lambdaMarg. Homogenous processes are generated if the intensity vectors in lambdaParent or in lambdaMarg are constant (that is if all the values in the vector are equal).

If a seed must be fixed in the generation process, function DistSimfix has to be used. The functions DistSim and DistSimfix are similar, the difference is that the first one uses a random seed to generate the processes, while the second one uses a seed set by the argument fixed.seed.

Value

DistTri	Vector of the mean distances of each point t_{x_i} calculated in the generated processes.

References

Abaurrea, J. Asin, J. and Cebrian, A.C. (2015). A Bootstrap Test of Independence Between Three Temporal Nonhomogeneous Poisson Processes and its Application to Heat Wave Modeling. *Environmental and Ecological Statistics*, 22(1), 127-144.

See Also

TestIndNH, DistObs, IndNHNeyScot, simNHPc

Examples

```
#Calculation of the distances in a vector of  three independent Poisson processes
#conditionally to the first one

set.seed(123)
lambdax<-runif(200, 0.01,0.15)
posaux<-simNHPc(lambda=lambdax, fixed.seed=123)$posNH

set.seed(124)
lambday<-runif(200, 0.005,0.1)
set.seed(125)
lambdaz<-runif(200, 0.005,0.2)

DistSimfix(posx=posaux, type = "Poisson", lambdaMarg = cbind(lambday,lambdaz),
 fixed.seed=123, info=TRUE)
#DistSim(posx=posaux, type = "Poisson", lambdaMarg = cbind(lambday,lambdaz))
```

DutilleulPlot *A graphical test to assess independence between two point processes*

Description

This function applies the Diggle's randomization testing procedure extended by Dutilleul(2011), and performs a plot which graphically assesses the independence between two point proceses. It is implemented for homogenous and non homogenous Poisson processes.

Usage

```
DutilleulPlot(posx, posy, lambday, nsim = 1000, lenve = c(0.025, 0.975), ...)
```

Arguments

posx	Numeric vector. Occurrence times of the points in the first point process.
posy	Numeric vector. Occurrence times of the points in the second point process.
lambday	Numeric vector. Intensity vector of the second point process. If the process is homogeneous, a vector of length T, with equal values must be provided; see Details.
nsim	Optional. Positive integer. Number of simulations to calculate the confidence band.
lenve	Optional. Numeric vector. The order of the lower and the upper percentiles to build the confidence band.
...	Further arguments to be passed to the function plot.

Details

This graphical approach is based on the comparison of the cumulative relative frequency of the nearest neighbour distances between the points in the two observed processes, with their counterpart in two independent processes with the same marginal distributions, which are obtained by simulation.

The function plots the cumulative relative frequency of the observed processes and a confidence band calculated from *nsim* simulated independent processes.

The length of the observed period T is determined by the length of the argument `lambday`.

Value

A list with the elements:

`quantobs`	Vector of the observed percentiles of the nearest neighbour distances.
`enve1`	Vector of the lower bounds of the confidence band.
`enve2`	Vector of the upper bounds of the confidence band.

References

Dutilleul, P. (2011), *Spatio-temporal heterogeneity: Concepts and analyses*, Cambridge University Press.

See Also

`TestIndNH`, `CondTest`,`nearestdist`

Examples

```
#Two independent NHPPs
set.seed(123)
lambdax<-runif(200, 0.01,0.1)
set.seed(124)
lambday<-runif(200, 0.015,0.15)
posx<-simNHPc(lambdax,fixed.seed=123)$posNH
posy<-simNHPc(lambday, fixed.seed=123)$posNH

aux<-DutilleulPlot(posx, posy, lambday,  nsim = 100)


#Two dependent Neyman Scott processes
#set.seed(123)
#lambdaParent<-runif(200)/10
#DepPro<-DepNHNeyScot(lambdaParent=lambdaParent, d=2, lambdaNumP = 3,
# dist = "normal", sigmaC = 3,fixed.seed=123)
#posx<-DepPro$PP1
#posy<-DepPro$PP2
#aux<-DutilleulPlot(posx, posy, lambday, nsim = 100)
```

IndNHNeyScot *Generating a vector of independent Neyman-Scott cluster processes*

Description

This function generates a vector of d independent (homogeneous or nonhomogeneous) Neyman-Scott cluster processes with independent trajectories of cluster centers with the same intensity.

It calls the auxiliary function GenSons (not intended for the users), see Details.

Usage

```
IndNHNeyScot(lambdaParent, d, lambdaNumP = 1,  dist = "normal",
 sigmaC = 1, minC = -1, maxC = 1, dplot=TRUE,fixed.seed=NULL,...)
```

Arguments

lambdaParent	Numeric vector. Intensity of the Poisson process used to generate the independent trajectories of the cluster centres of the Neyman-Scott process.
d	Integer. Number of independent processes to be generated.
lambdaNumP	Optional. Numeric vector. Mean values of the number of sons of each marginal process. If its length is equal to 1, the same value is used to generate all the processes.
dist	Optional. Label "normal" or "uniform". Distribution used to generate the point locations of each cluster.
sigmaC	Optional. Numeric vector. Standard deviation of the normal distribution. Only used if dist="normal". If its length is equal to 1, the same value is used in the d processes.
minC	Optional. Numeric vector. Lower bounds of the Uniform distribution. Only used if dist="uniform". If its length is equal to 1, the same value is used in the d processes.
maxC	Optional. Numeric vector. Upper bounds of the Uniform distribution. Only used if dist="uniform". If its length is equal to 1, the same value is used in the d processes.
dplot	Optional. A logical flag. If it is TRUE, the generated marginal processes are plotted.
fixed.seed	Optional. An integer or NULL. Value used to set the seed in random generation processes; if it is NULL, a random seed is used.
...	Further arguments to be passed to the function plot.

Details

A Neyman-Scott process is a Poisson cluster process where the points in each cluster are randomly distributed around the cluster center, see Neyman and Scott (1958) and Entekhabi et al. (1989).

To generate each process in the vector, an independent trajectory of the Poisson process of the cluster centres is generated first. Then, the number of points in each cluster is generated using a Poisson distribution with mean value μ_{P_i} (i=1,...d). Finally, the distances to the centre of each point in the cluster is be generated using one of the two distributions available, N(0, sigmaC) or Uniform(minC, maxC).

The lenght of the period where the processes are generated is determined by the length of the argument `lambdaParent`.

Homogenous processes are generated if the intensity vector `lambdaParent` is constant (that is if all the values are equal).

The marginal processes of the generated vector can be optionally plotted.

Value

A list with elements:

`posNH`	A list of d vectors, each one containing the time occurrences of one of the marginal processes. The name of the elements of the list are N1, N2,...,Nd.

References

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*.

Neyman, J., & Scott, E. L. (1958). Statistical approach to problems of cosmology. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1-43.

Entekhabi, D., Rodriguez-Iturbe, I., & Eagleson, P. S. (1989). Probabilistic representation of the temporal rainfall process by a modified Neyman-Scott Rectangular Pulses Model: Parameter estimation and validation. *Water Resources Research*, 25(2), 295-302.

See Also

`DepNHNeyScot`, `IndNHPP`

Examples

```
set.seed(123)
lambda<-runif(1000)/10

IndNHNeyScot(lambdaParent=lambda, d=3, lambdaNumP = c(2,3,2),  dist = "normal",
 sigmaC = 2, fixed.seed=123)
```

IndNHPP *Generates trajectories of independent Poisson processes*

Description

This function generates independent Poisson processes, which can be homogeneous or nonhomogeneous depending on the value of the intensity vectors.

Usage

```
IndNHPP(lambdas,dplot=TRUE, fixed.seed=NULL,...)
```

Arguments

lambdas	Matrix where each column contains the intensity vector to generate a Poisson process.
dplot	Optional. A logical flag. If it is TRUE, the marginal processes are plotted.
fixed.seed	An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.
...	Further arguments to be passed to the function plot.

Details

The number of generated processes is determined by the number of columns of the argument lambdas. The lenght of the period where the processes are generated is determined by the number of rows of lambdas.

Homogenous processes are generated if the corresponding intensity vector is constant (that is if all the rows of the corresponding column are equal).

