

Package ‘PtProcess’

February 2, 2010

Version 3.2-2

Date 2010-02-02

Title Time Dependent Point Process Modelling

Author David Harte

Maintainer David Harte <david@statsresearch.co.nz>

Description This package fits and analyses time dependent marked point process models with an emphasis on earthquake modelling. For a more detailed introduction to the package, see the topic “PtProcess”. A list of recent changes can be found in the topic “Changes”.

Suggests snow

License GPL (>= 2)

URL <http://cran.at.r-project.org/web/packages/PtProcess>,
<http://www.statsresearch.co.nz/software.html>

Repository CRAN

Date/Publication 2010-02-02 10:08:27

R topics documented:

Change Log	2
distribution	5
dpareto	9
etas_gif	13
etas_spatial	15
gif	18
linksrn	20
linksrn_convert	22
linksrn_gif	23
logLik	25
marks	27
mpp	29

neglogLik	31
NthChina	33
Ogata	34
Phuket	34
plot	36
PtProcess	37
residuals	39
simple_gif	41
simulate	43
srm_gif	45
summary	46
Tangshan	47

Index	49
--------------	-----------

Change Log	<i>Changes Made to the Package</i>
------------	------------------------------------

Description

This page contains a listing of recent changes made to functions, and known general problems.

Recent Changes

1. Version 3 contains major changes, and code that worked in Version 2 will no longer work in Version 3. The models included in Version 2 are also contained in Version 3, but the framework has been extended so that the original models can now contain a variety of mark distributions. This has been achieved by giving a more general structure and utilising the object orientated aspects of the R language. *Examples are given below that show how models were defined in Version 2 and how the corresponding models are now defined in Version 3.* (28 Apr 2008)
2. Naming changes to the `*.cif` functions. In Version 2, these were referred to as “conditional intensity functions”, which is really a slightly more general class. In keeping with Daley & Vere-Jones (2003) we now call them ground intensity functions, with a suffix of “gif”. Further, the dot has been replaced by an underscore, e.g. `etas.cif` to `etas_gif`. This is to lessen the possibility of future conflicts with object orientated naming conventions in the R language. (28 Apr 2008)
3. Arguments `eval.pts` and `t.plus` in the ground intensity functions have been renamed to `evalpts` and `tplus`, respectively. This is to lessen the possibility of future conflicts with object orientated naming conventions in the R language. (28 Apr 2008)
4. `logLik`: the log-likelihood calculated in package Versions before Version 3 did not have the sum over the mark density term (see topic `logLik`, under “Details”). This term can also be excluded in this Version of the package by placing `NULL` for the mark density in the `mpp` object, see example below. (28 Apr 2008)
5. Version 2 had a framework to assign prior densities to the estimated parameters. This has not been retained in Version 3. However, some of the features like holding a parameter at a fixed value, and restricting it to an open or closed interval can be achieved in Version 3; see `neglogLik` for further details. (28 Apr 2008)

6. `neglogLik`: the format of this function has been changed to be consistent with that in package **HiddenMarkov**. Argument `updatep` renamed as `pmap`. (07 Aug 2008)
7. `simulate`: manual page revised to include more information about controlling the length of the simulated series. (18 Nov 2008)
8. `mpp`: example modified due to warning messages caused by negative $\lambda_g(t|\mathcal{H}_t)$. (18 Nov 2008)
9. `marks`: manual page revised to include more information. (18 Nov 2008)
10. `mpp`: fuller description to argument `marks` on manual page. (19 Nov 2008)
11. `Phuket`: new dataset added. (4 Dec 2008)
12. `linkstrm_gif`, `marks`: remove some LaTeX specific formatting to be compatible with R 2.9.0. (26 Jan 2009)
13. `Phuket`: clarify magnitude scale used in the dataset. (11 Jul 2009)
14. Attribute `type` is no longer required on the `gif` functions, removed. (7 Oct 2009)
15. `logLik`, `neglogLik`: Parallel processing support, using package **snow**, has been added. (8 Oct 2009)
16. `plot`: Correct hyperlink to generic plot function. (10 Oct 2009)
17. `etas_normal0`: New function. Test version of a spatial ETAS conditional intensity function. (12 Oct 2009)
18. `logLik`: Fixed bug when using parallel processing on only two nodes. (22 Oct 2009)
19. Tidied HTML representation of equations in manual pages. Removal of “synopsis” on manual pages of functions with multiple forms of usage. (26 Jan 2010)
20. `logLik.mpp`, `summary.mpp`: Changed to `inherits` to determine class. (27 Jan 2010)
21. `Phuket`: Additional data, until the beginning of 2009, have been added. The magnitude is now the maximum of the body wave and surface wave magnitudes, m_b and M_s , respectively. Earlier it was simply m_b . (01 Feb 2010)

Future Development

1. Currently spatial versions of the ETAS model are being written and tested.
2. In the model object, allow one to alternatively specify the name of the `gif` function.
3. Function `linkstrm_gif`: Use of `St1` and `St2`. Is there a tidier way? Also utilise this feature in `strm_gif`.
4. Want a generic function, possibly called `forecast`, to produce probability forecasts. This would be based on simulating empirical probability distributions.
5. Want a function like `linkstrm_convert` to map between the two main parametrisations of the ETAS model.
6. Add general forms of the truncated exponential and gamma distributions as marks for the magnitude of the event.
7. A tidy way to pass the values of the `gif` function into the mark distributions, if required.

Examples

```

# SRM: magnitude is iid exponential with bvalue=1
# simulate and calculate the log-likelihood

TT <- c(0, 1000)
bvalue <- 1
params <- c(-1.5, 0.01, 0.8, bvalue*log(10))

# --- Old Method ---
# x <- pp.sim(NULL, params[1:3], srm.cif, TT, seed=5, magn.sim=1)
# print(pp.LL(x, srm.cif, params[1:3], TT))
# [1] -601.3941

# --- New Method, no mark density ---
x1 <- mpp(data=NULL,
          gif=srm_gif,
          mark=list(NULL, rexp_mark),
          params=params,
          gmap=expression(params[1:3]),
          mmap=expression(params[4]),
          TT=TT)
x1 <- simulate(x1, seed=5)
print(logLik(x1))

# An advantage of the object orientated format is that it
# simplifies further analysis, e.g. plot intensity function:
plot(x1)
# plot the residual process:
plot(residuals(x1))

#-----
# SRM: magnitude is iid exponential with bvalue=1
# simulate then estimate parameters from data

# --- Old Method ---
# TT <- c(0, 1000)
# bvalue <- 1
# params <- c(-2.5, 0.01, 0.8)
#
# x <- pp.sim(NULL, params, srm.cif, TT, seed=5, magn.sim=1)
#
# posterior <- make.posterior(x, srm.cif, TT)
#
# neg.posterior <- function(params){
#   x <- -posterior(params)
#   if (is.infinite(x) | is.na(x)) return(1e15)
#   else return(x)
# }
#
# z <- nlm(neg.posterior, params, typsize=abs(params),
#         iterlim=1000, print.level=2)
#

```

```

# print(z$estimate)
# [1] -2.83900091  0.01242595  0.78880647

# --- New Method, no mark density ---
#   maximise only SRM parameters (like old method)

TT <- c(0, 1000)
bvalue <- 1
params <- c(-2.5, 0.01, 0.8, bvalue*log(10))

x1 <- mpp(data=NULL,
          gif=srm_gif,
          mark=list(dexp_mark, rexp_mark),
          params=params,
          gmap=expression(params[1:3]),
          mmap=expression(params[4]),
          TT=TT)
# note that dexp_mark above is not used below
# and could alternatively be replaced by NULL

x1 <- simulate(x1, seed=5)

# maximise only SRM parameters
onlysrm <- function(y, p){
  # maps srm parameters into model object
  # the exp rate for magnitudes is unchanged
  y$params[1:3] <- p
  return(y)
}

params <- c(-2.5, 0.01, 0.8)

z1 <- nlm(neglogLik, params, object=x1, pmap=onlysrm,
         print.level=2, iterlim=500, typsize=abs(params))
print(z1$estimate)

```

Description

This page contains general notes about fitting probability distributions to datasets.

Details

We give examples of how the maximum likelihood parameters can be estimated using standard optimisation routines provided in the R software ([nlm](#) and [optim](#)). We simply numerically maximise the sum of the logarithms of the density evaluated at each of the data points, i.e. log-likelihood function. In fact, by default, the two mentioned optimizers find the *minimum*, and hence we minimise the negative log-likelihood function.

Both optimization routines require initial starting values. The optimisation function `optim` uses a grid search technique, and is therefore more robust to poor starting values. The function `nlm` uses derivatives and the Hessian to determine the size and direction of the next step, which is generally more sensitive to poor initial values, but faster in the neighbourhood of the solution. One possible strategy is to start with `optim` and then use its solution as a starting value for `nlm`. This is done below in the example for the tapered Pareto distribution.

The function `nlm` numerically calculates the Hessian and derivatives, by default. If the surface is very flat, the numerical error involved may be larger in size than the actual gradient. In this case the process will work better if analytic derivatives are supplied. This is done in the tapered Pareto example below. Alternatively, one could simply use the Newton-Raphson algorithm (again, see the tapered Pareto example below).

We also show that parameters can be constrained to be positive (or negative) by transforming the parameters with the exponential function during the maximisation procedure. Similarly, parameters can be restricted to a finite interval by using a modified logit transform during the maximisation procedure. The advantage of using these transformations is that the entire real line is mapped onto the positive real line or the required finite interval, respectively; and further, they are differentiable and monotonic. This eliminates the “hard” boundaries which are sometimes enforced by using a penalty function when the estimation procedure strays into the forbidden region. The addition of such penalty functions causes the function that is being optimised to be non-differentiable at the boundaries, which can cause considerable problems with the optimisation routines.

