Package 'stppResid'

February 20, 2015

Type Package

Title Perform residual analysis on space-time point process models.

Version 1.1

Date 2011-02-14

Author Robert Clements

Maintainer Robert Clements <rclements17@gmail.com>

Description Implement transformation-based and pixel-based residual analysis of spatial-temporal point process models.

License GPL (>= 2)

LazyLoad yes

Depends deldir, splancs, cubature

Repository CRAN

Date/Publication 2012-11-20 13:13:46

NeedsCompilation no

R topics documented:

`add.stpoints` *Add space-time points to a stpp object*

Description

This function adds new points to a `stpp` object.

Usage

```
add.stpoints(X, points)
```

Arguments

X	A space-time object of class "stpp" to add points to.
points	A vector, data frame, matrix or list of points to be added to X.

Details

points can be a vector of length three describing one x, y, and t coordinate, or a matrix or data frame where the first column is a column of x coordinates, the second column is a column of y coordinates, and the third column is a column of t coordinates. If points is a list, the first entry is a vector of x coordinates, the second entry is a vector of y coordinates, and the third entry is a vector of t coordinates.

Value

add.stpoints returns an object of class "stpp".

Author(s)

Robert Clements

See Also

stpp

Examples

```
#===> create a stpp object <===#
x <- rnorm(30, mean = 10, sd = 1)
y <- rnorm(30, mean = 100, sd = 10)
t <- runif(30, 0, 100)
stw <- stwin(xcoord = c(0, 20), ycoord = c(50, 150), tcoord = c(0, 100))
X <- stpp(x, y, t, stw = stw)

#===> add new points to X <===#
x.new <- rnorm(10, mean = 10, sd = 1)
y.new <- rnorm(10, mean = 100, sd = 10)
t.new <- runif(10, 0, 100)
all.new <- data.frame(cbind(x.new, y.new, t.new))
Y <- add.stpoints(X, all.new)
```

---

deviance *Pre-computed set of deviance residuals*

Description

A "devresid" object.

Usage

```
data(deviance)
```

Format

A "devresid" object.

---

devresid *Calculate deviance residuals*

Description

devresid divides the space-time window into a grid of bins and calculates the deviance residuals within each bin between two competing conditional intensity models.

Usage

```
devresid(X, cifunction1, cifunction2, theta1 = NULL, theta2 = NULL,
lambda1 = NULL, lambda2 = NULL, grid = c(10, 10), gf = NULL, algthm1 =
c("cubature", "mc", "miser", "none"), algthm2 = c("cubature", "mc", "miser", "none"),
n = 100, n1.miser = 10000, n2.miser = 10000, tol = 1e-05, maxEval = 0,
absError = 0, ints1 = NULL, ints2 = NULL)
```

Arguments

X	A "stpp" object.
cifunction1	A function returning estimates of the conditional intensity at all points in X, according to model 1 (cifunction1). The function should take arguments X and an optional vector of parameters theta1.
cifunction2	A function returning estimates of the conditional intensity at all points in X, according to model 2 (cifunction2) which should be different than model 1 (cifunction1). The function should take arguments X and an optional vector of parameters theta2.
theta1	Optional: A vector of parameters to be passed to cifunction1.
theta2	Optional: A vector of parameters to be passed to cifunction2.
lambda1	Optional: A vector of conditional intensities based on cifunction1 at each point in X.
lambda2	Optional: A vector of conditional intensities based on cifunction2 at each point in X.
grid	A vector representing the number of columns and rows in the grid.
gf	Optional: A "stgrid" object.
algthm1	The algorithm used for estimating the integrals in each grid cell for model 1. The three algorithms are "cubature", "mc", "miser", and "none".

algthm2	The algorithm used for estimating the integrals in each grid cell for model 2. The three algorithms are "cubature", "mc", "miser", and "none".
n	Initial number of sample points in each grid cell for approximating integrals. The number of sample points are iteratively increased by n until some accuracy threshold is reached.
n1.miser	The total number of sample points for estimating all integrals for model 1 if the miser algorithm is selected.
n2.miser	The total number of sample points for estimating all integrals for model 2 if the miser algorithm is selected.
tol	The maximum tolerance.
maxEval	The maximum number of function evaluations needed (default 0 implies no limit).
absError	The maximum absolute error tolerated.
ints1	An optional vector of integrals for model 1. Must be the same length as the number of rows in grid, and each element of ints1 should correspond to each row in grid.
ints2	An optional vector of integrals for model 2. Must be the same length as the number of rows in grid, and each element of ints2 should correspond to each row in grid.

Details

The deviance residuals are the differences in the log-likelihoods of model 1 vs. model 2 within each space-time bin, denoted here as B_i (see Wong and Schoenberg (2010)). The deviance residual is given by

$$R_D(B_i) = \sum_{i:(x_i)\in B_i} log\hat{\lambda}_1(x_i) - \int_{B_i} \hat{\lambda}_1(x_i)dx - \left(\sum_{i:(x_i)\in B_i} log\hat{\lambda}_2(x_i) - \int_{B_i} \hat{\lambda}_2(x_i)dx \right),$$

where $\hat{\lambda}(x)$ is the fitted conditional intensity model.

The conditional intensity functions, cifunction1 and cifunction2, should take X as their first argument, and an optional theta as their second argument, and return a vector of conditional intensity estimates with length equal to the number of points in X, i.e. the length of X$x. Both cifunction1 and cifunction2 are required. lambda1 and lambda2 are optional, and if passed will eliminate the need for devresid to calculate the conditional intensities at each observed point in X.

The integrals in $R(B_i)$ are approximated using one of three algorithms: the adaptIntegrate function from the cubature pakcage, a simple Monte Carlo (mc) algorithm, or the miser algorithm. The default is cubature and should be the fastest approximation. The approximation continues until either the maximum number of evaluations is reached, the error is less than the absolute error, or is less than the tolerance times the integral.

The simple Monte Carlo iteratively adds n sample points to each grid cell to approximate the integral, and the iteration stops when some threshold in the accuracy of the approximation is reached. The MISER algorithm samples a total number of n1.miser and/or n2.miser points in a recursive way, sampling the points in locations that have the highest variance. This part can be very slow

and the approximations can be very inaccurate. For highest accuracy these algorithms will require a very large n or n1.miser/n2.miser depending on the complexity of the conditional intensity functions (some might say ~1 billion sample points are needed for a good approximation). Passing the argument ints1 and/or ints2 eliminates the need for approximating the integrals using either of these two algorithms.

Passing gf will eliminate the need for devresid to create a "stgrid" object. If neither grid or gf is specified, the default grid is 10 by 10.

Value

Prints to the screen the number of simulated points used to approximate the integrals.

Outputs an object of class "devresid", which is a list of

X	An object of class "stpp".
grid	An object of class "stgrid".
residuals	A vector of deviance residuals. The order of the residuals corresponds with the order of the bins in grid.

The following elements are named by model number, e.g. n.1, n.2, integral.1, integral.2, etc..

n	Total number of points used for approximating all integrals.
integral	Vector of actual integral approximations in each grid cell.
mean.lambda	Vector of the approximate final mean of lambda in each grid cell.
sd.lambda	Vector of the approximate standard deviation of lambda in each grid cell.

If the miser algorithm is selected, the following is also returned:

app.pts	A data frame of the x,y, and t coordinates of a sample of 10,000 of the sampled points for integral approximation, along with the value of lambda (l).

