

# Package ‘VarianceGamma’

March 1, 2010

**Version** 0.3-0

**Date** 2010-02-10

**Title** The Variance Gamma Distribution

**Author** David Scott <d.scott@auckland.ac.nz> and Christine Yang Dong  
<c.dong@auckland.ac.nz>

**Maintainer** David Scott <d.scott@auckland.ac.nz>

**Depends** R (>= 2.3.0), DistributionUtils, GeneralizedHyperbolic

**Encoding** latin1

**Description** This package provides functions for the variance gamma distributions. Density, distribution and quantile functions. Functions for random number generation and fitting of the variance gamma to data. Also, functions for computing moments of the variance gamma distribution of any order about any location. In addition, there are functions for checking the validity of parameters and to interchange different sets of parameterizations for the variance gamma distribution.

**License** GPL (>= 2)

**URL** <http://www.r-project.org>

**Repository** CRAN

**Date/Publication** 2010-03-01 06:57:51

## R topics documented:

summary.vgFit . . . . .	2
Variance Gamma Mean, Variance, Skewness, Kurtosis and Mode . . . . .	3
VarianceGammaDistribution . . . . .	4
VarianceGammaPlots . . . . .	9
vgCalcRange . . . . .	11
vgChangePars . . . . .	13
vgCheckPars . . . . .	15

vgFit . . . . .	16
vgFitStart . . . . .	19
vgMom . . . . .	21
vgParam . . . . .	23

<b>Index</b>	<b>24</b>
--------------	-----------

---

summary.vgFit	<i>Summarizing Variance Gamma Distribution Fit</i>
---------------	--

---

## Description

summary Method for class "vgFit".

## Usage

```
## S3 method for class 'vgFit':
summary(object, ...)

## S3 method for class 'summary.vgFit':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

## Arguments

object	An object of class "vgFit", resulting from a call to <a href="#">vgFit</a> .
x	An object of class "summary.vgFit", resulting from a call to <a href="#">summary.vgFit</a> .
digits	The number of significant digits to use when printing.
...	Further arguments passed to or from other methods.

## Details

summary.vgFit calculates standard errors for the estimates of  $c$ ,  $\sigma$ ,  $\theta$ , and  $\nu$  of the variance gamma distribution parameter vector `param` if the Hessian from the call to `optim` or `nlm` is available. Because the parameters in the call to the optimiser are  $c$ ,  $\log(\sigma)$ ,  $\theta$  and  $\log(\nu)$ , the delta method is used to obtain the standard errors for  $\sigma$  and  $\nu$ .

## Value

If the Hessian is available, `summary.vgFit` computes standard errors for the estimates of  $c$ ,  $\sigma$ ,  $\theta$ , and  $\nu$ , and adds them to `object` as `object$sds`. Otherwise, no calculations are performed and the composition of `object` is unaltered.

`summary.vgFit` invisibly returns `x` with class changed to `summary.vgFit`.

See [vgFit](#) for the composition of an object of class `vgFit`.

`print.summary.vgFit` prints a summary in the same format as [print.vgFit](#) when the Hessian is not available from the fit. When the Hessian is available, the standard errors for the parameter estimates are printed in parentheses beneath the parameter estimates, in the manner of `fitdistr` in the package `MASS`.

**See Also**

[vgFit](#), [summary](#).

**Examples**

```
### Continuing the vgFit(.) example:
param <- c(0,0.5,0,0.5)
dataVector <- rvg(500, param = param)
fit <- vgFit(dataVector)
print(fit)
summary(fit)
```

---

Variance Gamma Mean, Variance, Skewness, Kurtosis and Mode  
*Moments and Mode of the Variance Gamma Distribution*

---

**Description**

Functions to calculate the mean, variance, skewness, kurtosis and mode of a specific variance gamma distribution.

**Usage**

```
vgMean(vgC = 0, sigma = 1, theta = 0, nu = 1, param = c(vgC, sigma, theta, nu))
vgVar(vgC = 0, sigma = 1, theta = 0, nu = 1, param = c(vgC, sigma, theta, nu))
vgSkew(vgC = 0, sigma = 1, theta = 0, nu = 1, param = c(vgC, sigma, theta, nu))
vgKurt(vgC = 0, sigma = 1, theta = 0, nu = 1, param = c(vgC, sigma, theta, nu))
vgMode(vgC = 0, sigma = 1, theta = 0, nu = 1, param = c(vgC, sigma, theta, nu))
```

**Arguments**

<code>vgC</code>	The location parameter $c$ , default is equal to 0.
<code>sigma</code>	The spread parameter $\sigma$ , default is equal to 1, must be positive.
<code>theta</code>	The asymmetry parameter $\theta$ , default is equal to 0.
<code>nu</code>	The shape parameter $\nu$ , default is equal to 1, must be positive.
<code>param</code>	Specifying the parameters as a vector which takes the form <code>c(vgC, sigma, theta, nu)</code> .

**Value**

`vgMean` gives the mean of the variance gamma distribution, `vgVar` the variance, `vgSkew` the skewness, `vgKurt` the kurtosis, and `vgMode` the mode. The formulae used for the mean and variance are as given in Seneta (2004). If  $\nu$  is greater than or equal to 2, the mode is equal to the value of the parameter  $c$ . Otherwise, it is found by a numerical optimisation using [optim](#).

The parameterisation of the variance gamma distribution used for these functions is the  $(c, \sigma, \theta, \nu)$  one. See [vgChangePars](#) to transfer between parameterisations.

**Author(s)**

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

**References**

Seneta, E. (2004). Fitting the variance-gamma model to financial data. *J. Appl. Prob.*, 41A:177–187. Kotz, S, Kozubowski, T. J., and Podgórski, K. (2001). The Laplace Distribution and Generalizations. *Birkhauser*, Boston, 349 p.

**See Also**

[dvg](#), [vgChangePars](#), [vgCalcRange](#), [besselK](#).

**Examples**