Examples

```
# Random number generation method
RNGkind("Mersenne-Twister", "Inversion")
set.seed(5)

#-----
# Exponential Distribution

# simulate a sample
p <- 1
x <- rexp(n=1000, rate=p)

# Transform to a log scale so that -infty < log(p) < infty.
# Hence no hard boundary, and p > 0.
# If LL is beyond machine precision, LL <- 1e20.

neg.LL <- function(logp, data){
  x <- -sum(log(dexp(data, rate=exp(logp))))
  if (is.infinite(x)) x <- 1e20
  return(x)
}

p0 <- 5
logp0 <- log(p0)
z <- nlm(neg.LL, logp0, print.level=0, data=x)
print(exp(z$estimate))

# Compare to closed form solution
```

```

print(exp(z$estimate)-1/mean(x))

#-----
#   Normal Distribution

#   simulate a sample
x <- rnorm(n=1000, mean=0, sd=1)

neg.LL <- function(p, data){
  x <- -sum(log(dnorm(data, mean=p[1], sd=exp(p[2]))))
  if (is.infinite(x)) x <- 1e20
  return(x)
}

p0 <- c(2, log(2))
z <- nlm(neg.LL, p0, print.level=0, data=x)
p1 <- c(z$estimate[1], exp(z$estimate[2]))
print(p1)

#   Compare to closed form solution
print(p1 - c(mean(x), sd(x)))

#-----
#   Gamma Distribution
#   shape > 0 and rate > 0
#   use exponential function to ensure above constraints

#   simulate a sample
x <- rgamma(n=2000, shape=1, rate=5)

neg.LL <- function(p, data){
  #   give unreasonable values a very high neg LL, i.e. low LL
  if (any(exp(p) > 1e15)) x <- 1e15
  else{
    x <- -sum(log(dgamma(data, shape=exp(p[1]), rate=exp(p[2]))))
    if (is.infinite(x)) x <- 1e15
  }
  return(x)
}

p0 <- c(2, 2)
z <- optim(p0, neg.LL, data=x)
print(exp(z$par))

z <- nlm(neg.LL, p0, print.level=0, data=x)
print(exp(z$estimate))

#-----
#   Beta Distribution
#   shapel > 0 and shape2 > 0
#   use exponential function to ensure above constraints

#   simulate a sample

```

```

x <- rbeta(n=5000, shapel=0.5, shape2=0.2)

#   exclude those where x=0
x <- x[x!=1]

neg.LL <- function(p, data)
  -sum(log(dbeta(data, shapel=exp(p[1]), shape2=exp(p[2]))))

p0 <- log(c(0.1, 0.1))

z <- optim(p0, neg.LL, data=x)
print(exp(z$par))

z <- nlm(neg.LL, p0, typsize=c(0.01, 0.01), print.level=0, data=x)
print(exp(z$estimate))

#-----
#   Weibull Distribution
#   shape > 0 and scale > 0
#   use exponential function to ensure above constraints

#   simulate a sample
x <- rweibull(n=2000, shape=2, scale=1)

neg.LL <- function(p, data)
  -sum(log(dweibull(data, shape=exp(p[1]), scale=exp(p[2]))))

p0 <- log(c(0.1, 0.1))
z <- optim(p0, neg.LL, data=x)
print(exp(z$par))

#-----
#   Pareto Distribution
#   lambda > 0
#   Use exponential function to enforce constraint

#   simulate a sample
x <- rpareto(n=2000, lambda=2, a=1)

neg.LL <- function(p, data){
  #   give unreasonable values a very high neg LL, i.e. low LL
  if (exp(p) > 1e15) x <- 1e15
  else x <- -sum(log(dpareto(data, lambda=exp(p), a=1)))
  if (is.infinite(x)) x <- 1e15
  return(x)
}

p0 <- log(0.1)
z <- nlm(neg.LL, p0, print.level=0, data=x)
print(exp(z$estimate))

#-----
#   Tapered Pareto Distribution

```

```

#   lambda > 0   and   theta > 0

# simulate a sample
x <- rtappareto(n=2000, lambda=2, theta=4, a=1)

neg.LL <- function(p, data){
  x <- -ltappareto(data, lambda=p[1], theta=p[2], a=1)
  attr(x, "gradient") <- -attr(x, "gradient")
  attr(x, "hessian") <- -attr(x, "hessian")
  return(x)
}

#   use optim to get approx initial value
p0 <- c(3, 5)
z1 <- optim(p0, neg.LL, data=x)
p1 <- z1$par
print(p1)
print(neg.LL(p1, x))

#   nlm with analytic gradient and hessian
z2 <- nlm(neg.LL, p1, data=x, hessian=TRUE)
p2 <- z2$estimate
print(z2)

#   Newton Raphson Method
p3 <- p1
iter <- 0
repeat{
  LL <- ltappareto(data=x, lambda=p3[1], theta=p3[2], a=1)
  p3 <- p3 - as.numeric(solve(attr(LL, "hessian")) %*%
    matrix(attr(LL, "gradient"), ncol=1))
  iter <- iter + 1
  if ((max(abs(attr(LL, "gradient")))) < 1e-8) |
    (iter > 100)) break
}
print(iter)
print(LL)
print(p3)

```

Description

Density, cumulative probability, quantiles and random number generation for the Pareto and tapered Pareto distributions with shape parameter λ , tapering parameter θ and range $a \leq x < \infty$; and log-likelihood of the tapered Pareto distribution.

Usage

```

dpareto(x, lambda, a, log=FALSE)
ppareto(q, lambda, a, lower.tail=TRUE, log.p=FALSE)
qpareto(p, lambda, a, lower.tail=TRUE, log.p=FALSE)
rpareto(n, lambda, a)

dtappareto(x, lambda, theta, a, log=FALSE)
ltappareto(data, lambda, theta, a)
ptappareto(q, lambda, theta, a, lower.tail=TRUE, log.p=FALSE)
qtappareto(p, lambda, theta, a, lower.tail=TRUE, log.p=FALSE,
           tol=1e-8)
rtappareto(n, lambda, theta, a)

```

Arguments

<code>x, q</code>	vector of quantiles.
<code>p</code>	vector of probabilities.
<code>data</code>	vector of sample data.
<code>n</code>	number of observations to simulate.
<code>lambda</code>	shape parameter.
<code>theta</code>	tapering parameter.
<code>a</code>	the random variable takes values on the interval $a \leq x < \infty$.
<code>log, log.p</code>	logical; if TRUE, probabilities <code>p</code> are given as $\log(p)$.
<code>lower.tail</code>	logical; if TRUE (default), probabilities are $\Pr\{X \leq x\}$, otherwise, $\Pr\{X > x\}$.
<code>tol</code>	convergence criteria for the Newton Raphson algorithm for solving the quantiles of the tapered Pareto distribution.

Details

Let Y be an exponential random variable with parameter $\lambda > 0$. Then the distribution function of Y is

$$F_Y(y) = \Pr\{Y < y\} = 1 - \exp(-\lambda y),$$

and the density function is

$$f_Y(y) = \lambda \exp(-\lambda y).$$

Further, the mean and variance of the distribution of Y is $1/\lambda$ and $1/\lambda^2$, respectively.

Now transform Y as

$$X = a \exp(Y),$$

where $a > 0$. Then X is a Pareto random variable with shape parameter λ and distribution function

$$F_X(x) = \Pr\{X < x\} = 1 - \left(\frac{a}{x}\right)^\lambda,$$

where $a \leq x < \infty$, and density function

$$f_X(x) = \frac{\lambda}{a} \left(\frac{a}{x}\right)^{\lambda+1}.$$

We simulate the Pareto deviates by generating exponential deviates, and then transforming as described above.

As above, let X be Pareto with shape parameter λ , and define $W - a$ to be exponential with parameter $1/\theta$, i.e.

$$\Pr\{X > x\} = \left(\frac{a}{x}\right)^\lambda$$

and

$$\Pr\{W > w\} = \exp\left(-\frac{w-a}{\theta}\right),$$

where $a \leq w < \infty$. Say we sample one independent value from each of the distributions X and W , then

$$\Pr\{X > z \& W > z\} = \Pr\{X > z\} \Pr\{W > z\} = \left(\frac{a}{z}\right)^\lambda \exp\left(-\frac{a-z}{\theta}\right).$$

We say that Z has a tapered Pareto distribution if it has the above distribution, i.e.

$$F_Z(z) = \Pr\{Z < z\} = 1 - \left(\frac{a}{z}\right)^\lambda \exp\left(-\frac{a-z}{\theta}\right).$$

The above relationship shows that a tapered Pareto deviate can be simulated by generating independent values of X and W , and then letting $Z = \min(X, W)$. This minimum has the effect of ‘‘tapering’’ the tail of the Pareto distribution.

The tapered Pareto variable Z has density

$$f_Z(z) = \left(\frac{\lambda}{z} + \frac{1}{\theta}\right) \left(\frac{a}{z}\right)^\lambda \exp\left(-\frac{a-z}{\theta}\right).$$

Given a sample of data z_1, z_2, \dots, z_n , we write the log-likelihood as

$$\log L = \sum_{i=1}^n \log f_Z(z_i).$$

Hence the gradients are calculated as

$$\frac{\partial \log L}{\partial \lambda} = \theta \sum_{i=1}^n \frac{1}{\lambda\theta + z_i} - \sum_{i=1}^n \log(z_i/a)$$

and

$$\frac{\partial \log L}{\partial \theta} = \frac{-1}{\theta} \sum_{i=1}^n \frac{z_i}{\lambda\theta + z_i} - \frac{1}{\theta^2} \sum_{i=1}^n (a - z_i).$$

Further, the Hessian is calculated using

$$\begin{aligned} \frac{\partial^2 \log L}{\partial \lambda^2} &= -\theta^2 \sum_{i=1}^n \frac{1}{(\lambda\theta + z_i)^2}, \\ \frac{\partial^2 \log L}{\partial \theta^2} &= \frac{1}{\theta^2} \sum_{i=1}^n \frac{z_i(2\lambda\theta + z_i)}{(\lambda\theta + z_i)^2} - \frac{2}{\theta^3} \sum_{i=1}^n (a - z_i), \end{aligned}$$

and

$$\frac{\partial^2 \log L}{\partial \theta \partial \lambda} = \frac{\partial^2 \log L}{\partial \lambda \partial \theta} = \sum_{i=1}^n \frac{z_i}{(\lambda\theta + z_i)^2}.$$

See the section ‘‘Seismological Context’’ (below), which outlines its application in Seismology.

Value

dpareto and dtappareto give the densities; ppareto and ptappareto give the distribution functions; qpareto and qtappareto give the quantile functions; and rpareto and rtappareto generate random deviates.

ltappareto returns the log-likelihood of a sample using the tapered Pareto distribution. It also calculates, using analytic expressions (see “Details”), the derivatives and Hessian which are attached to the log-likelihood value as the attributes "gradient" and "hessian", respectively.

Seismological Context

The Gutenberg-Richter (GR) Law says that if we plot the base 10 logarithm of the number of events with magnitude greater than M (vertical axis) against M (horizontal axis), there should be a straight line. This is equivalent to magnitudes having an exponential distribution.

Assume that the magnitude cutoff is M_0 , and let $Y = M - M_0$. Given that Y has an exponential distribution with parameter λ , it follows that

$$\log_{10}(1 - F_Y(y)) = \frac{-\lambda y}{\log_e 10}.$$

The coefficient $\lambda/(\log_e 10)$ is often referred to as the b -value, and its negative value is the slope of the line in the GR plot.

Now define S as

$$S = 10^{\gamma(M - M_0)} = 10^{\gamma Y}.$$

When $\gamma = 0.75$, S is the “stress”; and when $\gamma = 1.5$, S is the “seismic moment”. Still assuming that Y is exponential with parameter λ , then $Y\gamma \log_e 10$ is also exponential with parameter $\lambda/(\gamma \log_e 10)$. Hence, by noting that S can be rewritten as

$$S = \exp\{Y\gamma \log_e 10\},$$

it is seen that S is Pareto with parameter $\lambda/(\gamma \log_e 10)$, and $1 \leq S < \infty$.

While the empirical distribution of magnitudes appears to follow an exponential distribution for smaller events, it provides a poor approximation for larger events. This is because it is not physically possible to have events with magnitudes much greater than about 9.5. Consequently, the tail of the Pareto distribution will also be too long. Hence the tapered Pareto distribution provides a more realistic description.

See Also

See [dexp](#) for the exponential distribution. Generalisations of the exponential distribution are the gamma distribution [dgamma](#) and the Weibull distribution [dweibull](#).

See the topic [distribution](#) for examples of estimating parameters.

