Package ‘rpgm’

March 17, 2017

Type Package

Title Fast Simulation of Normal/Exponential Random Variables and Stochastic Differential Equations

Version 1.0.0

Date 2017-03-16

Author Nicolas Baradel

Maintainer Nicolas Baradel - PGM Solutions <nicolas.baradel@pgm-solutions.com>

Description Fast simulation of some random variables than the usual native functions, including rnorm() and rexp(), using Ziggurat method, reference: MARSAGLIA, George, TSANG, Wai Wan, and al. (2000) <doi:10.18637/jss.v005.i08>, and fast simulation of stochastic differential equations.

License GPL-3

URL https://pgm-solutions.com/packages

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-03-16 23:17:10 UTC

R topics documented:

`rpgm-package` *rpgm*

Description

Fast Simulation of Normal Random Variables

Details

Ziggurat method in order to simulate normal random variables approximately four times faster than the usual rnorm(), reference : MARSAGLIA, George, TSANG, Wai Wan, et al. The ziggurat method for generating random variables. Journal of statistical software, 2000, vol. 5, no 8, p. 1-7.

Author(s)

Nicolas Baradel

Maintainer: Nicolas Baradel - PGM Solutions

References

http://pgm-solutions.com/packages

Examples

```
rpgm.rnorm(5)
```

`bound` *Set a Minimum or a Maximum or Both to a Vector.*

Description

The function `lbound` sets a minimum to the elements of a vector, ubound a maximum and bound both.

Usage

```
lbound(x, m)
ubound(x, M)
bound(x, m, M)
```

Arguments

`x`	double, vector to put in [m, M].
`m`	double, the minimum.
`M`	double, the maximum.

Value

A vector with the values of `x` on which all values lower that `m` has been replaced by `m` and all values greater than `M` has been replace by `M`.

Note

`x <- lbound(x, a)` replaces `x <- x*(x >= a) + a*(x < a)` and `x[x < a] <- a` in a much faster way.

Author(s)

Nicolas Baradel - PGM Solutions

See Also

http://pgm-solutions.com/packages

Examples

```
K <- 1
x <- rpgm.rnorm(12, 0.5)
lbound(x-K, 0)
```

`colMaxs - rowMaxs - colMins - rowMins`

The Maximum or Minimum of each Column or each Row of a Matrix.

Description

Form row and column maxs and mins for numeric arrays (or data frames).

Usage

```
colMaxs(x)
```

Arguments

`x`	numeric or integer, matrix.

Details

These functions are equivalent, per example for `colMaxs(X)`, to `apply(X, 1, max)`, but are a lot faster.

Value

A numeric vector.

Author(s)

Nicolas Baradel - PGM Solutions

See Also

http://pgm-solutions.com/packages

Examples

```
X <- matrix(rpgm.rnorm(36), 6, 6)
colMaxs(X)
rowMaxs(X)
colMins(X)
rowMins(X)
```

`jarquebera` *The Jarque-Bera Test for Normality*

Description

This test, based on the `skewness` and the `kurtosis` of the vector, computes the p-value associated to the normality of the distribution.

Usage

```
jarquebera(x)
```

Arguments

`x`	numeric, vector of independent and identical random variables

Details

The Jarque-Bera test is based of the convergence, if the vector is i.i.d. normal random variables, of

$$\texttt{skewness}(x) \to \mathcal{N}(0, 6); \texttt{kurtosis}(x) \to \mathcal{N}(3, 24)$$

and moreover, both are asymptotically independent. Then, we have the statistic

$$J = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right) \to \chi^2(2)$$

Value

The p-value associated to the test : `1-pchisq(J, 2)`.

Note

If you choose a test of level alpha, then you reject the null hypothesis of a normal distribution if the p-value returned by the function is lower than alpha.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Jarque

See Also

http://pgm-solutions.com/packages

Examples

```
jarquebera(rpgm.rnorm(10^5))
```

`kurtosis` *The Kurtosis of a Vector of Random Variables*

Description

The function computes the centred and reduced moment of order 4 of the vector `x`.

Usage

```
kurtosis(x)
```

Arguments

`x`	numeric, vector of independent and identical random variables

Details

The function returns the value of

$$\frac{1}{n}\sum_{k=1}^{n}[(x_i - \mu_x)/\sigma_x]^4$$

Value

A vector of i.i.d. normal random variable.

