Package ‘nhppp’

January 9, 2025

Title Simulating Nonhomogeneous Poisson Point Processes

Version 1.0.2

Description Simulates events from one dimensional nonhomogeneous Poisson point processes (NH-PPPs) as per Trikalinos and Sereda (2024, <doi:10.48550/arXiv.2402.00358> and 2024, <doi:10.1371/journal.pone.0311311>). Functions are based on three algorithms that provably sample from a target NHPPP: the time-transformation of a homogeneous Poisson process (of intensity one) via the inverse of the integrated intensity function (Cinlar E, ``Theory of stochastic processes'' (1975, ISBN:0486497996)); the generation of a Poisson number of order statistics from a fixed density function; and the thinning of a majorizing NHPPP via an acceptance-rejection scheme (Lewis PAW, Shedler, GS (1979) <doi:10.1002/nav.3800260304>).

License GPL (>= 3)

Imports lifecycle, rstream, Rcpp (>= 1.0.12)

LinkingTo Rcpp

Encoding UTF-8

Language es

RoxygenNote 7.3.2

Suggests data.table, knitr, rlecuyer, rmarkdown, testthat, tictoc, truncnorm, withr

Config/Needs/website rmarkdown

URL `https://bladder-ca.github.io/nhppp/`,
`https://github.com/bladder-ca/nhppp`

BugReports `https://github.com/bladder-ca/nhppp/issues`

VignetteBuilder knitr

Depends R (>= 2.10)

LazyData true

NeedsCompilation yes

Author Thomas Trikalinos [aut, cre, cph]
(<`https://orcid.org/0000-0002-3990-1848`>),
Yuliia Sereda [aut] (<`https://orcid.org/0000-0002-4017-4561`>)

Maintainer Thomas Trikalinos <thomas_trikalinos@brown.edu>

Repository CRAN

Date/Publication 2025-01-09 16:20:02 UTC

Contents

`draw` *Generic function for simulating from NHPPPs given the intensity function or the cumulative intensity function.*

Description

This is a wrapper to the package's specific functions, and thus somewhat slower. For time-intensive simulations prefer one of the specific functions.

Usage

```
draw(
  Lambda = NULL,
  Lambda_inv = NULL,
  lambda = NULL,
  line_majorizer_intercept = NULL,
  line_majorizer_slope = NULL,
  line_majorizer_is_loglinear = FALSE,
```

```
    step_majorizer_vector = NULL,
    t_min = NULL,
    t_max = NULL,
    atmost1 = FALSE,
    atleast1 = FALSE
  )
```

Arguments

`Lambda` (function, double vector) the integrated (cumulative) rate of the NHPPP

`Lambda_inv` (function, double vector) the inverse of 'Lambda()'

`lambda` (function) the instantaneous rate

`line_majorizer_intercept`
The intercept `alpha` of the loglinear majorizer function: `alpha + beta * t` or `exp(alpha + beta * t)`

`line_majorizer_slope`
The slope `beta` of the loglinear majorizer function: `alpha + beta * t` or `exp(alpha + beta * t)`

`line_majorizer_is_loglinear`
(boolean) if `TRUE` the majorizer is loglinear `exp(alpha + beta * t)`; if `FALSE` it is a linear function

`step_majorizer_vector`
(vector, double) `K` constant majorizing rates, one per interval; all intervals are of equal length (regular)

`t_min` (double) the lower bound of the interval

`t_max` (double) the upper bound of the interval

`atmost1` boolean, draw at most 1 event time

`atleast1` boolean, draw at least 1 event time in interval

Value

a vector of event times

`draw_cumulative_intensity` *Simulate from a non homogeneous Poisson Point Process (NHPPP) over an interval when you know the cumulative intensity and its inverse.*

Description

Sample NHPPP times using the inversion method

Usage

```
draw_cumulative_intensity(Lambda, Lambda_inv, t_min, t_max, atmost1 = FALSE)
```

Arguments

`Lambda`	(function, double vector) a continuous increasing R to R map which is the integrated rate of the NHPPP
`Lambda_inv`	(function, double vector) the inverse of `Lambda()`
`t_min`	(double) the lower bound of the time interval
`t_max`	(double) the upper bound of the time interval
`atmost1`	boolean, draw at most 1 event time

Value

a vector of event times (t_); if no events realize, a vector of length 0

`draw_intensity` *Generic function for simulating from NHPPPs given the intensity function.*

Description

Sample from NHPPPs given the intensity function This is a wrapper to the package's specific functions, and thus somewhat slower. For time-intensive simulations prefer one of the specific functions.