Examples

```
# Simulate and plot histogram with density for Pareto Distribution
a0 <- 2
lambda0 <- 2
```

```

x <- rpareto(1000, lambda=lambda0, a=a0)
x0 <- seq(a0, max(x)+0.1, length=100)
hist(x, freq=FALSE, breaks=x0, xlim=range(x0),
      main="Pareto Distribution")
points(x0, dpareto(x0, lambda0, a0), type="l", col="red")

#-----
#   Calculate probabilities and quantiles for Pareto Distribution

a0 <- 2
lambda0 <- 2
prob <- ppareto(seq(a0, 8), lambda0, a0)
quan <- qpareto(prob, lambda0, a0)
print(quan)

#-----
#   Simulate and plot histogram with density for tapered Pareto Distribution

a0 <- 2
lambda0 <- 2
theta0 <- 3
x <- rtappareto(1000, lambda=lambda0, theta=theta0, a=a0)
x0 <- seq(a0, max(x)+0.1, length=100)
hist(x, freq=FALSE, breaks=x0, xlim=range(x0),
      main="Tapered Pareto Distribution")
points(x0, dtappareto(x0, lambda0, theta0, a0), type="l", col="red")

#-----
#   Calculate probabilities and quantiles for tapered Pareto Distribution

a0 <- 2
lambda0 <- 2
theta0 <- 3
prob <- ptappareto(seq(a0, 8), lambda0, theta0, a0)
quan <- qtappareto(prob, lambda0, theta0, a0)
print(quan)

#-----
#   Calculate log-likelihood for tapered Pareto Distribution
#   note the Hessian and gradient attributes

a0 <- 2
lambda0 <- 2
theta0 <- 3
x <- rtappareto(1000, lambda=lambda0, theta=theta0, a=a0)
LL <- ltappareto(x, lambda=lambda0, theta=theta0, a=a0)
print(LL)

```

Description

This function calculates the value of the ground intensity of a time-magnitude Epidemic Type Aftershock Sequence (ETAS) model. Spatial coordinates of the events are not taken into account.

Usage

```
etas_gif(data, evalpts, params, TT=NA, tplus=FALSE)
```

Arguments

data	a data frame containing the event history, where each row represents one event. There must be columns named "time", usually the number of days from some origin; and "magnitude" which is the event magnitude less the magnitude threshold, i.e. $M_i - M_0$.
evalpts	a vector , matrix or data.frame . If a vector, the elements will be assumed to represent the required evaluation times. Other objects must include a column named "time" that can be referred to as <code>evalpts[, "time"]</code> , at which the intensity function will be evaluated.
params	vector of parameter values in the following order: (μ, A, α, c, p) .
TT	vector of length 2, being the time interval over which the integral of the ground intensity function is to be evaluated.
tplus	logical, $\lambda_g(t \mathcal{H}_t)$ is evaluated as $\lambda_g(t^+ \mathcal{H}_t)$ if TRUE, else $\lambda_g(t^- \mathcal{H}_t)$.

Details

The ETAS model was proposed by Ogata (1988, 1998, 1999) for the modelling of earthquake mainshock-aftershock sequences. The form of the ground intensity function used here is given by

$$\lambda_g(t|\mathcal{H}_t) = \mu + A \sum_{i:t_i < t} e^{\alpha(M_i - M_0)} \left(1 + \frac{t - t_i}{c}\right)^{-p},$$

where t_i denotes the event times and the summation is taken over those i such that $t_i < t$.

Value

Two usages are as follows.

```
etas_gif(data, evalpts, params, tplus=FALSE)
etas_gif(data, evalpts=NULL, params, TT)
```

The first usage returns a vector containing the values of $\lambda_g(t)$ evaluated at the specified points. In the second usage, it returns the value of the integral.

Function Attributes

rate is "decreasing".

See Also

General details about the structure of ground intensity functions are given in the topic [gif](#).

Examples

```
# Tangshan: ground intensity and magnitude time plots

data(Tangshan)
p <- c(0.007, 2.3, 0.98, 0.008, 0.94)
bvalue <- 1
TT <- c(0, 4018)

x <- mpp(data=Tangshan,
         gif=etas_gif,
         mark=list(rexp_mark, rexp_mark),
         params=p,
         gmap=expression(params),
         mmap=expression(bvalue*log(10)),
         TT=TT)

par.default <- par(mfrow=c(1,1), mar=c(5.1, 4.1, 4.1, 2.1))
par(mfrow=c(2,1), mar=c(4.1, 4.1, 0.5, 1))

plot(x, log=TRUE, xlab="")

plot(Tangshan$time, Tangshan$magnitude+4, type="h",
     xlim=c(0, 4018),
     xlab="Days Since 1 January 1974", ylab="Magnitude")

par(par.default)
```

 etas_spatial

Template Function for Spatial ETAS

Description

This is a template function for a spatial ETAS model. The spatial component is based on the shape of the bivariate normal density function. *This function is currently in development and may change, see “Warnings” below.*

Usage

```
etas_normal0(data, evalpts, params, fixedparams, TT=NA,
             tplus=FALSE)
```

Arguments

data	a data frame containing the event history, where each row represents one event. There must be columns named "time", usually the number of days from some origin; "magnitude" which is the event magnitude less the magnitude threshold, i.e. $M_i - M_0$; "latitude", denoted below as y ; and "longitude", denoted below as x .
evalpts	a <code>matrix</code> or <code>data.frame</code> containing columns named "time", "magnitude", "latitude", and "longitude".
params	vector of parameter values in the following order: $(\mu, A, \alpha, c, p, d_x, d_y, \beta)$.
fixedparams	vector of values defining the spatial boundaries (degrees) of the region, assumed to be rectangular: (X_1, X_2, Y_1, Y_2) .
TT	vector of length 2, being the time interval over which the integral of the ground intensity function is to be evaluated.
tplus	logical, $\lambda(t, x, y \mathcal{H}_t)$ is evaluated as $\lambda(t^+, x, y \mathcal{H}_t)$ if TRUE, else $\lambda(t^-, x, y \mathcal{H}_t)$.

Details

Let

$$f(t) = (1 + t/c)^{-p}$$

and

$$g(x, y) = \exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right),$$

where $\sigma_x^2 = d_x e^{\alpha(M_i - M_0)}$ and $\sigma_y^2 = d_y e^{\alpha(M_i - M_0)}$. Then the conditional intensity function is

$$\lambda(t, x, y|\mathcal{H}_t) = \mu + A \sum_{i:t_i < t} e^{\beta(M_i - M_0)} g(x - x_i, y - y_i) f(t - t_i).$$

Note that the parameters μ and A are not the same as those in `etas_gif`. That function is the spatial integral of the above, and so, for example, $\tilde{\mu} = (X_2 - X_1)(Y_2 - Y_1)\mu$ where $\tilde{\mu}$ is the μ in `etas_gif`. Ogata & Zhuang (2006) have discussed this and other formulations of the spatial ETAS models.

This can be thought of as a template function, i.e. a function that can be used to define the required spatial intensity function. It is probably most efficient to embed the boundaries of the region under analysis into the function. Say we want the scaling for latitude and longitude to be the same (i.e. $d = d_x = d_y$), $\beta = 0$, and the analysed region is contained within 164°E–182°E, and 48°S–35°S. This is then the same as case (5) in Ogata & Zhuang (2006) with their S_i being the identity matrix. The conditional intensity function would then be defined as follows.

```
etas_tmp <- function(data, evalpts, params, TT=NA, tplus=FALSE)
  etas_normal0(data, evalpts, params=c(params, params[6], 0),
    fixedparams=c(164, 182, -48, -35),
    TT=TT, tplus=tplus)
```

The vector `params` of the new function `etas_tmp` is $(\mu, A, \alpha, c, p, d)$.

Value

The function has two forms of usage as described in [gif](#). The first usage returns a vector containing the values of $\lambda(t, x, y)$ evaluated at the specified points. In the second usage, it returns the value of the space-time integral.

Warnings

This function is still in development and may often change. The code may well be inefficient, though the current purpose is to keep it transparent and hence easier to change.

When this function is used to build a model object ([mpp](#)), the model object will work correctly with the generic functions [summary](#), [residuals](#), [neglogLik](#) and [logLik](#), but will not yet work with [simulate](#) and [plot](#).

See Also

General details about the structure of ground intensity functions are given in the topic [gif](#).

Examples

```
data(Phuket)
# magnitudes are rounded to 1 dp
Phuket$magnitude <- Phuket$magnitude - 4.95

TT <- c(0, 1827)
params <- c(0.000121, 34.66, 0.61, 0.0081, 1.0, 0.023)

# params <- c(nrow(Phuket)/(TT[2]-TT[1])/2, 100, 1, 0.01, 1.2, 0.01)

etas_tmp <- function(data, evalpts, params, TT = NA, tplus = FALSE){
  # params = c(mu, A, alpha, c, p, dx, dy, beta)
  # dx=dy and alpha=beta
  params <- params[c(1:5, 6, 6, 3)]
  etas_normal0(data, evalpts, params, c(89, 105, -5, 16), TT, tplus)
}

x <- mpp(data=Phuket,
         gif=etas_tmp,
         mark=list(NULL, NULL),
         params=params,
         gmap=expression(params),
         mmap=NULL,
         TT=TT)

allmap <- function(y, p){
  y$params <- exp(p)
  return(y)
}

# Note: the "Not run" blocks below are not run during package checks
# as the makeSOCKcluster definition is specific to my network,
# modify accordingly if you want parallel processing.
```

```

cl <- NULL
## Not run:
if (require(snow)){
  cl <- makeSOCKcluster(c("localhost", "horoeka.localdomain",
                        "horoeka.localdomain", "localhost"))
  clusterExport(cl, c("etas_normal0"))
}
## End(Not run)

initial <- log(params)
# set iterlim larger to converge satisfactorily
# it will take about 10 minutes
z <- nlm(neglogLik, initial, object=x, pmap=allmap,
        print.level=2, iterlim=5, SNOWcluster=cl)

# z <- nlm(neglogLik, params, object=x, typsize=abs(params),
#         print.level=2, iterlim=5, hessian=TRUE,
#         SNOWcluster=cl)
# stderr <- sqrt(diag(solve(z$hessian)))

## Not run:
if (!is.null(cl)){
  stopCluster(cl)
  rm(cl)
}
## End(Not run)

x <- allmap(x, z$estimate)
print(z$code)

print(x$params)
# MLE = (0.0001210 34.662 0.6143 0.008103 1.0012 0.02294)

print(logLik(x))

```

gif

General Notes on Ground Intensity Functions

Description

This page contains general notes about the required structure of ground intensity functions (including those that are not conditional on their history) to be used with this package.

Forms of Usage

The usage of a ground intensity function takes two forms, one to evaluate the gif at specified evalpts, or to evaluate the integral of the gif on the interval TT, each shown below, respectively.

```
gif(data, evalpts, params, tplus=FALSE)
```

```
gif(data, NULL, params, TT)
```

Arguments

All ground intensity functions should be defined to contain the following arguments, in the order below, even though they may not be required (see Details below).

`data` a data frame containing the history of the process, denoted below as \mathcal{H}_t . It should contain all variables that are required to evaluate the `gif` function, though can contain others too. No history is represented as `NULL`.

`evalpts` a object containing the values at which the `gif` function is to be evaluated, consistent with what is required by the `gif` function.

`params` vector containing values of the parameters required by the `gif` function.

`TT` vector of length 2, being the time interval over which the integral of the ground intensity function is to be evaluated.

