

Package ‘rpgm’

July 5, 2017

Type Package

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

Version 1.0.1

Date 2017-07-05

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 / Poisson processes.

License GPL-3

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-07-05 06:19:48 UTC

R topics documented:

<code>rpgm-package</code>	2
<code>bound</code>	3
<code>colMaxs - rowMaxs - colMins - rowMins</code>	4
<code>evalpoisson</code>	5
<code>jarquebera</code>	6
<code>kurtosis</code>	7
<code>rbernou</code>	8
<code>rbrownian</code>	9
<code>rcantor</code>	10
<code>reuler</code>	11
<code>rgpd</code>	12
<code>rinpoisson</code>	13
<code>rmilstein</code>	14

rpgm.rexp	15
rpgm.rgeom	16
rpgm.rlnorm	17
rpgm.rnorm	18
rpgm.rt	19
rpoisson	20
skewness	21
vasicek	22

Index	24
--------------	-----------

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	<i>Set a Minimum or a Maximum or Both to a Vector.</i>
-------	--

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

<code>x</code>	double, vector to put in <code>[m, M]</code> .
<code>m</code>	double, the minimum.
<code>M</code>	double, the maximum.

Value

A vector with the values of `x` on which all values lower than `m` has been replaced by `m` and all values greater than `M` has been replaced by `M`.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

`evalpoisson`*Evaluate the Poisson Process paths at date t*

Description

The function `evalpoisson` evaluates all paths of Poisson processes at date `t` and returns a vector with the corresponding values

Usage

```
evalpoisson(P, t = 1)
```

Arguments

<code>P</code>	list of double vector, a Poisson process simulation of <code>n</code> paths from <code>rpoisson</code> or <code>rinpoisson</code> .
<code>t</code>	double, atomic vector for evaluating each path at date <code>t</code> (do not support yet a vector).

Value

`evalpoisson` returns an integer vector of the size of the number of paths. The `i`th element is the value of the `i`th path of the Poisson process at date `t`.

Author(s)

Nicolas Baradel - PGM Solutions

References

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

See Also

<https://pgm-solutions.com/packages>

Examples

```
P <- rpoisson(5, 5)
evalpoisson(P, 0.5)
```

`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 on the convergence, if the vector is i.i.d. normal random variables, of

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

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

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

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

<code>n</code>	integer, number of simulations.
<code>prob</code>	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

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

Value

Returns a $n \times m+1$ 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 discrete 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 $\sum_{j=1, +\infty} c_j / 3^j$ with c_j in $\{0, 2\}$ and the corresponding probability distribution simulates uniformly the c_j (here, to $j=32$).

Value

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

Note

This distribution is provided only for theoretical 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 $X[t+h] = X[t] + b(X[t])h + s(X[t])\text{sqrt}(h)Z$, where $Z \sim N(0,1)$.

Usage

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

Arguments

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

Value

If `all_dates = TRUE`, it returns a $n \times m+1$ 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

<code>n</code>	integer, number of simulations.
<code>xi</code>	double, the shape.
<code>mu</code>	double, the location.
<code>sigma</code>	double, the scale.

Value

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

Note

For $\xi \neq 0$, the cumulative distribution function is:

$$F(x) = 1 - (1 + \xi(x - \mu)/\sigma)^{-1/\xi}$$

for $x \geq \mu$ when $\xi > 0$ and $\mu \leq x \leq \mu - \sigma/\xi$ 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)
```

`rinpoisson`*Simulation of inhomogeneous Poisson Processes*

Description

The function `rinpoisson` is a R-level function which simulates the jumping times of an inhomogeneous Poisson process, returning each path as a vector of a list.

Usage

```
rinpoisson(n, lambda, T = 1, drop = TRUE)
```

Arguments

<code>n</code>	integer, number of paths.
<code>lambda</code>	double, function of the intensity of the inhomogeneous Poisson processes over the time.
<code>T</code>	double, end time of the simulations.
<code>drop</code>	logical, if <code>n = 1</code> and <code>drop = TRUE</code> , returns the single path as a vector instead of a list.

Value

`rinpoisson` returns a list of `n` paths of an inhomogeneous Poisson process of intensity function `lambda`. Each element of the list is the vector of the jumping times.

Note

The function `lambda` must be vectorial in the sense that, if given an argument `x`, it returns a vector of the same size.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Poisson_point_process#Inhomogeneous_Poisson_point_process

See Also

<https://pgm-solutions.com/packages>

Examples