For the generation algorithm of each Poisson process, see simNHPc.

Value

posNHs	A list of d vectors, each one containing the time occurrences of the independent NHPPs. The name of the elements of the list are N1, N2,..., Nd.

See Also

IndNHNeyScot, simNHPc

Examples

```
set.seed(123)
lambdas<-cbind(runif(500)/10, rep(0.05,500))

IndNHPP(lambdas=lambdas, fixed.seed=123)
```

IntMPP *Simulated intervals in a vector of point processes*

Description

This function calculates a point estimation and a confidence interval for a given parameter related to a vector of point processes using a Monte Carlo (or parametric bootstrap) approach. The estimator of the parameter must be a function of the occurrence points of the (possibly dependent) marginal processes of the vector of processes.

It calls the auxiliary function `funMPPGen` (not intended for the users), see Details.

Usage

```
IntMPP(funMPP.name, funMPP.args, fun.name, fun.args = NULL, nsim=1000, clevel = 0.95,
    cores = 1, fixed.seed = NULL)
```

Arguments

`funMPP.name`	Name of the function defining the distribution of the vector of point processes.
`funMPP.args`	Additional arguments for the function `funMPP.name`.
`fun.name`	Name of the function to calculate the estimation of the parameters. The first argument of this function must be a list called posNH.
`fun.args`	A list whose elements are the additional arguments for the function `fun.name`.
`nsim`	Number of simulations to be carried out.
`clevel`	Confidence level of the interval. A value in (0,1).
`cores`	Optional. Number of cores of the computer to be used in the calculations. Default: one core is used.
`fixed.seed`	An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.

Details

This function calculates a point estimation and a confidence interval of a parameter related to a vector of point processes. It calls the auxiliary function `funMPPGen`, which generates a sample of vectors of processes using a parametric model. The parameter of interest is estimated using each process in that sample, so that a sample of values of the estimator is obtained. The mean of that sample is the point estimator, and the adequate sample percentiles give the lower and upper bounds of the confidence interval.

The parametric model is specified by the arguments `funMPP.name` and `funMPP.args`. Functions `DepNHCPSP`, `DepNHNeyScot`, `DepNHPPqueue` and `DepNHPPMarked` can be used as input of the argument `funMPP.name` to generate the corresponding vector of processes.

The considered estimator must be a function of the occurrence points of the vector of processes and any additional arguments, provided by argument `fun.args`, which must be a list. The first argument of the function `fun.name` must be a list called `posNH` whose elements are numeric vectors containing the occurrence points of each point process in the vector. For example, the first element of the output list of `DepNHCPSP` can be used as first argument of `fun.name`.

Value

A list with elements:

valmed	Point estimation (mean value) of the parameter.
valinf	Lower bound of the generated interval.
valsup	Upper bound of the generated interval.
nsim	Input argument.
fixed.seed	Input argument.

Examples

```
# Calculation of the point estimation and  95% intervals based on 1000 simulations
#of the  number of accurrences in each marginal process of a bivariate  Neyman-Scot process
# in  the time interval [100,200]
#NumI calculates the  number of occurrences in interval I in each element of  the list posNH

set.seed(123)
lambdai<-runif(1000,0.01,0.02)

aux<-IntMPP(funMPP.name="DepNHNeyScot", funMPP.args=list(lambdaParent=lambdai,d=2,
  lambdaNumP=c(2,1), dplot=FALSE), fun.name="NumI", fun.args = list(I=c(100,200)),
  fixed.seed = 125)


# Calculation of the point estimation and a 95% interval based on 1000 simulations
#of the  first occurrence time in  a multivariate  CPSP with d=3
#firstt calculates the  minimim occurrence time of  all the elements in the list posNH

#set.seed(124)
#lambdaij<-runif(1000,0.005,0.02)
#set.seed(125)
#lambdaijk<-runif(1000,0.001,0.02)
#lambdaiM<-cbind(lambdai,lambdai, lambdai, lambdaij, lambdaij, lambdaij, lambdaijk)
#aux<-IntMPP(funMPP.name="DepNHCPSP",funMPP.args=list(lambdaiM=lambdaiM,d=3,dplot=FALSE),
#  fun.name="firstt", fixed.seed = 125)
```

---

nearestdist *Distance to the nearest point*

Description

Given the occurrence points in two point processes, this function calculates for each point in the first process, the distance to the nearest occurrence point in the second process.

Usage

```
nearestdist(posx, posy)
```

Arguments

posx	Numeric vector. Occurrence times of the points in the first point process.
posy	Numeric vector. Occurrence times of the points in the second point process.

Details

The distance between two points x_i and y_i in a point process in time, is the absolute value of their difference: $|x_i - y_i|$.

To obtain the vector of nearest points, this function applies to each point in posx, the function pdist, which calculates the distance to its nearest point in posy.

Value

Vector of the distances to the nearest point in the second process for each point in the first process.

See Also

DutilleulPlot

Examples

```
posx<-c(3,8,23,54,57,82)
posy<-c(2,8,14,16,29,32,45,55,65)
nearestdist(posx, posy)
```

NHD *Estimating the D-function*

Description

This function estimates the cross nearest neighbour distance distribution function, D, between two sets, C and D, of (homogenous or nonhomogeneous) point processes. The D-function is evaluated in a grid of values r, and it can be optionally plotted.

It calls the auxiliary functions NHDaux and other functions, not intended for users.

Usage

```
NHD(lambdaC, lambdaD, T=NULL,Ptype='inhom', posC, typeC=1, posD, typeD=1,
 r = NULL, dplot = TRUE, tit = "D(r)",...)
```

Arguments

lambdaC	A matrix of positive values. Each column is the intensity vector of one of the point processes in C. If there is only one process in C, it can be a vector or even a numeric value if the process is homogeneous.
lambdaD	A matrix of positive values. Each column is the intensity vector of one of the point process in D. If there is only one process in D, it can be a vector or even a numeric value if the process is homogeneous.
T	Numeric value. Length of the observed period. It only must be specified if the number of rows in lambdaC and lambdaD is 1.
Ptype	Optional. Label: "hom" or "inhom". The first one indicates that all the point processes in sets C and D are homogeneous.
posC	Numeric vector. Occurrence times of the points in all the point processes in C.
typeC	Numeric vector with the same length as posC. Code of the point process in C where points in posC have occurred. See Details.
posD	Numeric vector. Occurrence times of the points in all the point processes in D.
typeD	Numeric vector with the same length as posD. Code of the point process in D where points in posD have occurred.
r	Numeric vector. Values where the D-function must be evaluated. If it is NULL, a default vector is used, see Details.
dplot	Optional. A logical flag. If it is TRUE, the D-function is plotted.
tit	Optional. The title to be used in the plot of the D-function.
...	Further arguments to be passed to the function plot.

Details

The information about the processes is provided by arguments posC, the vector of all the occurrence times in the processes in C, and typeC, the vector of the code of the point process in set C where each point in posC has occurred; the second set D is characterized analogously by typeD and posD.

This function estimates the D-function between two sets, C and D, of (homogenous or nonhomogeneous) point processes, see Cebrian et al (2020) for details of the estimation. The D-function is the distribution function of the distances from a point in a process in C to the nearest point in a process D. In homogeneous proceesses, it estimates the probability that at least one point in a process in set D occurs at a distance lower than r of a given point in a process in set C. If the processes are nonhomogenous, the inhomogenous version of the function, adjusted for time varying intensities, is used. It is calculated using the Hanisch estimator, see Van Lieshout (2006) Small values of the D-function suggest few points in processes in D in the r-neighbourhood of points of processes in C. Large values indicate that points in processes in D are attracted by those of processes in C.

For inference about independence of the processes, K and J-functions should be used.

If argument r is NULL, the following grid is used to evaluate the function

r1<-max(20, floor(T/20))

r<-seq(1,r1,by=2)

if (length(r)>200) r<-seq(1,r1,length.out=200)

Value

A list with elements:

r	Vector of values r where the D-function is estimated.
NHDr	Estimated values of $D_{CD}(r)$.
T	Length of the observed period.

References

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*.

