Package 'nhppp'

May 29, 2024

Title Simulating Nonhomogeneous Poisson Point Processes

Version 0.1.4

Description Simulates events from one dimensional nonhomogeneous Poisson point processes (NHPPPs) as per Trikalinos and Sereda (2024, <doi:10.48550/arXiv.2402.00358>). 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 7.3.1

Suggests data.table, rlecuyer, testthat, withr

Config/Needs/website rmarkdown

URL https://github.com/bladder-ca/nhppp

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

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 2024-05-28 22:50:02 UTC

R topics documented:

check_ppp_sample_validity . . . 3
check_ppp_vector_validity . . . 4
compare_ppp_vectors . . . 4
draw . . . 5
draw_cumulative_intensity_inversion . . . 6
draw_cumulative_intensity_orderstats . . . 7
draw_intensity . . . 8
draw_intensity_step . . . 9
draw_sc_linear . . . 10
draw_sc_loglinear . . . 11
draw_sc_step . . . 12
draw_sc_step_regular . . . 13
expect_no_error . . . 14
get_step_majorizer . . . 14
inverse_with_uniroot . . . 15
inverse_with_uniroot_sorted . . . 16
Lambda_exp_form . . . 17
Lambda_inv_exp_form . . . 17
Lambda_inv_linear_form . . . 18
Lambda_linear_form . . . 18
make_cumulative_Lambda_matrix . . . 19
make_lambda_matrix . . . 19
make_range_t_matrix . . . 20
mat_cumsum_columns . . . 20
mat_cumsum_columns_with_scalar_ceiling . . . 21
mat_cumsum_columns_with_vector_ceiling . . . 21
mat_diff_columns . . . 22
ppp_n . . . 22
ppp_next_n . . . 23
ppp_orderstat . . . 24
ppp_sequential . . . 24
read_code . . . 25
rng_stream_rexp . . . 26
rng_stream_rpois . . . 26
rng_stream_runif . . . 27
rng_stream_rztpois . . . 27
rztpois . . . 28
simpson_num_integr . . . 28
vdraw . . . 29
vdraw_intensity_step_regular . . . 30
vdraw_intensity_step_regular_cpp . . . 31
vdraw_intensity_step_regular_R . . . 32
vdraw_sc_step_regular . . . 34
vdraw_sc_step_regular_cpp . . . 35
vdraw_sc_step_regular_R . . . 36
vztdraw_intensity_step_regular . . . 37

vztdraw_intensity_step_regular_R 38
vztdraw_sc_step_regular 39
vztdraw_sc_step_regular_cpp 40
vztdraw_sc_step_regular_R 41
ztdraw_cumulative_intensity 42
ztdraw_intensity 43
ztdraw_intensity_step 44
ztdraw_sc_linear 45
ztdraw_sc_loglinear 46
ztppp 47

Index **48**

`check_ppp_sample_validity` *Check the validity of ppp samples*

Description

Standard checks for a vector of ordered times. Check that the `times` vector is sorted, has unique values, has all values in `[t_min, t_max]`, and has length `size` (if applicable).

Usage

```
check_ppp_sample_validity(
  times,
  t_min,
  t_max = NULL,
  size = NULL,
  atmost1 = FALSE,
  atleast1 = FALSE
)
```

Arguments

`times`	(vector, double | matrix) the times to be checked as vectors or matrices (time-vectors in rows)
`t_min`	(double) the start of the time nterval
`t_max`	(double) optional: the end of the time interval
`size`	(double) optional: the size of the vector
`atmost1`	(boolean) optional: at most one sample returned
`atleast1`	(boolean) optional: at least one sample returned

Value

None

`check_ppp_vector_validity` *Check the validity of a ppp vector.*

Description

Standard checks for a vector of ordered times. Check that the `times` vector is sorted, has unique values, has all values in `[t_min, t_max]`, and has length `size` (if applicable).

Usage

```
check_ppp_vector_validity(
  times,
  t_min,
  t_max = NULL,
  size = NULL,
  atmost1 = FALSE,
  atleast1 = FALSE
)
```

Arguments

`times`	(vector, double) the times to be checked
`t_min`	(double) the start of the time nterval
`t_max`	(double) optional: the end of the time interval
`size`	(double) optional: the size of the vector
`atmost1`	(boolean) optional: at most one sample returned
`atleast1`	(boolean) optional: at least one sample returned

Value

None

`compare_ppp_vectors` *Check that two ppp vectors Q-Q agree*

Description

Compare that the deciles of two vectors have absolute difference over average ratios less than `threshold`

Usage

```
compare_ppp_vectors(ppp1, ppp2, threshold = 0.15, showQQ = TRUE)
```

Arguments

ppp1	(vector, double) the first vector
ppp2	(vector, double) the second vector
threshold	(double) optional: the cutoff for a large absolute threshold
showQQ	(boolean) optional: show the QQ plot if the absolute value of the Difference vs Average ratio in any decile is bigger than the threshold