Author(s)

Robert Clements

References

Wong, K., Schoenberg, F.P. "On mainshock focal mechanisms and the spatial distribution of aftershocks", Bulletin of the Seismological Society of America, In review.

Clements, R.A., Schoenberg, F.P., and Schorlemmer, D. (2011) Residual analysis methods for space-time point processes with applications to earthquake forecast models in California. *Annals of Applied Statistics,* **5**, Number 4, 2549–2571.

See Also

make.grid

Examples

```
#===> load simulated data <===#
data(simdata)
X <- stpp(simdata$x, simdata$y, simdata$t)

#===> define two conditional intensity functions <===#
ci1 <- function(X, theta){theta*exp(-2*X$x - 2*X$y - 2*X$t)} #correct model

ci2 <- function(X, theta = NULL){rep(250, length(X$x))} #homogeneous Poisson model

## Not run:
deviance <- devresid(X, ci1, ci2, theta1 = 3000)

#===> plot results <===#
plot(deviance)

## End(Not run)
```

---

earthquake *Earthquake locations*

Description

Longitude, latitude and times (starting from zero) of a set of earthquakes in and around California.

Usage

```
data(earthquake)
```

Format

A dataframe with 3 columns.

---

eq *Earthquake dataset as "stpp" object*

Description

Locations and times of earthquakes of magnitude 3.5 and up, from 1/1/2002 - 12/10/2010.

Usage

```
data(eq)
```

Format

A "stpp" object

Details

Times are measured in minutes from first earthquake in the dataset.

Source

http://www.ncedc.org/anss/catalog-search.html

Examples

```
data(eq)
```

---

gresiduals *Pre-computed grid-based residuals*

Description

A "gridresid" object.

Usage

```
data(gresiduals)
```

Format

A "gridresid" object.

---

gridresid *Calculate grid-based residuals*

Description

gridresid divides the space-time window into a grid of bins and calculates residuals within each bin for a specified conditional intensity model.

Usage

```
gridresid(X, cifunction, theta = NULL, lambda = NULL, grid = c(10, 10), gf = NULL, resid = c("raw", "pe
```

Arguments

X	A "stpp" object.
cifunction	A function returning the value of the conditional intensity at all points in X. The function should take arguments X and an optional vector of parameters theta.
theta	Optional: A vector of parameters to be passed to cifunction.
lambda	Optional: A vector of conditional intensities at each point in X.
grid	A vector representing the number of columns and rows in the grid.
gf	Optional: A "stgrid" object.
resid	The residual type to be computed. The three types are "raw", "pearson", and "inverse".
algthm	The algorithm used for estimating the integrals in each grid cell. The three algorithms are "cubature", "mc", "miser", and "none"
n	Initial number of sample points in each grid cell for approximating integrals. The number of sample points are iteratively increased by n until some accuracy threshold is reached.
n.miser	The total number of sample points for estimating all integrals.
tol	The maximum tolerance.
maxEval	The maximum number of function evaluations needed (default 0 implies no limit).
absError	The maximum absolute error tolerated.
ints	An optional vector of integrals. Must be the same length as the number of rows in grid, and each element of ints should correspond to each row in grid.

Details

The grid-based residuals are well known residuals for temporal point processes, and purely spatial point processes (see Baddeley et al. (2005)), extended to space-time point processes in Clements et al. (2010). They consist of the raw residual, and rescaled versions of the raw residual called the pearson residual and the inverse residual.

The raw residual for bin i (B_i) is defined as the number of points in B_i minus the expected number of points in B_i,

$$R(B_i) = N(B_i) - \int_{B_i} \hat{\lambda}(x)dx,$$

where $\hat{\lambda}(x)$ is the fitted conditional intesity model.

The pearson residual is defined as

$$R_p(B_i) = \sum_{x_i \in B_i} 1/\sqrt{\hat{\lambda}(x_i)} - \int_{B_i} \sqrt{\hat{\lambda}(x)}dx.$$

The inverse residual is defined as

$$R_I(B_i) = \sum_{x_i \in B_i} 1/\hat{\lambda}(x_i) - \int_{B_i} I(\hat{\lambda}(x) > 0) dx.$$

If neither type of residual is specified, the default residual to be computed is the `raw` residual.

The conditional intensity function, `cifunction`, should take `X` as the first argument, and an optional `theta` as the second argument, and return a vector of conditional intensity estimates with length equal to the number of points in `X`, i.e. the length of `X$x`. `cifunction` is required, while `lambda` is optional. `lambda` eliminates the need for `gridresid` to calculate the conditional intensity at each observed point in `X`.

The integrals in $R(B_i)$ are approximated using one of three algorithms: the `adaptIntegrate` function from the `cubature` pakcage, a simple Monte Carlo (`mc`) algorithm, or the `miser` algorithm. The default is `cubature` and should be the fastest approximation. The approximation continues until either the maximum number of evaluations is reached, the error is less than the absolute error, or is less than the tolerance times the integral.

The simple Monte Carlo iteratively adds `n` sample points to each grid cell to approximate the integral, and the iteration stops when some threshold in the accuracy of the approximation is reached. The MISER algorithm samples a total number of `n.miser` points in a recursive way, sampling the points in locations that have the highest variance. This part can be very slow and the approximations can be very inaccurate. For highest accuracy these algorithms will require a very large `n` or `n.miser` depending on the complexity of the conditional intensity functions (some might say ~1 billion sample points are needed for a good approximation). Passing the argument `ints` eliminates the need for approximating the integrals using either of these two algorithms.

Passing `gf` will eliminate the need for `gridresid` to create a "`stgrid`" object. If neither `grid` or `gf` is specified, the default `grid` is 10 by 10.

Value

Prints to the screen the number of simulated points in each bin used to approximate the integrals.

Outputs an object of class "`gridresid`", which is a list of

`X`	An object of class "`stpp`".
`grid`	An object of class "`stgrid`".
`residuals`	A vector of grid-based residuals. The order of the residuals corresponds with the order of the bins in `grid`.
`n`	Total number of points used for approximating all integrals.
`integral`	Vector of actual integral approximations in each grid cell.
`mean.lambda`	Vector of the approximate final mean of lambda in each grid cell.
`sd.lambda`	Vector of the approximate standard deviation of lambda in each grid cell.

If the `miser` algorithm is selected, the following is also returned:

`app.pts`	A data frame of the x,y, and t coordinates of a sample of 10,000 of the sampled points for integral approximation, along with the value of lambda (l).

Author(s)

Robert Clements

References

Baddeley, A., Turner, R.,Moller, J., and Hazelton, M. (2005) Residual analysis for spatial point processes. *Journal of the Royal Statistical Society,* **67**, 617–666.

Clements, R.A., Schoenberg, F.P., and Schorlemmer, D. (2011) Residual analysis methods for space-time point processes with applications to earthquake forecast models in California. *Annals of Applied Statistics,* **5**, Number 4, 2549–2571.

See Also

make.grid

Examples

```
#===> load simulated data <===#
data(simdata)
X <- stpp(simdata$x, simdata$y, simdata$t)

#===> define two conditional intensity functions <===#
ci1 <- function(X, theta){theta*exp(-2*X$x - 2*X$y - 2*X$t)} #correct model

ci2 <- function(X, theta = NULL){rep(250, length(X$x))} #homogeneous Poisson model

## Not run:
gresiduals <- gridresid(X, ci1, theta = 3000)
plot(gresiduals)

## End(Not run)
gresiduals2 <- gridresid(X, ci2)
plot(gresiduals2)
```

---

make.grid *Make a grid of spatial bins*

Description

make.grid creates a space-time grid object of class "stgrid" for use with the functions gridresid and devresid.