Van Lieshout, M.N.M. (2006) A J-function for marked point patterns. AISM, 58, 235-259. DOI 10.1007/s10463-005-0015-7

See Also

NHK, NHJ, NHF

Examples

```
#Sets C and D with  independent NHPPs
set.seed(123)
lambda1<-runif(500, 0.05, 0.1)
set.seed(124)
lambda2<-runif(500, 0.01, 0.2)
pos1<-simNHPc(lambda=lambda1, fixed.seed=123)$posNH
pos2<-simNHPc(lambda=lambda2, fixed.seed=123)$posNH
aux<-NHD(lambdaC=lambda1, lambdaD=lambda2, posC=pos1, typeC=1, posD=pos2, typeD=1)
aux$NHDr

#Example with  independent NHPPs
#pos3<-simNHPc(lambda=lambda1, fixed.seed=321)$posNH
#pos4<-simNHPc(lambda=lambda2, fixed.seed=321)$posNH
#aux<-NHD(lambdaC=cbind(lambda1,lambda2),lambdaD=cbind(lambda1,lambda2),posC=c(pos1,pos2),
# typeC=c(rep(1, length(pos1)), rep(2, length(pos2))), posD=c(pos3, pos4),
# typeD=c(rep(1, length(pos3)), rep(2, length(pos4))))
#aux$NHDr
```

NHF *Estimating the F-function*

Description

This function estimates the F-function in a set of homogenous or nonhomogeneous point processes, D. The F-function is evaluated in a grid of values r, and it can be optionally plotted.

It calls the auxiliary functions NHFaux and other functions not intended for users.

Usage

```
NHF(lambdaD, T=NULL, Ptype='inhom', posD, typeD=1, r=NULL,L=NULL, dplot=TRUE,
tit='F(r)',...)
```

Arguments

lambdaD	A matrix of positive values. Each column is the intensity vector of one of the point process in D. If there is only one process in D, it can be a vector or even a numeric value if the process is homogeneous.
T	Numeric value. Length of the observed period. It only must be specified if the number of rows in lambdaC and lambdaD is 1.
Ptype	Optional. Label: "hom" or "inhom". The first one indicates that all the point processes in sets C and D are homogeneous.
posD	Numeric vector. Occurrence times of the points in all the point processes in D.
typeD	Numeric vector with the same length as posD. Code of the point process in D where the point in the same row in posD has occurred. The code must be the column number where the intensity of that process is in matrix lambdaD.
r	Numeric vector. Values where the F-function must be evaluated. If it is NULL, a default vector is used, see Details
L	Optional. Numeric vector. Values in the observed period used to calculate the F-function. If it is NULL, a default vector is used, see Details.
dplot	Optional. Logical flag. If it is true, the F-function is plotted.
tit	Optional. The title to be used in the plot of the F-function.
...	Further arguments to be passed to the function plot.

Details

The information about the processes is provided by arguments posD, the vector of all the occurrence times in the processes in C, and typeD, the vector of the code of the point process in set D where each point in posD has occurred.

This function estimates the F-function in a set D of homogenous or nonhomogeneous time point processes, see Cebrian et al (2020) for details of the estimation. The F-function, also known as empty space function, is the distribution function of the distances from an arbitray point in the space to the nearest point in a process in D. In homogeneous processes, it estimates the probability that at least one point in processes in D occurs at a distance lower than r of an arbitray point in the space. If the processes are nonhomogenous, the inhomogenous version of the function, adjusted for time varying intensities, is used.

If argument r is NULL, the following grid is used to evaluate the function

r1<-max(20, floor(T/20))

r<-seq(1,r1,by=2)

if (length(r)>200) r<-seq(1,r1,length.out=200)

If argument L is NULL, the following grid is used

L <- seq(1, T, by = 2) if (length(L) > 5000) L <- seq(1, T, by = round((T - 1)/199))

Value

A list with elements:

r	Vector of values r where the F-function is estimated.
NHFr	Estimated values of $F_D(r)$.
T	Length of the observed period of the process.
L	Grid of L values to calculate the F-funtion.

References

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*.

See Also

NHK, NHJ, NHD

Examples

```
set.seed(123)
lambda1<-runif(500, 0.05, 0.1)
pos1<-simNHPc(lambda=lambda1, fixed.seed=123)$posNH

aux<-NHF(lambdaD=lambda1, posD=pos1, typeD=1)
aux$NHFr

#Set D with two processes ***
#lambda2<-runif(1000, 0.01, 0.2)
#pos2<-simNHPc(lambda=lambda2, fixed.seed=123)$posNH
#aux<-NHF(lambdaD=cbind(lambda1,lambda2), posD=c(pos1,pos2),
# typeD=c(rep(1, length(pos1)), rep(2, length(pos2))) )
#aux$NHFr
```

NHJ *Estimating the cross J-function and testing independence*

Description

This function estimates the cross J-function between two sets, C and D, of (homogenous or nonhomogeneous) point processes in time. It is evaluated in a grid of distances r, and it can be optionally plotted. A test to assess the independence between the sets of processes, based on the cross J-function, is also implemented.

It calls the auxiliary functions NHJaux and Jenv, not intended for users.

Usage

```
NHJ(lambdaC, lambdaD,T=NULL, Ptype="inhom", posC, typeC=1, posD, typeD=1, r=NULL,
L=NULL,test=FALSE,nTrans=100, rTest=NULL, conf=0.95, dplot=NULL,
tit=c("J-function","D-function","F-function"),mfrow=NULL,cores=1,fixed.seed=NULL,...)
```

Arguments

lambdaC	A matrix of positive values. Each column is the intensity vector of one of the point processes in C. If there is only one process in C, it can be a vector or even a numeric value if the process is homogeneous.
lambdaD	A matrix of positive values. Each column is the intensity vector of one of the point process in D. If there is only one process in D, it can be a vector or even a numeric value if the process is homogeneous.
T	Numeric value. Length of the observed period. It only must be specified if the number of rows in lambdaC and lambdaD is 1.
Ptype	Optional. Label: "hom" or "inhom". The first one indicates that all the point processes in sets C and D are homogeneous.
posC	Numeric vector. Occurrence times of the points in all the point processes in C.
typeC	Numeric vector with the same length as posC. Code of the point process in C where the points in posC have occurred. See Details.
posD	Numeric vector. Occurrence times of the points in all the point processes in D.
typeD	Numeric vector with the same length as posD. Code of the point process in D where the points in posD have occurred.
r	Optional. Numeric vector. Values where J-function must be evaluated. If it is NULL, a default vector is used, see Details.
L	Optional. Numeric vector. Values in the observed period used to calculate the J-function. If it is NULL, a default vector is used, see Details.
test	Optional. Logical flag. If it is TRUE, a test of independence and a 95% envelope for the J-function are calculated.
nTrans	Optional. Numeric value. Only used if test=TRUE. Number of translations to be performed in the test and envelope calculation.
rTest	Optional. Numeric value. Maximum value of r used to calculate the independence test statistc, see Details.
conf	Optional. Numeric value in (0,1). Confidence level of the envelope for the J-function.
dplot	Optional. Label "JDF" or "J". If it is "JDF", plots of J, D and F-functions are displayed. If it is "J", only J-function is plotted.
tit	Optional. A vector with one or three titles to be used in the plots of J, D and F-functions.
mfrow	Optional. Argument to be passed to par for the plot of the J-function.
cores	Optional. Number of cores of the computer to be used in the calculations.
fixed.seed	An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.
...	Further arguments to be passed to the function plot.

Details

The information about the processes is provided by arguments posC, the vector of all the occurrence times in the processes in C, and typeC, the vector of the code of the point process in set C where each point in posC has occurred; the second set D is characterized analogously by typeD and posD.

This function estimates the cross J-function between two sets, C and D, of (homogenous or non-homogeneous) time point processes, see Cebrian et al (2020) for details of the estimation. The J-function measures the interpoint dependence between points in any of the processes in D, and points in any of the processes in C, adjusted for time varying intensity in the case of nonhomogenous processes. The cross J-function is defined as $J_{CD}(r) = (1 - D_{CD}(r))/(1 - F_D(r))$, if $F_D(r) < 1$ and it is not calculated otherwise. It compares $D_{CD}(r)$, the distribution function of the distances from a point in any of the processes in set C to the nearest point in any of the processes in set D, to $F_D(r)$, the distribution function of the distances from a fixed point in the space to the nearest point in any of the processes in set D.

