Package 'IHSEP'

January 20, 2025

Type Package

Title Inhomogeneous Self-Exciting Process

Version 0.3.1

Date 2022-9-16

Author Feng Chen <feng.chen@unsw.edu.au>

Maintainer Feng Chen <feng.chen@unsw.edu.au>

Description Simulate an inhomogeneous self-exciting process (IHSEP), or Hawkes process, with a given (possibly time-varying) baseline intensity and an excitation function. Calculate the likelihood of an IHSEP with given baseline intensity and excitation functions for an (increasing) sequence of event times. Calculate the point process residuals (integral transforms of the original event times). Calculate the mean intensity process.

License GPL (>= 2)

LinkingTo Rcpp

Imports lpint,Rcpp

NeedsCompilation yes

Encoding UTF-8

RoxygenNote 7.2.1

Repository CRAN

Date/Publication 2022-09-16 21:56:05 UTC

Contents

`IHSEP-package` *Inhomogeneous Self-exciting Process*

Description

Simulates the (inhomogeneous) Self-exciting point process (SEPP), or Hawkes process, on $[0, T]$ with a given (possibly time-varying) baseline/background intensity function nu and excitation function (fertility rate function) g. Or calculate the likelihood of an SEPP with baseline intensity nu and excitation function g for a given set of event times on $[0, T]$.

Details

Package:	IHSEP
Type:	Package
Version:	1.0
Date:	2014-05-12
License:	GPL(>=2)

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

Maintainer: Feng Chen <feng.chen@unsw.edu.au>

References

Feng Chen and Peter Hall (2013). Inference for a nonstationary self-exciting point process with an application in ultra-high frequency financial data modeling. Journal of Applied Probability 50(4):1006-1024.

Feng Chen and Peter Hall (2016). Nonparametric Estimation for Self-Exciting Point Processes – A Parsimonious Approach. Journal of Computational and Graphical Statistics 25(1): 209-224.

Alan G Hawkes (1971). Spectra of some self-exciting and mutually exciting point processes. Biometrika 58(1):83-90.

Examples

```
## Not run:
## simulate the times of a Poisson process on [0,1] with intensity
## function nu(t)=100*(2+cos(2*pi*t)).
tms <- simPois(int=function(x)100*(2+cos(2*pi*x)),int.M=300)
## calculate a nonparametric estimate of the intensity function
int <- lpint::lpint(tms,Tau=1)
matplot(int$x,int$y+qnorm(0.975)*outer(int$se,c(-1,0,1)),type="l",lty=c(2,1,2),
        col=1,xlab="t",ylab="nu(t)")
curve(100*(2+cos(2*pi*x)),add=TRUE,col="red")

## simulate an IHSEP on [0,1] with baseline intensity function
## nu(t)=100*(2+cos(2*pi*t)) and excitation function
## g(t)=0.5*8*exp(-8*t)
asep <- simHawkes1(nu=function(x)200*(2+cos(2*pi*x)),nuM=600,
                             g=function(x)8*exp(-16*x),gM=8)
## get the birth times of all generations and sort in ascending order
tms <- sort(unlist(asep))
## calculate the minus loglikelihood of an SEPP with the true parameters
mloglik1a(tms,TT=1,nu=function(x)200*(2+cos(2*pi*x)),
          g=function(x)8*exp(-16*x),Ig=function(x)8/16*(1-exp(-18*x)))
## calculate the MLE for the parameter assuming known parametric forms
## of the baseline intensity and excitation functions
est <- optim(c(400,200,2*pi,8,16),
             function(p){
               mloglik1a(jtms=tms,TT=1,
                         nu=function(x)p[1]+p[2]*cos(p[3]*x),
                         g=function(x)p[4]*exp(-p[5]*x),
                         Ig=function(x)p[4]/p[5]*(1-exp(-p[5]*x)))
             },
             hessian=TRUE,control=list(maxit=5000,trace=TRUE))
## point estimate by MLE
est$par
## standard error estimates:
diag(solve(est$hessian))^0.5

## End(Not run)
```

---

asep *An IHSEP data set*

Description

A simulated data set from the inhomegeneous self-exciting process model with baseline intensity, or immigration rate, $\nu(t) = 200(2 + \cos(2\pi t))$ and excitation function, or fertility $g(t) = 8\exp(-16t)$.

Usage

```
data(asep)
```

Format

The format is a list of the arrival/birth times of individuals of all generations, in the order of generation 0 individuals, generation 1 individuals, etc.