Usage

```
draw_intensity(
  lambda,
  line_majorizer_intercept = NULL,
  line_majorizer_slope = NULL,
  line_majorizer_is_loglinear = FALSE,
  step_majorizer_vector = NULL,
  t_min = NULL,
  t_max = NULL,
  atmost1 = FALSE
)
```

Arguments

`lambda`	(function) the instantaneous rate
`line_majorizer_intercept`	The intercept `alpha` of the loglinear majorizer function: `alpha + beta * t` or `exp(alpha + beta * t)`
`line_majorizer_slope`	The slope `beta` of the loglinear majorizer function: `alpha + beta * t` or `exp(alpha + beta * t)`
`line_majorizer_is_loglinear`	(boolean) if `TRUE` the majorizer is loglinear `exp(alpha + beta * t)`; if `FALSE` it is a linear function

`step_majorizer_vector`
: (vector, double) K constant majorizing rates, one per interval; all intervals are of equal length (regular)

`t_min` (double) the lower bound of the interval

`t_max` (double) the upper bound of the interval

`atmost1` boolean, draw at most 1 event time

Value

a vector of event times

`draw_sc_linear` *Special case: Simulate from a non homogeneous Poisson Point Process (NHPPP) from (t_min, t_max) with linear intensity function (inversion method)*

Description

Sample NHPPP times from a linear intensity function using the inversion method, optionally using an `rstream` generator

Usage

```
draw_sc_linear(intercept, slope, t_min, t_max, atmost1 = FALSE)
```

Arguments

`intercept` (double) the intercept

`slope` (double) the slope

`t_min` (double) lower bound of the time interval

`t_max` (double) upper bound of the time interval

`atmost1` boolean, draw at most 1 event time

Value

a vector of event times (t_); if no events realize, a vector of length 0

Examples

```
x <- draw_sc_linear(intercept = 0, slope = 0.2, t_min = 0, t_max = 10)
```

`draw_sc_loglinear` *Special case: Simulate from a non homogeneous Poisson Point Process (NHPPP) from (t_min, t_max) with log-linear intensity function (inversion method)*

Description

Sample NHPPP times from an log linear intensity function using the inversion method, optionally using an `rstream` generator

Usage

```
draw_sc_loglinear(intercept, slope, t_min, t_max, atmost1 = FALSE)
```

Arguments

`intercept`	(double) the intercept in the exponent
`slope`	(double) the slope in the exponent
`t_min`	(double) lower bound of the time interval
`t_max`	(double) upper bound of the time interval
`atmost1`	boolean, draw at most 1 event time

Value

a vector of event times (t_); if no events realize, a vector of length 0

Examples

```
x <- draw_sc_loglinear(intercept = 0, slope = 0.2, t_min = 0, t_max = 10)
```

`draw_sc_step` *Simulate a piecewise constant-rate Poisson Point Process over* `(t_min, t_max]` *(inversion method) The intervals need not have the same length.*

Description

Simulate a piecewise constant-rate Poisson Point Process over `(t_min, t_max]` (inversion method) The intervals need not have the same length.