`tplus` logical, $\lambda_g(t|\mathcal{H}_t)$ is evaluated as $\lambda_g(t^+|\mathcal{H}_t)$ if `TRUE`, else $\lambda_g(t^-|\mathcal{H}_t)$. It is important if a “jump” occurs at t .

Details

Note that the `gif` functions not only evaluate values of $\lambda_g(t_i|\mathcal{H}_t)$, but also the integral. The value of the ground intensity function is returned at each time point specified in `evalpts` when `TT==NA`. If `TT` is not missing, the integral between `TT[1]` and `TT[2]` of the ground intensity function is calculated. In this last situation, anything assigned to the argument `evalpts` will have no effect.

At the moment, we have the following types of processes: those jump processes that are conditional on their history (`etas_gif`, `srm_gif`, `linkstrm_gif`), and non-homogeneous Poisson processes that are not conditional on their history (`simple_gif`). Another case is where we have a collection of point like “regions” (or lattice nodes), each with their own ground intensity function, but where each is also dependent on what is happening in the other regions (`linkstrm_gif`).

Functions have been given an attribute “rate”, taking the values of “bounded”, “decreasing” or “increasing”. This is used within the simulation function `simulate.mpp` which uses the thinning method. This method requires a knowledge of the maximum of $\lambda_g(t|\mathcal{H}_t)$ in a given interval. The argument `tplus` is also used by the simulation routine, where it is necessary to determine the value of the intensity immediately after a simulated event.

Value

The returned value is either $\lambda_g(t_i|\mathcal{H}_t)$, where the t_i are specified within `evalpts`; or

$$\int \lambda_g(t|\mathcal{H}_t)dt$$

where the limits of the integral are specified by the function argument `TT`.

Function Attributes

Each function should have some of the following attributes if it is to be used in conjunction with `residuals.mpp` or `simulate.mpp`:

`rate` must be specified if the default method for `simulate.mpp` is to be used. Takes the values “bounded”, “decreasing” or “increasing”; see Details.

`regions` an expression giving the number of regions; required with `linkstrm_gif`.

See Also

[etas_gif](#), [expfourier_gif](#), [exppoly_gif](#), [fourier_gif](#), [linksrn_gif](#), [poly_gif](#), [simple_gif](#), [srm_gif](#)

Examples

```
# Ogata's Data: ground intensity function
# evaluate lambda_g(t) at certain times

data(Ogata)

p <- c(0.02, 70.77, 0.47, 0.002, 1.25)
times <- sort(c(seq(0, 800, 0.5), Ogata$time))
TT <- c(0, 800)

plot(times, log(etas_gif(Ogata, times, params=p)), type="l",
      ylab=expression(paste(log, " ", lambda[g](t))),
      xlab=expression(t), xlim=TT)

# Evaluate the integral
# The first form below is where the arguments are in their
# default positions, the 2nd is where they are not, hence
# their names must be specified

print(etas_gif(Ogata, NULL, p, TT))
# or
print(etas_gif(Ogata, params=p, TT=TT))
```

linksrn

Linked Stress Release Model Object

Description

Creates a point process model object with class "linksrn".

Usage

```
linksrn(data, gif, marks, params, gmap, mmap, TT)
```

Arguments

data	a data.frame containing the history of the process, denoted below as \mathcal{H}_t . It should contain all variables that are required to evaluate the <code>gif</code> function and the mark distribution, though can contain others too. No history is represented as <code>NULL</code> .
gif	ground intensity function. At this stage, this can only be linksrn_gif or modifications of that function; see “Details” below.
marks	mark distribution. See topic marks for further details.

params	numeric vector of <i>all</i> model parameters.
gmap	expression , maps the model parameters (params) into the parameter subspace of the ground intensity function; see “Details” below.
mmap	expression , maps the model parameters (params) into the parameter subspace of the mark distribution; see “Details” below.
TT	vector of length 2, being the time interval over which the integral of the ground intensity function is to be evaluated.

Details

The linked stress release model has a slightly peculiar structure which makes it difficult to fit into the `mpp` class. While the region should be thought of as a mark, it is completely defined by the function `linkstrm_gif`, and hence from the programming perspective the `region` mark is really tied in with the `gif` function. Hence at the moment, the linked stress release model is treated as a special case. There may be other models that could be grouped into this class.

Examples

```
p <- c(-1.5, -1.5, 0.01, 0.03, 2, -0.5, 0.2, 1, 1*log(10), 3)
TT <- c(0, 1000)

rexp trunc_mark <- function(ti, data, params){
  x <- rexp(n=1, params[1])
  x[x > params[2]] <- params[2]
  names(x) <- "magnitude"
  return(x)
}

x <- linkstrm(data=NULL,
             gif=linkstrm_gif,
             mark=list(NULL, rexp trunc_mark),
             params=p,
             gmap=expression(params[1:8]),
             mmap=expression(params[9:10]),
             TT=TT)

x <- simulate(x, seed=5)
print(logLik(x))

# estimate parameters
temp_map <- function(y, p){
  # map only gif parameters into model object
  y$params[1:8] <- p
  return(y)
}

# replace linkstrm_gif with linkstrm1_gif (faster)
if (exists('St1')) rm(St1)
if (exists('St2')) rm(St2)
x$gif <- linkstrm1_gif
```

```

weight <- c(0.1, 0.1, 0.005, 0.005, 0.1, 0.1, 0.1, 0.1)
z <- nlm(neglogLik, p[1:8], object=x, pmap=temp_map,
        hessian=TRUE, gradtol=1e-08, steptol=1e-10,
        print.level=2, iterlim=500, tysize=weight)

param.names <- c("a1", "a2", "b1", "b2", "c11", "c12", "c21", "c22")
param.est <- cbind(p[1:8], z$estimate, sqrt(diag(solve(z$hessian))))
dimnames(param.est) <- list(param.names,
                            c("Actual", "Estimate", "StdErr"))

print(param.est)

# place parameter estimates into model object
x <- temp_map(x, z$estimate)

# plot ground intensity function
par.default <- par(mfrow=c(2,1), mar=c(4.1, 4.1, 0.5, 1))
x$gif <- linksrm_gif
plot(x, 1, xlab="")
plot(x, 2)
par(par.default)

# plot "residuals" for each region
tau <- residuals(x)
par(mfrow=c(2,1))
plot(tau[[1]], ylab="Transformed Time",
      xlab="Event Number", main="Region 1")
abline(a=0, b=1, lty=2, col="red")
plot(tau[[2]], ylab="Transformed Time",
      xlab="Event Number", main="Region 2")
abline(a=0, b=1, lty=2, col="red")
par(mfrow=c(1,1))

```

linksrn_convert *Parameter Conversion for Linked Stress Release Model*

Description

Converts parameter values between two different parameterisations (described in Details below) of the linked stress release model.

Usage

```
linksrn_convert(params, abc=TRUE)
```

Arguments

params	a vector of parameter values of length $n^2 + 2n$, where n is the number of regions in the model.
abc	logical. If TRUE (default), then the input value of params is that of the abc parameterisation. See Details for further explanation.

Details

If `abc == TRUE`, the conditional intensity for the i th region is assumed to have the form

$$\lambda_g(t, i | \mathcal{H}_t) = \exp \left\{ a_i + b_i \left[t - \sum_{j=1}^n c_{ij} S_j(t) \right] \right\}$$

with `params = (a1, ..., an, b1, ..., bn, c11, c12, c13, ..., cnn)`.

If `abc == FALSE`, the conditional intensity for the i th region is assumed to have the form

$$\lambda_g(t, i | \mathcal{H}_t) = \exp \left\{ \alpha_i + \nu_i \left[\rho_i t - \sum_{j=1}^n \theta_{ij} S_j(t) \right] \right\}$$

where $\theta_{ii} = 1$ for all i , $n = \sqrt{\text{length}(\text{params}) + 1} - 1$, and `params`

$$= (\alpha_1, \dots, \alpha_n, \nu_1, \dots, \nu_n, \rho_1, \dots, \rho_n, \theta_{12}, \theta_{13}, \dots, \theta_{1n}, \theta_{21}, \theta_{23}, \dots, \theta_{n, n-1}).$$

Value

A list object with the following components is returned:

<code>params</code>	vector as specified in the function call.
<code>a</code>	vector of length n as in the <code>abc</code> parameterisation.
<code>b</code>	vector of length n as in the <code>abc</code> parameterisation.
<code>c</code>	n by n matrix as in the <code>abc</code> parameterisation.
<code>alpha</code>	vector of length n as in the alternative parameterisation.
<code>nu</code>	vector of length n as in the alternative parameterisation.
<code>rho</code>	vector of length n as in the alternative parameterisation.
<code>theta</code>	n by n matrix with ones on the diagonal as in the alternative parameterisation.

See Also

[linksrml_gif](#)

linksrml_gif

Ground Intensity for Linked Stress Release Model

Description

Calculates the value of the ground intensity of a Linked Stress Release Model (LSRM). This model allows for multiple linked regions, where the stress can be transferred between the regions.

Usage

```
linksrml_gif(data, evalpts, params, TT=NA, tplus=FALSE, eta=0.75)
linksrml1_gif(data, evalpts, params, TT=NA, tplus=FALSE, eta=0.75)
```

Arguments

data	a data frame containing the event history, where each row represents one event. There must be columns named "time", usually the number of days from some origin; "magnitude" which is the event magnitude less the magnitude threshold, i.e. $M_k - M_0$; and "region" which are consecutively numbered starting at 1.
evalpts	a <code>matrix</code> or <code>data.frame</code> . It must include two columns named "time" and "region" that can be referred to as <code>evalpts[, "time"]</code> and <code>evalpts[, "region"]</code> , respectively. The function will be evaluated at these points.
params	vector of parameters of length $n^2 + 2n$, where n is the number of regions, for the proposed LSRM in the following order: $(a_1, \dots, a_n, b_1, \dots, b_n, c_{11}, c_{12}, c_{13}, \dots, c_{nn}).$
TT	vector of length 2, being the time interval over which the integral of the ground intensity function is to be evaluated.
tplus	logical, $\lambda_g(t, i \mathcal{H}_t)$ is evaluated as $\lambda_g(t^+, i \mathcal{H}_t)$ if TRUE, else $\lambda_g(t^-, i \mathcal{H}_t)$.
eta	a scalar used in the stress calculations, see Details below.

Details

The ground intensity for the i th region is assumed to have the form

$$\lambda_g(t, i|\mathcal{H}_t) = \exp \left\{ a_i + b_i \left[t - \sum_{j=1}^n c_{ij} S_j(t) \right] \right\}$$

with `params = c(a1, ..., an, b1, ..., bn, c11, c12, c13, ..., cnn)`; and

$$S_j(t) = \sum_k 10^{\eta(M_k - M_0)},$$

where the summation is taken over those events in region j with time $t_k < t$. This model has been discussed by Bebbington & Harte (2001, 2003). The default value of $\eta = \text{eta} = 0.75$.