If argument r is NULL, the following grid is used to evaluate the function

r1<-max(20, floor(T/20))

r<-seq(1,r1,by=2)

if (length(r)>200) r<-seq(1,r1,length.out=200)

If argument L is NULL, the following grid is used

L <- seq(1, T, by = 2) if (length(L) > 5000) L <- seq(1, T, by = round((T - 1)/199))

Testing independence:

If the processes in C are independent of the processes in D given the marginal structure of the processes, the J-funtion is equal to 1, since D(r)=F(r). Hence, deviations of J(r) estimations from 1, suggest dependence betweent the two sets of processes. The test statistic is based on the mean of values $|J(r) - 1|$ evaluated in a given grid of r values.

A test based on a Lotwick-Silverman approach, see Lotwick and Silverman (1982), is implemented. This test provides a nonparametric way to test independence given the marginal intensities of the processes. Using the Lotwick-Silverman approach, not only the p-value of the test but also an envelope for the $J(r)$ values is calculated.

In point processes, dependence often appears between close observations, and with high r values it is more difficult that the J-function is able to discriminate between dependent and independent processes. By this reason, the argument rTest allows us to fix a maximum value of r so that only $J(r)$ estimations for $r < rTest$ will be used to calculate the test statistic. The value rTest is drawn in the plot of the J-function as a vertical grey line.

Value

A list with elements:

r	Vector of values r where the J-function is estimated.
NHJr	Estimated values of $J_{CD}(r)$.
NHDr	Estimated values of $D_{CD}(r)$.
NHFr	Estimated values of $F_D(r)$.
JenvL	Lower bounds of the envelope of $J_{CD}(r)$.

JenvU	Upper bounds of the envelope for $J_{CD}(r)$.
JStatOb	Observed value of the statistic.
JStatTr	Sample of the values of the test statistic obtained by random translations.
pv	P-value of the independence test.
T	Length of the observed period of the process.
L	Grid of L values to calculate the F-funtion.

References

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*.

Cronie, O. and van Lieshout, M.N.M. (2015). Summary statistics for inhomogeneous marked point processes. *Ann Inst Stat Math*.

Lotwick, H.W. and Silverman, B.W. (1982). Methods for analysing Spatial processes of several types of points. *J.R. Statist. Soc. B*, 44(3), pp. 406-13

See Also

NHK, NHD, NHF

Examples

```
set.seed(120)
lambda1<-runif(100, 0.05, 0.1)
set.seed(121)
lambda2<-runif(100, 0.01, 0.2)
pos1<-simNHPc(lambda=lambda1,fixed.seed=123)$posNH
pos2<-simNHPc(lambda=lambda2,fixed.seed=123)$posNH

aux<-NHJ(lambdaC=lambda1, lambdaD=lambda2, posC=pos1,nTrans=50,
  posD=pos2, rTest=7, dplot='J', cores=1,test=TRUE)
aux$pv

#Sets with two processes
#pos3<-simNHPc(lambda=lambda1,fixed.seed=300)$posNH
#pos4<-simNHPc(lambda=lambda2,fixed.seed=30)$posNH
#aux<-NHJ(lambdaC=cbind(lambda1,lambda2), lambdaD=cbind(lambda1,lambda2),
# posC=c(pos1,pos2), typeC=c(rep(1, length(pos1)), rep(2, length(pos2))),
# posD=c(pos3, pos4), typeD=c(rep(1, length(pos3)), rep(2, length(pos4))),
# dplot='J', test=TRUE)
#aux$pv
```

NHK *Estimating cross K-function and testing independence*

Description

This function estimates the cross K-function between two sets, C and D, of (homogenous or nonhomogeneous) point processes in time. It is evaluated in a grid of distances r, and it can be optionally plotted. A test to assess the independence between the sets of processes, based on the cross K-function, is also implemented.

It calls the auxiliary functions NHKaux, NHKaux2, NHKaux3 and Kenv, not intended for users.

Usage

```
NHK(lambdaC, lambdaD, T=NULL, posC, typeC=1, posD, typeD=1, r=NULL, test=TRUE,
 typeEst=2,  nTrans=1000, conf=0.95, rTest=NULL, typePlot=" ",tit=NULL,
 cores=1,fixed.seed=NULL,...)
```

Arguments

lambdaC	A matrix of positive values. Each column is the intensity vector of one of the point processes in C. If there is only one process in C, it can be a vector or even a numeric value if the process is homogeneous.
lambdaD	A matrix of positive values. Each column is the intensity vector of one of the point process in D. If there is only one process in D, it can be a vector or even a numeric value if the process is homogeneous.
T	Numeric value. Length of the observed period. It only must be specified if all the processes are homogeneous, that is if the number of rows in lambdaC and lambdaD is 1.
posC	Numeric vector. Occurrence times of the points in all the point processes in C.
typeC	Numeric vector with the same length as posC. Code of the point process in C where the points in posC have occurred; see Details.
posD	Numeric vector. Occurrence times of the points in all the point processes in D.
typeD	Numeric vector with the same length as posD. Code of the point process in D where the points in posD have occurred.
r	Optional. Numeric vector. Grid values where the K-function must be evaluated. If it is NULL, a default vector is used; see Details.
test	Optional. Logical flag. If it is TRUE, a test of independence and a 95% envelope for the K-function are calculated.
typeEst	Optional. Two possible values: 1 or 2, which determines which one of the two available estimators of the function K_{ij} has to be used; see Details.
nTrans	Optional. Numeric value. Only used if test=TRUE. Number of translations to be performed in the test and envelope calculation.
conf	Optional. Numeric value in (0,1). Confidence level of the envelope for the K-function.

rTest	Optional. Numeric value. Maximum value of r used to calculate the test statistc, see Details.
typePlot	Optional. Character string. If it is "Kfun" or "Kest" a plot of the values $\hat{K}_{xy}(r)$ or $\hat{K}_{xy}(r)/2r$ is shown. With any other value, no plot is carried out.
tit	Optional. Title to be used in the plot of the K-function.
cores	Optional. Number of cores of the computer to be used in the calculations.
fixed.seed	An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.
...	Further arguments to be passed to the function plot.

Details

The information about the processes is provided by arguments posC, the vector of all the occurrence times in the processes in C, and typeC, the vector of the code of the point process in set C where each point in posC has occurred; the second set D is characterized analogously by typeD and posD.

This function estimates the cross K function between two sets, C and D, of (homogenous or non-homogeneous) point processes. Two different estimators are available, see Cebrian et al (2020) for details. The cross K-function measures the dependence between two point processes (or two sets of point processes) and counts the expected number of points in any of the processes in D, within a given distance of a point in any of the processes in C, adjusted for time varying intensity in the case of nonhomogenous processes. The cross K-function of independent Poisson processes is the length of the considered intervals, $K_{CD}(r) = 2r$. Then, values $K_{CD}(r)/2r > 1$ indicate attraction between the processes, while values lower than 1 indicate repulsion.

If argument r is NULL, the following r-grid is used to evaluate the function

r1<-max(20, floor(T/20))

r<-seq(1,r1,by=2)

if (length(r)>200) r<-seq(1,r1,length.out=200)

Testing independence:

In order to test the independence hypothesis using this function, a test based on a Lotwick-Silverman approach, see Lotwick and Silverman (1982), is implemented. This test provides a nonparametric way to test independence given the marginal intensities of the processes. Using the Lotwick-Silverman approach, not only the p-value of the test but also an envelope for the $K(r)$ values is calculated. The test statistic is based on the mean of values $K(r)/(2r)$ evaluated in a given grid of r values.

In point processes, dependence often appears between close observations, and with high r values it is more difficult that the K-function is able to discriminate between dependent and independent processes. By this reason, the argument rTest allows us to fix a maximum value of r so that only K(r) estimations for $r < rTest$ will be used to calculate the test statistic. The value rTest is drawn in the plot of the K-function as a vertical grey line.

Value

A list with elements:

r	Vector of values r where the cross K-function is estimated.