Usage

```
draw_sc_step(lambda_vector, time_breaks, atmost1 = FALSE, atleast1 = FALSE)
```

Arguments

`lambda_vector`	(scalar, double) `K` constant rates, one per interval
`time_breaks`	(vector, double) `K+1` time points defining `K` intervals of constant rates: `[t_1 = range_t[1], t_2)`: the first interval `[t_k, t_{k+1})`: the k-th interval `[t_{K}, t_{K+1} = range_t[2])`: the `K`-th (last) interval
`atmost1`	boolean, draw at most 1 event time
`atleast1`	boolean, draw at least 1 event time

Value

a vector of event times t if no events realize, it will have 0 length

Examples

```
x <- draw_sc_step(lambda_vector = rep(1, 5), time_breaks = c(0:5))
```

---

draw_sc_step_regular — *Sampling from NHPPPs with piecewise constant intensities with same interval lengths (non-vectorized)*

Description

Sampling from NHPPPs with piecewise constant intensities with same interval lengths (non-vectorized)

Usage

```
draw_sc_step_regular(
  Lambda_vector = NULL,
  lambda_vector = NULL,
  t_min = NULL,
  t_max = NULL,
  atmost1 = FALSE,
  atleast1 = FALSE
)
```

Arguments

`Lambda_vector`	(scalar, double) `K` integrated intensity rates at the end of each interval
`lambda_vector`	(scalar, double) `K` constant intensity rates, one per interval
`t_min`	(scalar, double) lower bound of the time interval
`t_max`	(scalar, double) upper bound of the time interval
`atmost1`	boolean, draw at most 1 event time
`atleast1`	boolean, draw at least 1 event time

Value

a vector of event times t if no events realize, it will have 0 length

Examples

```
x <- draw_sc_step_regular(Lambda_vector = 1:5, t_min = 0, t_max = 5)
```

`get_step_majorizer` *Piecewise constant (step) majorizer for K-Lipschitz functions over an interval (vectorized over the* `breaks` *argument).*

Description

Return a piecewise constant (step) majorizer for K-Lipschitz functions over an interval. The function is vectorized over the `breaks` argument. The returned object has the same dimensions as `breaks`.

Usage

```
get_step_majorizer(fun, breaks, is_monotone = TRUE, K = 0)
```

Arguments

`fun`	A function object with a single argument `x`. If `x` is a matrix, `fun` should be vectorized over it.
`breaks`	(vector or matrix) The set of `M+1` boundaries for the `M` subintervals in `x`. If breaks is a matrix, each row is treated as a separate set of breaks.
`is_monotone`	(boolean) Is the function monotone? (Default is `TRUE`.)
`K`	(double) A non-negative number for the Lipschitz cone. (Default is 0.)

Value

A vector of length `M` with the values of the piecewise constant majorizer

Examples

```
get_step_majorizer(fun = abs, breaks = -5:5, is_monotone = FALSE, K = 1)
```

ppp — *Simulate a homogeneous Poisson Point Process in (t_min, t_max]*

Description

Simulate a homogeneous Poisson Point Process in (t_min, t_max]

Usage

```
ppp(rate, t_min, t_max, atmost1 = FALSE, tol = 10^-6)
```

Arguments

`rate`	(scalar, double) constant instantaneous rate
`t_min`	(scalar, double) the lower bound of the time interval
`t_max`	(scalar, double) the upper bound of the time interval
`atmost1`	boolean, draw at most 1 event time
`tol`	the probability that we will have more than the drawn events in (t_min, t_max]

Value

a vector of event times t if no events realize, it will have 0 length

Examples

```
x <- ppp(rate = 1, t_min = 0, t_max = 10, tol = 10^-6)
```

ppp_exactly_n — *Simulate exactly* `n` *points from a homogeneous Poisson Point Process over (t_min, t_max]*

Description

Simulate exactly `n` points from a homogeneous Poisson Point Process over (t_min, t_max]

Usage

```
ppp_exactly_n(n, t_min, t_max)
```

Arguments

`n`	(int) the number of points to be simulated
`t_min`	(double) the lower bound of the time interval
`t_max`	(double) the upper bound of the time interval

Value

a vector of event times of size n

Examples

```
x <- ppp_exactly_n(n = 10, t_min = 0, t_max = 10)
```

---

ppp_next_n — *Simulate n events from a homogeneous Poisson Point Process.*

Description

Simulate n events from a homogeneous Poisson Point Process.