Usage

```
make.grid(stw, grid = c(10, 10))
```

Arguments

stw	A space-time window object of class "stwin" to be divided into space-time bins.
grid	A vector representing the number of rows and columns in the grid.

Details

The space-time window, stw, is divided into a grid of space-time bins using the grid argument to determine the number of intervals in the x and y directions. The total number of bins will be the product of the elements of grid.

If grid is not specified, the default grid is 10 x 10, resulting in 100 total bins.

Value

Outputs an object of class "stgrid", which is a list of

grid.full	A data frame where each row represents a bin. There are four columns, xmin, xmax, ymin, and ymax, which represent the x and y limits of the bins.

Author(s)

Robert Clements

See Also

stwin, gridresid

Examples

```
#===> create a space-time window  <===#
xcoord <- c(0, 10)
ycoord <- c(0, 20)
tcoord <- c(0, 100)
stw <- stwin(xcoord, ycoord, tcoord)

#===> create a 10 x 10 grid <===#
gf <- make.grid(stw)

#===> create a 10 x 20 grid <===#
gf <- make.grid(stw, c(10, 20))
nrow(gf$grid.full)
ncol(gf$grid.full)
```

`plot.devresid` *Plot deviance residuals*

Description

Plot an image of deviance residuals for a space-time point process.

Usage

```
## S3 method for class 'devresid'
plot(x, ..., col.key = rev(heat.colors(100)), cutoffs = NULL)
```

Arguments

`x`	A "`devresid`" object.
`...`	Arguments for use with `points`.
`col.key`	A vector of colors in hexadecimal format.
`cutoffs`	A vector of cut points for assigning the colors in `col.key` to the residuals in `X`. `cutoffs` should be a vector of length one more than the length of `col.key`.

Details

`cutoffs` must be a vector of increasing values of the same length as `col.key` plus 1. `cutoffs` divides the residual values in `x$residuals` into a number of intervals equal to the number of colors in `col.key`. The colors are assigned to the intervals in order, e.g. the first color in `col.key` will be plotted in the bins defined by the spatial grid in `x$grid` that contains a residual that falls anywhere in the first interval (lower bound inclusive, upper bound exclusive).

Default `col.key` is a vector of 100 `heat` colors in reverse. Default `cutoffs` is a vector of 101 equally spaced points that range from the minimum residual in `x$residuals`, minus a very small number, to the maximum residual, plus a very small number.

Note

The default `col.key` and `cutoffs` may not be useful if the residuals are highly skewed. In this case, there should be more values in `cutoffs` where the residuals are most dense.

These are simply default plots for quick illustration of the residuals, and may or may not be useful for detailed analysis of the residuals.

Author(s)

Robert Clements

See Also

`devresid`, `image`

Examples

```
data(deviance)
plot(deviance)

hist(deviance$residuals)
cutoffs <- c(seq(-1.55, 2, length.out = 85), seq(2.01, 8.7, length.out = 16))
plot(deviance, cutoffs = cutoffs)
```

`plot.gridresid` *Plot grid-based residuals*

Description

`plot.gridresid` is a generic function for plotting grid-based residuals.

Usage

```
## S3 method for class 'gridresid'
plot(x, ..., col.key = rev(heat.colors(100)), cutoffs = NULL)
```

Arguments

`x`	A "`gridresid`" object.
`...`	Arguments for use with `points`.
`col.key`	A vector of colors in hexadecimal format.
`cutoffs`	A vector of cut points for assigning the colors in `col.key` to the residuals in `X`. `cutoffs` should be a vector of length one more than the length of `col.key`.

Details

`cutoffs` must be a vector of increasing values of the same length as `col.key` plus 1. `cutoffs` divides the residual values in `x$residuals` into a number of intervals equal to the number of colors in `col.key`. The colors are assigned to the intervals in order, e.g. the first color in `col.key` will be plotted in the bins defined by the spatial grid in `x$grid` that contains a residual that falls anywhere in the first interval (lower bound inclusive, upper bound exclusive).

Default `col.key` is a vector of 100 `heat` colors in reverse. Default `cutoffs` is a vector of 101 equally spaced points that range from the minimum residual in `x$residuals`, minus a very small number, to the maximum residual, plus a very small number.

Note

The default `col.key` and `cutoffs` may not be useful if the residuals are highly skewed. In this case, there should be more values in `cutoffs` where the residuals are most dense.

These are simply default plots for quick illustration of the residuals, and may or may not be useful for detailed analysis of the residuals.

Author(s)

Robert Clements

See Also

gridresid, image

Examples

```
data(gresiduals)
plot(gresiduals)

hist(gresiduals$residuals)
cutoffs = c(seq(-2.74, -1.01, length.out = 15),
seq(-1, 1, length.out = 70), seq(1, 3.83, length.out = 16))
plot(gresiduals, cutoffs = cutoffs)
```

---

plot.stpp *Plot a space-time point pattern*

Description

Plot a space-time point pattern described by a "stpp" object.

Usage

```
## S3 method for class 'stpp'
plot(x, ..., pch = 1, asp = 1)
```

Arguments

x	A "stpp" object.
...	Arguments for use with plot.
pch	Character type for plotting the points.
asp	y/x aspect ratio.

Author(s)

Robert Clements

See Also

stpp, stwin

Examples

```
data(eq)
plot(eq)
```

`plot.superthin` *Plot super-thinned residuals*

Description

Plot super-thinned residuals for a space-time point process.

Usage

```
## S3 method for class 'superthin'
plot(x, ..., pch1 = 1, pch2 = 3, asp = 1)
```

Arguments

`x`	A "`superthin`" object.
`...`	Arguments for use with `plot`.
`pch1`	Character type for plotting the retained points from the original data set.
`pch2`	Character type for plotting the simulated points.
`asp`	y/x aspect ratio.

Details

Plots the super-thinned residuals, where the original points of the point pattern can be plotted as a separate symbol than the superposed (simulated) points.

Note

This function does not plot the time component from the super-thinned residuals, only the spatial coordinates.

Author(s)

Robert Clements

See Also

`superthin`

Examples

```
data(stresiduals1)
plot(stresiduals1, pch1 = 3, pch2 = 5)
```

`plot.supresid` *Plot superposed residuals*

Description

Plot superposed residuals for a space-time point process.

Usage

```
## S3 method for class 'supresid'
plot(x, ..., pch1 = 1, pch2 = 3, asp = 1)
```

Arguments

`x`	A "`supresid`" object.
`...`	Arguments for use with `plot`.
`pch1`	Character type for plotting the original data points.
`pch2`	Character type for plotting the simulated points.
`asp`	y/x aspect ratio.

Details

Plots the superposed residuals, where the original points of the point pattern can be plotted as a separate symbol than the superposed (simulated) points.

Author(s)

Robert Clements

See Also

`supresid`

Examples

```
data(sresiduals1)
plot(sresiduals1, pch1 = 2)

data(sresiduals2)
plot(sresiduals2)
```

`plot.tessdev` *Plot tessellation deviance residuals*

Description

Plot an image of tessellation deviance residuals for a space-time point process.