NHKr	Estimated values of $K_{ij}(r)$.
KenvL	Lower bounds of the envelope of$K_{ij}(r)$.
KenvU	Upper bounds of the envelope of $K_{ij}(r)$.
KStatOb	Observed value of the test statistic.
KStatTr	Sample of the values of the test statistic obtained by random translations.
pv	P-value of the test.
T	Length of the observed period of the processes.

References

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*.

Lotwick, H.W. and Silverman, B.W. (1982). Methods for analysing Spatial processes of several types of points. *J.R. Statist. Soc. B*, 44(3), pp. 406-13

See Also

NHD, NHJ, NHF

Examples

```
set.seed(122)
lambda1<-runif(100, 0.05, 0.1)
set.seed(121)
lambda2<-runif(100, 0.01, 0.2)
pos1<-simNHPc(lambda=lambda1, fixed.seed=123)$posNH
pos2<-simNHPc(lambda=lambda2, fixed.seed=123)$posNH

aux<-NHK(lambdaC=lambda1, lambdaD=lambda2, posC=pos1, posD=pos2, typePlot='Kest',
 nTrans=200, cores=1, typeEst=2, fixed.seed=120)
aux$pv

#Sets with two processes
#pos3<-simNHPc(lambda=lambda1, fixed.seed=321)$posNH
#pos4<-simNHPc(lambda=lambda2, fixed.seed=321)$posNH
#aux<-NHK(lambdaC=cbind(lambda1,lambda2), lambdaD=cbind(lambda1,lambda2), posC=c(pos1,pos2),
# typeC=c(rep(1, length(pos1)), rep(2, length(pos2))), posD=c(pos3, pos4),
# typeD=c(rep(1, length(pos3)), rep(2, length(pos4))),  typeplot='Kest', fixed.seed=120)
#aux$pv
```

`PlotICPSP` *Plotting the occurrence points of the indicator processes in a CPSP*

Description

This function plots the points in the three indicator processes $N_{(1)}$, $N_{(2)}$ and $N_{(12)}$ of a bivariate Common Poisson shock process (CPSP).

Usage

```
PlotICPSP(posi1,posi2, posi12, T, date=NULL,axispoints=NULL, ...)
```

Arguments

`posi1`	Numeric vector of the points in $N_{(1)}$
`posi2`	Numeric vector of the points in $N_{(2)}$
`posi12`	Numeric vector of the points in $N_{(12)}$
`T`	Numeric value. The length of the observed period of the CPSP.
`date`	Optional. A vector indicating the date of each observation to be used in the axis of the plot.
`axispoints`	Optional. Numeric vector with the points in the time index in which axis ticks and labels (from `date`) will be drawn.
`...`	Further arguments to be passed to the function `plot`.

Details

A CPSP N can be decomposed into three independent indicator processes: $N_{(1)}$, $N_{(2)}$ and $N_{(12)}$, the processes of the points occurring only in the first marginal process, only in the second and in both of them (simultaneous points).

The points in the three indicator processes are plotted versus the time index. If one of the arguments `date` and `axispoints` is NULL, default axis are used. Otherwise, the values in `axispoints` are used as the points in the time index in which axis ticks and labels from `date` are drawn.

Value

A plot.

See Also

`CPSPpoints`, `PlotMCPSP`, `PlotMargP`

Examples

```
data(TxBHZ)
T<-length(TxBHZ$TxH)
dateT<-cbind(TxBHZ$year,TxBHZ$month,TxBHZ$day)  #year, month and day of the month
marca<- c(1:length(TxBHZ$TxH))[c(1,diff(dateT[,1]))==1]  #points at first day of the year
BivEv<-CPSPPOTevents(N1=TxBHZ$TxH,N2=TxBHZ$TxZ,thres1=37.8, thres2=36.4, date=dateT,
                     axispoints=marca)
PlotICPSP(posi1=BivEv$Px1,posi2=BivEv$Px2, posi12=BivEv$Px12, T=T)
PlotICPSP(posi1=BivEv$Px1,posi2=BivEv$Px2, posi12=BivEv$Px12, T=T, date=dateT[,1],
          axispoints=marca)
```

`PlotMargP` *Plotting the occurrence points of a vector of point processes*

Description

This function plots the points in the marginal processes N_1, N_2,..., N_d of a vector of point processes.

Usage

```
PlotMargP(listpos, T, date=NULL,axispoints=NULL, tcex=1.2, ...)
```

Arguments

`listpos`	A list of vectors. Each element of the list is the vector of the occurrences in a marginal process.
`T`	Numeric value. The length of the observed period of the processes.
`date`	Optional. A vector indicating the date of each observation to be used in the axis of the plot.
`axispoints`	Optional. Numeric vector with the points in the time index in which axis ticks and labels (from `date`) will be drawn.
`tcex`	Optional. cex argument, see `par`, for the text labels in the plot.
`...`	Further arguments to be passed to the function `plot`.

Details

The points in the d marginal processes N_1, N_2 ,..., N_d of a vector of point processes are plotted versus the time index.

If one of the arguments `date` and `axispoints` is NULL, default axis are used. Otherwise, the values in `axispoints` are used as the points in the time index in which axis ticks and labels, from `date`, are drawn.

Value

A plot.

See Also

PlotMCPSP

Examples

```
set.seed(123)
N1<-runif(50,0,5000)
set.seed(124)
N2<-runif(42,0,5000)

PlotMargP(list(N1=N1, N2=N2),T=5000)
```

PlotMCPSP *Plotting the occurrence points of the marginal processes in a CPSP*

Description

This function plots the points in the two marginal processes N_1, N_2 of a bivariate Common Poisson shock process (CPSP).

Usage

```
PlotMCPSP(pos1,pos2, T, date=NULL, axispoints=NULL, ...)
```

Arguments

pos1	Numeric vector of the points in N_1
pos2	Numeric vector of the points in N_2
T	Numeric value. The length of the observed period of the CPSP.
date	Optional. A vector indicating the date of each observation to be used in the axis of the plot.
axispoints	Optional. Numeric vector with the points in the time index in which axis ticks and labels (from date) will be drawn.
...	Further arguments to be passed to the function plot.

Details

The points in the two marginal processes N_1, N_2 of a bivariate CPSP are plotted versus the time index. The simultaneous points (points of the indicator process $N_{(12)}$) are drawn in red.

If one of the arguments date and axispoints is NULL, default axis are used. Otherwise, the values in axispoints are used as the points in the time index in which axis ticks and labels, from date, are drawn.

Value

A plot.

See Also

CPSPpoints, PlotICPSP

Examples

```
data(TxBHZ)
T<-length(TxBHZ$TxH)
dateT<-cbind(TxBHZ$year,TxBHZ$month,TxBHZ$day)  #year, month and day of the month
marca<- c(1:T)[c(1,diff(dateT[,1]))==1]  # points at  first day  of the year
BivEv<-CPSPPOTevents(N1=TxBHZ$TxH,N2=TxBHZ$TxZ,thres1=37.8, thres2=36.4, date=dateT,
                     axispoints=marca)
PlotMCPSP(pos1=union(BivEv$Px1, BivEv$Px12),pos2=union(BivEv$Px2,BivEv$Px12), T=T)

marca<- c(1:T)[c(1,diff(dateT[,1]))==1]
PlotMCPSP(pos1=union(BivEv$Px1, BivEv$Px12),pos2=union(BivEv$Px2,BivEv$Px12), T=T,
 date=dateT[,1], axispoints=marca)
```

`simHPc` *Generating points in a homogenous Poisson process*

Description

This function generates a given number of occurrence points in a homogenous Poisson process (HPP) in continuous time.

Usage

```
simHPc(lambda, nEv, fixed.seed=NULL)
```

Arguments

`lambda`	Numeric positive value. Intensity λ used to generate the HPP.
`nEv`	Optional. Positive integer. Number of points to be generated in the HPPs.
`fixed.seed`	An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.

Details

The points in a HPP are generated using independent exponentials with mean λ.

Points in a HPP can also be generated using `simNHPc`. The main difference is that in `simHPc` the number of points to be generated is given, while `simNHPc` generates points in a period of a given length T.