Value

None

---

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_maj = NULL,
  Lambda = NULL,
  Lambda_inv = NULL,
  range_t = c(0, 10),
  rng_stream = NULL,
  atmost1 = FALSE,
  atleast1 = FALSE
)
```

Arguments

lambda	(function) the instantaneous rate of the NHPPP. A continuous function of time.
lambda_maj	(double, vector) the intercept and optional slope of the majorizing linear (if exp_maj = FALSE) or log-linear (if exp_maj = TRUE) function in range_t.
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()
range_t	(vector, double) min and max of the time interval.
rng_stream	(rstream) an rstream object or NULL
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_inversion` *Simulate from a non homogeneous Poisson Point Process (NHPPP) from (t_min, t_max) (inversion method)*

Description

Sample NHPPP times using the inversion method, optionally using an `rstream` generator object

Usage

```
draw_cumulative_intensity_inversion(
  Lambda,
  Lambda_inv = NULL,
  range_t = c(0, 10),
  range_L = c(Lambda(range_t[1]), Lambda(range_t[2])),
  rng_stream = NULL,
  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()`
`range_t`	(vector, double) min and max of the time interval
`range_L`	(vector, double) min and max of the transformed time interval
`rng_stream`	(`rstream`) an `rstream` object.
`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_cumulative_intensity_inversion(Lambda = function(t) t + cos(t) - 1)
```

`draw_cumulative_intensity_orderstats` *Simulate from a non homogeneous Poisson Point Process (NHPPP) from (t_min, t_max) (order statistics method)*

Description

Sample NHPPP times using the order statistics method, optionally using an `rstream` generator object.

Usage

```
draw_cumulative_intensity_orderstats(
  Lambda,
  Lambda_inv = NULL,
  range_t = c(0, 10),
  range_L = c(Lambda(range_t[1]), Lambda(range_t[2])),
  rng_stream = NULL,
  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()`
`range_t`	(vector, double) min and max of the time interval
`range_L`	(vector, double) min and max of the transformed time interval
`rng_stream`	(`rstream`) an `rstream` object or `NULL`.
`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_cumulative_intensity_orderstats(Lambda = function(t) t + cos(t) - 1)
```

`draw_intensity` *Simulate from a non homogeneous Poisson Point Process (NHPPP) from (t0, t_max) (thinning method)*

Description

Sample NHPPP times using the thinning method, optionally using an `rstream` generator

Usage

```
draw_intensity(
  lambda,
  lambda_maj = NULL,
  exp_maj = FALSE,
  range_t = c(0, 10),
  rng_stream = NULL,
  atmost1 = FALSE
)
```

Arguments

`lambda`	(function) the instantaneous rate of the NHPPP. A continuous function of time.
`lambda_maj`	(double, vector) the intercept and optional slope of the majorizing linear (if `exp_maj = FALSE`) or log-linear (if `exp_maj = TRUE`) function in `range_t`.
`exp_maj`	(boolean) if `TRUE` the majorizer is `exp(alpha + beta * t)`
`range_t`	(vector, double) min and max of the time interval.
`rng_stream`	(`rstream`) an `rstream` object or `NULL`
`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_intensity(lambda = function(t) 1 + sin(t))
```

`draw_intensity_step` *Simulate from a non homogeneous Poisson Point Process (NHPPP) from (t0, t_max) (thinning method) with piecewise constant_majorizer*

Description

Sample NHPPP times using the thinning method, optionally using an `rstream` generator

Usage

```
draw_intensity_step(
  lambda,
  lambda_maj_vector = lambda(1:10),
  times_vector = 0:10,
  rng_stream = NULL,
  atmost1 = FALSE
)
```

Arguments

`lambda`	(function) the instantaneous rate of the NHPPP. A continuous function of time.
`lambda_maj_vector`	(scalar, double) `K` constant majorizing rates, one per interval
`times_vector`	(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
`rng_stream`	(`rstream`) an `rstream` object or `NULL`
`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_intensity_step(lambda = function(t) exp(.02 * t))
```

`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(
  alpha = 1,
  beta = 0,
  range_t = c(0, 10),
  rng_stream = NULL,
  atmost1 = FALSE
)
```

Arguments

`alpha`	(double) the intercept
`beta`	(double) the slope
`range_t`	(vector, double) min and max of the time interval
`rng_stream`	(`rstream`) an `rstream` object.
`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(alpha = 0, beta = 0.2)
```

`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(
  alpha = 1,
  beta = 0,
  range_t = c(0, 10),
  rng_stream = NULL,
  atmost1 = FALSE
)
```

Arguments

`alpha`	(double) the intercept in the exponent
`beta`	(double) the slope in the exponent
`range_t`	(vector, double) min and max of the time interval
`rng_stream`	(`rstream`) an `rstream` object.
`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(alpha = 0, beta = 0.2)
```

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 = rep(1, 5),
  times_vector = c(0:5),
  rng_stream = NULL,
  atmost1 = FALSE,
  atleast1 = FALSE
)
```

Arguments

lambda_vector	(scalar, double) K constant rates, one per interval
times_vector	(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
rng_stream	an rstream object
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), times_vector = 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,
  range_t = c(0, 10),
  rng_stream = 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
`range_t`	(vector, double) `t_min` and `t_max`
`rng_stream`	an `rstream` object
`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, range_t = c(0, 5))
```

expect_no_error *Helper functions*

Description

helper function that augments `test_that::expect_no_error()` to expect no error. Copied from the `R6` source code.

Usage