Note

For the skewness, see the `skewness` function.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Kurtosis

See Also

http://pgm-solutions.com/packages

Examples

```
kurtosis(rpgm.rnorm(10^5))
```

---

rbernou — *Fast Simulation of Bernoulli Random Variables*

Description

The function rbernou generates Bernoulli Random Variables faster than using `rbinom(n, 1, prob)` or `runif(n) <= prob` by using a C-level integer comparison.

Usage

```
rbernou(n, prob=0.5)
```

Arguments

`n`	integer, number of simulations.
`prob`	double, probability.

Details

The case `prob = 0.5` is twice time faster than the general case `0 <= proba <= 1`, using a specific C-level binary algorithm.

Value

A vector of i.i.d. Bernoulli random variables.

Note

For a big number of simulations, it is in general thirteenth times faster than the usual `rbinom(n, 1, prob)`.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Bernoulli_distribution

See Also

http://pgm-solutions.com/packages

Examples

```
rbernou(5)
```

`rbrownian` *Simulation of Brownian Motions*

Description

The function `rbrownian` is a C-level function which uses the Ziggurat in order to simulate the normal random variables.

Usage

```
rbrownian(n, m, b0=0, mu=0, sd=1, T=1, drop = TRUE)
```

Arguments

`n`	integer, number of paths.
`m`	integer, number of steps, the step size will be T/m.
`b0`	double, the initial value (or a vector of initial values of size `n`).
`mu`	double, the mean.
`sd`	double, the standard deviation.
`T`	double, the final date on which the brownian motion is simulated.
`drop`	logical, if `n = 1` and `drop = TRUE` then the function returns the single path of the brownian motion as a vector instead of a matrix.

Value

Returns a n x m+1 matrix of n path of the brownian motion.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Brownian_motion

See Also

https://pgm-solutions.com/packages

Examples

```
rbrownian(5, 10)
```

`rcantor` *Fast Simulation of Cantor Random Variables*

Description

The function generates uniformly random variable on the Cantor set. The distribution provided is singular (neither dsicrite nor absolutely continuous nor a mixture).

Usage

```
rcantor(n)
```

Arguments

`n`	integer, number of simulations.

Details

The Cantor set is uncountable with Lebesgue's measure 0 which leads to a singular probability distribution. The corresponding cumulative probability distribution is the Devil's staircase. The cantor set can be viewed as the number of the form sum(j=1, +Inf) c_j / 3^j with c_j in {0, 2} and the corresping probability distribution simulates uniformely the c_j (here, to j=32).

Value

A vector of i.i.d. Bernoulli random variables.

Note

This distribution is provided only for theorical use, not practical.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Cantor_distribution

See Also

http://pgm-solutions.com/packages

Examples

```
rcantor(5)
```

`reuler` *Euler Scheme for Stochastic Differential Equations*

Description

If the process X_t is the unique strong solution of the process

dX_t = b(X_t)dt + s(X_t)dW_t,

then the Euler Scheme is X[t+h] = X[t] + b(X[t])h + s(X[t])sqrt(h)Z, where Z ~ N(0,1).

Usage

```
reuler(n, m, x0, b, s, t0 = 0, T = 1, all_dates = TRUE, delta = NULL)
```

Arguments

`n`	integer, number of paths.
`m`	integer, number of steps, the step size will be T/m.
`x0`	numeric, starting point of the process.
`b`	function, the drift, a function which can take a vector and returns a vector.
`s`	function, the volatility, a function which can take a vector and returns a vector.
`t0`	double, the starting date of the process.
`T`	double, the final date of the process.
`all_dates`	logical, if TRUE, returns all steps from all paths. If FALSE, only returns the n final value X_T.
`delta`	double, the step size.

Value

If `all_dates = TRUE`, it returns a n x m+1 matrix : n paths with m steps (+ the first value). Else, it returns a vector of length n with the simulations of the final dates X_T.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Euler

See Also

https://pgm-solutions.com/packages

Examples

```
mu <- 0.07
sigma <- 0.20
reuler(5, 10, 1, function(x) return(mu*x), function(x) return(sigma*x))
```

---

rgpd — *Fast Simulation of Generalized Pareto Distribution*

Description

The function rgpd generates Generalized Pareto Random Variables.