Details

Times of arrivals/births of the same generation is listed together in ascending order. Number of generations is given by the length of the object.

Source

Simulated by a call to the function `simHawkes1`.

Examples

```
data(asep)
## number of generations
length(asep)
## jump times of the observed self-exciting process
sort(unlist(asep))
```

`conv.seq` *Sequence convolution conv.seq calculates the convolution of two sequences*

Description

Sequence convolution conv.seq calculates the convolution of two sequences

Usage

```
conv.seq(x, y, real = TRUE)
```

Arguments

`x, y`	numeric vectors
`real`	logical value, indicating whether the result should be real valued

Value

a numeric vector of length equal to `length(x)+length(y)`, giving the convolution of `x` and `y`

See Also

`convolve`

Examples

```
conv.seq(1:5,1:4)
convolve(1:5,4:1,type="open")
```

h.fn *Mean Intensity Function of the Self-Exciting Point Process*

Description

h.fn calculate the values of the mean intensity function of the self-exciting process with given baseline event rate and excitation function at a (fairly large) number of points. Values of the function at other points can be obtained by interpolation (e.g. spline interpolation).

Usage

```
h.fn(nu, g, N = 2^12, to = 1, abs.tol = 1e-10, maxit = 100)
```

Arguments

nu	a (vectorized) function specifying the baseline invent rate of the SEPP
g	a (vectorized) function specifying the excitation function of the SEPP
N	an integer giving the number of equal sized intervals to partition the domain into during the calculation The larger this value is, the more accurately the solution approxmates the truth, and the more time requred to evaluate.
to	a numeric scalar, the end point of the estimation domain
abs.tol	a numeric scalar specifying the absolute tolerance of error
maxit	an integer specifying the maximal number of iterations allowed

Value

a list with elelents, x: the vector of the points where h is evaluated; y: the vector of the corresponding h values; nit: the number of iterations used; G.err: the approximation error in G.

See Also

h.fn.exp

Examples

```
## Not run:
nu <- function(x)(200+100*cos(pi*x))*(x>=0);
g <- function(x) 2*exp(-x)
h.l <- h.fn(nu=nu,g=g,to=5);
h <- splinefun(h.l$x,h.l$y);
x <- 1:500/100;
max(nu(x)+sapply(x,function(x)integrate(function(u)g(x-u)*h(u),0,x)$value) - h(x))

## End(Not run)
```

h.fn.exp *Mean Intensity of the Self-Exciting Point Process With an Exponential Excitation Function*

Description

h.fn.exp calculates the mean intensity function $h(t)$ which solves the integral equation

$$h(t) = \nu(t) + \int_0^t g(t-s)h(s)ds, t \geq 0$$

, where the excitation function is exponential: $g(t) = \gamma_1 e^{-\gamma_2 t}$.

Usage

```
h.fn.exp(x, nu, g.p = c(4, 8))
```

Arguments

x	numerical scalar, at which the mean intensity h is to be evaluated
nu	a function, which gives the baseline event rate
g.p	a numeric vector of two elements giving the two parameters γ_1, γ_2 of the exponential excitation function

Value

a numric scalar which gives the value of the function h at x.

See Also

h.fn

Examples

```
nu <- function(x)200+100*cos(pi*x);
x <- 1:500/100;
y <- sapply(x,h.fn.exp,nu=nu,g.p=c(2,1));
h <- splinefun(x,y);
g <- function(x)2*exp(-x)
round(nu(x)+sapply(x,function(x)integrate(function(u)g(x-u)*h(u),0,x)$value) - y,5)
```

mloglik0 *Minus loglikelihood of an IHSEP model*

Description

Calculates the minus loglikelihood of an IHSEP model with given baseline inensity function ν and excitation function g for event times `jtms` on interval $[0, TT]$.

Usage

```
mloglik0(jtms, TT = max(jtms), nu, g,
         Ig=function(x)sapply(x,function(y)integrate(g,0,y)$value))
```

Arguments

`jtms`	A numeric vector, with values sorted in ascending order. Jump times to fit the inhomogeneous self-exciting point process model on.
`TT`	A scalar. The censoring time, or the terminal time for observation. Should be (slightly) greater than the maximum of `jtms`.
`nu`	A (vectorized) function. The baseline intensity function.
`g`	A (vectorized) function. The excitation function.
`Ig`	A (vectorized) function. Its value at `t` gives the integral of the excitation function from 0 to `t`.