Usage

```
## S3 method for class 'tessdev'
plot(x, ..., col.key = rev(heat.colors(100)), cutoffs = NULL)
```

Arguments

`x`	A "`tessdev`" object.
`...`	Arguments for use with `points`
`col.key`	A vector of colors in hexadecimal format.
`cutoffs`	A vector of cut points for assigning the colors in `col.key` to the residuals in `X`. `cutoffs` should be a vector of length one more than the length of `col.key`.

Details

`cutoffs` must be a vector of increasing values of the same length as `col.key` plus 1. `cutoffs` divides the residual values in `x$residuals` into a number of intervals equal to the number of colors in `col.key`. The colors are assigned to the intervals in order, e.g. the first color in `col.key` will be plotted in the cells defined by the tessellation in `x$tile.list` that contains a residual that falls anywhere in the first interval (lower bound inclusive, upper bound exclusive).

Default `col.key` is a vector of 100 `heat` colors in reverse. Default `cutoffs` is a vector of 101 equally spaced points that range from the minimum residual in `x$residuals`, minus a very small number, to the maximum residual, plus a very small number.

Note

The default `col.key` and `cutoffs` may not be useful if the residuals are highly skewed. In this case, there should be more values in `cutoffs` where the residuals are most dense.

These are simply default plots for quick illustration of the residuals, and may or may not be useful for detailed analysis of the residuals.

Author(s)

Robert Clements

See Also

`tessresid`, `tessdev`

Examples

```
#For example, see ?plot.devresid
```

`plot.tessresid` *Plot tessellation residuals*

Description

Plot an image of tessellation residuals for a space-time point process.

Usage

```
## S3 method for class 'tessresid'
plot(x, ..., col.key = rev(heat.colors(100)), cutoffs = NULL)
```

Arguments

`x`	A "`tessresid`" object.
`...`	Arguments for use with `points`
`col.key`	A vector of colors in hexadecimal format.
`cutoffs`	A vector of cut points for assigning the colors in `col.key` to the residuals in `X`. `cutoffs` should be a vector of length one more than the length of `col.key`.

Details

`cutoffs` must be a vector of increasing values of the same length as `col.key` plus 1. `cutoffs` divides the residual values in `x$residuals` into a number of intervals equal to the number of colors in `col.key`. The colors are assigned to the intervals in order, e.g. the first color in `col.key` will be plotted in the cells defined by the tessellation in `x$tile.list` that contains a residual that falls anywhere in the first interval (lower bound inclusive, upper bound exclusive).

Default `col.key` is a vector of 100 `heat` colors in reverse. Default `cutoffs` is a vector of 101 equally spaced points that range from the minimum residual in `x$residuals`, minus a very small number, to the maximum residual, plus a very small number.

Note

The default `col.key` and `cutoffs` may not be useful if the residuals are highly skewed. In this case, there should be more values in `cutoffs` where the residuals are most dense.

These are simply default plots for quick illustration of the residuals, and may or may not be useful for detailed analysis of the residuals.

Author(s)

Robert Clements

See Also

tessresid

Examples

```
data(tsresiduals)
plot(tsresiduals)

hist(tsresiduals$residuals)
cutoffs = c(seq(-1.07, -.51, length.out = 15), seq(-.5, .5, length.out=70),
seq(.51, 2.42, length.out = 16))
plot(tsresiduals, cutoffs = cutoffs)
```

---

plot.thinresid *Plot thinned residuals*

Description

Plot thinned residuals for a space-time point process.

Usage

```
## S3 method for class 'thinresid'
plot(x, ..., pch = 1, asp = 1)
```

Arguments

x	A "thinresid" object.
...	Arguments for use with plot.
pch	Character type for plotting the points.
asp	y/x aspect ratio.

Author(s)

Robert Clements

See Also

thinresid

Examples

```
data(tresiduals1)
plot(tresiduals1)

data(tresiduals2)
plot(tresiduals2)
```

print.devresid *Print details of a* devresid *object*

Description

Prints the details of the results of finding deviance residuals for a space-time point process.

Usage

```
## S3 method for class 'devresid'
print(x, ...)
```

Arguments

x	A "devresid" object.
...	

Details

Prints the details of the space-time dataset, the deviance residuals, and the spatial grid.

Author(s)

Robert Clements

See Also

devresid

Examples

```
data(deviance)
deviance
```

print.gridresid *Print details of a* gridresid *object*

Description

Prints the details of the results of finding grid-based residuals for a space-time point process.

Usage

```
## S3 method for class 'gridresid'
print(x, ...)
```

Arguments

x	A "gridresid" object.
...	

Details

Prints the details of the space-time dataset, the grid-based residuals, and the spatial grid.

Author(s)

Robert Clements

See Also

gridresid

Examples

```
data(gresiduals)
gresiduals
```

---

print.stgrid *Print details of a* stgrid *object*

Description

Prints a full spatial grid.

Usage

```
## S3 method for class 'stgrid'
print(x, ...)
```

Arguments

x	A "stgrid" object.
...	

Author(s)

Robert Clements

See Also

make.grid

Examples

```
data(gresiduals)
gresiduals$grid
```

`print.stpp` *Print details of a* `stpp` *object*

Description

Prints the x, y, and t coordinates of a space-time dataset, and prints the limits of the space-time window.

Usage

```
## S3 method for class 'stpp'
print(x, ...)
```

Arguments

`x`	A "`stpp`" object.
`...`	

Author(s)

Robert Clements

See Also

`stpp`

Examples

```
data(eq)
eq
```

print.stwin *Print details of a* stwin *object*

Description

Prints the details of a space-time window.

Usage

```
## S3 method for class 'stwin'
print(x, ...)
```

Arguments

x	A "stwin" object.
...	

Author(s)

Robert Clements

See Also

stwin

Examples

```
stw <- stwin(c(0,2), c(0,5), c(0,10))
stw
```

print.superthin *Print details of a* superthin *object*

Description

Prints the details of the results of finding super-thinned residuals for a space-time point process.

Usage

```
## S3 method for class 'superthin'
print(x, ...)
```

Arguments

x	A "superthin" object.
...	

Details

Prints the details of the space-time dataset, the super-thinning rate, the super-thinned residuals, the simulated points, the automatically retained points, the retained points after thinning, and the removed points.

Author(s)

Robert Clements

See Also

superthin

Examples

```
data(stresiduals1)
stresiduals1
```

`print.supresid` *Print details of a* `supresid` *object*

Description

Prints the details of the results of finding superposed residuals for a space-time point process.

Usage

```
## S3 method for class 'supresid'
print(x, ...)
```

Arguments

`x`	A "supresid" object.
`...`	

Details

Prints the details of the space-time dataset, the superposition rate, the superposed residuals, and the simulated points.

Author(s)

Robert Clements

See Also

supresid

Examples

```
data(sresiduals1)
sresiduals1
```

`print.tessdev` *Print details of a* `tessdev` *object*

Description

Prints the details of the results of finding tessellation deviance residuals for a space-time point process.

Usage

```
## S3 method for class 'tessdev'
print(x, ...)
```

Arguments

`x`	A "`tessdev`" object.
`...`	

Details

Prints the details of the space-time dataset, the tessellation residuals, and the tile list.

The tile list is a list of the details of the tessellation cells.

Author(s)

Robert Clements

See Also

`tessdev`

print.tessresid *Print details of a* tessresid *object*

Description

Prints the details of the results of finding tessellation residuals for a space-time point process.

Usage

```
## S3 method for class 'tessresid'
print(x, ...)
```

Arguments

x	A "tessresid" object.
...	

Details

Prints the details of the space-time dataset, the tessellation residuals, and the tile list.