```
expect_no_error(expr)
```

Arguments

`expr`	Expression.

Details

Small utility functions. Not to be exported to the user.

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.

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)
```

`inverse_with_uniroot` *Numerically evaluate the inverse of a function at a specific point*

Description

Numerically evaluate the inverse of a function at a specific point

Usage

```
inverse_with_uniroot(
  f = f,
  y,
  min_x = 0,
  max_x = 1,
  min_y = f(min_x),
  max_y = f(max_x)
)
```

Arguments

`f`	(function) the function to be inverted; must be continuous and increasing
`y`	(scalar, double) the f(x)=y value in which to evaluate the inverse
`min_x`	(scalar, double) the min of the domain of f()
`max_x`	(scalar, double) the max of the domain of f()
`min_y`	(scalar, double) the min in the range of f()
`max_y`	(scalar, double) the max in the range of f()

Value

(scalar, double) vector of x=f^(-1)(y): the inverted value

Examples

```
inverse_with_uniroot(f = function(x) {
  2 * x
}, y = 0.5)
```

`inverse_with_uniroot_sorted` *Numerically evaluate the inverse of a monotonically increasing continuous function from R to R at specific points.*

Description

Numerically evaluate the inverse of a monotonically increasing continuous function from R to R at specific points.

Usage

```
inverse_with_uniroot_sorted(
  f,
  y,
  range_x = c(0, 10),
  range_y = c(f(range_x[1]), f(range_x[2]))
)
```

Arguments

`f`	(function) the function to be inverted; must be continuous and increasing
`y`	(vector, double) the f(x)=y values in which to evaluate the inverse; must be in ascending order
`range_x`	(vector, double) the min and max of the domain of f()
`range_y`	(vector, double) the min and max in the range of f()

Value

(vector, double) vector of x=f^(-1)(y): the inverted values

Examples

```
inverse_with_uniroot_sorted(f = function(x) {
  2 * x
}, y = c(0, 0.5))
```

Lambda_exp_form — *Definite integral of* `l = exp(alpha + beta*t)` *at time* `t` *with* `L(t0) = 0`

Description

Definite integral of `l = exp(alpha + beta*t)` starting at `t0` – only for `l+`.

Usage

```
Lambda_exp_form(t, alpha, beta, t0)
```

Arguments

`t`	(double) the time point
`alpha`	(double) the intercept
`beta`	(double) the slope
`t0`	(double) the starting time

Lambda_inv_exp_form — *Inverse of the definite integral of* `l = exp(alpha + beta*t)` *at time* `t`

Description

Inverse of the definite integral of `l = exp(alpha + beta*t)` only for `l+`.

Usage

```
Lambda_inv_exp_form(z, alpha, beta, t0)
```

Arguments

`z`	(double) the value of integrated rate for which you want to find the time
`alpha`	(double) the intercept
`beta`	(double) the slope
`t0`	(double) the starting time

`Lambda_inv_linear_form` *Inverse of the definite integral of* `l = alpha + beta*t` *at time* `t`

Description

Inverse of the definite integral of `l = alpha + beta*t` only for `l+`.

Usage

```
Lambda_inv_linear_form(z, alpha, beta, t0)
```

Arguments

`z`	(double) the value of integrated rate for which you want to find the time
`alpha`	(double) the intercept
`beta`	(double) the slope
`t0`	(double) the starting time

`Lambda_linear_form` *Definite integral of* `l = alpha + beta*t` *at time* `t` *with* `L(t0) = 0`

Description

Definite integral of `l = alpha + beta*t` starting at `t0` – only for `l+`.

Usage

```
Lambda_linear_form(t, alpha, beta, t0)
```

Arguments

`t`	(double) the time point
`alpha`	(double) the intercept
`beta`	(double) the slope
`t0`	(double) the starting time

`make_cumulative_Lambda_matrix`

Helper function for the vectorized versions of sampling functions. Takes the usual ways that `lambda_mat` *and* `Lambda_mat` *are specified and returns* `Lambda_mat`.

Description

Helper function for the vectorized versions of sampling functions. Takes the usual ways that `lambda_mat` and `Lambda_mat` are specified and returns `Lambda_mat`.

Usage

```
make_cumulative_Lambda_matrix(
  Lambda_mat = NULL,
  lambda_mat = NULL,
  interval_duration = NULL
)
```

Arguments

`Lambda_mat`	a matrix of cumulative intensities or missing
`lambda_mat`	a matrix of intensities or missing
`interval_duration`	a vector with the same number of elements as the rows of `Lambda_mat`

Value

A matrix (r x 2), row-expanded if needed

`make_lambda_matrix`

Helper function for the vectorized versions of sampling functions. Takes the usual ways that `lambda_mat` *and* `Lambda_mat` *are specified and returns* `lambda_mat`.

Description

Helper function for the vectorized versions of sampling functions. Takes the usual ways that `lambda_mat` and `Lambda_mat` are specified and returns `lambda_mat`.