Value

The value of the negative log-liklihood.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

Examples

```
## simulated data of an IHSEP on [0,1] with baseline intensity function
## nu(t)=200*(2+cos(2*pi*t)) and excitation function
## g(t)=8*exp(-16*t)
data(asep)

## get the birth times of all generations and sort in ascending order
tms <- sort(unlist(asep))
## calculate the minus loglikelihood of an SEPP with the true parameters
mloglik0(tms,TT=1,nu=function(x)200*(2+cos(2*pi*x)),
         g=function(x)8*exp(-16*x),Ig=function(x)8/16*(1-exp(-16*x)))
## calculate the MLE for the parameter assuming known parametric forms
## of the baseline intensity and excitation functions
## Not run:
```

```
system.time(est <- optim(c(400,200,2*pi,8,16),
                         function(p){
                           mloglik0(jtms=tms,TT=1,
                                    nu=function(x)p[1]+p[2]*cos(p[3]*x),
                                    g=function(x)p[4]*exp(-p[5]*x),
                                    Ig=function(x)p[4]/p[5]*(1-exp(-p[5]*x)))
                         },
                         hessian=TRUE,control=list(maxit=5000,trace=TRUE))
            )
## point estimate by MLE
est$par
## standard error estimates:
diag(solve(est$hessian))^0.5

## End(Not run)
```

`mloglik1a` *Minus loglikelihood of an IHSEP model*

Description

Calculates the minus loglikelihood of an IHSEP model with given baseline inensity function ν and excitation function g for event times `jtms` on interval $[0, TT]$.

Usage

```
mloglik1a(jtms, TT = max(jtms), nu, g,
          Ig = function(x) sapply(x, function(y) integrate(g, 0, y)$value),
          tol.abs = 1e-12, tol.rel = 1e-12, limit = 1000
          )
```

Arguments

`jtms`	A numeric vector, with values sorted in ascending order. Jump times to fit the inhomogeneous self-exciting point process model on.
`TT`	A scalar. The censoring time, or the terminal time for observation. Should be (slightly) greater than the maximum of `jtms`.
`nu`	A (vectorized) function. The baseline intensity function.
`g`	A (vectorized) function. The excitation function.
`Ig`	A (vectorized) function. Its value at `t` gives the integral of the excitation function from 0 to `t`.
`tol.abs`	A small positive number. The tolerance of the absolute error in the numerical integral of ν.
`tol.rel`	A small positive number. The tolerance of the relative error in the numerical integral of ν.
`limit`	An (large) positive integer. The maximum number of subintervals allowed in the adaptive quadrature method to find the numerical integral of ν.

Details

This version of the mloglik function uses external C code to speedup the calculations. Otherwise it is the same as the `mloglik0` function.

Value

The value of the negative log-liklihood.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

See Also

`mloglik0`

Examples

```
## simulated data of an IHSEP on [0,1] with baseline intensity function
## nu(t)=200*(2+cos(2*pi*t)) and excitation function
## g(t)=8*exp(-16*t)
data(asep)

## get the birth times of all generations and sort in ascending order
tms <- sort(unlist(asep))
## calculate the minus loglikelihood of an SEPP with the true parameters
mloglik1a(tms,TT=1,nu=function(x)200*(2+cos(2*pi*x)),
          g=function(x)8*exp(-16*x),Ig=function(x)8/16*(1-exp(-16*x)))
## calculate the MLE for the parameter assuming known parametric forms
## of the baseline intensity and excitation functions
## Not run:
system.time(est <- optim(c(400,200,2*pi,8,16),
                         function(p){
                           mloglik1a(jtms=tms,TT=1,
                                     nu=function(x)p[1]+p[2]*cos(p[3]*x),
                                     g=function(x)p[4]*exp(-p[5]*x),
                                     Ig=function(x)p[4]/p[5]*(1-exp(-p[5]*x)))
                         },
                         hessian=TRUE,control=list(maxit=5000,trace=TRUE))
            )
## point estimate by MLE
est$par
## standard error estimates:
diag(solve(est$hessian))^0.5

## End(Not run)
```

`mloglik1b` *Minus loglikelihood of an IHSEP model*

Description

Calculates the minus loglikelihood of an IHSEP model with given baseline inensity function ν and excitation function g for event times `jtms` on interval $[0, TT]$.