The tile list is a list of the details of the tessellation cells.

Author(s)

Robert Clements

See Also

tessresid

Examples

```
data(tsresiduals)
tsresiduals
```

print.thinresid *Print details of a* thinresid *object*

Description

Prints the details of the results of finding thinned residuals for a space-time point process.

Usage

```
## S3 method for class 'thinresid'
print(x, ...)
```

Arguments

x	A "thinresid" object.
...	

Details

Prints the details of the space-time dataset, the thinning rate, the thinned residuals, and the removed points.

Author(s)

Robert Clements

See Also

thinresid

Examples

```
data(tresiduals1)
tresiduals1
```

---

redbanana *Space-time data for red banana plants*

Description

Spatial locations and birth times of 788 red banana plants. Locations given in longitude and latitude.

Usage

```
data(redbanana)
```

Format

A data frame with 788 observations on the following 3 variables.

`longitude` a numeric vector

`latitude` a numeric vector

`birth` a numeric vector

Source

Balderama, E., Schoenberg, F.P., Murray, E., and Rundel, P.W. (2012) Application of branching point process models to the study of invasive red banana plants in Costa Rica. *Journal of the American Statistical Association,* to appear.

Examples

```
data(redbanana)
```

`simdata` *A simulated Poisson process.*

Description

x, y, and t coordinates of a simulated Poisson process.

Usage

```
data(simdata)
```

Format

A dataframe with 3 columns.

`sresiduals1` *Pre-computed superposed residuals*

Description

A "`supresid`" object.

Usage

```
data(sresiduals1)
```

Format

A "`supresid`" object.

`sresiduals2` *Pre-computed superposed residuals*

Description

A "`supresid`" object.

Usage

```
data(sresiduals2)
```

Format

A "`supresid`" object.

stpp *Convert data to class* stpp

Description

stpp creates a space-time point pattern of class "stpp" for use by the package stppResid.

Usage

```
stpp(x, y, t, stw)
```

Arguments

x	A vector of x coordinates.
y	A vector of y coordinates.
t	A vector of t coordinates.
stw	An object of class "stwin".

Details

x, y, and t represent the coordinates of all observed points in the space-time window described by stw. If any points fall outside of the window, a warning message is returned, and those points are removed from the point pattern. All inclusive points are then ordered in ascending order according to the t coordinates.

If no space-time window is specified, the default is a unit cube.

Value

Outputs an object of class "stpp", which is a list of

x	A vector of x coordinates.
y	A vector of y coordinates.
t	A vector of t coordinates.
stw	An object of class "stwin".

Author(s)

Robert Clements

See Also

stwin

Examples

```
#==> create a stpp object <===#
x <- rnorm(30, mean = 10, sd = 1)
y <- rnorm(30, mean = 100, sd = 10)
t <- runif(30, 0, 100)
stw <- stwin(xcoord = c(0, 20), ycoord = c(50, 150), tcoord = c(0, 100))
X <- stpp(x, y, t, stw = stw)

#===> create a stpp object from redbanana data <===#
data(redbanana)
attach(redbanana)
xcoord <- c(min(longitude)-.01, max(longitude)+.01)
ycoord <- c(min(latitude)-.01, max(latitude)+.01)
tcoord <- c(0, max(birth)+.01)
stw <- stwin(xcoord, ycoord, tcoord)
X <- stpp(longitude, latitude, birth, stw)
```

---

stresiduals1 *Pre-computed super-thinned residuals*

Description

A "superthin" object.

Usage

```
data(stresiduals1)
```

Format

A "superthin" object.

---

stresiduals2 *Pre-computed super-thinned residuals*

Description

A "superthin" object.

Usage

```
data(stresiduals2)
```

Format

A "superthin" object.

stwin *Create a space-time window*

Description

stwin creates an object of class "stwin" representing a three-dimensional space-time window.

Usage

```
stwin(xcoord = c(0, 1), ycoord = c(0, 1), tcoord = c(0, 1))
```

Arguments

xcoord	A vector of x coordinate limits.
ycoord	A vector of y coordinate limits.
tcoord	A vector of t coordinate limits.

Details

To create a space-time point process object of class "stpp", an enclosing space-time window of class "stwin" must be created and passed to stpp.

The window must be box shaped. Each vector of coordinates must be of length two and ordered from smallest to largest. Every combination of the entries of the three vectors represents the 8 corners of the space-time window.

If no coordinates are given, the default is a unit cube.

Value

Outputs an object of class "stwin" describing a three-dimensional space-time cuboid, which is a list of

xcoord	A vector of x limits.
ycoord	A vector of y limits.
tcoord	A vector of t limits.

Author(s)

Robert Clements

See Also

stpp

Examples

```
#===> create a stpp object <===#
x <- rnorm(30, mean = 10, sd = 1)
y <- rnorm(30, mean = 100, sd = 10)
t <- runif(30, 0, 100)
stw <- stwin(xcoord = c(0, 20), ycoord = c(50, 150), tcoord = c(0, 100))
X <- stpp(x, y, t, stw = stw)

#===> create a stpp object from redbanana data <===#
data(redbanana)
attach(redbanana)
xcoord <- c(min(longitude)-.01, max(longitude)+.01)
ycoord <- c(min(latitude)-.01, max(latitude)+.01)
tcoord <- c(0, max(birth)+.01)
stw <- stwin(xcoord, ycoord, tcoord)
X <- stpp(longitude, latitude, birth, stw)
```

---

summary.superthin *Summary of a set of super-thinned residuals*

Description

Outputs and prints a summary of a set of super-thinned residuals.

Usage

```
## S3 method for class 'superthin'
summary(object, ...)
```

Arguments

object	A "superthin" object.
...	

Details

Outputs and prints a summary of the superthin object.

Printed to the screen are the super-thinning rate, k, the number of super-thinned residuals, n, the expected number of residuals, n.exp, and the p-value for observing n residuals (p.val).

Value

A list of

k	The super-thinning rate.
n	The number of residuals.
n.exp	The expected number of residuals.
p.val	The p-value for n.

Author(s)

Robert Clements

See Also

superthin

Examples

```
data(stresiduals1)
summary(stresiduals1)

data(stresiduals2)
summary(stresiduals2)
```

`summary.supresid` *Summary of a set of superposed residuals*

Description

Outputs and prints a summary of a set of superposed residuals.

Usage

```
## S3 method for class 'supresid'
summary(object, ...)
```

Arguments

`object`	A "`supresid`" object.
`...`	

Details

Outputs and prints a summary of `supresid` object.

Printed to the screen are the superposition rate, `k`, the number of superposed residuals, `n`, the expected number of residuals, `n.exp`, and the p-value for observing `n` residuals (`p.val`).

Value

A list of

`k`	The superposition rate.
`n`	The number of residuals.
`n.exp`	The expected number of residuals.
`p.val`	The p-value for `n`.

Author(s)

Robert Clements

See Also

supresid

Examples

```
data(sresiduals1)
summary(sresiduals1)

data(sresiduals2)
summary(sresiduals2)
```

`summary.thinresid` *Summary of a set of thinned residuals*

Description

Outputs and prints a summary of a set of thinned residuals.

Usage

```
## S3 method for class 'thinresid'
summary(object, ...)
```

Arguments

`object`	A "`thinresid`" object.
`...`	

Details

Outputs and prints a summary of `thinresid` object.

Printed to the screen are the thinning rate, `k`, the number of thinned residuals, `n`, the expected number of residuals, `n.exp`, and the p-value for observing `n` residuals (`p.val`).