The difference between `linksrml_gif` and `linksrml1_gif` is that the stress reduction matrices `St1` and `St2` (internal to both functions) are calculated every time that the function is called in the case of `linksrml_gif`. If the event *history* is not changing between successive calls (e.g. parameter estimation), then this is unnecessary. However, in a simulation, the history changes with the addition of each new event. The function `linksrml1_gif` checks to see whether the matrices `St1` and `St2` exist. If so, these existing matrices are used, and new ones are not calculated. Therefore when using `linksrml1_gif` for parameter estimation, one **must** check for the existence of such matrices, and delete upon starting to fit a new model, for example:

```
if (exists("St1")) rm(St1)
```

```
if (exists("St2")) rm(St2)
```

Value

Two usages of each function are as follows.

```
linksrml_gif(data, evalpts, params, tplus=FALSE, eta=0.75)
linksrml_gif(data, evalpts=NULL, params, TT, eta=0.75)
```

```
linksrml_gif(data, evalpts, params, tplus=FALSE, eta=0.75)
linksrml_gif(data, evalpts=NULL, params, TT, eta=0.75)
```

The first usage returns a vector containing the values of $\lambda_g(t, i)$ evaluated at the specified “time-region” points. In the second usage, it returns a vector containing the value of the integral for each region.

Function Attributes

rate is "increasing".

regions is `expression(sqrt(length(params) + 1) - 1)`.

Problems and Inconsistencies

It would be better if the objects `St1` and `St2` could be dealt with in a tidier manner. This is the only difference between `linksrml_gif` and `linksrml_gif`.

See Also

General details about the structure of ground intensity functions are given in the topic [gif](#).

logLik

Log Likelihood of a Point Process Model

Description

Calculates the log-likelihood of a point process. Provides methods for the generic function `logLik`.

Usage

```
## S3 method for class 'mpp':
logLik(object, SNOWcluster=NULL, ...)
## S3 method for class 'linksrml':
logLik(object, ...)
```

Arguments

<code>object</code>	an object with class " <code>mpp</code> " or " <code>linksrml</code> ".
<code>SNOWcluster</code>	an object of class " <code>cluster</code> " created by the package <code>snow</code> ; default is <code>NULL</code> . Enables parallel processing if not <code>NULL</code> . See “Parallel Processing” below for further details.
<code>...</code>	other arguments.

Value

Value of the log-likelihood.

Parallel Processing

Parallel processing can be enabled to calculate the term $\sum_i \log \lambda_g(t_i | \mathcal{H}_{t_i})$. Generally, the amount of computational work involved in calculating $\lambda_g(t | \mathcal{H}_t)$ is much greater if there are more events in the process history prior to t than in the case where there are fewer events. Given m nodes, the required evaluation points are divided into m groups, not of equal number, but of roughly equal amount of work, taking into account the amount of “history” prior to each event. Currently this division is very basic, and assumes that all nodes run at a roughly comparable speed.

If the communication between nodes is slow and the dataset is small, then the time taken to allocate the work to the various nodes may in fact take more time than simply using one processor to perform all of the calculations.

The required steps in initiating parallel processing are as follows.

```
# load the "snow" package
library(snow)

# define the SNOW cluster object, e.g. a SOCK cluster
# where each node has the same R installation.
cl <- makeSOCKcluster(c("localhost", "horoeka.localdomain",
                       "horoeka.localdomain", "localhost"))

# A more general setup: Totara is Fedora, Rimu is Debian:
# Use 2 procesors in Totara, 1 in Rimu:
totara <- list(host="localhost",
              rscript="/usr/lib/R/bin/Rscript",
              snowlib="/usr/lib/R/library")
rimu <- list(host="rimu.localdomain",
            rscript="/usr/lib/R/bin/Rscript",
            snowlib="/usr/local/lib/R/site-library")
cl <- makeCluster(list(totara, totara, rimu), type="SOCK")

# then define the required model object, e.g. see topic "mpp"
# say the model object is called x

# then calculate the log-likelihood as
print(logLik(x, SNOWcluster=cl))

# stop the R jobs on the slave machines
stopCluster(cl)
```

Note that the communication method does not need to be SOCKS; see the **snow** package documentation, topic [snow-startstop](#), for other options. Further, if some nodes are on other machines, the firewalls may need to be tweaked. The master machine initiates the R jobs on the slave machines by communicating through port 22 (use of security keys are needed rather than passwords),

and subsequent communications through port 10187. Again, these details can be tweaked in the options settings within the **snow** package.

Examples

```
# SRM: magnitude iid exponential with bvalue=1

TT <- c(0, 1000)
bvalue <- 1
params <- c(-2.5, 0.01, 0.8, bvalue*log(10))

# calculate log-likelihood excluding the mark density term
x1 <- mpp(data=NULL,
          gif=srm_gif,
          mark=list(NULL, rexp_mark),
          params=params,
          gmap=expression(params[1:3]),
          mmap=expression(params[4]),
          TT=TT)
x1 <- simulate(x1, seed=5)
print(logLik(x1))

# calculate log-likelihood including the mark density term
x2 <- mpp(data=x1$data,
          gif=srm_gif,
          mark=list(dexp_mark, rexp_mark),
          params=params,
          gmap=expression(params[1:3]),
          mmap=expression(params[4]),
          TT=TT)
print(logLik(x2))

# contribution from magnitude marks
print(sum(dexp(x1$data$magnitude, rate=bvalue*log(10), log=TRUE)))
```

marks

Mark Distributions

Description

Contains densities and random number generators for some example mark distributions. The mark distributions can be multi-dimensional. Users can write their own functions, and general rules are given under “Details”.

Usage

```
dexp_mark(x, data, params)
rexp_mark(ti, data, params)
```

Arguments

<code>ti</code>	scalar, time of an event.
<code>x</code>	a <code>data.frame</code> of mark values at given times, often a subset of the history.
<code>data</code>	a <code>data.frame</code> containing the history of the process, denoted below as \mathcal{H}_t .
<code>params</code>	numeric vector of parameters.

Details

The example functions listed under “Usage” calculate the *logarithm* of the (mark) density and simulate earthquake magnitudes assuming an exponential distribution that is independent of the history of the process. This corresponds to the Gutenberg-Richter law. They assume that the history contains a variable named "magnitude".

All mark densities and random number generators must have the three arguments as shown in the examples above. Multi-parameter distributions have their parameters specified as a vector in the `params` argument. Other ancillary data or information can be passed into the function non formally, though one needs to be careful about possible conflict with names of other objects.

Value

Mark density functions must return a vector with length being equal to the number of rows in `x`. Each element contains the *logarithm* of the joint density of the marks corresponding to each time (row) in `x`.

The random number generator simulates each mark for a *single value* of `ti`. It must return a `list` of simulated marks corresponding to the specified time `ti`. Further, the list must have its elements named the same as those in the history. Note that each component in the list will be of length one. A list is used (rather than a vector) because it allows marks to be character as well as numeric.

Example 1

This is an example where the density of the magnitude distribution is dependent on the value of the ground intensity function (assumed to be `etas_gif`), and in this case, the history of the process. The history is assumed to contain a variable named "magnitude". In this mark distribution, it is assumed that after large events, there is a deficit of smaller magnitude events with more larger magnitude events. It has seven parameters with parameters p_1, \dots, p_5 relating to `etas_gif`. It assumes that the magnitude distribution is gamma (`GammaDist`), with a shape parameter given by

$$\text{shape} = 1 + \sqrt{\lambda_g(t|\mathcal{H}_t)} p_7,$$

where p_7 ($p_7 > 0$) is a free estimable parameter, and parameter p_6 is the scale parameter. Hence when $\lambda_g(t|\mathcal{H}_t)$ is small, the magnitude distribution returns to an approximate exponential distribution with an approximate rate of p_6 (i.e. Gutenberg Richter law).

```
dexample1_mark <- function(x, data, params){
  lambda <- etas_gif(data, x[, "time"], params=params[1:5])
  y <- dgamma(x[, "magnitude"], rate=params[6],
             shape=1+sqrt(lambda)*params[7], log=TRUE)
  return(y)
```

```

}

rexample1_mark <- function(ti, data, params){
  # Gamma distribution
  # exponential density when params[7]=0
  lambda <- etas_gif(data, ti, params=params[1:5])
  y <- rgamma(1, shape=1+sqrt(lambda)*params[7],
             rate=params[6])
  return(list(magnitude=y))
}

```

Example 2

This an example of a 3-D mark distribution. Each component is independent of each other and the history, hence the arguments `ti` and `data` are not utilised in the functions. The history is assumed to contain the three variables "magnitude", "longitude" and "latitude". The event magnitudes are assumed to have an exponential distribution with rate `params[1]`, and the longitudes and latitudes to have normal distributions with means `params[2]` and `params[3]`, respectively.

```

dexample2_mark <- function(x, data, params)
  return(dexp(x[, "magnitude"], rate=params[1], log=TRUE) +
         dnorm(x[, "longitude"], mean=params[2], log=TRUE) +
         dnorm(x[, "latitude"], mean=params[3], log=TRUE))

rexample2_mark <- function(ti, data, params)
  return(list(magnitude=rexp(1, rate=params[1]),
             longitude=rnorm(1, mean=params[2]),
             latitude=rnorm(1, mean=params[3])))

```

mpp

Marked Point Process Object

Description

Creates a marked point process model object with class "mpp".

Usage

```
mpp(data, gif, marks, params, gmap, mmap, TT)
```

Arguments

`data` a `data.frame` containing the history of the process, denoted below as \mathcal{H}_t . It should contain all variables that are required to evaluate the `gif` function and the mark distribution, though can contain others too. No history is represented as `NULL`.

<code>gif</code>	ground intensity function. See topic gif for further details.
<code>marks</code>	a list containing the mark distribution. The first component (i.e. <code>marks[[1]]</code>) is the mark density and the second (i.e. <code>marks[[2]]</code>) is the random number generator. If either of these functions are not required, the particular component can be set to <code>NULL</code> . See topic marks for further details.
<code>params</code>	numeric vector of <i>all</i> model parameters.
<code>gmap</code>	expression , maps the model parameters (<code>params</code>) into the parameter subspace of the ground intensity function; see “Details” below.
<code>mmap</code>	expression , maps the model parameters (<code>params</code>) into the parameter subspace of the mark distribution; see “Details” below.
<code>TT</code>	vector of length 2, being the time interval over which the integral of the ground intensity function is to be evaluated.

Details

Let $\lambda_g(t|\mathcal{H}_t)$ denote the ground intensity function and $f(y|\mathcal{H}_t)$ denote the joint mark densities, where $y \in \mathcal{Y}$. The log-likelihood of a marked point process is given by

$$\log L = \sum_i \log \lambda_g(t_i|\mathcal{H}_{t_i}) + \sum_i \log f(y_i|\mathcal{H}_{t_i}) - \int \lambda_g(t|\mathcal{H}_t) dt,$$

where the summation is taken over those events contained in the interval $(TT[1], TT[2])$, and the integral is also taken over that interval. However, all events in the data frame `data` before t , even those before $TT[1]$, form the history of the process \mathcal{H}_t . This allows an initial period for the process to reach a “steady state” or “equilibrium”.

The parameter spaces of the ground intensity function and mark distribution are not necessarily disjoint, and can have common parameters. Hence, when the model parameters are estimated, these relationships must be known, and are specified by the arguments `gmap` and `mmap`. The mapping expressions can also contain arithmetic expressions. The i th element in the `params` argument is addressed in the expressions as `params[i]`. Here is an example of a five parameter model, where the `gif` has 4 parameters, and the mark distribution has 2, with mappings specified as:

```
gmap = expression(c(params[1:3], exp(params[4]+params[5])))
mmap = expression(c(log(params[2]/3), params[5]))
```

Note the inclusion of the combine (`c`) function, because the [expression](#) must create a vector of parameters. Care must be taken specifying these expressions are they are embedded directly into the code of various functions.