Usage

```
mloglik1b(jtms, TT = max(jtms), nu, g,
          Ig=function(x)sapply(x,function(y)integrate(g,0,y,
               rel.tol=1e-12,abs.tol=1e-12,subdivisions=1000)$value),
          Inu=function(x)sapply(x,function(y)integrate(nu,0,y)$value))
```

Arguments

`jtms`	A numeric vector, with values sorted in ascending order. Jump times to fit the inhomogeneous self-exciting point process model on.
`TT`	A scalar. The censoring time, or the terminal time for observation. Should be (slightly) greater than the maximum of `jtms`.
`nu`	A (vectorized) function. The baseline intensity function.
`g`	A (vectorized) function. The excitation function.
`Ig`	A (vectorized) function. Its value at `t` gives the integral of the excitation function from 0 to `t`.
`Inu`	A (vectorized) function. Its value at `t` gives the integral of the baseline intensity function ν from 0 to `t`.

Details

This version of the mloglik function uses external C code to speedup the calculations. When given the analytical form of `Inu` or a quickly calculatable `Inu`, it should be (probably slightly) faster than `mloglik1a`. Otherwise it is the same as `mloglik0` and `mloglik1a`.

Value

The value of the negative log-liklihood.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

See Also

`mloglik0` and `mloglik1a`

Examples

```
## simulated data of an IHSEP on [0,1] with baseline intensity function
## nu(t)=200*(2+cos(2*pi*t)) and excitation function
## g(t)=8*exp(-16*t)
data(asep)

## get the birth times of all generations and sort in ascending order
tms <- sort(unlist(asep))
## calculate the minus loglikelihood of an SEPP with the true parameters
mloglik1b(tms,TT=1,nu=function(x)200*(2+cos(2*pi*x)),
          g=function(x)8*exp(-16*x),Ig=function(x)8/16*(1-exp(-16*x)))
## calculate the MLE for the parameter assuming known parametric forms
## of the baseline intensity and excitation functions
## Not run:
system.time(est <- optim(c(400,200,2*pi,8,16),
                         function(p){
                           mloglik1b(jtms=tms,TT=1,
                                     nu=function(x)p[1]+p[2]*cos(p[3]*x),
                                     g=function(x)p[4]*exp(-p[5]*x),
                                     Ig=function(x)p[4]/p[5]*(1-exp(-p[5]*x)))
                         },
                         hessian=TRUE,control=list(maxit=5000,trace=TRUE))
            )

## point estimate by MLE
est$par
## standard error estimates:
diag(solve(est$hessian))^0.5

## End(Not run)
```

`mloglik1c` *Minus loglikelihood of an IHSEP model*

Description

Calculates the minus loglikelihood of an IHSEP model with given baseline inensity function ν and excitation function $g(x) = aexp(-bx)$ for event times `jtms` on interval `[0,TT]`.

Usage

```
mloglik1c(jtms, TT, nu, gcoef, Inu)
```

Arguments

`jtms`	A numeric vector, with values sorted in ascending order. Jump times to fit the inhomogeneous self-exciting point process model on.
`TT`	A scalar. The censoring time, or the terminal time for observation. Should be (slightly) greater than the maximum of `jtms`.

nu	A (vectorized) function. The baseline intensity function.
gcoef	A numeric vector (of two elements), giving the parameters (a,b) of the exponential excitation function g(x)=a*exp(-b*x).
Inu	A (vectorized) function. Its value at t gives the integral of the baseline intensity function ν from 0 to t.

Details

This version of the mloglik function uses external C code to speedup the calculations. When given the analytical form of Inu or a quickly calculatable Inu, it should be (substantially) faster than mloglik1a when calculating the (minus log) likelihood when the excitation function is exponential. Otherwise it is the same as mloglik0, mloglik1a, mloglik1b.

Value

The value of the negative log-liklihood.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

See Also

mloglik0, mloglik1a and mloglik1b

Examples

```
## simulated data of an IHSEP on [0,1] with baseline intensity function
## nu(t)=200*(2+cos(2*pi*t)) and excitation function
## g(t)=8*exp(-16*t)
data(asep)

## get the birth times of all generations and sort in ascending order
tms <- sort(unlist(asep))
## calculate the minus loglikelihood of an SEPP with the true parameters
mloglik1c(tms,TT=1,nu=function(x)200*(2+cos(2*pi*x)),
          gcoef=8*1:2,
          Inu=function(y)integrate(function(x)200*(2+cos(2*pi*x)),0,y)$value)
## calculate the MLE for the parameter assuming known parametric forms
## of the baseline intensity and excitation functions
## Not run:
system.time(est <- optim(c(400,200,2*pi,8,16),
                         function(p){
                           mloglik1c(jtms=tms,TT=1,
                                     nu=function(x)p[1]+p[2]*cos(p[3]*x),
                                     gcoef=p[-(1:3)],
                                     Inu=function(y){
                                      integrate(function(x)p[1]+p[2]*cos(p[3]*x),
                                                0,y)$value
                                     })
                         },hessian=TRUE,control=list(maxit=5000,trace=TRUE))
```

```
            )
## point estimate by MLE
est$par
## standard error estimates:
diag(solve(est$hessian))^0.5

## End(Not run)
```

