Package 'mppa'

February 20, 2015

Type Package

Title Statistics for analysing multiple simultaneous point processes on the real line

Version 1.0

Date 2014-08-16

Author Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk> and Nicholas A Heard <n.heard@imperial.ac.uk>

Maintainer Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk>

Description

A procedure to test for dependence between point processes on the real line, e.g. causal dependence, correlation, inhibition or anti-correlation. The package also provides a number of utilities for plotting simultaneous point processes, and combining p-values.

License GPL (>= 2)

Depends R (>= 2.2.0), methods, stats

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-23 08:06:06

R topics documented:

mppa-package *Statistics for analysing multiple simultaneous point processes on the real line*

Description

A procedure to test for dependence between point processes on the real line, e.g. causal dependence, correlation, inhibition or anti-correlation. The package also provides a number of utilities for plotting simultaneous point processes, and combining p-values.

Details

Package:	mppa
Type:	Package
Version:	1.0
Date:	2014-08-16
License:	GPL (>= 2)
Depends:	methods

`corrtest`: testing for dependence between point processes (can be causal dependence, correlation, inhibition or anti-correlation).

`mproc`: a class for plotting multiple simultaneous point processes.

`TMT.test`, `F.test`, `simes.test`: methods for combining p-values.

Author(s)

Patrick Rubin-Delanchy <patrick.rubin-delanchy> and Nicholas Heard <n.heard@imperial.ac.uk>

Maintainer: Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk>

References

Patrick Rubin-Delanchy and Nicholas A Heard. “A test for dependence between two point processes on the real line”. arXiv:1408.3845.

Examples

```
A = runif(20)
B=runif(20)
##around ten B events are caused by A
B=c(B, sample(A, 10)+abs(rnorm(10, 0,.01))); B = B[B>0&B<1]
## Not run: plot(mp(list(A=A,B=B)))
corrtest(A,B)
```

corrtest *A test for dependence between two point processes on the real line.*

Description

Given two point processes, A and B, simultaneously observed on the real line, this function computes a p-value for B's dependence on A. The function can be parameterised to detect triggering ('A causes B'), correlation, inhibition or anti-correlation. Further details are available in the corresponding paper.

Usage

```
corrtest(A, B, start = 0, end = 1, F = NULL, alternative = "causes",
      method = "timeout", transform = FALSE, usebeforeA1 = FALSE,
             maxtau = NA, careful = TRUE)
```

Arguments

A	a vector of event times. We condition on A and test B given A.
B	a second vector of event times. B is tested conditional on A.
start	the start of the observation period. Must be smaller than all of the event times in A and in B.
end	the end of the observation period. Must be larger than all of the event times in A and in B.
F	an increasing function, giving the cumulative intensity of B under the null hypothesis. If F=NULL, the cumulative intensity is assumed to be $F(t) = t$ (a homogeneous Poisson process). See details.
alternative	can be "causes" (testing for A causing B), "inhibits" (testing for A inhibiting B), "correlation" or "anticorrelation".
method	can be "timeout", "simes" or "fisher". These are different ways of testing the cumulative response times. See details.
transform	if TRUE, F is first used to 'homogenise' A and B by time change. See details.
usebeforeA1	if TRUE, the event times before A[1] are assumed to follow the null model and included in the test. Ignored if alternative is "correlation" or "anticorrelation".
maxtau	if method="timeout" then the user can set maxtau to limit the range of dependence being tested for. For example, if alternative is "causes" and maxtau is .1, then only events in B within .1 to the right of an event in A can be triggered.
careful	if TRUE a number of error checks are made before starting the procedure.

