Package ‘emhawkes’

August 26, 2025

Title Exponential Multivariate Hawkes Model

Version 0.9.8

Maintainer Kyungsub Lee <kyungsub@gmail.com>

Description Simulate and fitting exponential multivariate Hawkes model.
This package simulates a multivariate Hawkes model, introduced by Hawkes (1971) <doi:10.2307/2334319>, with an exponential kernel and fits the parameters from the data.
Models with the constant parameters, as well as complex dependent structures, can also be simulated and estimated.
The estimation is based on the maximum likelihood method, introduced by introduced by Ozaki (1979) <doi:10.1007/BF02480272>, with 'maxLik' package.

Depends R (>= 4.0.0)

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Imports methods, maxLik

Collate 'hspec.R' 'harrival.R' 'utilities.R' 'hmoment.R' 'hllf.R'
'hfit.R' 'hgfit.R' 'hreal.R' 'hsim.R' 'script.R' 'tzexp.R'
'zzz.R'

Suggests knitr, rmarkdown, miscTools

VignetteBuilder knitr

URL https://github.com/ksublee/emhawkes,
https://ksublee.github.io/emhawkes/

NeedsCompilation no

Author Kyungsub Lee [aut, cre]

Repository CRAN

Date/Publication 2025-08-26 04:20:11 UTC

Contents

`expected_tau` *Expected Inter-Arrival Time*

Description

Computes the conditional expected time until the next event.

Usage

```
expected_tau(
  object,
  rambda_component,
  type = 1,
  mu = NULL,
  beta = NULL,
  tol = .Machine$double.eps^0.25,
  max_upper = Inf,
  subdivisions = 400L
)
```

Arguments

`object`	An object of class `hspec`.
`rambda_component`	Rambda component.
`type`	Process dimension index (default is 1).
`mu`	Optional mu value (overrides object@mu if provided).
`beta`	Optional beta value (overrides object@beta if provided).
`tol`	Relative tolerance for numerical integration.
`max_upper`	Upper integration limit.
`subdivisions`	Number of subdivisions for numerical integration.

Value

Expected value of next inter-arrival time.

---

hfit *Perform Maximum Likelihood Estimation*

Description

This is a generic function named hfit designed for estimating the parameters of the exponential Hawkes model. It is implemented as an S4 method for two main reasons:

Usage

```
hfit(
  object,
  inter_arrival = NULL,
  type = NULL,
  mark = NULL,
  N = NULL,
  Nc = NULL,
  lambda_component0 = NULL,
  N0 = NULL,
  mylogLik = NULL,
  reduced = TRUE,
  grad = NULL,
  hess = NULL,
  constraint = NULL,
  method = "BFGS",
  verbose = FALSE,
  ...
)

## S4 method for signature 'hspec'
hfit(
  object,
  inter_arrival = NULL,
  type = NULL,
  mark = NULL,
  N = NULL,
  Nc = NULL,
  lambda_component0 = NULL,
  N0 = NULL,
  mylogLik = NULL,
  reduced = TRUE,
  grad = NULL,
  hess = NULL,
  constraint = NULL,
  method = "BFGS",
  verbose = FALSE,
  ...
)
```

Arguments

`object` An `hspec-class` object containing the parameter values.

`inter_arrival` A vector of inter-arrival times for events across all dimensions, starting with zero.

`type` A vector indicating the dimensions, represented by numbers like 1, 2, 3, etc., starting with zero.

`mark` A vector of mark (jump) sizes, starting with zero.

`N` A matrix representing counting processes.

`Nc` A matrix of counting processes weighted by mark sizes.

`lambda_component0`
Initial values for the lambda component λ_{ij}. Can be a numeric value or a matrix. Must have the same number of rows and columns as `alpha` or `beta` in `object`.

`N0` Initial values for the counting processes matrix `N`.

`mylogLik` A user-defined log-likelihood function, which must accept an `object` argument consistent with `object`.

`reduced` Logical; if `TRUE`, performs reduced estimation.