```
param <- c(2,2,2,0.5)
vgMean(param = param)
## Or to specify parameter values individually, use:
vgMean (2,2,2,0.5)

vgVar(param = param)
vgSkew(param = param)
vgKurt(param = param)
vgMode(param = param)
maxDens <- dvg(vgMode(param = param), param = param)
vgRange <- vgCalcRange(param = param, tol = 10^(-2)*maxDens)
curve(dvg(x, param = param), vgRange[1], vgRange[2])
abline(v = vgMode(param = param), col = "blue")
abline(v = vgMean(param = param), col = "red")
```

---

VarianceGammaDistribution

*The Variance Gamma Distribution*

---

**Description**

Density function, distribution function, quantiles and random number generation for the variance gamma distribution with parameters  $c$  (location),  $\sigma$  (spread),  $\theta$  (asymmetry) and  $\nu$  (shape). Utility routines are included for the derivative of the density function and to find suitable break points for use in determining the distribution function.

**Usage**

```
dvg(x, vgC = 0, sigma = 1, theta = 0, nu = 1,
    param = c(vgC, sigma, theta, nu), log = FALSE,
    tolerance = .Machine$double.eps ^ 0.5, ...)
pvg(q, vgC = 0, sigma = 1, theta = 0, nu = 1,
    param = c(vgC, sigma, theta, nu), lower.tail = TRUE, log.p = FALSE,
```

```

    small = 10^(-6), tiny = 10^(-10), deriv = 0.3, subdivisions = 100,
    accuracy = FALSE, ...)
qvg(p, vgC = 0, sigma = 1, theta = 0, nu = 1,
    param = c(vgC, sigma, theta, nu), lower.tail = TRUE, log.p = FALSE,
    small = 10^(-6), tiny = 10^(-10), deriv = 0.3, nInterpol = 100,
    subdivisions = 100, ...)
rvg(n, vgC = 0, sigma = 1, theta = 0, nu = 1,
    param = c(vgC, sigma, theta, nu))
ddvg(x, vgC = 0, sigma = 1, theta = 0, nu = 1,
    param = c(vgC, sigma, theta, nu), log = FALSE,
    tolerance = .Machine$double.eps ^ 0.5, ...)
vgBreaks(vgC = 0, sigma = 1, theta = 0, nu = 1,
    param = c(vgC, sigma, theta, nu), small = 10^(-6), tiny = 10^(-10),
    deriv = 0.3, ...)

```

### Arguments

<code>x, q</code>	Vector of quantiles.
<code>p</code>	Vector of probabilities.
<code>n</code>	Number of observations to be generated.
<code>vgC</code>	The location parameter $c$ , default is 0.
<code>sigma</code>	The spread parameter $\sigma$ , default is 1, must be positive.
<code>theta</code>	The asymmetry parameter $\theta$ , default is 0.
<code>nu</code>	The shape parameter $\nu$ , default is 1, must be positive.
<code>param</code>	Specifying the parameters as a vector which takes the form <code>c(vgC, sigma, theta, nu)</code> .
<code>log, log.p</code>	Logical; if TRUE, probabilities $p$ are given as $\log(p)$ ; not yet implemented.
<code>lower.tail</code>	If TRUE (default), probabilities are $P[X \leq x]$ , otherwise, $P[X > x]$ ; not yet implemented.
<code>small</code>	Size of a small difference between the distribution function and zero or one. See <b>Details</b> .
<code>tiny</code>	Size of a tiny difference between the distribution function and zero or one. See <b>Details</b> .
<code>deriv</code>	Value between 0 and 1. Determines the point where the derivative becomes substantial, compared to its maximum value. See <b>Details</b> .
<code>accuracy</code>	Uses accuracy calculated by <code>~integrate</code> to try and determine the accuracy of the distribution function calculation.
<code>subdivisions</code>	The maximum number of subdivisions used to integrate the density returning the distribution function.
<code>nInterpol</code>	The number of points used in <code>qvg</code> for cubic spline interpolation (see <code>splinefun</code> ) of the distribution function.
<code>tolerance</code>	Size of a machine difference between two values. See <b>Details</b> .
<code>...</code>	Passes arguments to <code>uniroot</code> . See <b>Details</b> .

## Details

Users may either specify the values of the parameters individually or as a vector. If both forms are specified but with different values, then the values specified by vector `param` will always overwrite the other ones.

The variance gamma distribution has density

$$f(x) = c(c, \sigma, \theta, \nu) \times e^{[\theta(x-c)/\sigma^2]} |x - c|^{1/\nu-1/2} K_{1/\nu-1/2} \left( \frac{|x - c| \sqrt{2\sigma^2/\nu + \theta^2}}{\sigma^2} \right)$$

where  $K_\nu(\cdot)$  is the modified Bessel function of the third kind of order  $\nu$ , and

$$c(c, \sigma, \theta, \nu) = \frac{2}{\sigma \sqrt{2\pi} \nu^{1/\nu} \Gamma(1/\nu)} \left( \frac{1}{\sqrt{2\sigma^2/\nu + \theta^2}} \right)^{1/\nu-1/2}$$

Special cases:

1. If  $\nu < 2$  and  $x = c$ , then the density function is approximate to

$$f(x) = \frac{\Gamma(1/\nu - 1/2)}{\sigma \sqrt{2\pi} \nu^{1/\nu} \Gamma(1/\nu)} \left( \frac{2\sigma^2}{\sqrt{2\sigma^2/\nu + \theta^2}} \right)^{1/\nu-1/2}$$

2. If  $\nu \geq 2$  and  $x = c$ , then the density function is taken the value `Inf`.

Use `vgChangePars` to convert from the  $(\mu, \sigma, \theta, \tau)$ , or  $(\theta, \sigma, \kappa, \tau)$  parameterisations given in Kotz *et al.* (2001) to the  $(c, \sigma, \theta, \nu)$  parameterisation used above.

`pvg` breaks the real line into eight regions in order to determine the integral of `dvf`. The break points determining the regions are found by `vgBreaks`, based on the values of `small`, `tiny`, and `deriv`. In the extreme tails of the distribution where the probability is `tiny` according to `vgCalcRange`, the probability is taken to be zero. In the inner part of the distribution, the range is divided in 6 regions, 3 above the mode, and 3 below. On each side of the mode, there are two break points giving the required three regions. The outer break point is where the probability in the tail has the value given by the variable `small`. The inner break point is where the derivative of the density function is `deriv` times the maximum value of the derivative on that side of the mode. In each of the 6 inner regions the numerical integration routine `safeIntegrate` (which is a wrapper for `integrate`) is used to integrate the density `dvf`.

`qvg` uses the breakup of the real line into the same 8 regions as `pvg`. For quantiles which fall in the 2 extreme regions, the quantile is returned as `-Inf` or `Inf` as appropriate. In the 6 inner regions `splinefun` is used to fit values of the distribution function generated by `pvg`. The quantiles are then found using the `uniroot` function.

`pvg` and `qvg` may generally be expected to be accurate to 5 decimal places.

The variance gamma distribution is discussed in Kotz *et al.* (2001). It can be seen to be the weighted difference of two i.i.d. gamma variables shifted by the value of  $\theta$ . `rvg` uses this representation to generate observations from the variance gamma distribution.

**Value**

`dvg` gives the density function, `pvg` gives the distribution function, `qvg` gives the quantile function and `rvvg` generates random variates. An estimate of the accuracy of the approximation to the distribution function may be found by setting `accuracy=TRUE` in the call to `pvg` which then returns a list with components `value` and `error`.

`ddvg` gives the derivative of `dvg`.

`vgBreaks` returns a list with components:

<code>xTiny</code>	Value such that probability to the left is less than <code>tiny</code> .
<code>xSmall</code>	Value such that probability to the left is less than <code>small</code> .
<code>lowBreak</code>	Point to the left of the mode such that the derivative of the density is <code>deriv</code> times its maximum value on that side of the mode.
<code>highBreak</code>	Point to the right of the mode such that the derivative of the density is <code>deriv</code> times its maximum value on that side of the mode.
<code>xLarge</code>	Value such that probability to the right is less than <code>small</code> .
<code>xHuge</code>	Value such that probability to the right is less than <code>tiny</code> .
<code>modeDist</code>	The mode of the given variance gamma distribution.

**Author(s)**

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

**References**

Seneta, E. (2004). Fitting the variance-gamma model to financial data. *J. Appl. Prob.*, 41A:177–187. Kotz, S, Kozubowski, T. J., and Podgórski, K. (2001). The Laplace Distribution and Generalizations. *Birkhauser*, Boston, 349 p.

**See Also**

[vgChangePars](#), [vgCalcRange](#)

**Examples**