`mloglik1d` *Minus loglikelihood of an IHSEP model*

Description

Calculates the minus loglikelihood of an IHSEP model with given baseline intensity function ν and excitation function $g(x) = \sum a_i exp(-b_i x)$ for event times `jtms` on interval `[0,TT]`.

Usage

```
mloglik1d(jtms, TT, nu, gcoef, Inu)
```

Arguments

`jtms`	A numeric vector, with values sorted in ascending order. Jump times to fit the inhomogeneous self-exciting point process model on.
`TT`	A scalar. The censoring time, or the terminal time for observation. Should be (slightly) greater than the maximum of `jtms`.
`nu`	A (vectorized) function. The baseline intensity function.
`gcoef`	A numeric vector (of 2k elements), giving the parameters `(a_1,...,a_k,b_1,...,b_k)` of the exponential excitation function $g(x) = \sum_{i=1}^{k} a_i * exp(-b_i * x)$.
`Inu`	A (vectorized) function. Its value at `t` gives the integral of the baseline intensity function ν from 0 to `t`.

Details

This function calculates the minus loglikelihood of the inhomegeneous Hawkes model with background intensity function $\nu(t)$ and excitation kernel function $g(t) = \sum_{i=1}^{k} a_i e^{-b_i t}$ relative to continuous observation of the process from time 0 to time `TT`. Like `mloglik1c`, it takes advantage of the Markovian property of the intensity process and uses external C++ code to speed up the computation.

Value

The value of the negative log-liklihood.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

See Also

mloglik1c

Examples

```
## simulated data of an IHSEP on [0,1] with baseline intensity function
## nu(t)=200*(2+cos(2*pi*t)) and excitation function
## g(t)=8*exp(-16*t)
data(asep)

## get the birth times of all generations and sort in ascending order
tms <- sort(unlist(asep))
## calculate the minus loglikelihood of an SEPP with the true parameters
mloglik1d(tms,TT=1,nu=function(x)200*(2+cos(2*pi*x)),
          gcoef=8*1:2,
          Inu=function(y)integrate(function(x)200*(2+cos(2*pi*x)),0,y)$value)
## calculate the MLE for the parameter assuming known parametric forms
## of the baseline intensity and excitation functions
## Not run:
system.time(est <- optim(c(400,200,2*pi,8,16),
                         function(p){
                           mloglik1d(jtms=tms,TT=1,
                                     nu=function(x)p[1]+p[2]*cos(p[3]*x),
                                     gcoef=p[-(1:3)],
                                     Inu=function(y){
                                      integrate(function(x)p[1]+p[2]*cos(p[3]*x),
                                                0,y)$value
                                     })
                         },hessian=TRUE,control=list(maxit=5000,trace=TRUE),
                         method="BFGS")
            )
## point estimate by MLE
est$par
## standard error estimates:
diag(solve(est$hessian))^0.5

## End(Not run)
```

---

mloglik1e *Minus loglikelihood of an IHSEP model*

Description

Calculates the minus loglikelihood of an IHSEP model with given baseline inensity function ν and excitation function $g(x) = \sum a_i exp(-b_i x)$ for event times `jtms` on interval `[0,TT]`.

Usage

```
mloglik1e(jtms, TT, nuvs, gcoef, InuT)
```

Arguments

jtms	A numeric vector, with values sorted in ascending order. Jump times to fit the inhomogeneous self-exciting point process model on.
TT	A scalar. The censoring time, or the terminal time for observation. Should be (slightly) greater than the maximum of jtms.
nuvs	A numeric vector, giving the values of the baseline intensity function ν at the jumptimes jtms.
gcoef	A numeric vector (of 2k elements), giving the parameters (a_1,...,a_k,b_1,...,b_k) of the exponential excitation function $g(x) = \sum_{i=1}^{k} a_i * exp(-b_i * x)$.
InuT	A numeric value (scalar) giving the integral of ν on the interval [0,TT].