Usage

```
make_lambda_matrix(
  lambda_mat = NULL,
  Lambda_mat = NULL,
  interval_duration = NULL
)
```

Arguments

lambda_mat	a matrix of intensities or missing
Lambda_mat	a matrix of cumulative intensities or missing
interval_duration	a vector with the same number of elements as the rows of Lambda_mat

Value

A matrix (r x 2), row-expanded if needed

---

make_range_t_matrix *Helper function for the vectorized versions of sampling functions. Takes the usual ways that* range_t *is specified (a 2-vector, a 1 x 2 or an r x 2 matrix) and returns a r x 2 matrix.*

Description

Helper function for the vectorized versions of sampling functions. Takes the usual ways that range_t is specified (a 2-vector, a 1 x 2 or an r x 2 matrix) and returns a r x 2 matrix.

Usage

```
make_range_t_matrix(range_t, n_rows)
```

Arguments

range_t	a 2-vector, a 1 x 2 or an r x 2 matrix
n_rows	the number of rows in the fully expanded matrix (r)

Value

A matrix (r x 2), row-expanded if needed

---

mat_cumsum_columns *Return matrix with column-wise cumulative sum No checks for arguments is done.*

Description

Return matrix with column-wise cumulative sum No checks for arguments is done.

Usage

```
mat_cumsum_columns(X)
```

Arguments

X	(matrix)

Value

matrix

`mat_cumsum_columns_with_scalar_ceiling`
Return matrix with column-wise cumulative sum replacing cells larger than `ceil` *with* `NA`*. No checks for arguments is done.*

Description

Return matrix with column-wise cumulative sum replacing cells larger than `ceil` with `NA`. No checks for arguments is done.

Usage

```
mat_cumsum_columns_with_scalar_ceiling(X, ceil = Inf)
```

Arguments

X	(matrix)
ceil	(double or Inf) the ceiling to be applied

Value

matrix

`mat_cumsum_columns_with_vector_ceiling`
Return matrix with column-wise cumulative sum replacing cells larger than `ceil` *with* `NA`*. No checks for arguments is done.*

Description

Return matrix with column-wise cumulative sum replacing cells larger than `ceil` with `NA`. No checks for arguments is done.

Usage

```
mat_cumsum_columns_with_vector_ceiling(X, ceil = Inf)
```

Arguments

X	(matrix)
ceil	(vector or Inf) the set of ceilings to be applied, per row of X

Value

matrix

---

mat_diff_columns *Return matrix with column-wise differencing. No checks for arguments is done.*

Description

Return matrix with column-wise differencing. No checks for arguments is done.

Usage

```
mat_diff_columns(X)
```

Arguments

X	(matrix)

Value

matrix

---

ppp_n *Simulate specific number of points from a homogeneous Poisson Point Process over (t_min, t_max]*

Description

Simulate specific number of points from a homogeneous Poisson Point Process over (t_min, t_max]

Usage

```
ppp_n(size, range_t = c(0, 10), rng_stream = NULL)
```

Arguments

size	(int) the number of points to be simulated
range_t	(vector, double) min and max of the time interval
rng_stream	an rstream object

Value

a vector of event times of size `size`

Examples

```
x <- ppp_n(size = 10, range_t = c(0, 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 = NULL)
```

Arguments

`n`	scalar number of samples
`rate`	scalar instantaneous rate
`t_min`	scalar for the starting time value
`rng_stream`	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)
```

ppp_orderstat *Simulate a homogeneous Poisson Point Process over (t_min, t_max] (order statistics method)*

Description

Simulate a homogeneous Poisson Point Process over (t_min, t_max] (order statistics method)

Usage

```
ppp_orderstat(range_t = c(0, 10), rate = 1, rng_stream = NULL, atmost1 = FALSE)
```

Arguments

range_t	(vector, double) min and max of the time interval
rate	(scalar, double) constant instantaneous rate
rng_stream	an rstream object
atmost1	boolean, draw at most 1 event time

Value

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

Examples

```
x <- ppp_orderstat(range_t = c(0, 10), rate = 1)
```

ppp_sequential *Simulate a homogeneous Poisson Point Process over (t_min, t_max]*

Description

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

Usage

```
ppp_sequential(
  range_t = c(0, 10),
  rate = 1,
  tol = 10^-6,
  rng_stream = NULL,
  atmost1 = FALSE
)
```

Arguments

range_t	(vector, double) min and max of the time interval
rate	(scalar, double) constant instantaneous rate
tol	the probability that we will have more than the drawn events in (t_min, t_max]
rng_stream	an rstream object
atmost1	boolean, draw at most 1 event time

Value

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

Examples

```
x <- ppp_sequential(range_t = c(0, 10), rate = 1, tol = 10^-6)
```

---

read_code *Read code from text file as string*

Description

Read code from text file as string

Usage

```
read_code(codeFile)
```

Arguments

codeFile	Path to file

Value

codeFile contents as a character string

rng_stream_rexp *Exponential random samples from* rstream *objects*

Description

Sample from rstream objects

Usage

```
rng_stream_rexp(size = 1, rate = 1, rng_stream = NULL)
```

Arguments

size	Integer, number of samples
rate	Positive number, the rate (i.e., 1/mean)
rng_stream	(rstream) an rstream object or NULL

Value

a vector of exponential variates of size size

Examples

```
rng_stream_rexp(10)
```

rng_stream_rpois *Poisson random samples from* rstream *objects*

Description

Sample from rstream objects

Usage