Details

Under the null hypothesis the events of B are assumed to follow a non-homogeneous Poisson process. This has an intensity function $\lambda(t)$. The intensity is provided by the user via the input `F`. If `F` is `NULL` (the default), B follows a homogeneous Poisson process under the null hypothesis. Otherwise `F` is interpreted as $F(t) = \int_{\text{start}}^{t} \lambda(x)dx$. The tests are invariant to any affine transformation of the input function $F^*(t) = \alpha + \beta F(t)$. Note that `F=NULL` is equivalent to `F=function(t)t`, although the latter is not recommended because the algorithm is faster if it knows the null hypothesis is homogeneous. To choose `F` the users are invited to estimate the non-homogeneous intensity of B via their favourite method, or try `mkF` which uses R's in-built `density` estimation. Note that the non-homogeneous Poisson assumption is a bit stronger than needed, see corresponding paper.

The procedure computes so-called cumulative response times. These are a sequence of values between 0 and 1, one for each event time in B (or each B after `A[1]` in the case of causes/inhibition with `usebeforeA1=FALSE`), which should be ordered uniform variables under the null hypothesis but 'closer to zero' under the alternative.

To test the cumulative response times, the user has the choice between three methods: a timeout test (see `TMT.test`), Fisher's method (see `F.test`) and Simes's test (see `simes.test`).

By default A and B are analysed as given. If `transform=TRUE`, F is first used to transform A and B so that B is homogeneous, and then the transformed processes are tested against a homogeneous null hypothesis.

For further details see corresponding paper.

Value

An S3 object of class "htest" with slots

`p`	the p-value.
`T`	the test statistic.
`data.name`	the input processes.
`t`	the cumulative response times.
`TMTval`	if method="timeout", the largest cumulative response time estimated to have been affected by A, `NA` otherwise.
`TMTi`	if method="timeout", the index of the largest cumulative response time estimated to have been affected by `A`, `NA` otherwise.

Author(s)

Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk> and Nicholas A Heard <n.heard@imperial.ac.uk>

References

Patrick Rubin-Delanchy and Nicholas A Heard. "A test for dependence between two point processes on the real line". arXiv:1408.3845.

See Also

`mkF`, `TMT.test`, `F.test`, `simes.test`

Examples

```
A = runif(20)
Braw=runif(20)
##around ten B events are caused by A
B1=c(Braw, sample(A, 10)+abs(rnorm(10, 0,.01))); B1 = B1[B1>0&B1<1]
##about ten B events are correlated to A
B2=c(Braw, sample(A, 10)+rnorm(10, 0,.01)); B2 = B2[B2>0&B2<1]
##the ten closest B events to A are deleted (anticorrelation)
d=sapply(Braw, function(b) min(abs(b-A))); B3=Braw[-order(d)[1:10]]
##The ten B events that are closest to but after an A event are deleted (inhibition).
##(Adding an A event at zero to be sure there are 10.)
A=c(0,A); d=sapply(Braw, function(b) b-max(A[A<b])); B4=Braw[-order(d)[1:10]]

alternatives=c("causes", "correlation", "anticorrelation", "inhibits")
ps=c()
for (B in list(B1, B2, B3, B4)){
    for (alternative in alternatives){
        p=corrtest(A,B, alternative=alternative)$p
        ps = c(ps, p)
    }
}
M=matrix(ps, nrow=4, ncol=4)
colnames(M) = c("causal_data", "correlated_data", "anticorrelated_data", "inhibited_data")
rownames(M) = c("causes_test", "correlation_test", "anticorrelation_test", "inhibition_test")
##should (hopefully!) see low p-values on the diagonal:
M
## and high p-values for opposite data versus test combinations,
## e.g. testing for A inhibiting B when in fact A causes B:
M["inhibition_test", "causal_data"]

##Now for a non-homogeneous example. A and B have a common daily pattern:
##their intensity is a sinusoidal curve lambda(t) = 1+sin(2*pi*t)
start=0; end=365 #A year
##the cumulative intensity is
F=function(t){
    t%/%1+t%%1+(1-cos(2*pi*t%%1))/(2*pi)
}
##Dropping 365 A and B points according to F
A=sapply(runif(365), function(u){uniroot(function(x) F(x)/365-u, interval=c(0,365))$root})
##B is independent of A aside from common periodicity
B=sapply(runif(365), function(u){uniroot(function(x) F(x)/365-u, interval=c(0,365))$root})
##Bc also has some caused events
Bc=c(B, sample(A, 10)+abs(rnorm(10, 0,.001))); Bc = Bc[Bc>start&Bc<end]

##If we do not account for the common periodicity of A and B we get
##spuriously strong evidence for A causing B (using Fisher's method for speed):
corrtest(A,B, start=start, end=end, method="fisher")
##On the other hand with the correct F, the p-value is uniformly distributed:
corrtest(A,B, start=start, end=end, F=F, method="fisher")

##For reference, with the truly caused process Bc we get
corrtest(A,Bc, start=start, end=end, method="fisher")
```

```
corrtest(A,Bc, start=start, end=end, F=F, method="fisher")
```