Usage

```
ppp_next_n(n = 1, rate = 1, t_min = 0, rng_stream = deprecated())
```

Arguments

n	scalar number of samples
rate	scalar instantaneous rate
t_min	scalar for the starting time value
rng_stream	**[Deprecated]** an rstream object

Value

a vector with event times t (starting from t_min)

Examples

```
x <- ppp_next_n(n = 10, rate = 1, t_min = 0)
```

vdraw *Vectorized generic function for simulating from NHPPPs given the intensity function or the cumulative intensity function*

Description

This is a wrapper to the package's specific functions, and thus slightly slower. For time-intensive simulations prefer one of the specific functions.

Usage

```
vdraw(
  lambda = NULL,
  lambda_args = NULL,
  Lambda_maj_matrix = NULL,
  lambda_maj_matrix = NULL,
  Lambda = NULL,
  Lambda_inv = NULL,
  Lambda_args = NULL,
  Lambda_inv_args = NULL,
  t_min = NULL,
  t_max = NULL,
  rate_matrix_t_min = NULL,
  rate_matrix_t_max = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  atleast1 = FALSE,
  atmostB = NULL
)
```

Arguments

`lambda` (function) intensity function, vectorized

`lambda_args` (list) optional arguments to pass to `lambda`

`Lambda_maj_matrix` (matrix) integrated intensity rates at the end of each interval

`lambda_maj_matrix` (matrix) intensity rates, one per interval

`Lambda` (function, double vector) an increasing function which is the integrated rate of the NHPPP. It should take a vectorized argument t for times and an optional arguments list.

`Lambda_inv` (function, double vector) the inverse of `Lambda()`, also in vectorized form It should take a vectorized argument z and an optional arguments list.

`Lambda_args` (list) optional arguments to pass to Lambda.

`Lambda_inv_args`
: (list) optional arguments to pass to Lambda_inv().

`t_min`
: (scalar | vector | column matrix) is the lower bound of a subinterval of (rate_matrix_t_min, rate_matrix_t_max]. If set, times are sampled from the subinterval. If omitted, it is equivalent to `rate_matrix_t_min`.

`t_max`
: (scalar | vector | column matrix) is the upper bound of a subinterval of (rate_matrix_t_min, rate_matrix_t_max]. If set, times are sampled from the subinterval. If omitted, it is equivalent to `rate_matrix_t_max`.

`rate_matrix_t_min`
: (scalar | vector | column matrix) is the lower bound of the time interval for each row of (Lambda|lambda)_maj_matrix. The length of this argument is the number of point processes that should be drawn.

`rate_matrix_t_max`
: (scalar | vector | column matrix) the upper bound of the time interval for each row of (Lambda|lambda)_maj_matrix. The length of this argument is the number of point processes that should be drawn.

`tol`
: (scalar, double) tolerance for the number of events

`atmost1`
: boolean, draw at most 1 event time

`atleast1`
: boolean, draw at least 1 event time

`atmostB`
: If not NULL, draw at most B (B>0) event times. NULL means ignore.

Value

a vector of event times

`vdraw_cumulative_intensity`
Vectorized simulation from a non homogeneous Poisson Point Process (NHPPP) from (t_min, t_max) given the cumulative intensity function and its inverse

---

Description

Sample NHPPP times using the cumulative intensity function and its inverse.

Usage

```
vdraw_cumulative_intensity(
  Lambda,
  Lambda_inv,
  t_min,
  t_max,
  Lambda_args = NULL,
  Lambda_inv_args = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  atleast1 = FALSE
)
```

Arguments

`Lambda`	(function, double vector) an increasing function which is the integrated rate of the NHPPP. It should take a vectorized argument t for times and an optional arguments list.
`Lambda_inv`	(function, double vector) the inverse of `Lambda()`, also in vectorized form It should take a vectorized argument z and an optional arguments list.
`t_min`	(scalar | vector | column matrix) the lower bound of the interval for each sampled point process The length of this argument is the number of point processes that should be drawn.
`t_max`	(scalar | vector | column matrix) the upper bound of the interval for each sampled point process The length of this argument is the number of point processes that should be drawn.
`Lambda_args`	(list) optional arguments to pass to Lambda.
`Lambda_inv_args`	(list) optional arguments to pass to Lambda_inv().
`tol`	the tolerange for the calulations.
`atmost1`	boolean, draw at most 1 event time per sampled point process.
`atleast1`	boolean, draw at least 1 event time

Value

a matrix of event times with one row per sampled point process.