```
rng_stream_rpois(size = 1, lambda = 1, rng_stream = NULL)
```

Arguments

size	Integer, number of samples
lambda	Positive number, the mean
rng_stream	(rstream) an rstream object or NULL

Value

a vector of counts of size size

Examples

```
rng_stream_rpois(10)
```

`rng_stream_runif` *Uniform random samples from* `rstream` *objects*

Description

Sample from `rstream` objects

Usage

```
rng_stream_runif(size = 1, minimum = 0, maximum = 1, rng_stream = NULL)
```

Arguments

`size`	Integer, number of samples
`minimum`	Lower bound
`maximum`	Upper bound
`rng_stream`	(`rstream`) an `rstream` object or `NULL`

Value

a vector of uniform variates of size `size`

Examples

```
rng_stream_runif(10)
```

`rng_stream_rztpois` *Zero-truncated Poisson random samples from* `rstream` *objects*

Description

Sample from `rstream` objects

Usage

```
rng_stream_rztpois(size = 1, lambda = 1, rng_stream = NULL)
```

Arguments

`size`	Integer, number of samples
`lambda`	Positive number, the mean of the original (untruncated) Poisson distribution
`rng_stream`	(`rstream`) an `rstream` object or `NULL`

Value

a vector of non zero counts of size `size`

Examples

```
rng_stream_rztpois(10)
```

`rztpois` *Zero-truncated Poisson random samples (basic R)*

Description

Sample zero-truncated Poisson random samples (basic R)

Usage

```
rztpois(size = 1, lambda = 1)
```

Arguments

`size`	Integer, number of samples
`lambda`	Positive number, the mean of the original (untruncated) Poisson distribution

Value

a vector of non zero counts of size `size`

Examples

```
rztpois(10, 1)
rztpois(10, 1:10)
```

`simpson_num_integr` *Simpson's method to integrate a univariate function.*

Description

Simpson's method to integrate a univariate continuous function. Faster that R's `integrate()` and precise enough, but does not do any checks. The error is at most `M (b-a)^5/(180 n^4)` where `M` is the maximum of the fourth derivative of the integrand in the interval `[a, b]`.

Usage

```
simpson_num_integr(f, a, b, n)
```

Arguments

f	function that takes a single argument
a	the lower limit of integration
b	the upper limit of integration
n	integer, number of integration points with a and b

Value

numeric, the integration value examples #expect 1 simpson_num_integr(sin, 0, pi/2, 100) #max error for simpson_num_integr(sin, 0, pi/2, 100) is 5.312842e-10 1 * (pi/2 - 0)^5/(180 * 100^4)

---

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,
  range_t = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  atleast1 = FALSE,
  use_cpp = FALSE
)
```

Arguments

lambda	(function) a vectorized intensity function, with one or two arguments. The first is time. The optional second (should be named lambda_args) is a named list with additional arguments.
lambda_args	(list) optional list of named arguments for lambda()
Lambda_maj_matrix	(matrix) for the majorizeintegrated intensity rates at the end of each interval
lambda_maj_matrix	(matrix) intensity rates, one per interval
range_t	(vector, or matrix) t_min and t_max, possibly vectorized

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 in interval
use_cpp	boolean, use C++ code

Value

a vector of event times

Examples

```
Z <- vdraw(
  lambda = function(x, lambda_args = NULL) 0.1 * x,
  range_t = c(1, 10),
  lambda_maj_matrix = matrix(rep(1, 5), nrow = 1),
  atmost1 = FALSE, atleast1 = FALSE, use_cpp = TRUE
)
```

---

vdraw_intensity_step_regular

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_step_regular(
  lambda = NULL,
  lambda_args = NULL,
  Lambda_maj_matrix = NULL,
  lambda_maj_matrix = NULL,
  range_t = NULL,
  subinterval = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  atmostB = NULL
)
```

Arguments

lambda	(function) a vectorized intensity function, with one or two arguments. The first is time. The optional second is a named list with additional arguments.
lambda_args	(list) optional list of named arguments for lambda()
Lambda_maj_matrix	(matrix) for the majorizeintegrated intensity rates at the end of each interval
lambda_maj_matrix	(matrix) intensity rates, one per interval
range_t	(vector, or matrix) t_min and t_max, possibly vectorized
subinterval	(matrix, double) subinterval of range_t to sample from
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.

Value

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

Examples

```
Z <- vdraw_intensity_step_regular(
  lambda = function(x, lambda_args = NULL) 0.1 * x,
  range_t = c(1, 10),
  lambda_maj_matrix = matrix(rep(1, 5), nrow = 1)
)
```

---

vdraw_intensity_step_regular_cpp

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_step_regular_cpp(
  lambda = NULL,
  lambda_args = NULL,
  Lambda_maj_matrix = NULL,
  lambda_maj_matrix = NULL,
```

```
  range_t = NULL,
  subinterval = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  atmostB = NULL
)
```

Arguments

lambda	(function) a vectorized intensity function, with one or two arguments. The first is time. The optional second is a named list with additional arguments.
lambda_args	(list) optional list of named arguments for lambda()
Lambda_maj_matrix	(matrix) for the majorizeintegrated intensity rates at the end of each interval
lambda_maj_matrix	(matrix) intensity rates, one per interval
range_t	(vector, or matrix) t_min and t_max, possibly vectorized
subinterval	(matrix, double) subinterval of range_t to sample from
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.