`F.test` *Fisher's method for combining p-values*

Description

Compute the p-value for $-2\sum \log p_i$.

Usage

```
F.test(x, returnstat = FALSE)
```

Arguments

`x`	a vector of p-values.
`returnstat`	if TRUE then return the statistic as well as the p-value.

Value

if returnstat=TRUE then a pair (p,T) where p is the p-value and T is the test, otherwise just p.

Author(s)

Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk>

References

Mosteller, F. and Fisher, R. A. (1948). Questions and answers. The American Statistician, 2(5):pp. 30-31.

See Also

`TMT.test`, `simes.test`

Examples

```
## This should be uniformly distributed
F.test(runif(10))
```

mkF *Estimation of the cumulative intensity of a process using native* density

Description

A function that estimates the density of the points and then creates a valid cumulative intensity function F for input to corrtest.

Usage

```
mkF(x, start = 0, end = 1, adjust = 1, disallow.zero = TRUE)
```

Arguments

x	a vector of event times.
start	the start of the observation period: must be smaller than all elements of x.
end	the end of the observation period: must be greater than all elements of x.
adjust	parameter passed on to density.
disallow.zero	if TRUE do not allow F to be constant. (Useful to avoid spurious results when using corrtest.)

Value

A non-decreasing function that can serve as input to corrtest.

Author(s)

Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk>

References

Patrick Rubin-Delanchy and Nicholas A Heard. "A test for dependence between two point processes on the real line". arXiv:1408.3845.

See Also

corrtest

Examples

```
start=0; end=1
A=rbeta(100, 1, 10)
B=rbeta(100, 1, 10)
##This will be extreme because A and B are not homogeneous over [0,1]
corrtest(A,B)
##If we use an estimate of F the p-value is less aggressive
```

```
corrtest(A,B,F=mkF(c(A,B), start=0, end=1))
##But we can still find evidence of A causing B:
Bc=c(B, sample(A, 10)+abs(rnorm(10, 0,.0001))); Bc = Bc[Bc>start&Bc<end]
corrtest(A,Bc,F=mkF(c(A,Bc), start=0, end=1))
```

---

mp *Create objects of class* mproc

Description

A function to create objects of class mproc, allowing manipulation of multiple simultaneous point processes.

Usage

```
mp(..., start = numeric(0), end = numeric(0))
```

Arguments

...	either a list of vectors of event times, or the same vectors given as a sequence of arguments.
start	the start of observations, must be smaller than all event times. If not specified the start will be set to the smallest event time.
end	the end of observations, must be greater than all event times. If not specified the end will be set to the greatest event time.

Value

An object of class mproc with slots:

events	a list of vectors of event times.
start	the start of observations.
end	the end of observations.

Author(s)

Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk>

Examples

```
mp(runif(10))
mp(runif(10), runif(10), start=0, end=1)
mp(list(runif(10), runif(10)), start=0, end=1)
```

mproc-class *Class* mproc

Description

Manipulation of sets of one-dimensional processes - Internal.

