

# Package ‘mvtnorm’

May 9, 2026

**Title** Multivariate Normal and t Distributions

**Version** 1.3-7

**Date** 2026-04-14

**Description** Computes multivariate normal and t probabilities, quantiles, random deviates, and densities. Log-likelihoods for multivariate Gaussian models and Gaussian copulae parameterised by Cholesky factors of covariance or precision matrices are implemented for interval-censored and exact data, or a mix thereof. Score functions for these log-likelihoods are available. A class representing multiple lower triangular matrices and corresponding methods are part of this package.

**Imports** stats

**Depends** R(>= 3.5.0)

**Suggests** qrng, numDeriv, bibtex

**License** GPL-2

**URL** <http://mvtnorm.R-forge.R-project.org>

**NeedsCompilation** yes

**Author** Alan Genz [aut],  
Frank Bretz [aut],  
Tetsuhisa Miwa [aut],  
Xuefei Mi [aut],  
Friedrich Leisch [ctb],  
Fabian Scheipl [ctb],  
Bjoern Bornkamp [ctb] (ORCID: <<https://orcid.org/0000-0002-6294-8185>>),  
Martin Maechler [ctb] (ORCID: <<https://orcid.org/0000-0002-8685-9910>>),  
Torsten Hothorn [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-8301-0471>>)

**Maintainer** Torsten Hothorn <Torsten.Hothorn@R-project.org>

**Repository** CRAN

**Date/Publication** 2026-04-15 07:10:13 UTC

## Contents

mvtnorm-package . . . . .	2
algorithms . . . . .	3
interface . . . . .	4
lpmvnorm . . . . .	6
lpRR . . . . .	8
ltMatrices . . . . .	10
margcond . . . . .	13
Mvnorm . . . . .	14
Mvt . . . . .	16
pmvnorm . . . . .	18
pmvt . . . . .	21
qmvnorm . . . . .	24
qmvt . . . . .	26
<b>Index</b>	<b>29</b>

---

mvtnorm-package	<i>Multivariate Normal and t Distributions</i>
-----------------	--

---

## Description

Computes multivariate normal and t probabilities, quantiles, random deviates, and densities. Log-likelihoods for multivariate Gaussian models and Gaussian copulae parameterised by Cholesky factors of covariance or precision matrices are implemented for interval-censored and exact data, or a mix thereof. Score functions for these log-likelihoods are available. A class representing multiple lower triangular matrices and corresponding methods are part of this package.

## Details

Package **mvtnorm** provides functionality for dealing with multivariate normal and t-distributions. The package interfaces FORTRAN and C code for evaluating multivariate normal probabilities written by Alan Genz and Tetsuhisa Miwa. Functions `pmvnorm`, `pmvt`, `qmvnorm`, and `qmvt` return normal and t probabilities or corresponding quantiles computed by these original implementations. Users interested in the computation of such probabilities or quantiles, for example for multiple testing purposes, should use this functionality.

When the multivariate normal log-likelihood function, defined by the log-probability in the discrete or interval-censored case or by the log-density for exact real observations, or a mix thereof, shall be computed, functions `lpmvnorm`, `ldmvnorm`, and `ldpmvnorm` are better suited. They rely on an independent implementation of Genz' algorithm (for log-probabilities), can be customised (different quasi-Monte Carlo schemes), and are a bit faster. Most importantly, the corresponding score functions are available through functions `slpmvnorm`, `sldmvnorm`, or `sldpmvnorm`, which help to speed-up parameter estimation considerably. Users interested in this functionality should consult the `lpmvnorm_src` package vignette.

**See Also**

```
vignette("lmvnorm_src", package = "mvtnorm")
```

algorithms

*Choice of Algorithm and Hyper Parameters***Description**

Choose between three algorithms for evaluating normal (and t-) distributions and define hyper parameters.