Details

This version of the mloglik function uses external C code to speedup the calculations. When given the analytical form of Inu or a quickly calculatable Inu, it should be (substantially) faster than mloglik1a when calculating the (minus log) likelihood when the excitation function is exponential. Otherwise it is the same as mloglik0, mloglik1a, mloglik1b.

Value

The value of the negative log-liklihood.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

See Also

mloglik0, mloglik1a and mloglik1b

Examples

```
## simulated data of an IHSEP on [0,1] with baseline intensity function
## nu(t)=200*(2+cos(2*pi*t)) and excitation function
## g(t)=8*exp(-16*t)
data(asep)

## get the birth times of all generations and sort in ascending order
tms <- sort(unlist(asep))
## calculate the minus loglikelihood of an SEPP with the true parameters
mloglik1e(tms,TT=1,nuvs=200*(2+cos(2*pi*tms)),
          gcoef=8*1:2,
          InuT=integrate(function(x)200*(2+cos(2*pi*x)),0,1)$value)
## calculate the MLE for the parameter assuming known parametric forms
## of the baseline intensity and excitation functions
## Not run:
system.time(est <- optim(c(400,200,2*pi,8,16),
                         function(p){
                           mloglik1e(jtms=tms,TT=1,
```

```
                                    nuvs=p[1]+p[2]*cos(p[3]*tms),
                                    gcoef=p[-(1:3)],
                                    InuT=integrate(function(x)p[1]+p[2]*cos(p[3]*x),
                                                   0,1)$value
                                    )
                            },hessian=TRUE,control=list(maxit=5000,trace=TRUE),
                            method="BFGS")
            )
## point estimate by MLE
est$par
## standard error estimates:
diag(solve(est$hessian))^0.5

## End(Not run)
```

`sepp.resid` *Calculate the self exciting point process residuals*

Description

Calculate the self exciting point process residuals

Usage

```
sepp.resid(jtms, Tau, Inu, Ig)
```

Arguments

`jtms`	A numerical vector containing the event times of the SEPP in ascending order
`Tau`	A numerical scalar giving the censoring time, which should be greater than or equal to the event time
`Inu`	A function that gives the integral of the baseline event rate function $\nu(t)$ from 0 to the argument of the function
`Ig`	A function that gives the integral of the excitation function $g(t)$ from 0 to the argument of the function

Value

A numerical vector containing the SEPP residuals $\Lambda(t_i)$ where Λ is the cumulative intensity process.

simchildren *Simulate the child events* simchildren *simulates the birth times of all child events spawned from an event relative the birth time of the parent event. This function is to be called by the simulator function for offspring events and is not meant for external use.*

Description

Simulate the child events simchildren simulates the birth times of all child events spawned from an event relative the birth time of the parent event. This function is to be called by the simulator function for offspring events and is not meant for external use.

Usage

```
simchildren(br = 0.5, dis = "exp", par.dis = list(rate = 1), cens = Inf, sorted = TRUE)
```

Arguments

br	numerical scalar in [0,1), the branching ratio, or the expected number of direct children due to an event
dis	character string, which gives the name of the common (positive) distribution of the birth times of the child events relative to the parent event (referred to as the child birthtime distribution), such as "exp", "gamma", "weibull", etc.
par.dis	list, which gives the values of the (named) parameter(s) of the child birthtime distribution)
cens	positive scalar, which gives the censoring time (termination of observation time). The default value of Inf means no censoring.
sorted	boolean scalar, which indicates whether the out child birth times should be sorted or not. The default value is TRUE.

Value

a numeric vector of length giving the birth times of child events relative to the parent event in ascending order

See Also

simoffspring

Examples

```
simchildren(br=0.9,dis="exp",par.dis=list(rate=1))
```

simHawkes0 *Simulate a Hawkes process, or Self-exciting point process*

Description

Simulate an (inhomogeneous) self-exciting process with given background intensity and excitation/fertility function.

Usage

```
simHawkes0(nu, g, cens = 1,
           nuM=max(optimize(nu,c(0,cens),maximum=TRUE)$obj,nu(0),nu(cens))*1.1,
           gM=max(optimize(g,c(0,cens),maximum=TRUE)$obj, g(0),g(cens))*1.1)
```

Arguments

nu	A (vectorized) function. The baseline intensity function.
g	A (vectorized) function. The excitation function.
cens	A scalar. The censoring time, or the time of termination of observations.
nuM	A scalar. The maximum time of the baseline intensity from 0 to cens.
gM	A scalar. The maximum time of the excitation function from 0 to cens.