```
## Use the following rules for vgCalcRange when plotting graphs for dvg,
## ddvg and pvg.
## if nu < 2, use:
##   maxDens <- dvg(vgMode(param = c(vgC, sigma, theta, nu)),
##   param = c(vgC, sigma, theta, nu), log = FALSE)
##   vgRange <- vgCalcRange(param = c(vgC, sigma, theta, nu),
##   tol = 10^(-2)*maxDens, density = TRUE)

## if nu >= 2 and theta < 0, use:
##   vgRange <- c(vgC-2,vgC+6)
## if nu >= 2 and theta > 0, use:
##   vgRange <- c(vgC-6,vgC+2)
## if nu >= 2 and theta = 0, use:
##   vgRange <- c(vgC-4,vgC+4)
```

```

# Example 1 (nu < 2)
## For dvg and pvg
param <- c(0,0.5,0,0.5)
maxDens <- dvg(vgMode(param = param), param = param, log = FALSE)
## Or to specify parameter values individually, use:
maxDens <- dvg(vgMode(0,0.5,0,0.5), 0,0.5,0,0.5, log = FALSE)

vgRange <- vgCalcRange(param = param, tol = 10^(-2)*maxDens, density = TRUE)
par(mfrow = c(1,2))
curve(dvg(x, param = param), from = vgRange[1], to = vgRange[2], n = 1000)
title("Density of the Variance Gamma Distribution")
curve(pvg(x, param = param), from = vgRange[1], to = vgRange[2], n = 1000)
title("Distribution Function of the Variance Gamma Distribution")

## For rvg
dataVector <- rvg(500, param = param)
curve(dvg(x, param = param), range(dataVector)[1], range(dataVector)[2],
      n = 500)
hist(dataVector, freq = FALSE, add = TRUE)
title("Density and Histogram of the Variance Gamma Distribution")
logHist(dataVector, main =
         "Log-Density and Log-Histogram of the Generalized Hyperbolic Distribution")
curve(log(dvg(x, param = param)), add = TRUE,
      range(dataVector)[1], range(dataVector)[2], n = 500)

## For dvg and ddvg
par(mfrow = c(2,1))
curve(dvg(x, param = param), from = vgRange[1], to = vgRange[2],
      n = 1000)
title("Density of the Variance Gamma Distribution")
curve(ddvg(x, param = param), from = vgRange[1], to = vgRange[2],
      n = 1000)
title("Derivative of the Density of the Variance Gamma Distribution")

# Example 2 (nu > 2 and theta = 0)
## For dvg and pvg
param <- c(0,0.5,0,3)
vgRange <- c(0-4,0+4)
par(mfrow = c(1,2))
curve(dvg(x, param = param), from = vgRange[1], to = vgRange[2],
      n = 1000)
title("Density of the Variance Gamma Distribution")
curve(pvg(x, param = param), from = vgRange[1], to = vgRange[2],
      n = 1000)
title("Distribution Function of the Variance Gamma Distribution")

## For rvg
dataVector <- rvg(500, param = param)
curve(dvg(x, param = param), range(dataVector)[1], range(dataVector)[2],
      n = 500)
hist(dataVector, freq = FALSE, add = TRUE)
title("Density and Histogram of the Variance Gamma Distribution")

```

```

logHist(dataVector, main =
  "Log-Density and Log-Histogram of the Generalized Hyperbolic Distribution")
curve(log(dvg(x, param = param)), add = TRUE,
  range(dataVector)[1], range(dataVector)[2], n = 500)