Objects from the Class

Objects can be created by calls of the form `new("mproc", ...)`, or use `mp` (recommended).

Slots

events: Object of class `list`.

start: Object of class `numeric`.

end: Object of class `numeric`.

Methods

plot `signature(x = "mproc", y = "missing")`: ...

Author(s)

Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk>

See Also

`mp`

Examples

```
showClass("mproc")
```

place *Function to place a point on a period-by-period plot generated by* `plot,mproc-method`

Description

Places a point at a specified event time on a period-by-period plot generated by `plot,mproc-method`.

Usage

```
place(e, m, period, ...)
```

Arguments

e	the event time - must be within the start and end times of m.
m	an object of class mproc.
period	the period of the period-by-period plot.
...	further arguments passed to points.

Author(s)

Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk>

See Also

plot,mproc-method, mp

Examples

```
#### A has a daily pattern: its intensity is a sinusoidal curve lambda(t) = 1+sin(2*pi*t)
start=0; end=365 #A year
##the cumulative intensity is
F=function(t){
    t%/%1+t%%1+(1-cos(2*pi*t%%1))/(2*pi)
}
##Dropping 365 A and B points according to F
A=sapply(runif(365), function(u){uniroot(function(x) F(x)/365-u, interval=c(0,365))$root})
m=mp(A,start=start, end=end)
##This plots A in a period by period plot
## Not run: plot(m, period=1)
##The middle of the day in the middle of the year:
x=365/2
## Not run: place(x,m, period=1, col="red", pch=8, cex=10)
```

---

plot-methods *Methods for function* plot *in package 'mproc'*

Description

Plot objects of class mproc.

Usage

```
## S4 method for signature 'mproc'
plot(x, period=NA, id=1, mks=NULL, palette=rainbow, cols=NULL, xlab="Time", ylab="",...)
```

Arguments

x	an object of class mproc.
period	if given, will wrap the data into periods, and plot each period on a horizontal line. See example.
id	the vector id to be plotted in periods. Ignored if period is not given.
mks	a vector or a list of vectors of marks. Points with the same mark will be shown in the same colour.
palette	a colour palette. Ignored if mks is NULL or cols is given.
cols	a vector or a list of vectors of colours for the points.
xlab	the label for the x-axis.
ylab	the label for the y-axis.
...	further arguments passed on to plot.

Methods

signature(x = "mproc")

Author(s)

Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk>

Examples

```
A = c(.1, .2, .3)
B = c(.1, .2, .3)
##Giving the last point in B a different colour from the rest
## Not run: plot(mp(list(A=A,B=B)), mks=list(c(1,1,1), c(1,1,2)))

#### A and B have a common daily pattern:
### their intensity is a sinusoidal curve lambda(t) = 1+sin(2*pi*t)
start=0; end=365 #A year
##the cumulative intensity is
F=function(t){
    t%/%1+t%%1+(1-cos(2*pi*t%%1))/(2*pi)
}
##Dropping 365 A and B points according to F
A=sapply(runif(365), function(u){uniroot(function(x) F(x)/365-u, interval=c(0,365))$root})
B=sapply(runif(365), function(u){uniroot(function(x) F(x)/365-u, interval=c(0,365))$root})
##This plots A and B
## Not run: plot(mp(A,B, start=start, end=end), pch=".")

##This plots B, one day represented on each horizontal line
## Not run: plot(mp(A,B, start=start, end=end), period=1, id=2)

##Same as above, but now colouring the seventh day of every week differently:
## Not run: plot(mp(A,B, start=start, end=end), period=1, id=2,mks=round(B%%7)==6)
##Now A and B are in the same plot, different colour
lA=length(A); lB=length(B);
## Not run: plot(mp(c(A,B),start=start,end=end),period=1,mks=c(rep(1,lA), rep(2,lB)))
```

`simes.test` *Simes's method for combining p-values*

Description

Computes the combined p-value $\min(np_{(i)}/i)$ where $p_{(1)}, \ldots, p_{(n)}$ are the ordered p-values.