Value

A list of

`k`	The thinning rate.
`n`	The number of residuals.
`n.exp`	The expected number of residuals.
`p.val`	The p-value for `n`.

Author(s)

Robert Clements

See Also

thinresid

Examples

```
data(tresiduals1)
summary(tresiduals1)

data(tresiduals2)
summary(tresiduals2)
```

---

superthin *Perform super-thinned residuals method*

Description

superthin takes a space-time point pattern and conditional intensity model and calculates a set of super-thinned residuals for further analysis.

Usage

```
superthin(X, cifunction, theta = NULL, k = NULL, lambda = NULL)
```

Arguments

X	A "stpp" object.
cifunction	A function returning the value of the conditional intensity at all points in X. The function should take arguments X and an optional vector of parameters theta.
theta	Optional: A vector of parameters to be passed to cifunction.
k	The super-thinning rate.
lambda	Optional: A vector of conditional intensities at each point in X.

Details

Super-thinned residuals (Clements et. al. (2012)) is a type of transformation based residuals for space-time point processes based on both thinned residuals (see Schoenberg (2003)) and superposed residuals (see Bremaud (1981)). The residuals consist of a set of points that should be homogeneous Poisson, with rate k, if the model for the conditional intensity is correct. Any patterns or inter-point interaction in the residuals indicates a lack of fit of the model. To test for homogeneity, a commonly used tool is Ripley's K-function, a version of which can be found in the spatstat package.

Super-thinned residuals are found as follows:

1. The super-thinning rate k is specified. This rate determines the amount of thinning and superposition conducted, and also determines the final rate of the super-thinned residual point process.

2. All observed points in X where $\hat{\lambda} <$ k are automatically kept.

3. All points in X where $\hat{\lambda} >=$ k are kept with probability k/$\hat{\lambda}$.

4. In all space-time locations where $\lambda <$ k, points are simulated with rate k $-\hat{\lambda}$.

The result should be a homogeneous Poisson process with rate k if the model is correct.

The conditional intensity function, `cifunction`, should take X as the first argument, and an optional `theta` as the second argument, and return a vector of conditional intensity estimates with length equal to the number of points in X, i.e. the length of `X$x`. `cifunction` is required, while `lambda` is optional. `lambda` eliminates the need for `superthin` to calculate the conditional intensity at each observed point in X.

If k is not specified, the default is the mean of $\hat{\lambda}$ estimated by the total number of points divided by the volume of the space-time window.

Value

Outputs an object of class "`superthin`", which is a list of

X	An object of class "`stpp`".
k	The super-thinning rate.
`residuals`	A data frame consisting of the x, y, and t coordinates of the super-thinned residuals.
`super`	A data frame consisting of the x, y, and t coordinates of the superposed points.
`keep1`	A data frame consisting of the x, y, and t coordinates of the automatically retained points.
`keep2`	A data frame consisting of the x, y, and t coordinates of the points remaining after the thinning has taken place.
`deleted`	A data frame consisting of the x, y, and t coordinates of the points removed during the thinning process.

Author(s)

Robert Clements

References

Bremaud, P. *Point Processes and Queues: Martingale Dynamics.* SpringerVerlag, New York, 1981.

Clements, R.A., Schoenberg, F.P., and Veen, A. (2012) Evaluation of space-time point process models using super-thinning. *Environmetrics*, to appear.

Schoenberg, F.P. (2003) Multi-dimensional residuals analysis of point process models for earthquake occurrences. *Journal of the American Statistical Association,* **98**, 789–795.

Clements, R.A., Schoenberg, F.P., and Schorlemmer, D. (2011) Residual analysis methods for space-time point processes with applications to earthquake forecast models in California. *Annals of Applied Statistics,* **5**, Number 4, 2549–2571.

See Also

stpp, thinresid, supresid, spatstat

Examples

```
#===> load simulated data <===#
data(simdata)
X <- stpp(simdata$x, simdata$y, simdata$t)

#===> define conditional intensity function <===#
ci1 <- function(X, theta){theta[1]*exp(-theta[2]*X$x -
theta[3]*X$y - theta[4]*X$t)} #correct model

stresiduals1 <- superthin(X, ci1, theta = c(3000, 2, 2, 2), k = 250)
stresiduals2 <- superthin(X, ci1, theta = c(2500, 5, 5, 10), k = 250)
#===> plot results <===#
par(mfrow = c(1,2))
plot(stresiduals1)
plot(stresiduals2)

summary(stresiduals1)
summary(stresiduals2)
```

---

supresid *Perform superposed residuals method*

Description

supresid takes a space-time point pattern and conditional intensity model and calculates a set of superposed residuals for further analysis.

Usage

```
supresid(X, cifunction, theta = NULL, k = NULL, lambda = NULL)
```

Arguments

X	A "stpp" object.
cifunction	A function returning the value of the conditional intensity at all points in X. The function should take arguments X and an optional vector of parameters theta.
theta	Optional: A vector of parameters to be passed to cifunction.
k	The superposition rate.
lambda	Optional: A vector of conditional intensities at each point in X.

Details

Superposed residuals is a type of transformation based residuals for space-time point processes (see Bremaud (1981)) which consists of superimposing a point process with rate k - $\hat{\lambda}$ onto the observed point process. k should be the maximum conditional intensity over the entire space-time window. If the model for the conditional intensity is correct, the residuals should be homogeneous Poisson with rate k. Any patterns or inter-point interaction in the residuals indicates a lack of fit of the model. To test for homogeneity, a commonly used tool is Ripley's K-function, a version of which can be found in the spatstat package.

The conditional intensity function, cifunction, should take X as the first argument, and an optional theta as the second argument, and return a vector of conditional intensity estimates with length equal to the number of points in X, i.e. the length of X$x. cifunction is required, while lambda is optional. lambda eliminates the need for supresid to calculate the conditional intensity at each observed point in X.

If k is not specified, the default is the maximum $\hat{\lambda}$ estimated at the points.

Value

Outputs an object of class "supresid", which is a list of

X	An object of class "stpp".
k	The superposition rate.
residuals	A data frame consisting of the x, y, and t coordinates of the superposed residuals.
super	A data frame consisting of the x, y, and t coordinates of the superposed points.

Author(s)

Robert Clements

References

Bremaud, P. *Point Processes and Queues: Martingale Dynamics.* SpringerVerlag, New York, 1981.

Clements, R.A., Schoenberg, F.P., and Schorlemmer, D. (2011) Residual analysis methods for space-time point processes with applications to earthquake forecast models in California. *Annals of Applied Statistics,* **5**, Number 4, 2549–2571.

See Also

stpp, thinresid, superthin

Examples

```
#===> load simulated data <===#
data(simdata)
X <- stpp(simdata$x, simdata$y, simdata$t)

#===> define conditional intensity function <===#
ci1 <- function(X, theta){theta[1]*exp(-theta[2]*X$x -
theta[3]*X$y - theta[4]*X$t)} #correct model
```

```
sresiduals1 <- supresid(X, ci1, theta = c(3000, 2, 2, 2))
sresiduals2 <- supresid(X, ci1, theta = c(2500, 5, 5, 10))
#===> plot results <===#
par(mfrow = c(1,2))
plot(sresiduals1)
plot(sresiduals2)

summary(sresiduals1)
summary(sresiduals2)
```

`tessdev` *Calculate tessellation deviance residuals*

Description

`tessdev` divides the space-time window into cells using a Voronoi tessellation and calculates the deviance residuals within each cell between two competing conditional intensity models.