`grad` A gradient matrix for the likelihood function. Refer to `maxLik` for more details.

`hess` A Hessian matrix for the likelihood function. Refer to `maxLik` for more details.

`constraint` Constraint matrices. Refer to `maxLik` for more details.

`method` The optimization method to be used. Refer to `maxLik` for more details.

`verbose` Logical; if `TRUE`, prints the progress of the estimation process.

`...` Additional parameters for optimization. Refer to `maxLik` for more details.

Details

Model Representation: To represent the structure of the model as an `hspec` object. The multivariate marked Hawkes model has numerous variations, and using an S4 class allows for a flexible and structured approach.

Optimization Initialization: To provide a starting point for numerical optimization. The parameter values assigned to the `hspec` slots serve as initial values for the optimization process.

This function utilizes the `maxLik` package for optimization.

Value

`maxLik` object

See Also

hspec-class, hsim,hspec-method

Examples

```
# example 1
mu <- c(0.1, 0.1)
alpha <- matrix(c(0.2, 0.1, 0.1, 0.2), nrow=2, byrow=TRUE)
beta <- matrix(c(0.9, 0.9, 0.9, 0.9), nrow=2, byrow=TRUE)
h <- new("hspec", mu=mu, alpha=alpha, beta=beta)
res <- hsim(h, size=100)
summary(hfit(h, inter_arrival=res$inter_arrival, type=res$type))


# example 2

mu <- matrix(c(0.08, 0.08, 0.05, 0.05), nrow = 4)
alpha <- function(param = c(alpha11 = 0, alpha12 = 0.4, alpha33 = 0.5, alpha34 = 0.3)){
  matrix(c(param["alpha11"], param["alpha12"], 0, 0,
           param["alpha12"], param["alpha11"], 0, 0,
           0, 0, param["alpha33"], param["alpha34"],
           0, 0, param["alpha34"], param["alpha33"]), nrow = 4, byrow = TRUE)
}
beta <- matrix(c(rep(0.6, 8), rep(1.2, 8)), nrow = 4, byrow = TRUE)

impact <- function(param = c(alpha1n=0, alpha1w=0.2, alpha2n=0.001, alpha2w=0.1),
                   n=n, N=N, ...){

  Psi <- matrix(c(0, 0, param['alpha1w'], param['alpha1n'],
                  0, 0, param['alpha1n'], param['alpha1w'],
                  param['alpha2w'], param['alpha2n'], 0, 0,
                  param['alpha2n'], param['alpha2w'], 0, 0), nrow=4, byrow=TRUE)

  ind <- N[,"N1"][n] - N[,"N2"][n] > N[,"N3"][n] - N[,"N4"][n] + 0.5

  km <- matrix(c(!ind, !ind, !ind, !ind,
                 ind, ind, ind, ind,
                 ind, ind, ind, ind,
                 !ind, !ind, !ind, !ind), nrow = 4, byrow = TRUE)

  km * Psi
}
h <- new("hspec",
         mu = mu, alpha = alpha, beta = beta, impact = impact)
hr <- hsim(h, size=100)
plot(hr$arrival, hr$N[,'N1'] - hr$N[,'N2'], type='s')
lines(hr$N[,'N3'] - hr$N[,'N4'], type='s', col='red')
fit <- hfit(h, hr$inter_arrival, hr$type)
summary(fit)


# example 3
```

```
mu <- c(0.15, 0.15)
alpha <- matrix(c(0.75, 0.6, 0.6, 0.75), nrow=2, byrow=TRUE)
beta <- matrix(c(2.6, 2.6, 2.6, 2.6), nrow=2, byrow=TRUE)
rmark <- function(param = c(p=0.65), ...){
  rgeom(1, p=param[1]) + 1
}
impact <- function(param = c(eta1=0.2), alpha, n, mark, ...){
  ma <- matrix(rep(mark[n]-1, 4), nrow = 2)
  alpha * ma * matrix( rep(param["eta1"], 4), nrow=2)
}
h1 <- new("hspec", mu=mu, alpha=alpha, beta=beta,
          rmark = rmark,
          impact=impact)
res <- hsim(h1, size=100, lambda_component0 = matrix(rep(0.1,4), nrow=2))

fit <- hfit(h1,
            inter_arrival = res$inter_arrival,
            type = res$type,
            mark = res$mark,
            lambda_component0 = matrix(rep(0.1,4), nrow=2))
summary(fit)

# For more information, please see vignettes.
```

hreal *Realization of Hawkes Process*

Description

`hreal` is a list containing the following components:

- `hspec`: An S4 object of class `hspec-class` that specifies the parameter values.
- `inter_arrival`: The time intervals between consecutive events.
- `arrival`: The cumulative sum of `inter_arrival` times.
- `type`: An integer representing the type of event.
- `mark`: The size of the mark, providing additional information associated with the event.
- `N`: A counting process that tracks the number of events.
- `Nc`: A counting process that tracks the number of events, weighted by mark.
- `lambda`: The left-continuous intensity process.
- `lambda_component`: The component of the intensity process, λ_{ij}, that excludes `mu`.
- `rambda`: The right-continuous intensity process.
- `rambda_component`: The right-continuous version of `lambda_component`.

Functions for printing `hreal` objects are provided.

Usage

```
## S3 method for class 'hreal'
print(x, n = 20, ...)

## S3 method for class 'hreal'
summary(object, n = 20, ...)

## S3 method for class 'hreal'
as.matrix(x, ...)
```

Arguments

`x`	An S3 object of class `hreal`.
`n`	The number of rows to display.
`...`	Additional arguments passed to or from other methods.
`object`	An S3 object of class `hreal`.

hsim *Simulate multivariate Hawkes process with exponential kernel.*

Description

The method simulate multivariate Hawkes processes. The object `hspec-class` contains the parameter values such as `mu`, `alpha`, `beta`. The mark (jump) structure may or may not be included. It returns an object of class `hreal` which contains `inter_arrival`, `arrival`, `type`, `mark`, `N`, `Nc`, `lambda`, `lambda_component`, `rambda`, `rambda_component`.

Usage

```
hsim(
  object,
  size = 100,
  lambda_component0 = NULL,
  N0 = NULL,
  Nc0 = NULL,
  verbose = FALSE,
  ...
)

## S4 method for signature 'hspec'
hsim(
  object,
  size = 100,
  lambda_component0 = NULL,
  N0 = NULL,
  Nc0 = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

object — hspec-class. S4 object that specifies the parameter values.

size — Number of observations.

lambda_component0 — Initial values for the lambda component λ_{ij}. Can be a numeric value or a matrix. Must have the same number of rows and columns as `alpha` or `beta` in `object`.

N0 — Starting values of N with default value 0.

Nc0 — Starting values of Nc with default value 0.

verbose — Logical. If `TRUE`, print progress messages during the simulation. Default is `FALSE`.

... — Further arguments passed to or from other methods.

Value

hreal S3-object, summary of the Hawkes process realization.

Examples

```
# example 1

mu <- 1; alpha <- 1; beta <- 2
h <- new("hspec", mu=mu, alpha=alpha, beta=beta)
hsim(h, size=100)