`vdraw_intensity` *Vectorized sampling from a non homogeneous Poisson Point Process (NHPPP) from an interval (thinning method) with piecewise constant_majorizers (C++)*

Description

Vectorized sampling from a non homogeneous Poisson Point Process (NHPPP) from an interval (thinning method) with piecewise constant_majorizers. The majorizers are step functions over equal-length time intevals.

Usage

```
vdraw_intensity(
  lambda = NULL,
  lambda_args = NULL,
  Lambda_maj_matrix = NULL,
  lambda_maj_matrix = NULL,
  rate_matrix_t_min = NULL,
  rate_matrix_t_max = NULL,
  t_min = NULL,
```

```
  t_max = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  atleast1 = FALSE,
  atmostB = NULL
)
```

Arguments

lambda	(function) intensity function, vectorized
lambda_args	(list) optional arguments to pass to lambda
Lambda_maj_matrix	(matrix) integrated intensity rates at the end of each interval
lambda_maj_matrix	(matrix) intensity rates, one per interval
rate_matrix_t_min	(scalar | vector | column matrix) is the lower bound of the time interval for each row of (Lambda|lambda)_maj_matrix. The length of this argument is the number of point processes that should be drawn.
rate_matrix_t_max	(scalar | vector | column matrix) the upper bound of the time interval for each row of (Lambda|lambda)_maj_matrix. The length of this argument is the number of point processes that should be drawn.
t_min	(scalar | vector | column matrix) is the lower bound of a subinterval of (rate_matrix_t_min, rate_matrix_t_max]. If set, times are sampled from the subinterval. If omitted, it is equivalent to rate_matrix_t_min.
t_max	(scalar | vector | column matrix) is the upper bound of a subinterval of (rate_matrix_t_min, rate_matrix_t_max]. If set, times are sampled from the subinterval. If omitted, it is equivalent to rate_matrix_t_max.
tol	(scalar, double) tolerance for the number of events
atmost1	boolean, draw at most 1 event time
atleast1	boolean, draw at least 1 event time
atmostB	If not NULL, draw at most B (B>0) event times. NULL means ignore.

Value

a matrix of event times (columns) per draw (rows) NAs are structural empty spots

Examples

```
x <- vdraw_intensity(
  lambda = function(x, ...) 0.1 * x,
  lambda_maj_matrix = matrix(rep(1, 5), nrow = 1),
  rate_matrix_t_min = 1,
  rate_matrix_t_max = 5
)
```

`vdraw_sc_step_regular` *Vectorized sampling from NHPPPs with piecewise constant intensities with same interval lengths*

Description

Simulate a piecewise constant-rate Poisson Point Process over `(t_min, t_max]` (inversion method) where the intervals have the same length (are "regular").

Usage

```
vdraw_sc_step_regular(
  lambda_matrix = NULL,
  Lambda_matrix = NULL,
  rate_matrix_t_min = NULL,
  rate_matrix_t_max = NULL,
  t_min = NULL,
  t_max = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  atmostB = NULL,
  atleast1 = FALSE
)
```

Arguments

`lambda_matrix` (matrix) intensity rates, one per interval

`Lambda_matrix` (matrix) integrated intensity rates at the end of each interval

`rate_matrix_t_min`
(scalar | vector | column matrix) is the lower bound of the time interval for each row of (Lambda|lambda)_maj_matrix. The length of this argument is the number of point processes that should be drawn.

`rate_matrix_t_max`
(scalar | vector | column matrix) the upper bound of the time interval for each row of (Lambda|lambda)_maj_matrix. The length of this argument is the number of point processes that should be drawn.