Usage

```
rgpd(n, xi, mu = 0, sigma = 1)
```

Arguments

`n`	integer, number of simulations.
`xi`	double, the shape.
`mu`	double, the location.
`sigma`	double, the scale.

Value

A vector of i.i.d. Generalized Pareto random variables.

Note

For xi != 0, the cumulative distribution function is:

F(x) = 1-(1+xi*(x-mu)/sigma)^(-1/xi)

for x >= mu when xi > 0 and mu <= x <= mu - sigma/xi when xi < 0.

If xi = 0, (X - mu)/sigma follows a Exponential distribution of parameter 1.

Author(s)

Nicolas Baradel - PGM Solutions

See Also

http://pgm-solutions.com/packages

Examples

```
x <- rgpd(5, 1)
```

`rmilstein` *Milstein Scheme for Stochastic Differential Equations*

Description

If the process X_t is the unique strong solution of the process

dX_t = b(X_t)dt + s(X_t)dW_t,

then the Milstein Scheme is X[t+h] = X[t] + b(X[t])h + s(X[t])Z + 0.5*s'(X[t])*(Z^2 - h), where Z ~ N(0,h) (variance h), and s' is the differential function of s.

Usage

```
rmilstein(n, m, x0, b, s, sx, t0 = 0, T = 1, all_dates = TRUE, delta = NULL)
```

Arguments

`n`	integer, number of paths.
`m`	integer, number of steps, the step size will be T/m.
`x0`	numeric, starting point of the process.
`b`	function, the drift, a function which can take a vector and returns a vector.
`s`	function, the volatility, a function which can take a vector and returns a vector.
`sx`	function, the differential of the volatility, a function which can take a vector and returns a vector.
`t0`	double, the starting date of the process.
`T`	double, the final date of the process.
`all_dates`	logical, if TRUE, returns all steps from all paths. If FALSE, only returns the n final value X_T.
`delta`	double, the step size.

Value

If `all_dates = TRUE`, it returns a n x m+1 matrix : n paths with m steps (+ the first value). Else, it returns a vector of length n with the simulations of the final dates X_T.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Milstein_method

See Also

https://pgm-solutions.com/packages

Examples

```
mu <- 0.07
sigma <- 0.20
rmilstein(5, 10, 1, function(x) return(mu*x),
function(x) return(sigma*x), function(x) return(sigma))
```

---

rpgm.rexp — *Fast Simulation of Exponential Random Variables*

Description

The function rpgm.rexp uses the Ziggurat algorithm with a 256-regions table, in order to simulate exponential random variables faster than `rexp`.

Usage

```
rpgm.rexp(n, lambda = 1)
```

Arguments

`n`	integer, number of simulations.
`lambda`	double, the parameter lambda.

Details

The density is lambda * exp(-lambda*x) for x > 0.

Value

A vector of i.i.d. exponential random variables.

Note

For a big number of simulations, it is in general three times faster than the usual `rexp`. For one simulution, it is around one half faster.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Ziggurat_algorithm

See Also

http://pgm-solutions.com/packages

Examples

```
rpgm.rnorm(5)
```

`rpgm.rgeom` *Fast Simulation of Geometric Random Variables*

Description

The function rpgm.geom uses `rpgm.exp` in order to simulate geometric random variables faster than `rgeom`.

Usage

```
rpgm.rgeom(n, prob)
```

Arguments

`n`	integer, number of simulations.
`prob`	double, probability.

Details

The argument `prob` must be in]0, 1], else, NA are produced.

Value

A vector of i.i.d. geometric random variables.

Note

For a big number of simulations, it is in general nine times faster than the usual `rgeom`.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Ziggurat_algorithm

See Also

http://pgm-solutions.com/packages

Examples

```
rpgm.rgeom(5, 0.5)
```

`rpgm.rlnorm` *Fast Simulation of Log-Normal Random Variables*

Description

The function rpgm.rlnorm uses `rpgm.rnorm` in order to simulate log-normal random variables faster than `rlnorm`.

Usage

```
rpgm.rlnorm(n, mean = 0, sd = 1)
```

Arguments

`n`	integer, number of simulations.
`mean`	double, the mean (or the vector of means) of the logarithm of the log-normal variable.
`sd`	double, the standard deviation (or the vector of standard deviations) of the logarithm of the log-normal variable.

Details

If `mean` or `sd` are not specified they assume the default values of 0 and 1, respectively.