Examples

```
data(Tangshan)

# increment magnitudes a fraction so none are zero
Tangshan[, "magnitude"] <- Tangshan[, "magnitude"] + 0.01

dmagn_mark <- function(x, data, params){
```

```

# Gamma distribution
# exponential density when params[7]=0
# See topic "marks" for further discussion
lambda <- etas_gif(data, x[, "time"], params=params[1:5])
y <- dgamma(x[, "magnitude"], shape=1+sqrt(lambda)*params[7],
            rate=params[6], log=TRUE)
return(y)
}

TT <- c(0, 4018)
params <- c(0.007, 2.3, 0.98, 0.008, 0.94, 1.25, 0.2)

x <- mpp(data=Tangshan,
         gif=etas_gif,
         mark=list(dmagn_mark, NULL),
         params=params,
         gmap=expression(params[1:5]),
         mmap=expression(params[1:7]),
         TT=TT)

allmap <- function(y, p){
  # one to one mapping, all p positive
  y$params <- exp(p)
  return(y)
}

# Parameters must be positive. Transformed so that nlm
# can use entire real line (no boundary problems, see
# topic "neglogLik" for further explanation).
z <- nlm(neglogLik, log(params), object=x, pmap=allmap,
        print.level=2, iterlim=500, tysize=abs(params))

x1 <- allmap(x, z$estimate)

# print parameter estimates
print(x1$params)

print(logLik(x))
print(logLik(x1))
plot(x1, log=TRUE)

```

neglogLik

Negative Log-Likelihood

Description

Calculates the log-likelihood multiplied by negative one. It is in a format that can be used with the functions `nlm` and `optim`.

Usage

```
neglogLik(params, object, pmap = NULL, SNOWcluster=NULL)
```

Arguments

<code>params</code>	a vector of revised parameter values.
<code>object</code>	an object of class "mpp".
<code>pmap</code>	a user provided function mapping the revised parameter values <code>params</code> into the appropriate locations in <code>object</code> . If <code>NULL</code> (default), an untransformed one to one mapping is used.
<code>SNOWcluster</code>	an object of class "cluster" created by the package <code>snow</code> ; default is <code>NULL</code> . Enables parallel processing if not <code>NULL</code> . See <code>logLik</code> for further details.

Details

This function can be used with the two functions `nlm` and `optim` (see “Examples” below) to maximise the likelihood function of a model specified in `object`. Both `nlm` and `optim` are *minimisers*, hence the “negative” log-likelihood. The topic `distribution` gives examples of their use in the relatively easy situation of fitting standard probability distributions to data assuming independence.

The maximisation of the model likelihood function can be restricted to be over a subset of the model parameters. Other parameters will then be fixed at the values stored in the model `object`. Let Θ_0 denote the full model parameter space, and let Θ denote the parameter sub-space ($\Theta \subseteq \Theta_0$) over which the likelihood function is to be maximised. The argument `params` contains values in Θ , and `pmap` is assigned a function that maps these values into the full model parameter space Θ_0 . See “Examples” below.

The mapping function assigned to `pmap` can also be made to impose restrictions on the domain of the parameter space Θ so that the minimiser cannot jump to values such that $\Theta \not\subseteq \Theta_0$. For example, if a particular parameter must be positive, one can work with a transformed parameter that can take any value on the real line, with the model parameter being the exponential of this transformed parameter. Similarly a modified logit like transform can be used to ensure that parameter values remain within a fixed interval with finite boundaries. Examples of these situations can be found in the topic `distribution` and the “Examples” below.

Value

Value of the log-likelihood times negative one.

See Also

`nlm`, `optim`

Examples

```
# SRM: magnitude is iid exponential with bvalue=1
# maximise exponential mark density too

TT <- c(0, 1000)
```

```

bvalue <- 1
params <- c(-2.5, 0.01, 0.8, bvalue*log(10))

x <- mpp(data=NULL,
         gif=srm_gif,
         mark=list(dexp_mark, rexp_mark),
         params=params,
         gmap=expression(params[1:3]),
         mmap=expression(params[4]),
         TT=TT)
x <- simulate(x, seed=5)

allmap <- function(y, p){
  #   map all parameters into model object
  #   transform exponential param so it is positive
  y$params[1:3] <- p[1:3]
  y$params[4] <- exp(p[4])
  return(y)
}

params <- c(-2.5, 0.01, 0.8, log(bvalue*log(10)))

z <- nlm(neglogLik, params, object=x, pmap=allmap,
        print.level=2, iterlim=500, typsize=abs(params))
print(z$estimate)

#   these should be the same:
print(exp(z$estimate[4]))
print(1/mean(x$data$magnitude))

```

NthChina

Historical Earthquakes of North China

Description

Contains 65 large historical earthquakes in North China between 1480 and 1997, as given by Bebbington & Harte (2003). Events are divided into 4 regions using the regionalisations given by Zheng & Vere-Jones (1991).

Usage

```
data(NthChina)
```

Format

A data frame with 65 rows, each representing an earthquake event, with the following variables:

time number of years since 1480 AD.

latitude number of degrees north.

longitude number of degrees east.
magnitude number of magnitude units *above* 6.
region 1, 2, 3, or 4; being the region of the event.

Ogata *Ogata's ETAS Test Data*

Description

A data frame containing the test data from Utsu and Ogata's (1997) software contained in the file `testetas.dat`. The first column is named "time", and the second column is named "magnitude".

Usage

```
data(Ogata)
```

Format

A data frame with 100 rows (earthquake events) in the time interval (0, 800). It contains the following variables:

time number of time units since time zero.
magnitude number of magnitude units *above* 3.5.

Examples

```
data(Ogata)
plot(Ogata$time, Ogata$magnitude + 3.5, type="h")
```

Phuket *Phuket Earthquake and Aftershock Sequence*

Description

The Phuket earthquake occurred on 26 December 2004 at 00:58:53.45 GMT. The `Phuket` data frame contains this event and its aftershock sequence.

Usage

```
data(Phuket)
```

Format

This data frame contains the following columns:

latitude number of degrees north.

longitude number of degrees east.

depth depth of event in kilometres.

mb body wave magnitude (m_b) rounded to one decimal place.

Ms surface wave magnitude (M_s) rounded to one decimal place.

magnitude event magnitude ($\max(m_b, M_s)$) rounded to one decimal place.

year year of event (numeric vector).

month month of event, 1 ... 12 (numeric vector).

day day of event, 1 ... 31 (numeric vector).

hour hour of event, 0 ... 23 (numeric vector).

minute minute of event, 0 ... 59 (numeric vector).

second second of event, 0 ... 59 (numeric vector).

time number of days (and fractions) from midnight on 1 January 2004.

Details

The `Phuket` data frame contains those events (1248) from the PDE Catalogue, within the spatial region 89°E – 105°E and 5°S – 16°N , with magnitude 5 or greater, occurring between midnight on 1 January 2004 and midnight on 1 January 2009 (1827 days later). The body wave magnitudes are determined by the amplitude of the initial primary wave, and these magnitudes tend to saturate for higher values. Consequently, the tabulated `magnitude` is taken as the maximum of the body wave magnitude (m_b) and surface wave magnitude (M_s).

Source

The data were extracted from the PDE (Preliminary Determination of Epicentres) catalogue provided by the US Geological Survey (<ftp://hazards.cr.usgs.gov/pde/>).

Examples

```
data(Phuket)
print(Phuket[1:10,])
```

 plot

Plot Point Process Ground Intensity Function

Description

Provides methods for the generic function `plot`.

Usage

```
## S3 method for class 'mpp':
plot(x, log=FALSE, ...)
## S3 method for class 'linkstrm':
plot(x, region, log=FALSE, ...)
```

Arguments

<code>x</code>	an object with class " <code>mpp</code> " or " <code>linkstrm</code> ".
<code>region</code>	scalar, specifies the required region.
<code>log</code>	plot $\log \lambda_g(t \mathcal{H}_t)$, default is FALSE.
<code>...</code>	other arguments.

Examples

```
data(Ogata)

p <- c(0.02, 70.77, 0.47, 0.002, 1.25)
TT <- c(0, 800)
bvalue <- 1

# Note that the plot function does not utilise the
# information about mark distributions, hence these
# arguments can be NULL

x <- mpp(data=Ogata,
        gif=etas_gif,
        mark=list(NULL, NULL),
        params=p,
        gmap=expression(params[1:5]),
        mmap=NULL,
        TT=TT)

plot(x, log=TRUE)
```

Description

This topic gives an introductory overview to the package **PtProcess**. Links are given to follow up topics where more detail can be found.

Introduction

This package contains routines for the fitting of *time dependent* point process models, particularly marked processes with “jumps”. These models have particular application to earthquake data. A detailed theoretical background to these and other point process models can be found in Daley & Vere-Jones (2003, 2008).

The direction of the development of the package has been influenced by our research on the application of point process models to seismology. The package was originally written for S-PLUS, being part of the Statistical Seismology Library (Harte, 1998; Brownrigg & Harte, 2005). The package **ptproc** by Peng (2002, 2003) analyses multi-dimensional point process models, and the package **spatstat** by Baddeley et al (2005, 2005a, 2008) analyses spatial point processes.

The topic [Changes](#) lists recent changes made to the package. Version 3 of the package has some major changes from Version 2, and code for Version 2 will not work in Version 3 without modification. Some examples giving the old code and the required new code are given in the topic [Changes](#). Changes made in Version 3 enable one to fit a more general class of model.

Classes of Point Process Models Analysed

The classes of models currently fitted by the package are listed below. Each are defined within an object that contains the data, current parameter values, and other model characteristics.

Marked Point Process Model: is described under the topic [mpp](#). This model can be simulated or fitted to data by defining the required model structure within an object of class "[mpp](#)".

Linked Stress Release Model: is described under the topic [linksrm](#). This model is slightly peculiar, and doesn't fit naturally in the [mpp](#) framework.

Main Tasks Performed by the Package

The main tasks performed by the package are listed below. These can be achieved by calling the appropriate generic function.

Simulation: can be performed by the function [simulate](#).

Parameter Estimation: can be achieved by using the function [neglogLik](#).

Model Residuals: can be calculated with the function [residuals](#).

Model Summary: can be extracted with the function [summary](#).

Log-Likelihood: can be calculated with the function [logLik](#).

Ground Intensity Plot: can be performed by the function [plot](#).

Organisation of Topics in the Package

Cited References: anywhere in the manual are only listed within this topic.

General Documentation: topics summarising general structure are indexed under the keyword “documentation” in the Index.

Acknowledgements

The package is based on an S-PLUS package which was commenced at Victoria University of Wellington in 1996. Contributions and suggestions have been made by many, including: Mark Bebbington, Ray Brownrigg, Edwin Choi, Robert Davies, Michael Eglinton, Dongfeng Li, Li Ma, Alistair Merrifield, Andrew Tokeley, David Vere-Jones, Wenzheng Yang, Leon Young, Irina Zhdanova and Jiancang Zhuang.