## For dvg and ddvg
par(mfrow = c(2,1))
curve(dvg(x, param = param), from = vgRange[1], to = vgRange[2],
  n = 1000)
title("Density of the Variance Gamma Distribution")
curve(ddvg(x, param = param), from = vgRange[1], to = vgRange[2],
  n = 1000)
title("Derivative of the Density of the Variance Gamma Distribution")

## Use the following rules for vgCalcRange when plotting graphs for vgBreaks.
## if (nu < 2), use:
##   maxDens <- dvg(vgMode(param = c(vgC, sigma, theta, nu)),
##     param = c(vgC, sigma, theta, nu), log = FALSE)
##   vgRange <- vgCalcRange(param = param, tol = 10^(-6)*maxDens, density = TRUE)
## if (nu >= 2) and theta < 0, use:
##   vgRange <- c(vgC-2,vgC+6)
## if (nu >= 2) and theta > 0, use:
##   vgRange <- c(vgC-6,vgC+2)
## if (nu >= 2) and theta = 0, use:
##   vgRange <- c(vgC-4,vgC+4)

## Example 3 (nu < 2)
## For vgBreaks
param <- c(0,0.5,0,0.5)
maxDens <- dvg(vgMode(param = param), param = param, log = FALSE)
vgRange <- vgCalcRange(param = param, tol = 10^(-6)*maxDens, density = TRUE)
curve(dvg(x, param = param), from = vgRange[1], to = vgRange[2],
  n = 1000)
bks <- vgBreaks(param = param)
abline(v = bks)
title("Density of the Variance Gamma Distribution with breaks")

## Example 4 (nu > 2 and theta = 0)
## For vgBreaks
param <- c(0,0.5,0,3)
vgRange <- c(0-4,0+4)
curve(dvg(x, param = param), from = vgRange[1], to = vgRange[2],
  n = 1000)
bks <- vgBreaks(param = param)
abline(v = bks)
title("Density of the Variance Gamma Distribution with breaks")

```

**Description**

`qqvg` produces a variance gamma Q-Q plot of the values in `y`.

`ppvg` produces a variance gamma P-P (percent-percent) or probability plot of the values in `y`.

Graphical parameters may be given as arguments to `qqvg` and `ppvg`.

**Usage**

```
qqvg(y, vgC = NULL, sigma = NULL, theta = NULL, nu = NULL,
     param = c(vgC, sigma, theta, nu), main = "Variance Gamma Q-Q Plot",
     xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
     plot.it = TRUE, line = TRUE, ...)
```

```
ppvg(y, vgC = NULL, sigma = NULL, theta = NULL, nu = NULL,
     param = c(vgC, sigma, theta, nu), main = "Variance Gamma P-P Plot",
     xlab = "Uniform Quantiles",
     ylab = "Probability-integral-transformed Data", plot.it = TRUE,
     line = TRUE, ...)
```

**Arguments**

<code>y</code>	The data sample.
<code>vgC</code>	The location parameter $c$ , default is 0.
<code>sigma</code>	The spread parameter $\sigma$ , default is 1, must be positive.
<code>theta</code>	The asymmetry parameter $\theta$ , default is 0.
<code>nu</code>	The shape parameter $\nu$ , default is 1, must be positive.
<code>param</code>	An optional option, specifying the parameters as a vector which takes the form <code>c(vgC, sigma, theta, nu)</code> if known.
<code>main</code>	Plot title.
<code>xlab, ylab</code>	Plot labels.
<code>plot.it</code>	Logical. Should the result be plotted?
<code>line</code>	Add line through origin with unit slope.
<code>...</code>	Further graphical parameters.

**Details**

Users may specify the parameter values of the data sample `y` using argument `param`. If `param` is not specified by users, then the values are estimated from `y` by `vgFit`. For more details of fitting a variance gamma distribution to data, see [vgFit](#).

**Value**

For `qqvg` and `ppvg`, a list with components:

<code>x</code>	The x coordinates of the points that are to be plotted.
<code>y</code>	The y coordinates of the points that are to be plotted.

**Author(s)**

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

**References**

Wilk, M. B. and Gnanadesikan, R. (1968) Probability plotting methods for the analysis of data. *Biometrika*. **55**, 1–17.

**See Also**

[ppoints](#), [dvg](#).

**Examples**

```
## Example 1: the parameter values are known
par(mfrow = c(1,2))
y <- rvg(200, param = c(2,2,1,2))
qqvg(y, param = c(2,2,1,2), line = FALSE)
abline(0, 1, col = 2)
ppvg(y, param = c(2,2,1,2))

## Example 2: the parameter values are unknown
par(mfrow = c(1,2))
y <- rvg(200, param = c(2,2,1,2))
qqvg(y, line = FALSE)
abline(0, 1, col = 2)
ppvg(y)
```

---

vgCalcRange

*Range of a Variance Gamma Distribution*

---

**Description**

Given the parameter vector `param` or the individual parameter values  $(c, \sigma, \theta, \nu)$  of a variance gamma distribution, this function determines the range outside of which the density function is negligible, to a specified tolerance. The parameterization used is the  $(c, \sigma, \theta, \nu)$  one (see [dvg](#)). To use another parameterization, use [vgChangePars](#).

**Usage**