Value

A vector of i.i.d. log-normal random variables.

Note

For a big number of simulations, it is in general two times faster than the usual `rnorm`.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Ziggurat_algorithm

See Also

http://pgm-solutions.com/packages

Examples

```
rpgm.rlnorm(5)
```

`rpgm.rnorm` *Fast Simulation of Normal Random Variables*

Description

The function rpgm.rnorm uses the Ziggurat algorithm with a 128-regions table, in order to simulate normal random variables faster than `rnorm`.

Usage

```
rpgm.rnorm(n, mean = 0, sd = 1)
```

Arguments

`n`	integer, number of simulations.
`mean`	double, the mean (or the vector of means).
`sd`	double, the standard deviation (or the vector of standard deviations).

Details

If `mean` or `sd` are not specified they assume the default values of 0 and 1, respectively.

Value

A vector of i.i.d. normal random variables.

Note

For a big number of simulations, it is in general between three and four times faster than the usual `rnorm`. For one simulution, it is around one half faster.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Ziggurat_algorithm

See Also

http://pgm-solutions.com/packages

Examples

```
rpgm.rnorm(5)
```

`rpgm.rt` *Fast Simulation of Student Random Variables*

Description

The function rpgm.rt uses `rpgm.rnorm` in order to simulate student random variables faster than `rt`.

Usage

```
rpgm.rt(n, df)
```

Arguments

`n`	integer, number of simulations.
`df`	double, degrees of freedom (> 0, maybe non-integer).

Details

If `df = 1`, the distribition if the Cauchy one. The mean exists when `df > 1` and the variance when `df > 2`.

Value

A vector of i.i.d. student random variables.

Note

For a big number of simulations, it is in general two times faster than the usual `rt`.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Ziggurat_algorithm

See Also

http://pgm-solutions.com/packages

Examples

```
rpgm.rt(5, 4)
```

---

skewness *The Skewness of a Vector of Random Variables*

Description

The function computes the centred and reduced moment of order 3 of the vector `x`.

Usage

```
skewness(x)
```

Arguments

`x`	numeric, vector of independent and identical random variables

Details

The function returns the value of

$$\frac{1}{n}\sum_{k=1}^{n}[(x_i - \mu_x)/\sigma_x]^3$$

Value

A vector of i.i.d. normal random variable.

Note

For the kurtosis, see the `kurtosis` function.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Skewness

See Also

http://pgm-solutions.com/packages

Examples

```
skewness(rpgm.rnorm(10^5))
```

`vasicek` *Simulation and Density of Vasicek Process*

Description

The definition of the process used here is:

dX_t = -a(X_t - mu) + sd*dW_t,

where (mu, a, sd) are the three real parameters.

Usage

```
rvasicek(n, m, x0 = 0, mu = 0, a = 1, sd = 1, T = 1, drop = TRUE)
dvasicek(x, mu=0, a=1, sd=1, T=1, log = FALSE)
lvasicek(x, mu=0, a=1, sd=1, T=1)
evasicek(x, a0=1, T=1)
```

Arguments

`n`	integer, number of paths.
`m`	integer, number of steps, the step size will be T/m.
`x`	double, the vector of the observed values of a Vasicek process.
`x0`	double, the initial value.
`mu`	double, the value on which the process is centered and has an attraction when it is away.
`a`	double, the coefficient of how strong is the mean reversion when the process is away from `mu`.
`sd`	double, the volatility.
`T`	double, the final date on which the brownian motion is simulated.
`drop`	logical, if `n = 1` and `drop = TRUE` then the function returns the single path of the brownian motion as a vector instead of a matrix.
`log`	logical, if TRUE, returns the log-density, if FALSE, returns the density.
`a0`	double, starting value of `a` in the estimation algorithm.

Value

`rvasicek` returns a `(n, m+1)` matrix of n path of the Vasicek process. `dvasicek` returns a vector of size `length(x)-1`. Note that the first value has no density. `lvasicek` returns the log-liklihood associated to `dvasicek` and `evasicek` returns the Maximum Likelihood Estimator of the parameters `(mu, a, sd)`.

Note

If `mu = 0`, the process coincides with the Ornstein-Uhlenbeck process.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Vasicek_model

See Also

https://pgm-solutions.com/packages

Examples

```
x <- rvasicek(5, 10)
dvasicek(x[1L, ])
```

Index