Usage

```
tessdev(X, cifunction1, cifunction2, theta1 = NULL, theta2 = NULL,
lambda1 = NULL, lambda2 = NULL, algthm1 = c("mc", "miser", "none"),
algthm2 = c("mc", "miser", "none"), n = 100, n1.miser = 10000,
n2.miser = 10000, ints1 = NULL, ints2 = NULL)
```

Arguments

`X`	A "stpp" object.
`cifunction1`	A function returning estimates of the conditional intensity at all points in `X`, according to model 1 (`cifunction1`). The function should take arguments `X` and an optional vector of parameters `theta1`.
`cifunction2`	A function returning estimates of the conditional intensity at all points in `X`, according to model 2 (`cifunction2`) which should be different than model 1 (`cifunction1`). The function should take arguments `X` and an optional vector of parameters `theta2`.
`theta1`	Optional: A vector of parameters to be passed to `cifunction1`.
`theta2`	Optional: A vector of parameters to be passed to `cifunction2`.
`lambda1`	Optional: A vector of conditional intensities based on `cifunction1` at each point in `X`.
`lambda2`	Optional: A vector of conditional intensities based on `cifunction2` at each point in `X`.
`algthm1`	The algorithm used for estimating the integrals in each cell for model 1. The three algorithms are "`mc`", "`miser`", and "`none`"
`algthm2`	The algorithm used for estimating the integrals in each cell for model 2. The three algorithms are "`mc`", "`miser`", and "`none`"

n	Initial number of sample points in each cell for approximating integrals. The number of sample points are iteratively increased by n until some accuracy threshold is reached.
n1.miser	The total number of sample points for estimating all integrals for model 1 if the miser algorithm is selected.
n2.miser	The total number of sample points for estimating all integrals for model 2 if the miser algorithm is selected.
ints1	An optional vector of integrals for model 1. Must be the same length as the number of tessellation cells, and each element of ints1 should correspond to each cell in the tile.list that is returned using the deldir function, which can be called separately.
ints2	An optional vector of integrals for model 2. Must be the same length as the number of tessellation cells, and each element of ints2 should correspond to each cell in the tile.list that is returned using the deldir function, which can be called separately.

Details

The tessellation deviance residuals are the differences in the tessellation residuals of model 1 vs. model 2 within each Voronoi tessellation cell, denoted here as V_i. The tessellation deviance residual is given by

$$R_{TD}(V_i) = \left(1 - \int_{V_i} \hat{\lambda}_1(x)dx\right) / \sqrt{\int_{V_i} \hat{\lambda}_1(x)dx} - \left(1 - \int_{V_i} \hat{\lambda}_2(x)dx\right) / \sqrt{\int_{V_i} \hat{\lambda}_2(x)dx},$$

where $\hat{\lambda}(x)$ is the fitted conditional intensity model.

The conditional intensity functions, cifunction1 and cifunction2, should take X as their first argument, and an optional theta as their second argument, and return a vector of conditional intensity estimates with length equal to the number of points in X, i.e. the length of X$x. Both cifunction1 and cifunction2 are required. lambda1 and lambda2 are optional, and if passed will eliminate the need for devresid to calculate the conditional intensities at each observed point in X.

The integrals in $R_{TD}(V_i)$ are approximated using one of two algorithms: a simple Monte Carlo (mc) algorithm, or the MISER algorithm. The simple Monte Carlo iteratively adds n sample points to each tessellation cell to approximate the integral, and the iteration stops when some threshold in the accuracy of the approximation is reached. The MISER algorithm samples a total number of n.miser points in a recursive way, sampling the points in locations that have the highest variance. This part can be very slow and the approximations can be very inaccurate. For highest accuracy these algorithms will require a very large n or n.miser depending on the complexity of the conditional intensity functions (some might say ~1 billion sample points are needed for a good approximation).

Passing the arguments ints1 and/or ints2 eliminates the need for approximating the integrals using either of the two algorithms here. However, the tile.list must first be obtained in order to assure that each element of ints1 and/or ints2 corresponds to the correct cell. The tile.list can be obtained, either by using the deldir function separately, or by using tessresid with one of the included algorithms first (the tile.list is returned along with the residuals). tessresid can then be called again with ints1 and/or ints2 included and algthm = "none".

Note that if `miser` is selected, and if the points in the point pattern are very densely clustered, the integral in some cells may end up being approximated based on only the observed point in the point pattern that is contained in that cell. This happens because the cells in these clusters of points will be very small, and so it may be likely that sampled points based on the MISER algorithm will miss these cells entirely. For this reason, the simple Monte Carlo algorithm might be preferred.

Value

Prints to the screen the number of simulated points used to approximate the integrals.

Outputs an object of class "`tessdev`", which is a list of

`X`	An object of class "`stpp`".
`tile.list`	An object of class "`tile.list`".
`residuals`	A vector of tessellation deviance residuals. The order of the residuals corresponds with the order of the cells (tiles) in `tile.list`.

If `algthm` = "`mc`", then a list of the following elements are also included for each model that uses the `mc` algorithm:

`n`	Vector of the total number of points used for approximating integrals.
`integral`	Vector of actual integral approximations in each grid cell.
`mean.lambda`	Vector of the approximate final mean of lambda in each grid cell.
`sd.lambda`	Vector of the approximate standard deviation of lambda in each grid cell.

If the `miser` algorithm is selected, then a list of the following elements are also included for each model that uses the `miser` algorithm:

`n`	Total number of points used for approximating integrals.
`integral`	Vector of actual integral approximations in each grid cell.
`mean.lambda`	Vector of the approximate final mean of lambda in each grid cell.
`sd.lambda`	Vector of the approximate standard deviation of lambda in each grid cell.
`app.pts`	A data frame of the x,y, and t coordinates of a sample of 10,000 of the sampled points for integral approximation, along with the value of lambda (l).

Author(s)

Robert Clements

See Also

`tessresid`

Examples

```
#===> load simulated data <===#
data(simdata)
X <- stpp(simdata$x, simdata$y, simdata$t)

#===> define two conditional intensity functions <===#
ci1 <- function(X, theta){theta*exp(-2*X$x - 2*X$y - 2*X$t)} #correct model

ci2 <- function(X, theta = NULL){rep(250, length(X$x))} #homogeneous Poisson model

## Not run:
deviance <- tessdev(X, ci1, ci2, theta1 = 3000)
#===> plot results <===#
plot(deviance)

## End(Not run)
```

---

tessresid *Calculate tessellation residuals*

Description

`tessresid` divides the space-time window into bins using a Voronoi tessellation and calculates residuals within each bin for a specified conditional intensity model.

Usage

```
tessresid(X, cifunction, theta = NULL, algthm = c("mc", "miser", "none"),
n = 100, n.miser = 10000, ints = NULL)
```

Arguments

`X`	A "`stpp`" object.
`cifunction`	A function returning the value of the conditional intensity at all points in `X`. The function should take arguments `X` and an optional vector of parameters `theta`.
`theta`	Optional: A vector of parameters to be passed to `cifunction`.
`algthm`	The algorithm used for estimating the integrals in each tessellation cell. The three algorithms are "`mc`", "`miser`", and "`none`"
`n`	Initial number of sample points in each grid tessellation for approximating integrals. The number of sample points are iteratively increased by `n` until some accuracy threshold is reached.
`n.miser`	The total number of sample points for estimating all integrals.
`ints`	An optional vector of integrals. Must be the same length as the number of tessellation cells, and each element of `ints` should correspond to each cell in the `tile.list` that is returned using the `deldir` function, which can be called separately.