```
vgCalcRange(vgC = 0, sigma = 1, theta = 0, nu = 1,
            param = c(vgC, sigma, theta, nu), tol = 10^(-5), density = TRUE, ...)
```

**Arguments**

vgC	The location parameter $c$ , default is 0.
sigma	The spread parameter $\sigma$ , default is 1, must be positive.
theta	The asymmetry parameter $\theta$ , default is 0.
nu	The shape parameter $\nu$ , default is 1, must be positive.
param	Specifying the parameters as a vector which takes the form <code>c(vgC, sigma, theta, nu)</code> .
tol	Tolerance.
density	Logical. If TRUE, the bounds are for the density function. If FALSE, they should be for the probability distribution, but this has not yet been implemented.
...	Extra arguments for calls to <a href="#">uniroot</a> .

**Details**

Users may either specify the values of the parameters individually or as a vector. If both forms are specified but with different values, then the values specified by vector `param` will always overwrite the other ones.

The particular variance gamma distribution being considered is specified by the value of the parameter `param`.

If `density = TRUE`, the function gives a range, outside of which the density is less than the given tolerance. Useful for plotting the density. Also used in determining break points for the separate sections over which numerical integration is used to determine the distribution function. The points are found by using [uniroot](#) on the density function.

If `density = FALSE`, the function returns the message: "Distribution function bounds not yet implemented".

**Value**

A two-component vector giving the lower and upper ends of the range.

**Author(s)**

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

**References**

Seneta, E. (2004). Fitting the variance-gamma model to financial data. *J. Appl. Prob.*, 41A:177–187. Kotz, S, Kozubowski, T. J., and Podgórski, K. (2001). The Laplace Distribution and Generalizations. *Birkhauser*, Boston, 349 p.

**See Also**

[dvg](#), [vgChangePars](#)

**Examples**

```
## Use the following rules for vgCalcRange when plotting graphs for dvg,
## ddvg and pvg.
## if nu < 2, use:
##   maxDens <- dvg(vgMode(param = c(vgC, sigma, theta, nu)),
##   param = c(vgC, sigma, theta, nu), log = FALSE)
##   vgRange <- vgCalcRange(param = c(vgC, sigma, theta, nu),
##   tol = 10^(-2)*maxDens, density = TRUE)

## if nu >= 2 and theta < 0, use:
##   vgRange <- c(vgC-2,vgC+6)
## if nu >= 2 and theta > 0, use:
##   vgRange <- c(vgC-6,vgC+2)
## if nu >= 2 and theta = 0, use:
##   vgRange <- c(vgC-4,vgC+4)

param <- c(0,0.5,0,0.5)
maxDens <- dvg(vgMode(param = param), param = param)
vgRange <- vgCalcRange(param = param, tol = 10^(-2)*maxDens)
vgRange
curve(dvg(x, param = param), vgRange[1], vgRange[2])
curve(dvg(x, param = param), vgRange[1], vgRange[2])

param <- c(2,2,0,3)
vgRange <- c(2-4,2+4)
vgRange
curve(dvg(x, param = param), vgRange[1], vgRange[2])
## Not run: vgCalcRange(param = param, tol = 10^(-3), density = FALSE)
```

**Description**

This function interchanges between the following 4 parameterizations of the variance gamma distribution:

1.  $c, \sigma, \theta, \nu$
2.  $\theta, \sigma, \mu, \tau$
3.  $\theta, \sigma, \kappa, \tau$
4.  $\lambda, \alpha, \beta, \mu$

The first set of parameterizations is given in Seneta (2004). The second and third ones are the parameterizations given in Kotz *et al.* (2001). The last set takes the form of the generalized hyperbolic distribution parameterization.  $\delta$  is not included since the variance gamma distribution is a limiting case of generalized hyperbolic distribution with  $\delta$  always equal to 0.

**Usage**

```
vgChangePars(from, to, param, noNames = FALSE)
```

**Arguments**

from	The set of parameters to change from.
to	The set of parameters to change to.
param	"from" parameter vector consisting of 4 numerical elements.
noNames	Logical. When TRUE, suppresses the parameter names in the output.

**Details**

In the 3 parameterizations, the following must be positive:

1.  $\sigma, \nu$
2.  $\sigma, \tau$
3.  $\sigma, \tau$
4.  $\lambda, \alpha$

In addition in the 4th parameterization, the absolute value of  $\beta$  must be less than  $\alpha$ .

**Value**

A numerical vector of length 4 representing `param` in the `to` parameterization.

**Author(s)**

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

**References**

Seneta, E. (2004). Fitting the variance-gamma model to financial data. *J. Appl. Prob.*, 41A:177–187. Kotz, S, Kozubowski, T. J., and Podgórski, K. (2001). The Laplace Distribution and Generalizations. *Birkhauser*, Boston, 349 p.

**See Also**

[dvg](#), [vgMom](#)

**Examples**

```
param1 <- c(2, 2, 1, 3) # Parameterization 1
param2 <- vgChangePars(1, 2, param1) # Convert to parameterization 2
param2 # Parameterization 2
vgChangePars(2, 1, as.numeric(param2)) # Convert back to parameterization 1

param3 <- c(1, 2, 0, 0.5) # Parameterization 3
param1 <- vgChangePars(3, 1, param3) # Convert to parameterization 1
param1 # Parameterization 1
vgChangePars(1, 3, as.numeric(param1)) # Convert back to parameterization 3
```

---

 vgCheckPars

*Check Parameters of the Variance Gamma Distribution*


---

## Description

Given a putative set of parameters for the variance gamma distribution, the functions checks if the parameters are in the correct range, and if the set has the correct length of 4.

## Usage

```
vgCheckPars(param, ...)
```

## Arguments

param	Numeric. Putative parameter values for a Variance Gamma distribution.
...	Further arguments for calls to <code>all.equal</code> .

## Details

The vector `param` takes the form `c(c, sigma, theta, nu)`. If either `sigma` or `nu` is negative, then an error message is returned.

If the vector `param` has a length not equal to 4, then an error message is returned.

## Value

A list with components:

case	Whichever of 'error' or 'normal' is identified by the function.
errMessage	An appropriate error message if an error was found, the empty string "" otherwise.

## Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

## See Also

[dvg](#), [vgMom](#)

## Examples

```
vgCheckPars(c(0,1,0,1))      # normal
vgCheckPars(c(0,0,0,1))      # error
vgCheckPars(c(0,1,0,-2))     # error
vgCheckPars(c(0,1,0))        # error
```

**Description**

Fits a variance gamma distribution to data. Displays the histogram, log-histogram (both with fitted densities), Q-Q plot and P-P plot for the fit which has the maximum likelihood.

**Usage**

```
vgFit(x, freq = NULL, breaks = NULL, paramStart = NULL,
      startMethod = "Nelder-Mead", startValues = "SL",
      method = "Nelder-Mead", hessian = FALSE,
      plots = FALSE, printOut = FALSE,
      controlBFGS = list(maxit = 200),
      controlNM = list(maxit = 1000), maxitNLM = 1500, ...)