```
lambda <- function(t) return(400*(1+sin(-pi/2+2*pi*t)))
P <- rinpoisson(5, lambda, T=4)

plot(density(P[[1]], bw = 0.05))
lines((0:400)/100, lambda((0:400)/100)/integrate(lambda, 0, 4)$value, col="red")
length(P[[1]])
integrate(lambda, 0, 4)
```

 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 $Z \sim N(0,h)$ (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 final value X_T .
delta	double, the step size.

Value

If `all_dates = TRUE`, it returns a $n \times m+1$ 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 simulation, 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

<code>n</code>	integer, number of simulations.
<code>prob</code>	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

<code>n</code>	integer, number of simulations.
<code>mean</code>	double, the mean (or the vector of means) of the logarithm of the log-normal variable.
<code>sd</code>	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

<code>n</code>	integer, number of simulations.
<code>mean</code>	double, the mean (or the vector of means).
<code>sd</code>	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 simulation, 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 distribution is 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)
```

rpoisson

Simulation of homogeneous Poisson Processes

Description

The function `rpoisson` is a C-level function which simulates the jumping times of a Poisson process, returning each path as a vector of a list.

Usage

```
rpoisson(n, lambda = 1, T = 1, drop = TRUE)
```

Arguments

<code>n</code>	integer, number of paths.
<code>lambda</code>	double, intensity of the Poisson process.
<code>T</code>	double, end time of the simulations.
<code>drop</code>	logical, if <code>n = 1</code> and <code>drop = TRUE</code> , returns the single path as a vector instead of a list.

Value

`rpoisson` returns a list of `n` paths of a Poisson process of intensity `lambda`. Each element of the list is the vector of the jumping times.

Author(s)

Nicolas Baradel - PGM Solutions

References

https://en.wikipedia.org/wiki/Poisson_point_process#Homogeneous_Poisson_point_process

See Also

<https://pgm-solutions.com/packages>

Examples

```
x <- rpoisson(5, 5)
```

`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

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

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))
```

Description

The definition of the process used here is:

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

where (μ, 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 $\text{length}(x)-1$. Note that the first value has no density. lvasicek returns the log-likelihood associated to dvasicek and evasicek returns the Maximum Likelihood Estimator of the parameters (μ, 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

*Topic `\textasciitildekwd1`

bound, [3](#)
colMaxs - rowMaxs - colMins -
 rowMins, [4](#)
evalpoisson, [5](#)
jarquebera, [6](#)
kurtosis, [7](#)
rbernou, [8](#)
rbrownian, [9](#)
rcantor, [10](#)
reuler, [11](#)
rgpd, [12](#)
rinpoisson, [13](#)
rmilstein, [14](#)
rpgm.rexp, [15](#)
rpgm.rgeom, [16](#)
rpgm.rlnorm, [17](#)
rpgm.rnorm, [18](#)
rpgm.rt, [19](#)
rpoisson, [20](#)
skewness, [21](#)
vasicek, [22](#)

*Topic `\textasciitildekwd2`

bound, [3](#)
colMaxs - rowMaxs - colMins -
 rowMins, [4](#)
evalpoisson, [5](#)
jarquebera, [6](#)
kurtosis, [7](#)
rbernou, [8](#)
rbrownian, [9](#)
rcantor, [10](#)
reuler, [11](#)
rgpd, [12](#)
rinpoisson, [13](#)
rmilstein, [14](#)
rpgm.rexp, [15](#)
rpgm.rgeom, [16](#)
rpgm.rlnorm, [17](#)

rpgm.rnorm, [18](#)
rpgm.rt, [19](#)
rpoisson, [20](#)
skewness, [21](#)
vasicek, [22](#)

*Topic `package`

rpgm-package, [2](#)

bound, [3](#)

colMaxs (colMaxs - rowMaxs - colMins -
 rowMins), [4](#)

colMaxs - rowMaxs - colMins - rowMins,
[4](#)

colMins (colMaxs - rowMaxs - colMins -
 rowMins), [4](#)

dvasicek (vasicek), [22](#)

evalpoisson, [5](#)

evasicek (vasicek), [22](#)

jarquebera, [6](#)

kurtosis, [7](#)

lbound (bound), [3](#)

lvasicek (vasicek), [22](#)

rbernou, [8](#)

rbrownian, [9](#)

rcantor, [10](#)

reuler, [11](#)

rgpd, [12](#)

rinpoisson, [13](#)

rmilstein, [14](#)

rowMaxs (colMaxs - rowMaxs - colMins -
 rowMins), [4](#)

rowMins (colMaxs - rowMaxs - colMins -
 rowMins), [4](#)

rpgm (rpgm-package), [2](#)

rpgm-package, 2
rpgm.rexp, 15
rpgm.rgeom, 16
rpgm.rlnorm, 17
rpgm.rnorm, 18
rpgm.rt, 19
rpoisson, 20
rvasicek (vasicek), 22

skewness, 21

ubound (bound), 3

vasicek, 22