Value

A list with elements:

posN	Numeric vector. Occurrence points of the HPP.
T	Length of the period where the given number of points are generated.
fixed.seed	Input argument.

References

Ross, S.M. (2006). *Simulation.* Academic Press.

See Also

simNHPc, IndNHPP

Examples

```
aux<-simHPc(lambda=0.01, nEv=50,fixed.seed=123)
aux$posH
```

---

simNHPc *Generating points in a Poisson process*

Description

This function generates the occurrence points in a homogenous or nonhomogeneous Poisson process (NHPP) with a given intensity $\lambda(t)$, in a continuous period of time (0, T).

It calls the auxiliary function buscar (not intended for the users), see Details.

Usage

```
simNHPc(lambda, fixed.seed=NULL, algor="Thinning")
```

Arguments

lambda	Numeric vector. Intensity $\lambda(t)$ used to generate the Poisson process. Its length determines the length of the observed period.
fixed.seed	An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.
algor	Optional. Character string. The algorithm used to generate the process, it can be "Inversion" or "Thinning"; see Details.

Details

Two algorithms to generate the NHPP points are implemented. "Inversion" is based on the inversion algortihm, see Ross(2006), and it consists in two steps. First, the points of a homogeneous Poisson process of intensity one are generated using independent exponentials. Then, the homogeneous occurrence times are transformed into the points of a nonhomogeneous process with intensity $\lambda(t)$. This transformation is performed by the auxiliary function `buscar` (not intended for the user).

The algorithm "Thinning", see Banerjee et al. (2014), generates the occurrences times in a homogeneous Poisson process with intensity $\lambda_{max} = \max_t \lambda(t)$ and the resulting points are retained with probability $\lambda(t_i)/\lambda_{max}$.

The "Inversion" algorithm requires positive values of the argument `lambda` and it is slower, but the "Thinning" algorithm may yield excesive rejection according to Ross (2006).

The lenght of the period where the processes are generated is determined by the length of the argument `lambda`.

Homogenous processes are generated if the intensity vector `lambda` is constant (that is if all the values are equal).

Value

A list with elements:

`posNH`	Numeric vector. Occurrence points of the Poisson process.
`lambda`	Input argument.
`fixed.seed`	Input argument.

References

Banerjee, S., Carlin, B.P. and Gelfand, A. E. (2014) *Hierarchical modeling and analysis for spatial data*.CRC Press.

Ross, S.M. (2006). *Simulation.* Academic Press.

See Also

`simHPc`, `IndNHPP`

Examples

```
#Generation  of a Homogeneous Poisson process
aux<-simNHPc(lambda=rep(0.1,200),fixed.seed=123, algor='Inversion')
aux$posNH

#Generation of a NHPP
set.seed(123)
lambdat<-runif(500, 0.01,0.1)
aux<-simNHPc(lambda=lambdat,fixed.seed=123, algor='Thinning')
aux$posNH
```

SpecGap *Stationary distribution of a matrix and its spectral gap*

Description

This function calculates the stationary distribution of the transition matrix of a Markov chain process and its spectral gap.

Usage

```
SpecGap(P)
```

Arguments

P	Matrix. It must be a markovian matrix.

Details

The spectral gap of a matrix P measures the convergence speed of P to a matrix P_I with all the rows equal to $(\pi_1, \pi_2, ...\pi_k)$, the stationary distribution of P. It takes values in [0,1].

The spectral gap of a transition matrix can be used as a dependence measure between the marginal processes defined by a marked Poisson procces with discrete marks generated by a Markov chain with that transition matrix, see Cebrian et al (2020) for details.

Value

A list with elements

SG	Spectral gap value of the matrix.
pi	Vector of the stationary distribution of the matrix.

References

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*.

See Also

DepNHPPMarked

Examples

```
P<-cbind(c(0.7, 0.1, 0.2), c(0.2, 0.7, 0.1), c(0.1, 0.2, 0.7))
SpecGap(P)
```

`TestIndLS` *Lotwick-Silverman test of independence between point processes*

Description

This function calculates a test based on the Lotwick-Silverman (LoS) approach to study the independence between two or three homogeneous point processes in time. The statistic is based on the close point sets of the points in the first process.

Usage

```
TestIndLS(posx, posy, posz=NULL, T,  alpha = 0.05, nTrans = 100, PA = FALSE,
 cores=1,fixed.seed=NULL)
```

Arguments

`posx`	Numeric vector. Position of the occurrence points in the first process.
`posy`	Numeric vector. Position of the occurrence points in the second process.
`posz`	Numeric vector. Position of the occurrence points in the third process. Only used if there are 3 processes.
`T`	Numeric value. Length of the observed period of the processes.
`alpha`	Optional. Significance level used to obtain a decision (reject-no reject) based on the test p-value.
`nTrans`	Optional. Positive integer. Number of translations to calculate the test.
`PA`	Optional. Logical flag. If it is TRUE, the close point relation is broadened by including the previous and the following points to the overlapping intervals.
`cores`	Optional. Number of cores of the computer to be used in the calculations.
`fixed.seed`	Optional. An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.

Details

The underlying idea of the test is to compare, for each point in the first process, the behavior of its set of close points in the vector of observed processes (N_x, N_y, N_z), and in new vectors of independent processes with the same marginal distribution. The new independent vectors are obtained using a LoS approach, see Lotwick and Silverman (1982): the process N_x is fixed and second and third processes are obtained by shifting the original ones a random amount. This translation keeps the distribution of the homogeneous processes, but breaks any dependence between them. If the observed behavior is significantly different, independence is rejected. More details can be foun in Cebrian et al. (2020).

The test statistic is the one used in `TestIndNH`, but the p-value is obtained using a LoS approach, so that it does not require any assumption about the marginal distribution of the processes, not even the marginal intensities. The test `TestIndNH`, can be applied to study nonhomogeneous processes, but it requires a parametric model for the second process.

Value

A list with elements:

pv	P-value of the independence test.
reject	Binary variable indicating if the test is rejected (1) or not (0) at an alpha significance level.
est	Sample of the KS statistics. The first value corresponds to the observed processes and the others to the generated processes.

References

Cebrian, A.C., Abaurrea, J. and Asin, J. (2020). Testing independence between two point processes in time. *Journal of Simulation and Computational Statistics*.

Lotwick, H.W. and Silverman, B.W. (1982). Methods for analysing Spatial processes of several types of points. *J.R. Statist. Soc. B*, 44(3), pp. 406-13

See Also

TestIndNH, CondTest, DutilleulPlot, DistShift

Examples

```
#Test applied to three independent HPP

posx<-simNHPc(lambda=rep(0.1,200),fixed.seed=123)$posNH
posz<-simNHPc(lambda=rep(0.15,200),fixed.seed=124)$posNH
posy<-simNHPc(lambda=rep(0.1,200),fixed.seed=125)$posNH


aux<-TestIndLS(posx, posy, posz,T=200,
 cores=1,fixed.seed=321)
aux$pv
```

TestIndNH *Parametric bootstrap test of independence between point processes*

Description

This function calculates a parametric bootstrap test (PaB) to study the independence between two or three homogeneous or nonhomogeneous point processes in time. The statistic is based on the close point sets of the points in the first process. Currently, it is implemented for Poisson processes and for Neyman-Scott cluster processes.