Details

Tessellation residuals are residuals calculated in bins that are created by dividing up the spatial window using a Voronoi tessellation. Because the bins are based on a tessellation, each bin contains at most one point. The residual in bin i (V_i) is defined by

$$R_T(V_i) = \left(1 - \int_{V_i} \hat{\lambda}(x)dx\right) / \sqrt{\int_{V_i} \hat{\lambda}(x)dx},$$

where $\hat{\lambda}(x)$ is the fitted conditional intensity model.

The conditional intensity function, `cifunction`, should take `X` as the first argument, and an optional `theta` as the second argument, and return a vector of conditional intensity estimates with length equal to the number of points in `X`, i.e. the length of `X$x`. `cifunction` is required, while `lambda` is optional. `lambda` eliminates the need for `tessresid` to calculate the conditional intensity at each observed point in `X`.

The integrals in $R_T(V_i)$ are approximated using one of two algorithms: a simple Monte Carlo (`mc`) algorithm, or the MISER algorithm. The simple Monte Carlo iteratively adds `n` sample points to each tessellation cell to approximate the integral, and the iteration stops when some threshold in the accuracy of the approximation is reached. The MISER algorithm samples a total number of `n.miser` points in a recursive way, sampling the points in locations that have the highest variance. This part can be very slow and the approximations can be very inaccurate. For highest accuracy these algorithms will require a very large `n` or `n.miser` depending on the complexity of the conditional intensity functions (some might say ~1 billion sample points are needed for a good approximation).

Passing the argument `ints` eliminates the need for approximating the integrals using either of the two algorithms here. However, the `tile.list` must first be obtained in order to assure that each element of `ints` corresponds to the correct cell. The `tile.list` can be obtained, either by using the `deldir` function separately, or by using `tessresid` with one of the included algorithms first (the `tile.list` is returned along with the residuals). `tessresid` can then be called again with `ints` included and `algthm` = "none".

Note that if `miser` is selected, and if the points in the point pattern are very densely clustered, the integral in some cells may end up being approximated based on only the observed point in the point pattern that is contained in that cell. This happens because the cells in these clusters of points will be very small, and so it may be likely that sampled points based on the MISER algorithm will miss these cells entirely. For this reason, the simple Monte Carlo algorithm might be preferred.

Value

Outputs an object of class "`tessresid`", which is a list of

`X`	An object of class "`stpp`".
`tile.list`	An object of type "`tile.list`", which is itself a list with one entry for each point in `X`. Each entry is a list of pt: x and y coordinates of the point. x: x coordinates of the vertices of the tessellation cell. y: y coordinates of the vertices of the tessellation cell.
`residuals`	A vector of tessellation residuals. The order of the residuals corresponds with the order of the cells in `tile.list`.

Author(s)

Robert Clements

See Also

gridresid, deldir, tile.list

Examples

```
#===> load simulated data <===#
data(simdata)
X <- stpp(simdata$x, simdata$y, simdata$t)

#===> define two conditional intensity functions <===#
ci1 <- function(X, theta){theta*exp(-2*X$x - 2*X$y - 2*X$t)} #correct model

ci2 <- function(X, theta = NULL){rep(250, length(X$x))} #homogeneous Poisson model

## Not run:
tsresiduals <- tessresid(X, ci1, theta = 3000)
tsresiduals2 <- tessresid(X, ci2)
#===> plot results <===#
plot(tsresiduals)
plot(tsresiduals2)

## End(Not run)
```

---

thinresid *Perform thinned residuals method*

Description

thinresid takes a space-time point pattern and conditional intensity model and calculates a set of thinned residuals for further analysis.

Usage

```
thinresid(X, cifunction = NULL, theta = NULL, k = NULL, lambda = NULL)
```

Arguments

X	A "stpp" object.
cifunction	A function returning the value of the conditional intensity at all points in X. The function should take arguments X and an optional vector of parameters theta.
theta	Optional: A vector of parameters to be passed to cifunction.
k	The thinning rate.
lambda	Optional: A vector of conditional intensities at each point in X.

Details

Thinned residuals is a type of transformation based residuals for space-time point processes (see Schoenberg (2003)) which consists of thinning out the observed points using the fitted conditional intensity model, $\hat{\lambda}$. Each point is kept with probability $k/\hat{\lambda}$, where k should be the minimum conditional intensity over the entire space-time window. If the model for the conditional intensity is correct, the residuals should be homogeneous Poisson with rate k. Any patterns or inter-point interaction in the residuals indicates a lack of fit of the model. To test for homogeneity, a commonly used tool is Ripley's K-function, a version of which can be found in the `spatstat` package.

The conditional intensity function, `cifunction`, should take `X` as the first argument, and an optional `theta` as the second argument, and return a vector of conditional intensity estimates with length equal to the number of points in `X`, i.e. the length of `X$x`. `cifunction` is required, while `lambda` is optional. `lambda` eliminates the need for `thinresid` to calculate the conditional intensity at each observed point in `X`.

If k is not specified, the default is the minimum $\hat{\lambda}$ estimated at the points.

Value

Outputs an object of class "`thinresid`", which is a list of

`X`	An object of class "`stpp`".
`k`	The thinning rate.
`residuals`	A data frame consisting of the x, y, and t coordinates of the thinned residuals.
`deleted`	A data frame consisting of the x, y, and t coordinates of the points removed during the thinning process.

Author(s)

Robert Clements

References

Schoenberg, F.P. (2003) Multi-dimensional residuals analysis of point process models for earthquake occurrences. *Journal of the American Statistical Association,* **98**, 789–795.

Clements, R.A., Schoenberg, F.P., and Schorlemmer, D. (2011) Residual analysis methods for space-time point processes with applications to earthquake forecast models in California. *Annals of Applied Statistics,* **5**, Number 4, 2549–2571.

See Also

`stpp`, `supresid`, `superthin`

Examples

```
#===> load simulated data <===#
data(simdata)
X <- stpp(simdata$x, simdata$y, simdata$t)

#===> define conditional intensity function <===#
```

```
ci1 <- function(X, theta){theta[1]*exp(-theta[2]*X$x -
theta[3]*X$y - theta[4]*X$t)} #correct model

tresiduals1 <- thinresid(X, ci1, theta = c(3000, 2, 2, 2))
tresiduals2 <- thinresid(X, ci1, theta = c(2500, 5, 5, 10))
#===> plot results <===#
par(mfrow = c(1,2))
plot(tresiduals1)
plot(tresiduals2)

summary(tresiduals1)
summary(tresiduals2)
```

`tresiduals1` *Pre-computed thinned residuals*

Description

A "`thinresid`" object.

Usage

```
data(tresiduals1)
```

Format

A "`thinresid`" object.

`tresiduals2` *Pre-computed thinned residuals*

Description

A "`thinresid`" object.

Usage

```
data(tresiduals2)
```

Format

A "`thinresid`" object.

tsresiduals *Pre-computed tessellation residuals*

Description

A "tessresid" object.

Usage

```
data(tsresiduals)
```

Format

A "tessresid" object.

tsresiduals2 *Pre-computed tessellation residuals*

Description

A "tessresid" object.

Usage

```
data(tsresiduals2)
```

Format

A "tessresid" object.

Index