References

- Aalen, O.O. & Hoem, J.M. (1978). Random time changes for multivariate counting processes. *Scandinavian Journal of Statistics* **5**, 81–101.
- Baddeley, A. (2008). *Open source software for spatial statistics*. <http://www.spatstat.org/>.
- Baddeley, A. & Turner, R. (2005) Spatstat: an R package for analyzing spatial point patterns. *Journal of Statistical Software* **12(6)**, 1–42. <http://www.jstatsoft.org>, ISSN: 1548-7660.
- Baddeley, A.; Turner, R.; Moller, J. & Hazelton, M. (2005a). Residual analysis for spatial point processes (with discussion). *J. R. Statist. Soc. B* **67(5)**, 617–666. DOI: <http://dx.doi.org/10.1111/j.1467-9868.2005.00519.x>.
- Bebbington, M.S. & Harte, D.S. (2001). On the statistics of the linked stress release model. *Journal of Applied Probability* **38A**, 176–187. DOI: <http://dx.doi.org/10.1239/jap/1085496600>.
- Bebbington, M.S. & Harte, D.S. (2003). The linked stress release model for spatio-temporal seismicity: formulations, procedures and applications. *Geophysical Journal International* **154**, 925–946. DOI: <http://dx.doi.org/10.1046/j.1365-246X.2003.02015.x>.
- Brownrigg, R. & Harte, D.S. (2005). Using R for statistical seismology. *R News* **5(1)**, 31–35.
- Daley, D.J. & Vere-Jones, D. (2003). *An Introduction to the Theory of Point Processes. Volume I: Elementary Theory and Methods. Second Edition*. Springer-Verlag, New York. <http://books.google.com/?isbn=0387955410>.
- Daley, D.J. & Vere-Jones, D. (2008). *An Introduction to the Theory of Point Processes. Volume II: General Theory and Structure. Second Edition*. Springer-Verlag, New York. <http://books.google.com/?isbn=9780387213378>.
- Harte, D. (1998). Documentation for the Statistical Seismology Library. School of Mathematical and Computing Sciences Research Report No. 98–10 (Updated Edition June 1999), Victoria University of Wellington. (ISSN 1174–4545)
- Kagan, Y. & Schoenberg, F. (2001). Estimation of the upper cutoff parameter for the tapered Pareto distribution. *Journal of Applied Probability* **38A**, 158–175. DOI: <http://dx.doi.org/10.1239/jap/1085496599>.

- Lewis, P.A.W. & Shedler, G.S. (1979). Simulation of nonhomogeneous Poisson processes by thinning. *Naval Research Logistics Quarterly* **26**(3), 403–413. DOI: <http://dx.doi.org/10.1002/nav.3800260304>
- Ogata, Y. (1981). On Lewis' simulation method for point processes. *IEEE Transactions on Information Theory* **27**(1), 23–31.
- Ogata, Y. (1988). Statistical models for earthquake occurrences and residual analysis for point processes. *J. Amer. Statist. Assoc.* **83**(401), 9–27. DOI: <http://dx.doi.org/10.2307/2288914>.
- Ogata, Y. (1998). Space-time point-process models for earthquake occurrences. *Ann. Instit. Statist. Math. (Tokyo)* **50**(2), 379–402. DOI: <http://dx.doi.org/10.1023/A:1003403601725>.
- Ogata, Y. (1999). Seismicity analysis through point-process modeling: a review. *Pure and Applied Geophysics* **155**, 471–507. DOI: <http://dx.doi.org/10.1007/s000240050275>.
- Ogata, Y. & Zhuang, J.C. (2006). Space-time ETAS models and an improved extension. *Tectonophysics* **413**(1-2), 13–23. DOI: <http://dx.doi.org/10.1016/j.tecto.2005.10.016>.
- Peng, R. (2002). Multi-dimensional Point Process Models. Package “ptproc”, <http://www.biostat.jhsph.edu/~rpeng>.
- Peng, R. (2003). Multi-dimensional point process models in R. *Journal of Statistical Software* **8**(16), 1–27. <http://www.jstatsoft.org>.
- Reid, H.F. (1910). The mechanism of the earthquake. In *The California Earthquake of April 18, 1906, Report of the State Earthquake Investigation Commission* **2**, 16–28. Carnegie Institute of Washington, Washington D.C.
- Utsu, T. and Ogata, Y. (1997). Statistical analysis of seismicity. In: *Algorithms for Earthquake Statistics and Prediction* (Edited by: J.H. Healy, V.I. Keilis-Borok and W.H.K. Lee), pp 13–94. IASPEI, Menlo Park CA.
- Vere-Jones, D. (1978). Earthquake prediction - a statistician's view. *Journal of Physics of the Earth* **26**, 129–146.
- Vere-Jones, D.; Robinson, R. & Yang, W. (2001). Remarks on the accelerated moment release model: problems of model formulation, simulation and estimation. *Geophysical Journal International* **144**(3), 517–531. DOI: <http://dx.doi.org/10.1046/j.1365-246x.2001.01348.x>.
- Zheng, X.-G. & Vere-Jones, D. (1991). Application of stress release models to historical earthquakes from North China. *Pure and Applied Geophysics* **135**(4), 559–576. DOI: <http://dx.doi.org/10.1007/BF01772406>.
- Zhuang, J.C. (2006). Second-order residual analysis of spatiotemporal point processes and applications in model evaluation. *J. R. Statist. Soc. B* **68**(4), 635–653. DOI: <http://dx.doi.org/10.1111/j.1467-9868.2006.00559.x>.

Description

Provides methods for the generic function `residuals`.

Usage

```
## S3 method for class 'mpp':
residuals(object, ...)
## S3 method for class 'linkstrm':
residuals(object, ...)
```

Arguments

```
object      an object with class mpp or linkstrm.
...         other arguments.
```

Details

Let t_i be the times of the observed events. Then the transformed times are defined as

$$\tau_i = \int_0^{t_i} \lambda_g(t|\mathcal{H}_t) dt.$$

If the proposed point process model is correct, then the transformed time points will form a stationary Poisson process with rate parameter one. A plot of transformed time points versus the cumulative number of events should then roughly follow the straight line $y = x$. Significant departures from this line indicate a weakness in the model. Further details can be found in Ogata (1988) and Aalen & Hoem (1978).

See Baddeley et al (2005) and Zhuang (2006) for extensions of these methodologies.

Value

Returns a time series object with class "ts" in the case of `mpp`. In the case of `linkstrm` a list is returned with the number of components being equal to the number of regions, and with each component being a time series object.

Examples

```
TT <- c(0, 1000)
bvalue <- 1
params <- c(-2.5, 0.01, 0.8, bvalue*log(10))

x <- mpp(data=NULL,
         gif=srm_gif,
         mark=list(NULL, rexp_mark),
         params=params,
         gmap=expression(params[1:3]),
         mmap=expression(params[4]),
         TT=TT)
x <- simulate(x, seed=5)

tau <- residuals(x)

plot(tau, ylab="Transformed Time", xlab="Event Number")
abline(a=0, b=1, lty=2, col="red")
```

Description

The functions listed here are intensity functions that are not conditional on the history of the process. Each has exactly the same “Usage” and calling format (see section “Value”) as the function `simple_gif`. They are: `expfourier_gif`, `exppoly_gif`, `fourier_gif`, `poly_gif`, and `simple_gif`.

Usage

```
simple_gif(data, evalpts, params, TT=NA, tplus=FALSE)
```

Arguments

<code>data</code>	NULL or a data frame. The contents of this object are not used by these functions, though they retain this argument for consistency with other <code>gif</code> functions.
<code>evalpts</code>	a <code>vector</code> , <code>matrix</code> or <code>data.frame</code> . If a vector, the elements will be assumed to represent the required evaluation times. Other objects must include a column named "time" that can be referred to as <code>evalpts[, "time"]</code> , at which the intensity function will be evaluated.
<code>params</code>	vector of parameter values as required by the particular intensity function, see Details below.
<code>TT</code>	vector of length 2, being the time interval over which the integral of the intensity function is to be evaluated.
<code>tplus</code>	logical, $\lambda_g(t \mathcal{H}_t)$ is evaluated as $\lambda_g(t^+ \mathcal{H}_t)$ if TRUE, else $\lambda_g(t^- \mathcal{H}_t)$. Included for compatibility with others conditional intensity functions.

Details

The models are parameterised as follows.

`expfourier_gif` The vector of parameters is

$$(p, a_0, a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_n)$$

and the intensity function is

$$\lambda_g(t) = \exp \left\{ a_0 + \sum_{j=1}^n a_j \cos \left(\frac{2j\pi t}{p} \right) + \sum_{j=1}^n b_j \sin \left(\frac{2j\pi t}{p} \right) \right\}.$$

The length of `params` is $2n + 2$, and determines the order of the fitted Fourier series. The numbers of specified sine and cosine coefficients must be the same. The integral is evaluated using numerical integration, using the R function `integrate`.

exppoly_gif The vector of parameters is $(b_0, b_1, b_2, \dots, b_n)$ and the intensity function is

$$\lambda_g(t) = \exp \left\{ b_0 + \sum_{j=1}^n b_j t^j \right\}.$$

The length of `params` determines the order of the fitted polynomial. The integral is evaluated using numerical integration, using the R function `integrate`.

fourier_gif The Fourier intensity function is the same as `expfourier_gif`, except the intensity function omits the exponential, and the integration is performed explicitly.

poly_gif The polynomial intensity function is the same as `exppoly_gif`, except the intensity function omits the exponential, and the integration is performed explicitly.

simple_gif The intensity function is $\lambda_g(t) = a + bt^g$ and the vector of parameters is (a, b, g) .

Value

Two usages are as follows.

```
simple_gif(data, evalpts, params, tplus=FALSE)
simple_gif(data, evalpts=NULL, params, TT=NA)
```

The first usage returns a vector containing the values of $\lambda_g(t)$ evaluated at the specified points. In the second usage, it returns the value of the integral.

Function Attributes

rate is "bounded".

See Also

General details about the structure of conditional intensity functions are given in the topic `gif`.

Examples

```
expfourier_gif(NULL, c(1.1,1.2,1.3), c(2,3,1,2,3,4), TT=NA)
# Evaluates: lambda_g(t) = exp(3 + 1*cos(2*pi*t/2) + 2*cos(4*pi*t/2) +
#                               3*sin(2*pi*t/2) + 4*sin(4*pi*t/2))
# lambda_g(1.1) = 162.56331
# lambda_g(1.2) = 127.72599
# lambda_g(1.3) = 23.83979

expfourier_gif(NULL, NULL, c(2,3,1,2,3,4), TT=c(3,4))
# Let: lambda_g(t) = exp(3 + 1*cos(2*pi*t/2) + 2*cos(4*pi*t/2) +
#                               3*sin(2*pi*t/2) + 4*sin(4*pi*t/2))
# Evaluates: integral_3^4 lambda_g(t) dt = 46.21920
```

simulate

*Simulate a Point Process***Description**

Provides methods for the generic function `simulate`.

Usage

```
## S3 method for class 'mpp':
simulate(object, nsim=1, seed=NULL, max.rate=NA,
         stop.condition=NULL, ...)
## S3 method for class 'linkstrm':
simulate(object, nsim=1, seed=NULL, max.rate=NA,
         stop.condition=NULL, ...)
```

Arguments

<code>object</code>	an object with class <code>"mpp"</code> or <code>"linkstrm"</code> .
<code>nsim</code>	has no effect, and is only included for compatibility with the generic function <code>simulate</code> . See section “Length of Simulated Series” below for control information.
<code>seed</code>	seed for the random number generator.
<code>max.rate</code>	maximum rate, only used if the attribute of <code>object\$gif</code> is <code>"bounded"</code> . It is the maximum value of <code>object\$gif</code> on the simulation interval <code>object\$TT</code> .
<code>stop.condition</code>	a function returning a logical value. It is called after the addition of each simulated event. The simulation continues until either <code>object\$TT[2]</code> is exceeded or <code>stop.condition</code> returns <code>TRUE</code> . See section “Length of Simulated Series” below for further information.
<code>...</code>	other arguments.