Usage

```
TestIndNH(posx, posy, posz=NULL,  alpha = 0.05, nsim = 100, PA = FALSE, cores = 1,
 type = "Poisson", lambdaMarg = NULL,   lambdaParent = NULL, lambdaNumP = NULL,
 dist = "normal", sigmaC = 1, minC = -1, maxC = 1,fixed.seed=NULL)
```

Arguments

posx	Numeric vector. Position of the occurrence points in the first process.
posy	Numeric vector. Position of the occurrence points in the second process.
posz	Numeric vector. Position of the occurrence points in the third process. By default is null, and only two processes are tested.
alpha	Optional. Significance level used to obtain a decision (reject-no reject) based on the test p-value.
nsim	Optional. Positive integer. Number of simulations to calculate the test.
PA	Optional. Logical flag. If it is TRUE, the close point relation is broadened by including the previous and the following points to the overlapping intervals.
cores	Optional. Number of cores of the computer to be used in the calculations.
type	Optional. Label "Poisson" or "PoissonCluster". Type of point processes to be generated in the parametric bootstrap. Up to now, only two types are available: Poisson processes ("Poisson") and Neyman-Scott cluster processes ("PoissonCluster").
lambdaMarg	Matrix of positive values and dimension $T \times N_P$ with N_P=1 or 2. Only used if *type="Poisson"*. Each column is the intensity vector to generate the processes N_y and N_z.
lambdaParent	Numeric vector. Only used if *type="PoissonCluster"*. Intensity vector of the process used to generate the centers of the clusters of the Neyman-Scott process.
lambdaNumP	Numeric vector with 1 or 2 values. Only used if *type="PoissonCluster"*. Mean values of the number of sons of the processes to be generated. If its length is equal to 1 and there are three processes, the same value is used to generate N_y and N_z.
dist	Optional. Label "normal" or "uniform". Only used if *type="PoissonCluster"*. Distribution used to generate the point distances to the centre in each cluster.
sigmaC	Optional. Numeric vector with 1 or 2 values. Only used if *type="PoissonCluster"* and *dist='normal'*. Standard deviation of the normal distribution. If its length is equal to 1, the same value is used in both processes.
minC	Optional. Numeric vector with 1 or 2 values. Only used if *type="PoissonCluster"* and *dist='uniform'*. Lower bounds of the Uniform distribution. If its length is equal to 1 and there are three processes, the same value is used to generate N_y and N_z.
maxC	Optional. Numeric vector with 1 or 2 values. Only used if *type="PoissonCluster"* and *dist='uniform'*. Upper bounds of the Uniform distribution. If its length is equal to 1 and there are three processes, the same value is used to generate N_y and N_z.
fixed.seed	Optional. An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.

Details

The underlying idea of the test is to compare, for each point in the first process, the behavior of its set of close points in the vector of observed processes (N_x, N_y, N_z), and in new vectors of

independent processes with the observed marginal distribution. The new independent vectors are obtained using a parametric bootstrap approach, see Abaurrea et al. (2015): the process N_x is fixed and second and third processes are generated using a parametric model with intensities λ_y and λ_z. Currently, it is implemented for Poisson processes and for Neyman-Scott cluster processes. If the observed behavior is significantly different, independence is rejected.

The test statistic is the one used in `TestIndLS`, but the p-value is obtained using a Monte Carlo approach if the intensities $\lambda_y(t)$ and $\lambda_z(t)$ are known, or a parametric bootstrap if they have been estimated. The test `TestIndLS` can only be applied to homogeneous processes, but it does not require any assumption about the distribution of the marginal processes.

It is noteworthy that when the test is applied, it is being assumed that the processes follow a parametric model with the given intensities. If necessary, validation of that assumption should be previously carried out.

The lenght of the observed period is determined by the length of the intensity vector λ, that is *lambdaParent* (if *type="PoissonCluster"*) or the first element of the dimension of *lambdaMarg* (if *type="PoissonC"*. It can be applied to homogeneous processes, using an intensity vector (`lambda`) with equal values.

Value

A list with elements:

`pv`	P-value of the independence test.
`reject`	Binary variable indicating if the test is rejected (1) or not (0) at an alpha significance level.
`est`	Sample of the KS statistics. The first value corresponds to the observed processes and the others to the generated processes.

References

Abaurrea, J. Asin, J. and Cebrian, A.C. (2015). A Bootstrap Test of Independence Between Three Temporal Nonhomogeneous Poisson Processes and its Application to Heat Wave Modeling. *Environmental and Ecological Statistics*.

See Also

`TestIndLS`,`CondTest`, `DutilleulPlot`, `DistSim`, `DistObs`, `uniongentri`

Examples

```
#Test applied to 3  independent NHPP
set.seed(123)
lambdax<-runif(150, 0.01,0.1)
set.seed(124)
lambday<-runif(150, 0.02,0.1)
set.seed(125)
lambdaz<-runif(150, 0.015,0.1)
posx<-simNHPc(lambdax, fixed.seed=123)$posNH
posy<-simNHPc(lambday, fixed.seed=124)$posNH
posz<-simNHPc(lambdaz, fixed.seed=125)$posNH
```

```
aux<-TestIndNH(posx, posy, posz, nsim=50, type='Poisson',
 lambdaMarg=cbind(lambday,lambdaz), fixed.seed=321)
aux$pv


#Test applied to 3 dependent NS cluster processes with 2 cores
#set.seed(123)
#lambdaParent<-runif(500,0,0.1)
#DepPro<-DepNHNeyScot(lambdaParent=lambdaParent, d=3, lambdaNumP = 3,
#  dist = "normal", sigmaC = 1, fixed.seed=123,cores=2)
#posx<-DepPro$PP1
#posy<-DepPro$PP2
#posz<-DepPro$PP3
#aux<-TestIndNH(posx, posy, posz, cores=1, type='PoissonCluster',
# lambdaParent = lambdaParent, lambdaNumP = 3,
# dist = "normal", sigmaC = 1, fixed.seed=123, nsim=200)
#aux$pv
```

`TranM` *Estimation of the transition matrix of a Markov chain*

Description

It estimates the transition matrix of a Makov chain to model the dependence between the discrete marks of a marked point process. The estimator is the MLE based on count data.

Usage

```
TranM(marcas = NULL, d = NULL, vecpro = NULL)
```

Arguments

`marcas`	Integer vector. It contains the discrete marks of the marked point process. The order of the marks in the vector must correspond to the points in the process sorted over time.
`d`	Integer. Number of states of the Markov chain, that is the number of different marks of the marked point process.
`vecpro`	A list with d elements. Element "i" of the list must be a vector including the occurrence times of the points in the marked point process with marks equal to "i".

Details

The input of this function must be a marked point process. It can be defined by the sequence of marks of all the points in the process (arguments `marcas` and `d`), or alternatively by a vector of `d` point processes (argument `vecpro`). If `marcas` or `d` are NULL, `vecpro` must be provided. If they are not NULL, they are used to define the marked Poisson process.

Value

prob The estimated transition matrix of probabilities.

See Also

DepNHPPMarked

Examples

```
TranM(marcas = c(1,3,3,2,1,2,1,1), d = 3)

TranM(vecpro=list(n1=c(2,7,9,23), n2=c(4,5,21), n3=c(2,8,9,12,16)))
```

TxBHZ *Daily maximum temperature at Barcelona, Huesca and Zaragoza*

Description

Daily maximum temperature series during the summer months (May, June, July, August and September) from 1951 to 2016 at three Spanish locations: Barcelona, Huesca and Zaragoza.

Usage

```
data(TxBHZ)
```

Details

Variables

year: Year, from 1951 to 2016.

month: Month, from 5 (May) to 9 (September).

day: Postion of the day in the month, from 1 up to 31.

dayyear: Postion of the day in the year, from 121 (1st of May) to 253 (30th of September).

TxB: Daily maximum temperature at Barcelona in Celsius degrees.

TxH: Daily maximum temperature at Huesca in Celsius degrees.

TxZ: Daily maximum temperature at Zaragoza in Celsius degrees.

Txm31B: Local maximum temperature signal in Celsius degrees. Moving average of TxB with a window of the last past 31 days.

Txm31H: Local maximum temperature signal in Celsius degrees. Moving average of TxH with a window of the last past 31 days.

Txm31Z: Local maximum temperature signal in Celsius degrees. Moving average of TxZ with a window of the last past 31 days.

lambdaOZ: estimated intensities of the first indicator processes of a CPSP fitted to model the occurrence times of the extreme events in the series of Zaragoza and Huesca, TxZ and TxH. The first indicator process includes the extreme events occurring only at TxZ.

lambdaOH: estimated intensities of the second indicator processes of a CPSP fitted to model the occurrence times of the extreme events in the series of Zaragoza and Huesca, TxZ and TxH. The second indicator process includes the extreme events occurring only at TxH.

lambdaZH: estimated intensities of the third indicator processes of a CPSP fitted to model the occurrence times of the extreme events in the series of Zaragoza and Huesca, TxZ and TxH. The third indicator process includes the simultaneous extreme events occurring both at TxZ and TxH.