# example 2
mu <- matrix(c(0.1, 0.1), nrow=2)
alpha <- matrix(c(0.2, 0.1, 0.1, 0.2), nrow=2, byrow=TRUE)
beta <- matrix(c(0.9, 0.9, 0.9, 0.9), nrow=2, byrow=TRUE)
h <- new("hspec", mu=mu, alpha=alpha, beta=beta)
res <- hsim(h, size=100)
print(res)
```

hspec-class — *An S4 Class Representing an Exponential Marked Hawkes Model*

Description

This class defines a marked Hawkes model with an exponential kernel. The intensity of the ground process is expressed as:

$$\lambda(t) = \mu + \int_{(-\infty,t)\times E} (\alpha + g(u,z)) e^{-\beta(t-u)} M(du \times dz).$$

For more details, refer to the vignettes.

Details

μ is base intensity, typically a constant vector or a function.

α is a constant matrix representing the impact on intensities after events, stored in the `alpha` slot.

β is a constant matrix for exponential decay rates, stored in the `beta` slot.

z represents the mark and can be generated by `rmark` slot.

g is represented by `eta` when it is linear function of z, and by `impact` when it is a genenral function.

`mu`, `alpha` and `beta` are required slots for every exponential Hawkes model. `rmark` and `impact` are additional slots.

Slots

`mu` A numeric value, matrix, or function. If numeric, it is automatically converted to a matrix.

`alpha` A numeric value, matrix, or function. If numeric, it is automatically converted to a matrix, representing the exciting term.

`beta` A numeric value, matrix, or function. If numeric, it is automatically converted to a matrix, representing the exponential decay.

`eta` A numeric value, matrix, or function. If numeric, it is automatically converted to a matrix, representing the impact of an additional mark.

`impact` A function describing the after-effects of the mark on λ, with the first argument always being `param`.

`dimens` The dimension of the model.

`rmark` A function that generates marks for the counting process, used in simulations.

`dmark` A density function for the mark, used in estimation.

`type_col_map` A mapping between type and column number of the kernel used in multi-kernel models.

`rresidual` A function for generating residuals, analogous to the R random number generator function, specifically for the discrete Hawkes model.

`dresidual` A density function for the residual.

`presidual` A distribution function for the residual.

`qresidual` A quantile function for the residual.

Examples

```
MU <- matrix(c(0.2), nrow = 2)
ALPHA <- matrix(c(0.75, 0.92, 0.92, 0.75), nrow = 2, byrow=TRUE)
BETA <- matrix(c(2.25, 2.25, 2.25, 2.25), nrow = 2, byrow=TRUE)
mhspec2 <- new("hspec", mu=MU, alpha=ALPHA, beta=BETA)
mhspec2
```

hvol *Compute Hawkes volatility*

Description

This function computes Hawkes volatility. Only works for bi-variate Hawkes process.

Usage

```
hvol(
  object,
  horizon = 1,
  inter_arrival = NULL,
  type = NULL,
  mark = NULL,
  dependence = FALSE,
  lambda_component0 = NULL,
  ...
)

## S4 method for signature 'hspec'
hvol(
  object,
  horizon = 1,
  inter_arrival = NULL,
  type = NULL,
  mark = NULL,
  dependence = FALSE,
  lambda_component0 = NULL,
  ...
)
```

Arguments

`object` hspec-class

`horizon` Time horizon for volatility.

`inter_arrival` Inter-arrival times of events which includes inter-arrival for events that occur in all dimensions. Start with zero.

`type` A vector of dimensions. Distinguished by numbers, 1, 2, 3, and so on. Start with zero.

`mark` A vector of mark (jump) sizes. Start with zero.

`dependence` Dependence between mark and previous sigma-algebra.

`lambda_component0` A matrix of the starting values of lambda component.

`...` Further arguments passed to or from other methods.

infer_lambda *Infer lambda process with given Hawkes model and realized path*

Description

This method compute the inferred lambda process and returns it as `hreal` form. If we have realized path of Hawkes process and its parameter value, then we can compute the inferred lambda processes. Similarly with other method such as `hfit`, the input arguments are `inter_arrival`, `type`, `mark`, or equivalently, `N` and `Nc`.

Usage

```
infer_lambda(
  object,
  inter_arrival = NULL,
  type = NULL,
  mark = NULL,
  N = NULL,
  Nc = NULL,
  lambda_component0 = NULL,
  N0 = NULL,
  Nc0 = NULL,
  ...
)