`t_min` (scalar | vector | column matrix) is the lower bound of a subinterval of (rate_matrix_t_min, rate_matrix_t_max]. If set, times are sampled from the subinterval. If omitted, it is equivalent to `rate_matrix_t_min`.

`t_max` (scalar | vector | column matrix) is the upper bound of a subinterval of (rate_matrix_t_min, rate_matrix_t_max]. If set, times are sampled from the subinterval. If omitted, it is equivalent to `rate_matrix_t_max`.

`tol` (scalar, double) tolerance for the number of events

`atmost1` boolean, draw at most 1 event time

`atmostB` If not NULL, draw at most B (B>0) event times. NULL means ignore.

`atleast1` boolean, draw at least 1 event time

Value

a vector of event times t if no events realize, it will have 0 length

Examples

```
x <- vdraw_sc_step_regular(
  Lambda_matrix = matrix(1:5, nrow = 1),
  rate_matrix_t_min = 100,
  rate_matrix_t_max = 110,
  atmost1 = FALSE
)
```

`ztdraw_cumulative_intensity` *Simulate from a zero-truncated non homogeneous Poisson Point Process (zt-NHPPP) from (t_min, t_max) (order statistics method)*

Description

Sample zero-truncated NHPPP times using the order statistics method, optionally using an `rstream` generator

Usage

```
ztdraw_cumulative_intensity(Lambda, Lambda_inv, t_min, t_max, atmost1 = FALSE)
```

Arguments

`Lambda`	(function, double vector) a continuous increasing R to R map which is the integrated rate of the NHPPP
`Lambda_inv`	(function, double vector) the inverse of `Lambda()`
`t_min`	(double) the lower bound of the time interval
`t_max`	(double) the upper bound of the time interval
`atmost1`	(boolean) draw at most 1 event time

Value

a vector of at least 1 event times

ztdraw_sc_linear

Simulate `size` *samples from a zero-truncated non homogeneous Poisson Point Process (zt-NHPPP) from (t_min, t_max) with linear intensity function*

Description

Sample zero-truncated NHPPP times from a linear intensity function using the inversion method, optionally using an `rstream` generator

Usage

```
ztdraw_sc_linear(intercept, slope, t_min, t_max, atmost1 = FALSE)
```

Arguments

`intercept`	(double) the intercept
`slope`	(double) the slope
`t_min`	(double) the lower bound of the time interval
`t_max`	(double) the upper bound of the time interval
`atmost1`	(boolean) draw 1 event time

Value

a vector of at least 1 event times

Examples

```
x <- ztdraw_sc_linear(intercept = 0, slope = 0.2, t_min = 0, t_max = 10)
```

ztdraw_sc_loglinear

Simulate from a zero-truncated non homogeneous Poisson Point Process (zt-NHPPP) from (t_min, t_max) with a log-linear intensity function

Description

Sample zt-NHPPP times from an log-linear intensity function

Usage

```
ztdraw_sc_loglinear(intercept, slope, t_min, t_max, atmost1 = FALSE)
```

Arguments

`intercept`	(double) the intercept in the exponent
`slope`	(double) the slope in the exponent
`t_min`	(double) the lower bound of the time interval
`t_max`	(double) the upper bound of the time interval
`atmost1`	boolean, 1 event time

Value

a vector of at least 1 event times

Examples

```
x <- ztdraw_sc_loglinear(intercept = 0, slope = 0.2, t_min = 0, t_max = 10)
```

---

ztppp — *Simulate a zero-truncated homogeneous Poisson Point Process over (t_min, t_max]*

Description

Simulate a zero-truncated homogeneous Poisson Point Process over (t_min, t_max]

Usage

```
ztppp(rate, t_min, t_max, atmost1 = FALSE)
```

Arguments

`rate`	(scalar, double) constant instantaneous rate
`t_min`	(scalar, double) lower bound of the time interval
`t_max`	(scalar, double) upper bound of the time interval
`atmost1`	boolean, draw at most 1 event time

Value

a vector of event times of size `size`

Examples

```
x <- ztppp(t_min = 0, t_max = 10, rate = 0.001)
```

Index