Details

The function works by simulating the birth times generation by generation according to inhomegenous Poisson processes with appropriate intensity functions (ν or g).

Value

A list of vectors of arrival/birth times of individuals/events of generations 0, 1, ...

Note

Same algorithm as in simHawkes1, though the latter might be more succinct and (very slightly) faster.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

See Also

simHawkes1

Examples

```
asepp <- simHawkes0(nu=function(x)200*(2+cos(2*pi*x)),nuM=600,
                                g=function(x)8*exp(-16*x),gM=8)
```

simHawkes1 *Simulate a Hawkes process, or Self-exciting point process*

Description

Simulate an (inhomogeneous) self-exciting process with given background intensity and excitation/fertility function.

Usage

```
simHawkes1(nu=NULL, g=NULL, cens = 1,
          nuM=max(optimize(nu,c(0,cens),maximum=TRUE)$obj, nu(0), nu(cens))*1.1,
            gM=max(optimize(g,c(0,cens),maximum = TRUE)$obj, g(0),
g(cens))*1.1,
            exp.g=FALSE,gp=c(1,2))
```

Arguments

nu	A (vectorized) function. The baseline intensity function.
g	A (vectorized) function. The excitation function.
cens	A scalar. The censoring time, or the time of termination of observations.
nuM	A scalar. The maximum time of the baseline intensity from 0 to cens.
gM	A scalar. The maximum time of the excitation function from 0 to cens.
exp.g	A logical. Whether the excitation function g should be treated as an exponential function.
gp	A vector of two elements, giving the two parameters a and b in the exponential excitation function $g(x) = a * exp(-b * x)$, which is used when exp.g is set to TRUE, and is ignored otherwise.

Details

The function works by simulating the birth times generation by generation according to inhomegenous Poisson processes with appropriate intensity functions (ν or g).

Value

A list of vectors of arrival/birth times of individuals/events of generations 0, 1, ... The length of the list is the total number of generations.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

See Also

simHawkes0

Examples

```
asepp <- simHawkes1(nu=function(x)200*(2+cos(2*pi*x)),nuM=600,
                    g=function(x)8*exp(-16*x),gM=8)
```

`simHawkes1a` *Simulate an (inhomogeneous) Hawkes self-exciting process* `simHawkes1a` *simulates the event times of an inhomogeneous Hawkes process (IHSEP) with background event intensity/rate* $nu(\cdot) \geq 0$*, branching ratio* $\eta \in [0, 1)$*, and offspring birthtime density* $g(\cdot)$*, up to a censoring time* T*.*

Description

Simulate an (inhomogeneous) Hawkes self-exciting process `simHawkes1a` simulates the event times of an inhomogeneous Hawkes process (IHSEP) with background event intensity/rate $nu(\cdot) \geq 0$, branching ratio $\eta \in [0, 1)$, and offspring birthtime density $g(\cdot)$, up to a censoring time T.

Usage

```
simHawkes1a(
  nu = function(x) rep(100, length(x)),
  cens = 1,
  nuM = max(optimize(nu, c(0, cens), maximum = TRUE)$obj, nu(0), nu(cens),
    .Machine$double.eps^0.5) * 1.1,
  br = 0.5,
  dis = "exp",
  par.dis = list(rate = 1)
)
```

Arguments

`nu`	a function, which gives the background event intensity function $\nu(\cdot)$; needs to be a bounded function on $[0, T]$.
`cens`	a positive scalar, which gives the censoring time.
`nuM`	positive scalar, optional argument giving the maximum of the background intensity function on $[0, T]$.
`br`	scalar in [0,1), giving the branching ratio.
`dis`	character string giving the name of the child birthtime distribution; 'd$dis' gives the density function $g(\cdot)$.
`par.dis`	a (named) list giving the values of the parameters of the child birthtime distribution.

Value

a vector giving the event times of an inhomogeneous Hawkes process up to the censoring time in ascending order.

`simoffspring` *Simulate the offspring events* `simoffspring` *simulates the birth times of offspring events of all generations spawned from an event relative the birth time of the parent event. This function is to be called by the simulator function for Hawkes processes and is not meant for external use.*

Description

Simulate the offspring events `simoffspring` simulates the birth times of offspring events of all generations spawned from an event relative the birth time of the parent event. This function is to be called by the simulator function for Hawkes processes and is not meant for external use.