## S3 method for class 'vgFit':
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'vgFit':
plot(x, which = 1:4,
      plotTitles = paste(c("Histogram of ", "Log-Histogram of ",
                           "Q-Q Plot of ", "P-P Plot of "), x$obsName,
                          sep = " "),
      ask = prod(par("mfcol")) < length(which) && dev.interactive(),
      ...)
```

**Arguments**

<code>x</code>	Data vector for <code>vgFit</code> . Object of class "vgFit" for <code>print.vgFit</code> and <code>plot.vgFit</code> .
<code>freq</code>	A vector of weights with length equal to <code>length(x)</code> .
<code>breaks</code>	Breaks for histogram, defaults to those generated by <code>hist(x, right = FALSE, plot = FALSE)</code> .
<code>paramStart</code>	A user specified starting parameter vector <code>param</code> taking the form <code>c(vgC, sigma, theta, nu)</code> .
<code>startMethod</code>	Method used by <code>vgFitStart</code> in calls to <code>optim</code> , default is "Nelder-Mead". See <b>Details</b> .
<code>startValues</code>	Code giving the method of determining starting values for finding the maximum likelihood estimate of <code>param</code> , default method is "SL". See <b>Details</b> .
<code>method</code>	Different optimisation methods to consider, default is "Nelder-Mead". See <b>Details</b> .
<code>hessian</code>	Logical. If TRUE the value of the hessian is returned.

plots	Logical. If FALSE suppresses printing of the histogram, log-histogram, Q-Q plot and P-P plot.
printOut	Logical. If FALSE suppresses printing of results of fitting.
controlBFGS	A list of control parameters for <code>optim</code> when using the "BFGS" optimisation.
controlNLM	A list of control parameters for <code>optim</code> when using the "Nelder-Mead" optimisation.
maxitNLM	A positive integer specifying the maximum number of iterations when using the "nlm" optimisation.
digits	Desired number of digits when the object is printed.
which	If a subset of the plots is required, specify a subset of the numbers 1 : 4.
plotTitles	Titles to appear above the plots.
ask	Logical. If TRUE, the user is <i>asked</i> before each plot, see <code>par(ask = .)</code> .
...	Passes arguments to <code>par</code> , <code>hist</code> , <code>logHist</code> , <code>qqhyperb</code> and <code>pphyperb</code> .

## Details

`startMethod` can be either "BFGS" or "Nelder-Mead".

`startValues` can be one of the following:

- "US" User-supplied.
- "SL" Based on a fitted skew-Laplace distribution.
- "MoM" Method of moments.

For the details concerning the use of `paramStart`, `startMethod`, and `startValues`, see [vgFitStart](#).

The three optimisation methods currently available are:

- "BFGS" Uses the quasi-Newton method "BFGS" as documented in [optim](#).
- "Nelder-Mead" Uses an implementation of the Nelder and Mead method as documented in [optim](#).
- "nlm" Uses the `nlm` function in R.

For details of how to pass control information for optimisation using [optim](#) and `nlm`, see [optim](#) and `nlm`.

When `method = "Nelder-Mead"` is used, very rarely, it would return an error message of "error in `optim(paramStart,...)`", use `method = "BFGS"` or `method = "nlm"` instead in that case.

When `method = "nlm"` is used, warnings may be produced. These do not appear to be a problem.

**Value**

A list with components:

<code>param</code>	A vector giving the maximum likelihood estimate of param, as $(c, \sigma, \theta, \nu)$ .
<code>maxLik</code>	The value of the maximised log-likelihood.
<code>hessian</code>	If <code>hessian</code> was set to <code>TRUE</code> , the value of the hessian. Not present otherwise.
<code>method</code>	Optimisation method used.
<code>conv</code>	Convergence code. See the relevant documentation (either <code>optim</code> or <code>nlm</code> ) for details on convergence.
<code>iter</code>	Number of iterations of optimisation routine.
<code>obs</code>	The data used to fit the hyperbolic distribution.
<code>obsName</code>	A character string with the actual <code>obs</code> argument name.
<code>paramStart</code>	Starting value of param returned by call to <code>vgFitStart</code> .
<code>svName</code>	Descriptive name for the method finding start values.
<code>startValues</code>	Acronym for the method of finding start values.
<code>breaks</code>	The cell boundaries found by a call to <code>hist</code> .
<code>midpoints</code>	The cell midpoints found by a call to <code>hist</code> .
<code>empDens</code>	The estimated density found by a call to <code>hist</code> .

**Author(s)**

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

**References**

Seneta, E. (2004). Fitting the variance-gamma model to financial data. *J. Appl. Prob.*, 41A:177–187.

**See Also**

`optim`, `nlm`, `par`, `hist`, `logHist`, `qqvg`, `ppvg`, `dskewlap` and `vgFitStart`.

**Examples**

```
param <- c(0,0.5,0,0.5)
dataVector <- rvg(500, param = param)
## See how well vgFit works
vgFit(dataVector)
vgFit(dataVector, plots = TRUE)
fit <- vgFit(dataVector)
par(mfrow = c(1,2))
plot(fit, which = c(1,3))

## Use nlm instead of default
param <- c(0,0.5,0,0.5)
dataVector <- rvg(500, param = param)
```

```
vgFit(dataVector, method = "nlm", hessian = TRUE)

## Use BFGS instead of default
param <- c(0,0.5,0,0.5)
dataVector <- rvg(500, param = param)
vgFit(dataVector, method = "BFGS", hessian = TRUE)
```

---

vgFitStart

*Find Starting Values for Fitting a Variance Gamma Distribution*


---

### Description

Finds starting values for input to a maximum likelihood routine for fitting variance gamma distribution to data.

### Usage