Value

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

Examples

```
Z <- vdraw_intensity_step_regular_cpp(
  lambda = function(x, lambda_args = NULL) 0.1 * x,
  range_t = c(1, 10),
  lambda_maj_matrix = matrix(rep(1, 5), nrow = 1)
)
```

---

vdraw_intensity_step_regular_R

Vectorized sampling from a non homogeneous Poisson Point Process (NHPPP) from an interval (thinning method) with piecewise constant_majorizers (R)

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_step_regular_R(
  lambda = NULL,
  lambda_args = NULL,
  Lambda_maj_matrix = NULL,
  lambda_maj_matrix = NULL,
  range_t = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  force_zt_majorizer = FALSE,
  ...
)
```

Arguments

lambda	(function) a vectorized intensity function, with one or two arguments. The first is time. The optional second is a named list with additional arguments.
lambda_args	(list) optional list of named arguments for lambda()
Lambda_maj_matrix	(matrix) for the majorizeintegrated intensity rates at the end of each interval
lambda_maj_matrix	(matrix) intensity rates, one per interval
range_t	(vector, or matrix) t_min and t_max, possibly vectorized
tol	(scalar, double) tolerance for the number of events
atmost1	boolean, draw at most 1 event time
force_zt_majorizer	boolean, force the use of the zero-truncated majorizer; - used for flexibility in calling from other functions. Keep the default FALSE unless you know what you are doing.
...	(any) other arguments (ignored – used for flexibility in calling from other functions)

Value

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

Examples

```
Z <- vdraw_intensity_step_regular_R(
  lambda = function(x, lambda_args = NULL) 0.1 * x,
  range_t = c(1, 10),
  lambda_maj_matrix = matrix(rep(1, 5), nrow = 1)
)
```

vdraw_sc_step_regular *Vectorized sampling from NHPPPs with piecewise constant intensities with same interval lengths (R)*

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,
  range_t = c(0, 10),
  tol = 10^-6,
  atmost1 = FALSE,
  atleast1 = FALSE,
  use_cpp = TRUE
)
```

Arguments

Lambda_matrix	(matrix) integrated intensity rates at the end of each interval
lambda_matrix	(matrix) intensity rates, one per interval
range_t	(vector, double) t_min and 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 (zero-truncated NHPPP)
use_cpp	(boolean, TRUE) use the C++ implementation of the function

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))
```

vdraw_sc_step_regular_cpp *Vectorized sampling from NHPPPs with piecewise constant intensities with same interval lengths (C++)*

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_cpp(
  lambda_matrix = NULL,
  Lambda_matrix = NULL,
  range_t = NULL,
  subinterval = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  atmostB = NULL
)
```

Arguments

`lambda_matrix`	(matrix) intensity rates, one per interval
`Lambda_matrix`	(matrix) integrated intensity rates at the end of each interval
`range_t`	(vector, double) `t_min` and `t_max`
`subinterval`	(matrix, double) subinterval of `range_t` to sample from
`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.

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))
```

vdraw_sc_step_regular_R *Vectorized sampling from NHPPPs with piecewise constant intensities with same interval lengths (R)*

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_R(
  Lambda_matrix = NULL,
  lambda_matrix = NULL,
  range_t = c(0, 10),
  tol = 10^-6,
  atmost1 = FALSE
)
```

Arguments

`Lambda_matrix`	(matrix) integrated intensity rates at the end of each interval
`lambda_matrix`	(matrix) intensity rates, one per interval
`range_t`	(vector, double) `t_min` and `t_max`
`tol`	(scalar, double) tolerance for the number of events
`atmost1`	boolean, draw at most 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_R(Lambda_matrix = matrix(1:5, nrow = 1))
```

vztdraw_intensity_step_regular *Vectorized sampling from a zero-truncated non homogeneous Poisson Point Process (NHPPP) from an interval (thinning method) with piecewise constant_majorizers*

Description

Vectorized sampling from a zero-truncated 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

```
vztdraw_intensity_step_regular(
  lambda = NULL,
  lambda_args = NULL,
  Lambda_maj_matrix = NULL,
  lambda_maj_matrix = NULL,
  range_t = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  ...
)
```

Arguments

`lambda`	(function) a vectorized intensity function, with one or two arguments. The first is time. The optional second is a named list with additional arguments.
`lambda_args`	(list) optional list of named arguments for `lambda()`
`Lambda_maj_matrix`	(matrix) for the majorizeintegrated intensity rates at the end of each interval
`lambda_maj_matrix`	(matrix) intensity rates, one per interval
`range_t`	(vector, or matrix) `t_min` and `t_max`, possibly vectorized
`tol`	(scalar, double) tolerance for the number of events
`atmost1`	boolean, draw at most 1 event time
`...`	(any) other arguments (ignored – used for flexibility in calling from other functions)

Value

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

Examples

```
Z <- vztdraw_intensity_step_regular(
  lambda = function(x, lambda_args = NULL) 0.1 * x,
  range_t = c(1, 10),
  lambda_maj_matrix = matrix(rep(1, 5), nrow = 1)
)
```

---

vztdraw_intensity_step_regular_R

Vectorized sampling from a zero-truncated non homogeneous Poisson Point Process (NHPPP) from an interval (thinning method) with piecewise constant_majorizers (R)

Description

Vectorized sampling from a zero-truncated 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