Usage

```
simoffspring(br = 0.5, dis = "exp", par.dis = list(rate = 1), cens = Inf, sorted = TRUE)
```

Arguments

`br`	numerical scalar in [0,1), the branching ratio, or the expected number of direct children due to an event
`dis`	character string, which gives the name of the common (positive) distribution of the birth times of the child events relative to the parent event (referred to as the child birthtime distribution), such as "exp", "gamma", "weibull", etc.
`par.dis`	list, which gives the values of the (named) parameter(s) of the child birthtime distribution)
`cens`	numeric scalar, which gives the censoring time (termination of observation time). The default value of Inf means no censoring.
`sorted`	boolean scalar, which indicates whether the out child birth times should be sorted or not. The default value is TRUE.

Details

This function uses recursion, so can break down when the branching ratio is close to 1, leading to very deep recursions. In this case, we should use `simHawkes1` for Hawkes process simulation.

Value

a numeric vector of giving the birth times of offspring events of all generations relative to the parent's birth time in ascending order

See Also

`simchildren`

Examples

```
simoffspring(br=0.9,dis="exp",par.dis=list(rate=1))
```

`simPois` *Simulate a Poisson process*

Description

Simulate an (inhomogeneous) Poisson process with a given intensity/rate function over the interval $[0, T]$

Usage

```
simPois(int=NULL,cens=1,int.M=optimize(int,c(0,cens),maximum=TRUE)$obj*1.1,
        B=max(as.integer(sqrt(int.M * cens)),10), exp.int=FALSE,par=c(1,2))
```

Arguments

`int`	A (vectorized) positive function. The intensity/rate function.
`cens`	A positive scalar. The censoring time, or time of termination of observations, T.
`int.M`	A positive scalar. Maximum value of the intensity function over $[0, T]$, or a larger value.
`B`	An integer. The block size to be used in generating exponential random variables in blocks.
`exp.int`	Logical. Set to TRUE, if the intensity function is exponential, i.e. a*exp(-b*x). If set to TRUE, the parameters a and b should also be supplied via the argument par.
`par`	A numerical vector of two elements, giving the parameters a and b of the exponential intensity function. The values are not ignored if exp.int is set to FALSE.

Details

The function works by first generating a Poisson process with constant rate `int.M` oever $[0, T]$, and then thinning the process with retention probability function

$$p(x) = \texttt{int}(x)/\texttt{int.M}$$

.

When generating the homoneous Poisson process, it first generates about $\Lambda + 1.96 * \sqrt{\Lambda}$ exponential variables, then, if the exponential variables are not enough (their sum has not reached the censoring time T yet), generates exponential variables in blocks of size `B` until the total of all the generated exponentials exceeds T. Then `cumsums` of the exponentials that are smaller than or equal to T are retained as the event times of the homogeneous Poisson process. This method apparantly does not produce tied event times.

Value

A numerical vector giving the event/jump times of the Poisson process in $[0, T]$, in ascending order.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

See Also

simPois0

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (int, cens = 1, int.M = optimize(int, c(0, cens), maximum = TRUE)$obj *
    1.1, B = max(as.integer(sqrt(int.M * cens)), 10))
{
    tms <- rexp(as.integer(int.M * cens + 2 * sqrt(int.M * cens)),
        rate = int.M)
    while (sum(tms) < cens) tms <- c(tms, rexp(B, rate = int.M))
    cumtms <- cumsum(tms)
    tms <- cumtms[cumtms <= cens]
    N <- length(tms)
    if (N == 0)
        return(numeric(0))
    tms[as.logical(mapply(rbinom, n = 1, size = 1, prob = int(tms)/int.M))]
  }
```

---

simPois0 *Simulate a Poisson process*

Description

Simulate an (inhomogeneous) Poisson process with a given intensity/rate function over the interval $[0, T]$

Usage

```
simPois0(int, cens = 1, int.M = optimize(int, c(0, cens), maximum = TRUE)$obj * 1.1)
```

Arguments

int	A (vectorized) positive function. The intensity/rate function.
cens	A positive scalar. The censoring time, or time of termination of observations, T.
int.M	A positive scalar. Maximum value of the intensity function over $[0, T]$, or a value larger than this.

Details

The function works by first generating a Poisson process with constant rate `int.M` oever $[0, T]$, and then thinning the process with retention probability function

$$p(x) = \texttt{int}(x)/\texttt{int.M}$$

.

Value

A numerical vector giving the event/jump times of the Poisson process in $[0, T]$, in ascending order.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

See Also

`simPois0`

Examples

```
aPP <- simPois(int=function(x)200*(2+cos(2*pi*x)),cens=1,int.M=600)
```

Index