Examples

```
data(TxBHZ)
```

`uniongentri` *Calculating the set of close points*

Description

This function calculates the set of close points of each occurence point in the first process of a vector of two or three processes.

Usage

```
uniongentri(posx, posy, posz=NULL,  info = FALSE, PA = FALSE,
 procName=c('X','Y','Z'),...)
```

Arguments

`posx`	Numeric vector. Position of the occurrence points in the first process.
`posy`	Numeric vector. Position of the occurrence points in the second process.
`posz`	Optional. Numeric vector. Position of the occurrence points in the third process. Only used when three processes are involved.
`info`	Optional. Logical flag. If it is TRUE, information about the generated points is shown on the screen and dotcharts and bivariate charts of the occurrence points in the processes are displayed.
`procName`	Vector of character strings. Names of the processes.
`PA`	Optional. Logical flag. If it is TRUE, the close point relation is broadened by including the previous and the following points to the overlapping intervals.
`...`	Further arguments to be passed to the function `plot` if `info=T`.

Details

A point in a process is close to a point in another process, if their time intervals overlap; the time interval of a point is the interval between itself and the previous point in the same process. If there are three processes, the set of close points of t_{x_k},$S_{x_i;xyz}$, is defined as the set of the pairs of points (t_{y_j}, t_{z_k}) such that t_{x_i} is close to t_{y_j} and t_{y_j} is close to t_{z_k}. If there are two processes, $S_{x_i;xy}$ is the set of points t_{y_j} such that t_{x_i} is close to t_{y_j}. This definition can be broadened, see argument `PA`, by adding to the set two more points, the previous and the following ones.

The algortihm to calculate the sets of close points (in the case of three processes) is the following, see Abaurrea et al. (2015) for details: First, given two processes, the pairs of close points in those processes are calculated. If the last point occurs in the first process, there is a censored time interval in the second process (the point overlaps a time interval whose occurrence point has not been observed) and that pair is not considered). This step is performed for all the combinations of pairs of processes. The basic close point relation is commutative, and only three different pairs (XY, YZ, XZ) must be considered. This is not the case of the broadened definition, where the six pairs (XY, YX, YZ, ZY, XZ, ZX) must be calculated.

Once all the pairs of close points are obtained, the set of close points for each point t_{x_i} is obtained by concatenating the adequate pairs of points from all the possible orders of the three processes: XYZ, XZY and YXZ for the basic definition, and the six possible permutations for the broadened definition. The final set of close points of t_{x_i} is the union of the different pairs from all the possible permutations.

Value

A list with elements:

`X`	First elements of the 3-tuples of points $(t_{x_i}, t_{y_i}, t_{z_i})$ in the sets of close points.
`iX`	Position i (=1,2,3....) of the point t_{x_i} in the first process.
`Y`	Second elements of the 3-tuples of points $(t_{x_i}, t_{y_i}, t_{z_i})$ in the sets of close points.
`iY`	Position i (=1,2,3....) of the point t_{y_i} in the second process.
`Z`	Third elements of the 3-tuples of points $(t_{x_i}, t_{y_i}, t_{z_i})$ in the sets of close points. It is NULL if `posz=NULL`.
`iZ`	Position i (=1,2,3....) of the point t_{z_i} in the third process. It is NULL if posz=NULL.

References

Abaurrea, J. Asin, J. and Cebrian, A.C. (2015). A Bootstrap Test of Independence Between Three Temporal Nonhomogeneous Poisson Processes and its Application to Heat Wave Modeling. *Environmental and Ecological Statistics*.

See Also

`TestIndNH`, `DistSim`, `DistObs`

Examples

```
set.seed(123)
posx<-sort(runif(20,0,1000))
posy<-sort(runif(25,0,1000))
posz<-sort(runif(40,0,1000))
aux<-uniongentri(posx, posy, posz, info=TRUE)
```

Index

∗ **Vectors of Point Processes in Time**
IndTestPP-package, 2

BinPer, 3, *8*, *13*
buscar *(*simNHPc*)*, 52

calcdist, *25*
calcdist *(*DistObs*)*, 25
calcNmu *(*CondTest*)*, 5
ComplPos, 4
CondTest, 5, *31*, *56*, *58*
cor.test, *7*
CountingCor, *3*, 7, *13*
countP *(*CountingCor*)*, 7
CPSPpoints, 8, *11*, *48*, *51*
CPSPPOTevents, 9, *9*

depchi, *3*, *8*, 12
DepCPSPKenv *(*DepCPSPNHK*)*, 13
DepCPSPNHK, 13
DepNHCPSP, *14*, 15, *15*, *19*, *20*, *22*, *35*
DepNHNeyScot, 17, *17*, *20*, *22*, *33*, *35*
DepNHPPMarked, *17*, 19, *19*, *22*, *35*, *54*, *60*
DepNHPPqueue, *17*, *19*, 20, *20*, *24*, *35*
DepNHPPqueue1 *(*DepNHPPqueue*)*, 20
DepNHPPqueueI *(*DepNHPPqueue*)*, 20
DepqueueKenv *(*DepqueueNHK*)*, 23
DepqueueNHK, 23
DistObs, 25, *30*, *58*, *62*
Distributions, *21*, *23*
DistShift, 26, *56*
DistSim, *26*, *27*, 28, *58*, *62*
DistSimfix *(*DistSim*)*, 28
dotchart, *27*, *29*
DutilleulPlot, *6*, 30, *37*, *56*, *58*

firstt *(*IntMPP*)*, 35
fn2 *(*TestIndNH*)*, 56
fn2B *(*TestIndLS*)*, 55
fn2fix *(*TestIndNH*)*, 56
fn3 *(*DutilleulPlot*)*, 30
funMPPGen *(*IntMPP*)*, 35

genbiPos *(*uniongentri*)*, 61
GenSons *(*DepNHNeyScot*)*, 17
gentriPos *(*uniongentri*)*, 61

HDFaux *(*NHD*)*, 37

IndNHNeyScot, *19*, *29*, *30*, 32, *34*
IndNHPP, *16*, *17*, *20*, *22*, *33*, 34, *52*, *53*
IndTestPP *(*IndTestPP-package*)*, 2
IndTestPP-package, 2
IntMPP, 35

Jenv *(*NHJ*)*, 41

Kenv *(*NHK*)*, 45

marca *(*CPSPpoints*)*, 8
miKS *(*TestIndNH*)*, 56
mirank *(*TestIndNH*)*, 56

nearestD *(*NHD*)*, 37
nearestdist, *31*, 36
NHD, *4*, 37, *41*, *44*, *47*
NHDaux *(*NHD*)*, 37
NHDFaux *(*NHJ*)*, 41
NHF, *4*, 39, *39*, *44*, *47*
NHFaux *(*NHF*)*, 39
NHJ, *4*, *6*, *39*, 41, *41*, *47*
NHJaux *(*NHJ*)*, 41
NHK, *4*, *6*, *14*, *15*, *24*, *39*, *41*, *44*, 45
NHKaux *(*NHK*)*, 45
NHKaux2 *(*NHK*)*, 45
NHKaux3 *(*NHK*)*, 45
nMenr *(*NHD*)*, 37
NumI *(*IntMPP*)*, 35

par, *8*, *10*, *16*, *42*, *49*
pdist *(*nearestdist*)*, 36

plot, *5*, *8*, *10*, *12*, *14*, *16*, *18, 19*, *21*, *24, 25*, *27*, *29, 30*, *32*, *34*, *38*, *40*, *42*, *46*, *48–50*, *61*
PlotICPSP, *9*, *11*, 48, *51*
PlotMargP, *48*, 49
PlotMCPSP, *9*, *11*, *48*, 50, *50*
prodN2 *(*NHD*)*, 37
PsimNHPc *(*DepNHCPSP*)*, 15

simHPc, 51, *51*, *53*
simNHPc, *29, 30*, *34*, *51*, 52, *52*
SpecGap, *20*, 54

TestIndLS, *27*, 55, *58*
TestIndNH, *6*, *26*, *29–31*, *55*, 56, *56*, *62*
TranM, 59
TxBHZ, 60

uniongentri, *25, 26*, *58*, 61