## S4 method for signature 'hspec'
infer_lambda(
  object,
  inter_arrival = NULL,
  type = NULL,
  mark = NULL,
  N = NULL,
  Nc = NULL,
  lambda_component0 = NULL,
  N0 = NULL,
  Nc0 = NULL,
  ...
)
```

Arguments

`object`	`hspec-class`. This object includes the parameter values.
`inter_arrival`	inter-arrival times of events. This includes inter-arrival for events that occur in all dimensions. Start with zero.
`type`	a vector of dimensions. Distinguished by numbers, 1, 2, 3, and so on. Start with zero.

mark a vector of mark (jump) sizes. Start with zero.

N Hawkes process. If not provided, then generate using inter_arrival and type.

Nc mark accumulated Hawkes process. If not provided, then generate using inter_arrival, type and mark.

lambda_component0

Initial values for the lambda component λ_{ij}. Can be a numeric value or a matrix. Must have the same number of rows and columns as alpha or beta in object.

N0 the initial values of N.

Nc0 the initial values of Nc.

... further arguments passed to or from other methods.

Value

hreal S3-object, with inferred intensity.

Examples

```
mu <- c(0.1, 0.1)
alpha <- matrix(c(0.2, 0.1, 0.1, 0.2), nrow=2, byrow=TRUE)
beta <- matrix(c(0.9, 0.9, 0.9, 0.9), nrow=2, byrow=TRUE)
h <- new("hspec", mu=mu, alpha=alpha, beta=beta)
res <- hsim(h, size=100)
summary(res)
res2 <- infer_lambda(h, res$inter_arrival, res$type)
summary(res2)
```

logLik,hspec-method *Compute the Log-Likelihood Function*

Description

Calculates the log-likelihood for the Hawkes model.

Usage

```
## S4 method for signature 'hspec'
logLik(
  object,
  inter_arrival,
  type = NULL,
  mark = NULL,
  N = NULL,
  Nc = NULL,
  N0 = NULL,
  Nc0 = NULL,
  lambda_component0 = NULL,
  infer = FALSE,
  ...
)
```

Arguments

`object` An `hspec-class` object containing parameter values for computing the log-likelihood.

`inter_arrival` A vector of inter-arrival times for events across all dimensions, starting with zero.

`type` A vector indicating the dimensions, represented by numbers (1, 2, 3, etc.), starting with zero.

`mark` A vector of mark (jump) sizes, starting with zero.

`N` A matrix representing counting processes.

`Nc` A matrix of counting processes weighted by mark sizes.

`N0` A matrix of initial values for `N`.

`Nc0` A matrix of initial values for `Nc`.

`lambda_component0` Initial values for the lambda component λ_{ij}. Can be a numeric value or a matrix. Must have the same number of rows and columns as `alpha` or `beta` in `object`.

`infer` Logical

`...` Additional arguments passed to or from other methods.

See Also

`hspec-class`, `hfit,hspec-method`

`residual_process` *Compute residual process*

Description

Using random time change, this function compute the residual process, which is the inter-arrival time of a standard Poisson process. Therefore, the return values should follow the exponential distribution with rate 1, if model and rambda are correctly specified.

Usage

```
residual_process(
  component,
  inter_arrival,
  type,
  rambda_component,
  mu,
  beta,
  dimens = NULL,
  mark = NULL,
  N = NULL,
  Nc = NULL,
  lambda_component0 = NULL,
  N0 = NULL,
  ...
)
```

Arguments

component The component of type to get the residual process.

inter_arrival Inter-arrival times of events. This includes inter-arrival for events that occur in all dimensions. Start with zero.

type A vector of types distinguished by numbers, 1, 2, 3, and so on. Start with zero.

rambda_component
Right continuous version of lambda process.

mu Numeric value or matrix or function. If numeric, automatically converted to matrix.

beta Numeric value or matrix or function. If numeric, automatically converted to matrix, exponential decay.

dimens Dimension of the model. If omitted, set to be the length of mu.

mark A vector of realized mark (jump) sizes. Start with zero.

N A matrix of counting processes.

Nc A matrix of counting processes weighted by mark.

lambda_component0
The initial values of lambda component. Must have the same dimensional matrix with hspec.

N0 The initial value of N

... Further arguments passed to or from other methods.

Examples