```
vgFitStart(x, breaks = NULL, startValues = "SL", paramStart = NULL,
           startMethodSL = "Nelder-Mead",
           startMethodMoM = "Nelder-Mead", ...)
vgFitStartMoM(x, startMethodMoM = "Nelder-Mead", ...)
```

### Arguments

x	Data vector.
breaks	Breaks for histogram. If missing, defaults to those generated by <code>hist(x, right = FALSE, plot = FALSE)</code> .
startValues	Vector of the different starting values to consider. See <b>Details</b> .
paramStart	Starting values for param if <code>startValues = "US"</code> .
startMethodSL	Method used by call to <code>optim</code> in finding skew Laplace estimates.
startMethodMoM	Method used by call to <code>optim</code> in finding method of moments estimates.
...	Passes arguments to <code>optim</code> .

### Details

Possible values of the argument `startValues` are the following:

- "US" User-supplied.
- "SL" Based on a fitted skew-Laplace distribution.
- "MoM" Method of moments.

If `startValues = "US"` then a value must be supplied for `paramStart`.

If `startValues = "MoM"`, `vgFitStartMoM` is called. These starting values are based on Barndorff-Nielsen *et al* (1985).

If `startValues = "SL"`, or `startValues = "MoM"` an initial optimisation is needed to find the starting values. These optimisations call `optim`.

## Value

`vgFitStart` returns a list with components:

<code>vgStart</code>	A vector with elements <code>vgC</code> , <code>lSigma</code> (log of sigma), <code>theta</code> and <code>lNu</code> (log of nu) giving the starting value of <code>param</code> .
<code>xName</code>	A character string with the actual <code>x</code> argument name.
<code>breaks</code>	The cell boundaries found by a call to <code>hist</code> .
<code>midpoints</code>	The cell midpoints found by a call to <code>hist</code> .
<code>empDens</code>	The estimated density found by a call to <code>hist</code> .

`vgFitStartMoM` returns only the method of moments estimates as a vector with elements `vgC`, `lSigma` (log of sigma), `theta` and `lNu` (log of nu).

## Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

## References

Seneta, E. (2004). Fitting the variance-gamma model to financial data. *J. Appl. Prob.*, 41A:177–187.

## See Also

`dvg`, `dskewlap`, `vgFit`, `hist`, and `optim`.

## Examples

```
param <- c(0, 0.5, 0, 0.5)
dataVector <- rvg(500, param = param)
vgFitStart(dataVector, startValues="SL")
vgFitStartMoM(dataVector)
vgFitStart(dataVector, startValues="MoM")
```

vgMom

*Calculate Moments of the Variance Gamma Distribution***Description**

This function can be used to calculate raw moments, mu moments, central moments and moments about any other given location for the variance gamma (VG) distribution.

**Usage**

```
vgMom(order, vgC = 0, sigma = 1, theta = 0, nu = 1,
      param = c(vgC, sigma, theta, nu), momType = "raw", about = 0)
```

**Arguments**

order	Numeric. The order of the moment to be calculated. Not permitted to be a vector. Must be a positive whole number except for moments about zero.
vgC	The location parameter $c$ , default is 0.
sigma	The spread parameter $\sigma$ , default is 1, must be positive.
theta	The asymmetry parameter $\theta$ , default is 0.
nu	The shape parameter $\nu$ , default is 1, must be positive.
param	Specifying the parameters as a vector which takes the form <code>c(vgC, sigma, theta, nu)</code> .
momType	Common types of moments to be calculated, default is "raw". See <b>Details</b> .
about	Numeric. The point around which the moment is to be calculated, default is 0. See <b>Details</b> .

**Details**

For the parameters of the variance gamma distribution, users may either specify the values individually or as a vector. If both forms are specified but with different values, then the values specified by vector `param` will always overwrite the other ones. In addition, the parameters values are examined by calling the function `vgCheckParams` to see if they are valid for the VG distribution.

`order` is also checked by calling the function `is.wholenumber` in `DistributionUtils` package to see whether a whole number is given.

`momType` can be either "raw" (moments about zero), "mu" (moments about `vgC`), or "central" (moments about mean). If one of these moment types is specified, then there is no need to specify the `about` value. For moments about any other location, the `about` value must be specified. In the case that both `momType` and `about` are specified and contradicting, the function will always calculate the moments based on `about` rather than `momType`.

To calculate moments of the VG distribution, the function first calculates mu moments by the formula defined below and then transforms mu moments to central moments or raw moments or moments about any other location as required by calling `momChangeAbout` in `DistributionUtils` package.

To calculate mu moments of the variance gamma distribution, the function first transforms the parameterization of  $c, \sigma, \theta, \nu$  to the generalized hyperbolic distribution's parameterization of  $\lambda, \alpha, \beta, \mu$  (see [vgChangePars](#) for details). Then, the mu moments of the variance gamma distribution are given by

$$\sum_{\ell=\lfloor(k+1)/2\rfloor}^k a_{k,\ell} \beta^{2\ell-k} [\Gamma(\lambda + \ell)/\Gamma(\lambda) 2^\ell / (\alpha^2 - \beta^2)^\ell]$$

where  $k = \text{order}$  and  $k > 0$  and  $a_{k,\ell}$  is the recursive coefficient (see [momRecursion](#) for details).

This formula is developed from the mu moments formula of the generalized hyperbolic distribution given in Scott, Würtz and Tran (2008). Note that the part in  $[\ ]$  of this equation is actually equivalent to the formula of raw moments of the gamma distribution. So the function calls `gammaRawMom` in `GeneralizedHyperbolic` package when implementing the computations.

### Value

The moment specified. In the case of raw moments, `Inf` is returned if the moment is infinite.

### Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

### References

Paolella, Marc S. (2007) *Intermediate Probability: A Computational Approach*, Chichester: Wiley  
 Scott, D. J., Würtz, D. and Tran, T. T. (2008) *Moments of the Generalized Hyperbolic Distribution*. Preprint.

### See Also

[vgCheckPars](#), [vgChangePars](#), [vgMean](#), [vgVar](#), [vgSkew](#), [vgKurt](#), [is.wholenumber](#), [momRecursion](#), [momChangeAbout](#) and [momIntegrated](#).

### Examples