```
vztdraw_intensity_step_regular_R(
  lambda = NULL,
  lambda_args = NULL,
  Lambda_maj_matrix = NULL,
  lambda_maj_matrix = NULL,
  range_t = NULL,
  tol = 10^-6,
  atmost1 = FALSE,
  ...
)
```

Arguments

`lambda`	(function) a vectorized intensity function, with one or two arguments. The first is time. The optional second is a named list with additional arguments.
`lambda_args`	(list) optional list of named arguments for `lambda()`
`Lambda_maj_matrix`	(matrix) for the majorizeintegrated intensity rates at the end of each interval
`lambda_maj_matrix`	(matrix) intensity rates, one per interval
`range_t`	(vector, or matrix) `t_min` and `t_max`, possibly vectorized
`tol`	(scalar, double) tolerance for the number of events
`atmost1`	boolean, draw at most 1 event time
`...`	(any) other arguments (ignored – used for flexibility in calling from other functions)

Value

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

Examples

```
Z <- vztdraw_intensity_step_regular_R(
  lambda = function(x, lambda_args = NULL) 0.1 * x,
  range_t = c(1, 10),
  lambda_maj_matrix = matrix(rep(1, 5), nrow = 1)
)
```

---

vztdraw_sc_step_regular

Vectorized sampling from zero-truncated 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

```
vztdraw_sc_step_regular(
  Lambda_matrix = NULL,
  lambda_matrix = NULL,
  range_t = c(0, 10),
  subinterval = NULL,
  atmost1 = FALSE,
  ...
)
```

Arguments

`Lambda_matrix`	(matrix) integrated intensity rates at the end of each interval
`lambda_matrix`	(matrix) intensity rates, one per interval
`range_t`	(vector, double) `t_min` and `t_max`
`subinterval`	(vector, double) optional – the subinterval of `range_t` to sample. If `NULL`, the whole range_t is used.
`atmost1`	boolean, draw at most 1 event time
`...`	(any) other arguments (ignored – used for flexibility in calling from other functions)

Value

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

Examples

```
x <- vztdraw_sc_step_regular(Lambda_matrix = matrix(1:5, nrow = 1))
```

`vztdraw_sc_step_regular_cpp` *Vectorized sampling from zero-truncated NHPPPs with piecewise constant intensities with same interval lengths (C++)*

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

```
vztdraw_sc_step_regular_cpp(
  Lambda_matrix = NULL,
  lambda_matrix = NULL,
  range_t = c(0, 10),
  subinterval = NULL,
  atmost1 = FALSE
)
```

Arguments

`Lambda_matrix`	(matrix) integrated intensity rates at the end of each interval
`lambda_matrix`	(matrix) intensity rates, one per interval
`range_t`	(vector, double) `t_min` and `t_max`
`subinterval`	(vector, double) optional – the subinterval of `range_t` to sample. If `NULL`, the whole range_t is used.
`atmost1`	boolean, draw at most 1 event time

Value

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

Examples

```
x <- vztdraw_sc_step_regular_cpp(Lambda_matrix = matrix(1:5, nrow = 1))
```

vztdraw_sc_step_regular_R *Vectorized sampling from zero-truncated NHPPPs with piecewise constant intensities with same interval lengths (R)*

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

```
vztdraw_sc_step_regular_R(
  Lambda_matrix = NULL,
  lambda_matrix = NULL,
  range_t = c(0, 10),
  atmost1 = FALSE,
  ...
)
```

Arguments

`Lambda_matrix`	(matrix) integrated intensity rates at the end of each interval
`lambda_matrix`	(matrix) intensity rates, one per interval
`range_t`	(vector, double) `t_min` and `t_max`
`atmost1`	boolean, draw at most 1 event time
`...`	(any) other arguments (ignored – used for flexibility in calling from other functions)

Value

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

Examples

```
x <- vztdraw_sc_step_regular_R(Lambda_matrix = matrix(1:5, nrow = 1))
```

`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 = NULL,
  range_t = c(0, 10),
  range_L = c(Lambda(range_t[1]), Lambda(range_t[2])),
  rng_stream = NULL,
  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()`
`range_t`	(vector, double) min and max of the time interval
`range_L`	(vector, double) min and max of the transformed time interval
`rng_stream`	(`rstream`) an `rstream` object or `NULL`.
`atmost1`	(boolean) draw at most 1 event time

Value

a vector of at least 1 event times

Examples

```
x <- ztdraw_cumulative_intensity(Lambda = function(t) t + cos(t) - 1)
```

`ztdraw_intensity` *Simulate* `size` *samples from a zero-truncated non homogeneous Poisson Point Process (zt-NHPPP) from (t0, t_max) (thinning method)*

Description

Sample zero-truncated NHPPP intensity times using the thinning method, optionally using an `rstream` generator

Usage