Details

The *thinning method* (Ogata, 1981; Lewis & Shedler, 1979) is used to simulate a point process with specified ground intensity function. The method involves calculating an upper bound for the intensity function, simulating a value for the time to the next *possible* event using a rate equal to this upper bound, and then calculating the intensity at this simulated point; hence these “events” are simulated too frequently. The ratio of this rate with the upper bound is compared with a uniform random number to randomly determine whether the simulated time is retained or not (i.e. thinned).

The functions need to calculate an upper bound for the intensity function. The ground intensity functions will usually be discontinuous at event times, but may be monotonically increasing or decreasing at other times. The ground intensity functions have an attribute called `rate` with values of `"bounded"`, `"increasing"` or `"decreasing"`. This information is used to determine the required upper bounded.

The function `simulate.linksrm` is currently only used in conjunction with `linksrm_gif`, or a variation of that function. It expects the `gif` function to have an attribute called `regions`, which may be an expression, being the number of regions. The method used by the function `simulate.linksrm` also assumes that the function is “increasing” (i.e. rate, summed over all regions, apart from discontinuous jumps), hence a positive tectonic input over the whole system.

Value

The returned value is an object of the same class as `object`. It will contain all events prior to `object$TT[1]` in `object` and all subsequently simulated events.

Length of Simulated Series

The interval of time over which events are simulated is determined by `object$TT`. Simulation starts at `object$TT[1]` and stops at `object$TT[2]`. The “current” dataset will consist of all events prior to `object$TT[1]` in `object`, plus subsequently simulated events. A more complicated stopping condition can be formulated by using the argument `stop.condition`.

The argument `stop.condition` can be assigned a function that returns a logical value. The assigned function is a function of the “current” dataset. It is executed near the bottom of `simulate.mpp` (check by printing the function). Simulation will then continue until either the stopping condition has been met or the current time exceeds `object$TT[2]`.

For example, we may want to simulate until the first earthquake with a magnitude of 8. Assume that the current dataset contains a variable with name “magnitude” (untransformed). We would then assign `Inf` to `object$TT[2]`, and write this condition as a function:

```
stop.cond <- function(data) {
  n <- nrow(data)
  # most recent event is the nth
  return(data$magnitude[n] >= 8)
}
```

Examples

```
TT <- c(0, 1000)
bvalue <- 1
params <- c(-2.5, 0.01, 0.8, bvalue*log(10))

x <- mpp(data=NULL,
  gif=srm_gif,
  mark=list(NULL, rexp_mark),
  params=params,
  gmap=expression(params[1:3]),
  mmap=expression(params[4]),
  TT=TT)
x <- simulate(x, seed=5)

y <- hist(x$data$magnitude, xlab="Magnitude", main="")

# overlay with an exponential density
magn <- seq(0, 3, length.out=100)
```

```
points(magn, nrow(x$data) * (y$breaks[2] - y$breaks[1]) *
      dexp(magn, rate=1/mean(x$data$magnitude)),
      col="red", type="l")
```

srm_gif

*Conditional Intensity for Stress Release Model***Description**

This function calculates the value of the conditional intensity of a Stress Release Model (SRM). Spatial coordinates of the events are not taken into account.

Usage

```
srm_gif(data, evalpts, params, TT=NA, tplus=FALSE)
```

Arguments

data	a data frame containing the event history, where each row represents one event. There must be columns named “time”, usually the number of days from some origin; and “magnitude” which is the event magnitude less the magnitude threshold, i.e. $M_i - M_0$.
evalpts	a vector , matrix or data.frame . If a vector, the elements will be assumed to represent the required evaluation times. Other objects must include a column named "time" that can be referred to as <code>evalpts[, "time"]</code> , at which the intensity function will be evaluated.
params	vector of parameters for the proposed SRM model in the order (a, b, c) .
TT	vector of length 2, being the time interval over which the integral of the conditional intensity function is to be evaluated.
tplus	logical, $\lambda_g(t \mathcal{H}_t)$ is evaluated as $\lambda_g(t^+ \mathcal{H}_t)$ if TRUE, else $\lambda_g(t^- \mathcal{H}_t)$.

Details

Vere-Jones (1978) proposed the stress release model, being a stochastic version of elastic rebound theory (Reid, 1910). The SRM assumes a deterministic increase in stress over time, and a stochastic release through earthquake events. The conditional intensity function is

$$\lambda_g(t) = \exp\{a + b[t - cS(t)]\},$$

where

$$S(t) = \sum_i 10^{0.75(M_i - M_0)}$$

and the summation is taken over those i such that $t_i < t$, where t_i denotes the event times.

Value

Two usages are as follows.

```
srm_gif(data, evalpts, params, tplus=FALSE)
srm_gif(data, evalpts=NULL, params, TT)
```

The first usage returns a vector containing the values of $\lambda_g(t)$ evaluated at the specified points. In the second usage, it returns the value of the integral.

Function Attributes

```
rate is "increasing".
```

Problems and Inconsistencies

Runs much slower than [linksrmsrm_gif](#). Should set up matrices `St1` and `St2` as in [linksrmsrm_gif](#).

See Also

General details about the structure of conditional intensity functions are given in the topic [gif](#).

Examples

```
# Treating North China as one region

data(NthChina)
p <- c(-2.46, 0.0113, 0.851)
times <- seq(0, 517, 0.5)

par.default <- par(mfrow=c(2,1), mar=c(4.1, 4.1, 0.5, 1))
plot(times+1480, srm_gif(NthChina, times, params=p), type="l",
      ylab=expression(lambda[g](t)),
      xlab="", xlim=c(1480, 2000))
plot(NthChina$time+1480, NthChina$magnitude+6, type="h",
      xlim=c(1480, 2000), ylim=c(5.8, 8.6),
      xlab="Year", ylab="Magnitude")

par(par.default)
```

Description

Provides methods for the generic function [summary](#).

Usage

```
## S3 method for class 'mpp':  
summary(object, ...)  
## S3 method for class 'linkstrm':  
summary(object, ...)
```

Arguments

object an object with class "mpp" or "linkstrm".
... other arguments.

Value

A list object with a reduced number of components, mainly the parameter values.

Examples

```
TT <- c(0, 1000)  
bvalue <- 1  
params <- c(-2.5, 0.01, 0.8, bvalue*log(10))  
  
x <- mpp(data=NULL,  
         gif=srm_gif,  
         mark=list(NULL, rexp_mark),  
         params=params,  
         gmap=expression(params[1:3]),  
         mmap=expression(params[4]),  
         TT=TT)  
x <- simulate(x, seed=5)  
  
print(summary(x))
```

Tangshan

Tangshan Earthquake and Aftershock Sequence

Description

The Tangshan earthquake occurred on 28 July 1976 at 03:42:53, with a magnitude of 7.9. The Tangshan data frame contains those events (455) from the Beijing Catalogue, within 100 km of the epicentre and with magnitude 4 or greater, from the beginning of 1974 to the end of 1984.

Usage

```
data(Tangshan)
```

Format

This data frame contains the following columns:

latitude number of degrees north.

longitude number of degrees east.

magnitude number of magnitude units *above* 4.

year year of event (numeric vector).

month month of event, 1 ... 12 (numeric vector).

day day of event, 1 ... 31 (numeric vector).

hour hour of event, 0 ... 23 (numeric vector).

minute minute of event, 0 ... 59 (numeric vector).

second second of event, 0 ... 59 (numeric vector).

time number of days (and fractions) from the beginning of 1974.

Source

These data originate from the Beijing Catalogue which is administered by the China Seismological Bureau, Beijing.

Examples

```
data(Tangshan)
print(Tangshan[1:10,])
```

Index

- *Topic **classes**
 - linkstrm, 20
 - mpp, 29
 - *Topic **datagen**
 - simulate, 43
 - *Topic **datasets**
 - NthChina, 33
 - Ogata, 34
 - Phuket, 34
 - Tangshan, 47
 - *Topic **distribution**
 - dpareto, 9
 - marks, 27
 - *Topic **documentation**
 - Change Log, 1
 - distribution, 5
 - gif, 18
 - PtProcess, 37
 - *Topic **methods**
 - logLik, 25
 - plot, 36
 - residuals, 39
 - simulate, 43
 - summary, 46
 - *Topic **models**
 - etas_gif, 13
 - etas_spatial, 15
 - linkstrm_convert, 22
 - linkstrm_gif, 23
 - simple_gif, 41
 - strm_gif, 45
 - *Topic **optimize**
 - neglogLik, 31
- c, 30
- Change Log, 1
- Changes, 37
- Changes (*Change Log*), 1
- data.frame, 14, 15, 20, 24, 28, 29, 41, 45
- dexp, 12
- dexp_mark (*marks*), 27
- dgamma, 12
- distribution, 5, 12, 32
- dpareto, 9
- dtappareto (*dpareto*), 9
- dweibull, 12
- etas_gif, 13, 16, 19
- etas_normal0, 2
- etas_normal0 (*etas_spatial*), 15
- etas_spatial, 15
- expfourier_gif, 19
- expfourier_gif (*simple_gif*), 41
- exppoly_gif, 19
- exppoly_gif (*simple_gif*), 41
- expression, 20, 30
- fourier_gif, 19
- fourier_gif (*simple_gif*), 41
- GammaDist, 28
- gif, 2, 14, 16, 17, 18, 25, 30, 41, 42, 46
- inherits, 3
- integrate, 41, 42
- linkstrm, 20, 25, 36, 37, 40, 43, 47
- linkstrm1_gif (*linkstrm_gif*), 23
- linkstrm_convert, 3, 22
- linkstrm_gif, 2, 3, 19–21, 23, 23, 44, 46
- list, 28, 30
- logLik, 2, 16, 25, 25, 32, 37
- logLik.mpp, 3
- ltappareto (*dpareto*), 9
- marks, 2, 20, 27, 30
- matrix, 14, 15, 24, 41, 45
- mpp, 2, 16, 21, 25, 29, 32, 36, 37, 40, 43, 47
- neglogLik, 2, 16, 31, 37

nlm, 5, 31, 32
NthChina, 33
NULL, 29, 30

Ogata, 34
optim, 5, 31, 32

Phuket, 2, 3, 34
plot, 2, 16, 36, 36, 37
poly_gif, 19
poly_gif(*simple_gif*), 41
ppareto(*dpareto*), 9
ptappareto(*dpareto*), 9
PtProcess, 37

qpareto(*dpareto*), 9
qtappareto(*dpareto*), 9

residuals, 16, 37, 39, 39
residuals.mpp, 19
rexp_mark(*marks*), 27
rpareto(*dpareto*), 9
rtappareto(*dpareto*), 9

simple_gif, 19, 41
simulate, 2, 16, 37, 43, 43
simulate.mpp, 19
snow-startstop, 26
srm_gif, 3, 19, 45
summary, 16, 37, 46, 46
summary.mpp, 3

Tangshan, 47
ts, 40

vector, 14, 41, 45