Usage

```
simes.test(x, returnstat = FALSE)
```

Arguments

`x`	a vector of p-values.
`returnstat`	if TRUE then return the statistic as well as the p-value. (Very unusually, they are the same with this method.)

Value

if returnstat=TRUE then a pair (p,T) where p is the p-value and T is the test, otherwise just p.

Author(s)

Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk>

References

Simes, R. J. (1986). An improved Bonferroni procedure for multiple tests of significance. Biometrika, 73(3):751-754.

See Also

`F.test`, `TMT.test`

Examples

```
## This should be uniformly distributed
simes.test(runif(10))
```

TMT.test *A method for combining p-values based on a changepoint model*

Description

Under the null hypothesis that $p_1, \ldots, p_n$ are independent and uniformly distributed on $[0, 1]$, this function computes the p-value for the maximum likelihood of the changepoint model $d(p) \propto \lambda_1$ if $p \leq \tau$ and $d(p) \propto \lambda_2$ otherwise, where d is a piecewise constant density with one changepoint, with unknown parameters $\lambda_1 \geq \lambda_2$ and $\tau \in [0, 1]$. The null hypothesis is $\lambda_1 = \lambda_2$.

Usage

```
TMT.test(x, method = "AUTO", maxtau = 1, samples = 1000)
```

Arguments

x	a vector of p-values.
method	if "AUTO" then the computation method will be chosen automatically. If "NOE", the p-value is computed exactly (effort is order n^2) by Noe's recursion. If "SIM", then the p-value is determined by simulation.
maxtau	restrict the changepoint to a maximum maxtau. Useful if we are only interested in testing small p-values.
samples	number of samples used for simulation. Ignored if method is "NOE" or method is "AUTO" with $n < 1000$.

Details

The p-value for the maximum likelihood can be restated as the solution to

$$P[u_1 \geq o_1, \ldots, u_n \geq o_n],$$

where $o_1, \ldots, o_n$ are a sequence determined from the maximum likelihood and $u_1, \ldots, u_n$ are ordered uniform random variables. Computing this probability is harder than it looks, because simple analytical recursions fail due to catastrophic cancellation. We have implemented Noe's recursions, which are safe but expensive for large n, so we recommend simulation for $n \geq 1000$. This will be done automatically if method="AUTO".

Value

lk	the maximum likelihood.
TMTi	the rank of the largest p-value to the left of the estimated changepoint (the rank of the largest p-value in the group of 'small' p-values).
TMTu	the largest p-value to the left of the changepoint (the largest p-value in the group of 'small' p-values).
p	the p-value of the test. If maxtau was set, and no inputs were lower than maxtau, the p-value returned is 1.

Author(s)

Patrick Rubin-Delanchy <patrick.rubin-delanchy@bristol.ac.uk>

References

Patrick Rubin-Delanchy and Nicholas A Heard. "A test for dependence between two point processes on the real line". arXiv:1408.3845.

Noe, M. (1972). The calculation of distributions of two-sided Kolmogorov-Smirnov type statistics. The Annals of Mathematical Statistics, pages 58-64.

Noe, M. and Vandewiele, G. (1968). The calculation of distributions of Kolmogorov-Smirnov type statistics including a table of significance points for a particular case. The Annals of Mathematical Statistics, 39(1):233-241.

See Also

F.test, simes.test

Examples

```
## This should be uniformly distributed
TMT.test(runif(10))$p
## Whenever no p-values fall inside [0,maxtau] the returned p-value is one
replicate(20,TMT.test(runif(10), maxtau=.1)$p)
##Use maxtau to gain extra detection power if only interested in a
##subset of very low p-values, e.g.
TMT.test(c(.04, .5))$p
##is larger than
TMT.test(c(.04, .5), maxtau=0.05)$p
```

Index