```
mu <- c(0.1, 0.1)
alpha <- matrix(c(0.2, 0.1, 0.1, 0.2), nrow=2, byrow=TRUE)
beta <- matrix(c(0.9, 0.9, 0.9, 0.9), nrow=2, byrow=TRUE)
h <- new("hspec", mu=mu, alpha=alpha, beta=beta)
res <- hsim(h, size=1000)
rp <- residual_process(component = 1, res$inter_arrival, res$type, res$rambda_component, mu, beta)
```

set_residual *Set Residual Distribution Functions for Hawkes Model Specification*

Description

Sets residual distribution functions (density, CDF, quantile, and random generation) for a Hawkes model specification object with fixed parameters.

Usage

```
set_residual(
  object,
  param,
  dresidual = NULL,
  presidual = NULL,
  qresidual = NULL,
  rresidual = NULL,
  ...
)

## S4 method for signature 'hspec'
set_residual(
  object,
  param,
  dresidual = NULL,
  presidual = NULL,
  qresidual = NULL,
  rresidual = NULL
)
```

Arguments

`object`	An object of class `hspec` (Hawkes model specification)
`param`	A named numeric vector of parameters for the residual distribution
`dresidual`	Density function of the residual distribution (optional)
`presidual`	Cumulative distribution function (CDF) of the residual distribution (optional)
`qresidual`	Quantile function of the residual distribution (optional)
`rresidual`	Random generation function of the residual distribution (optional)
`...`	Additional arguments for future extensions

Details

This method allows setting residual distribution functions for a flexible model.

The `param` argument in these functions defaults to the parameters provided during setup and is used for estimation.

Value

An updated hspec object with residual functions set

Examples

```
## Not run:
# Create basic Hawkes specification
hspec_obj <- new("hspec",
                 mu = matrix(0.1, nrow = 1),
                 alpha = matrix(0.5, nrow = 1),
                 beta = matrix(1.0, nrow = 1))

# Set residual distribution parameters
params <- c(a = 0.5, ell = 1.0)

# Apply residual functions
hspec_obj <- set_residual(
  hspec_obj,
  param = params,
  dresidual = dtzexp,
  presidual = ptzexp,
  qresidual = qtzexp,
  rresidual = rtzexp
)

# Check resulting functions
hspec_obj@dresidual
hspec_obj@rresidual

## End(Not run)
```

tzexp *Trapezoid + Exponential Distribution*

Description

These functions implement a custom distribution combining a trapezoidal section (0 < x < a) and an exponential tail ($x \geq a$). The distribution is parameterized by:

- a: transition point between trapezoid and exponential
- ell: rate parameter for the exponential tail

Usage

```
dtzexp(x, a, ell)

ptzexp(q, a, ell)
```

```
qtzexp(p, a, ell)

rtzexp(n, a, ell)
```

Arguments

`x, q`	vector of quantiles
`a`	location parameter for transition (must be > 0)
`ell`	rate parameter for exponential decay (must be > 0)
`p`	vector of probabilities
`n`	number of observations

Details

Density, distribution function, quantile function and random generation for a custom trapezoid + exponential distribution.

The trapezoid+exponential distribution has the probability density function:

$$f(x) = \begin{cases} 0 & \text{if } x \le 0 \\ \frac{(p\ell - c)}{a}x + c & \text{if } 0 < x < a \\ p\ell e^{-\ell(x-a)} & \text{if } x \ge a \end{cases}$$

where:

$$p = \frac{\ell - \frac{a\ell}{3}}{\frac{a^2\ell^2}{6} + \frac{2a\ell}{3} + 1}$$

$$c = \frac{2 - 2p - p\ell a}{a}$$

The trapezoid+exponential distribution has the following characteristics:

- Support on $[0, \infty)$
- Continuous probability distribution
- Linear density from 0 to a
- Exponential decay for x > a

Value

- `dtzexp` gives the density
- `ptzexp` gives the distribution function
- `qtzexp` gives the quantile function
- `rtzexp` generates random deviates

Index