```
### Raw moments of the VG distribution
vgMom(3, param=c(2,1,2,1), momType = "raw")

### Mu moments of the VG distribution
vgMom(2, param=c(2,1,2,1), momType = "mu")

### Central moments of the VG distribution
vgMom(4, param=c(2,1,2,1), momType = "central")

### Moments about any locations
vgMom(4, param=c(2,1,2,1), about = 1)
```

**Description**

These objects store different parameter sets of the Variance Gamma distribution for testing or demonstrating purpose as matrixes. Specifically, the parameter sets `vgSmallShape` and `vgLargeShape` have constant (standard) location and spread parameters of  $c=0$  and  $\sigma=1$ ; where asymmetry and shape parameters vary from  $\theta=(-2, 0, 2)$  and  $\nu=(0.5, 1, 2)$  for `vgSmallShape` and  $\theta=(-4, -2, 0, 2, 4)$  and  $\nu=(0.25, 0.5, 1, 2, 4)$  for `vgLargeShape`.

The parameter sets `vgSmallParam` and `vgLargeParam` have varied values of all 4 parameters. `vgSmallParam` contains all of the parameter combinations from  $c=(-2, 0, 2)$ ,  $\sigma=(0.5, 1, 2)$ ,  $\theta=(-2, 0, 2)$  and  $\nu=(0.5, 1, 2)$ . `vgLargeParam` contains all of the parameter combinations from  $c=(-4, -2, 0, 2, 4)$ ,  $\sigma=(0.25, 0.5, 1, 2, 4)$ ,  $\theta=(-4, -2, 0, 2, 4)$  and  $\nu=(0.25, 0.5, 1, 2, 4)$ .

**Usage**

```
vgSmallShape
vgLargeShape
vgSmallParam
vgLargeParam
```

**Format**

`vgSmallShape`: a 9 by 4 matrix; `vgLargeShape`: a 25 by 4 matrix; `vgSmallParam`: a 81 by 4 matrix; `vgLargeParam`: a 625 by 4 matrix.

**Author(s)**

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

**Examples**

```
data(vgParam)
## Testing the accuracy of vgMean
for (i in 1:nrow(vgSmallParam)) {
  param <- vgSmallParam[i,]
  x <- rvg(10000,param = param)
  sampleMean <- mean(x)
  funMean <- vgMean(param = param)
  difference <- abs(sampleMean - funMean)
  print(difference)
}
```

# Index

## \*Topic **distribution**

- vgCheckPars, [14](#)
- vgMom, [20](#)
  
- besselK, [4](#)
  
- ddvg (*VarianceGammaDistribution*),  
[4](#)
- dskewlap, [18](#), [20](#)
- dvg, [4](#), [11](#), [12](#), [14](#), [15](#), [20](#)
- dvg (*VarianceGammaDistribution*), [4](#)
  
- hist, [18–20](#)
  
- integrate, [5](#), [6](#)
- is.wholenumber, [22](#)
  
- logHist, [18](#)
  
- momChangeAbout, [22](#)
- momIntegrated, [22](#)
- momRecursion, [21](#), [22](#)
  
- nlm, [2](#), [17](#), [18](#)
  
- optim, [2](#), [3](#), [16–20](#)
  
- par, [17](#), [18](#)
- plot.vgFit (*vgFit*), [15](#)
- ppoints, [11](#)
- ppv, [18](#)
- ppv (*VarianceGammaPlots*), [9](#)
- print.summary.vgFit  
(*summary.vgFit*), [2](#)
- print.vgFit, [2](#)
- print.vgFit (*vgFit*), [15](#)
- pv (*VarianceGammaDistribution*), [4](#)
  
- qqvg, [18](#)
- qqvg (*VarianceGammaPlots*), [9](#)
- qvg (*VarianceGammaDistribution*), [4](#)
  
- rvg (*VarianceGammaDistribution*), [4](#)
  
- safeIntegrate, [6](#)
- summary, [2](#)
- summary.vgFit, [2](#), [2](#)
  
- uniroot, [12](#)
  
- Variance Gamma Mean, Variance,  
Skewness, Kurtosis and  
Mode, [3](#)
- VarianceGammaDistribution, [4](#)
- VarianceGammaPlots, [9](#)
- vgBreaks  
(*VarianceGammaDistribution*),  
[4](#)
- vgCalcRange, [4](#), [7](#), [11](#)
- vgChangePars, [3](#), [4](#), [7](#), [11](#), [12](#), [13](#), [21](#), [22](#)
- vgCheckPars, [14](#), [22](#)
- vgFit, [2](#), [10](#), [15](#), [20](#)
- vgFitStart, [17](#), [18](#), [18](#)
- vgFitStartMoM (*vgFitStart*), [18](#)
- vgKurt, [22](#)
- vgKurt (*Variance Gamma Mean,  
Variance, Skewness,  
Kurtosis and Mode*), [3](#)
- vgLargeParam (*vgParam*), [22](#)
- vgLargeShape (*vgParam*), [22](#)
- vgMean, [22](#)
- vgMean (*Variance Gamma Mean,  
Variance, Skewness,  
Kurtosis and Mode*), [3](#)
- vgMode (*Variance Gamma Mean,  
Variance, Skewness,  
Kurtosis and Mode*), [3](#)
- vgMom, [14](#), [15](#), [20](#)
- vgParam, [22](#)
- vgSkew, [22](#)
- vgSkew (*Variance Gamma Mean,  
Variance, Skewness,  
Kurtosis and Mode*), [3](#)

`vgSmallParam(vgParam)`, [22](#)  
`vgSmallShape(vgParam)`, [22](#)  
`vgVar`, [22](#)  
`vgVar(Variance Gamma Mean,  
Variance, Skewness,  
Kurtosis and Mode)`, [3](#)