```
ztdraw_intensity(
  lambda,
  lambda_maj = NULL,
  exp_maj = FALSE,
  range_t = c(0, 10),
  rng_stream = NULL,
  atmost1 = FALSE
)
```

Arguments

`lambda`	(function) the instantaneous rate of the NHPPP. A continuous function of time.
`lambda_maj`	(double, vector) the intercept and optional slope of the majorizing linear (if `exp_maj = FALSE`) or log-linear (if `exp_maj = TRUE`) function in `range_t`.
`exp_maj`	(boolean) if `TRUE` the majorizer is `exp(alpha + beta * t)`
`range_t`	(vector, double) min and max of the time interval.
`rng_stream`	(`rstream`) an `rstream` object or `NULL`
`atmost1`	(boolean) draw at most 1 event time

Value

a vector of at least 1 event times

Examples

```
x <- ztdraw_intensity(lambda = function(t) 1 + sin(t))
```

`ztdraw_intensity_step` *Simulate from a zero-truncated non homogeneous Poisson Point Process (NHPPP) from (t0, t_max) (thinning method) with piecewise constant_majorizer*

Description

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

Usage

```
ztdraw_intensity_step(
  lambda,
  lambda_maj_vector = lambda(1:10),
  times_vector = 0:10,
  rng_stream = NULL,
  atmost1 = FALSE
)
```

Arguments

`lambda`	(function) the instantaneous rate of the NHPPP. A continuous function of time.
`lambda_maj_vector`	(scalar, double) `K` constant majorizing rates, one per interval
`times_vector`	(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
`rng_stream`	(`rstream`) an `rstream` object or `NULL`
`atmost1`	boolean, draw at most 1 event time

Value

a vector of event times (t_) with at least one element

Examples

```
x <- ztdraw_intensity_step(lambda = function(t) exp(.02 * t))
```

`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(
  alpha = 1,
  beta = 0,
  range_t = c(0, 10),
  rng_stream = NULL,
  atmost1 = FALSE
)
```

Arguments

`alpha`	(double) the intercept
`beta`	(double) the slope
`range_t`	(vector, double) min and max of the time interval
`rng_stream`	(`rstream`) an `rstream` object
`atmost1`	(boolean) draw 1 event time

Value

a vector of at least 1 event times

Examples

```
x <- ztdraw_sc_linear(alpha = 0, beta = 0.2)
```

`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 (inversion method)*

Description

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

Usage

```
ztdraw_sc_loglinear(
  alpha = 1,
  beta = 0,
  range_t = c(0, 10),
  rng_stream = NULL,
  atmost1 = FALSE
)
```

Arguments

`alpha`	(double) the intercept in the exponent
`beta`	(double) the slope in the exponent
`range_t`	(vector, double) min and max of the time interval
`rng_stream`	(`rstream`) an `rstream` object.
`atmost1`	boolean, 1 event time

Value

a vector of at least 1 event times

Examples

```
x <- ztdraw_sc_loglinear(alpha = 0, beta = 0.2)
```

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(range_t = c(0, 10), rate = 1, rng_stream = NULL, atmost1 = FALSE)
```

Arguments

range_t	(vector, double) min and max of the time interval
rate	(scalar, double) constant instantaneous rate
rng_stream	an rstream object
atmost1	boolean, draw at most 1 event time

Value

a vector of event times of size size

Examples

```
x <- ztppp(range_t = c(0, 10), rate = 0.001)
```

Index

check_ppp_sample_validity, 3
check_ppp_vector_validity, 4
compare_ppp_vectors, 4

draw, 5
draw_cumulative_intensity_inversion, 6
draw_cumulative_intensity_orderstats, 7
draw_intensity, 8
draw_intensity_step, 9
draw_sc_linear, 10
draw_sc_loglinear, 11
draw_sc_step, 12
draw_sc_step_regular, 13

expect_no_error, 14

get_step_majorizer, 14

inverse_with_uniroot, 15
inverse_with_uniroot_sorted, 16

Lambda_exp_form, 17
Lambda_inv_exp_form, 17
Lambda_inv_linear_form, 18
Lambda_linear_form, 18

make_cumulative_Lambda_matrix, 19
make_lambda_matrix, 19
make_range_t_matrix, 20
mat_cumsum_columns, 20
mat_cumsum_columns_with_scalar_ceiling, 21
mat_cumsum_columns_with_vector_ceiling, 21
mat_diff_columns, 22

ppp_n, 22
ppp_next_n, 23
ppp_orderstat, 24
ppp_sequential, 24

read_code, 25
rng_stream_rexp, 26
rng_stream_rpois, 26
rng_stream_runif, 27
rng_stream_rztpois, 27
rztpois, 28

simpson_num_integr, 28

vdraw, 29
vdraw_intensity_step_regular, 30
vdraw_intensity_step_regular_cpp, 31
vdraw_intensity_step_regular_R, 32
vdraw_sc_step_regular, 34
vdraw_sc_step_regular_cpp, 35
vdraw_sc_step_regular_R, 36
vztdraw_intensity_step_regular, 37
vztdraw_intensity_step_regular_R, 38
vztdraw_sc_step_regular, 39
vztdraw_sc_step_regular_cpp, 40
vztdraw_sc_step_regular_R, 41

ztdraw_cumulative_intensity, 42
ztdraw_intensity, 43
ztdraw_intensity_step, 44
ztdraw_sc_linear, 45
ztdraw_sc_loglinear, 46
ztppp, 47