**Usage**

```
GenzBretz(maxpts = 25000, abseps = 0.001, releps = 0)
```

```
Miwa(steps = 128, checkCorr = TRUE, maxval = 1e3)
```

```
TVPACK(abseps = 1e-6)
```

**Arguments**

maxpts	maximum number of function values as integer. The internal FORTRAN code always uses a minimum number depending on the dimension. (for example 752 for three-dimensional problems).
abseps	absolute error tolerance; for TVPACK only used for dimension 3.
releps	relative error tolerance as double.
steps	number of grid points to be evaluated; cannot be larger than 4097.
checkCorr	logical indicating if a check for singularity of the correlation matrix should be performed (once per function call to <code>pmvt()</code> or <code>pmvnorm()</code> ).
maxval	replacement for <code>Inf</code> when non-orthant probabilities involving <code>Inf</code> shall be computed.

**Details**

There are three algorithms available for evaluating normal (and two algorithms for t-) probabilities: The default is the randomized Quasi-Monte-Carlo procedure by Genz (1992); Genz (1993) and Genz and Bretz (2002) applicable to arbitrary covariance structures and dimensions up to 1000.

For normal probabilities, smaller dimensions (up to 20) and non-singular covariance matrices, the algorithm by Miwa, Hayter, and Kuriki (2003) can be used as well. This algorithm can compute orthant probabilities (lower being `-Inf` or upper equal to `Inf`). Non-orthant probabilities are computed from the corresponding orthant probabilities, however, infinite limits are replaced by `maxval` along with a warning.

For two- and three-dimensional problems and semi-infinite integration region, TVPACK implements an interface to the methods described by Genz (2004).

**Value**

An object of class `"GenzBretz"`, `"Miwa"`, or `"TVPACK"` defining hyper parameters.

## References

- Genz A (1992). “Numerical Computation of Multivariate Normal Probabilities.” *Journal of Computational and Graphical Statistics*, **1**(2), 141–149. doi:10.1080/10618600.1992.10477010.
- Genz A (1993). “Comparison of Methods for the Computation of Multivariate Normal Probabilities.” *Computing Science and Statistics*, **25**, 400–405.
- Genz A (2004). “Numerical Computation of Rectangular Bivariate and Trivariate Normal and  $t$  Probabilities.” *Statistics and Computing*, **14**(3), 251–260. doi:10.1023/B:STCO.0000035304.20635.31.
- Genz A, Bretz F (2002). “Methods for the Computation of Multivariate  $t$  Probabilities.” *Journal of Computational and Graphical Statistics*, **11**(4), 950–971. doi:10.1198/106186002394.
- Miwa T, Hayter AJ, Kuriki S (2003). “The Evaluation of General Non-centred Orthant Probabilities.” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **65**(1), 223–234. doi:10.1111/14679868.00382.

---

interface

*User Interface to Multiple Multivariate Normal Distributions*

---

## Description

A simple user interface for computing on multiple multivariate normal distributions.

## Usage

```

mvnorm(mean, invcholmean, chol, invchol)
## S3 method for class 'mvnorm'
aperm(a, perm, ...)
margDist(object, which, ...)
## S3 method for class 'mvnorm'
margDist(object, which, ...)
condDist(object, which_given, given, ...)
## S3 method for class 'mvnorm'
condDist(object, which_given, given, ...)
## S3 method for class 'mvnorm'
simulate(object, nsim = dim(object$scale)[1L], seed = NULL,
          standardize = FALSE, as.data.frame = FALSE, ...)
## S3 method for class 'mvnorm'
logLik(object, obs, lower, upper, standardize = FALSE, ...)
## S3 method for class 'mvnorm'
llgrad(object, obs, lower, upper, standardize = FALSE, ...)

```

## Arguments

- `chol` either an `ltMatrices` object specifying (multiple) Cholesky factors of the covariance matrix or one single numeric lower triangular square matrix.
- `invchol` either an `ltMatrices` object specifying (multiple) inverse Cholesky factors of the covariance matrix or one single numeric lower triangular square matrix.

<code>a</code> , object	objects of class <code>mvnorm</code> .
<code>perm</code>	a permutation of the covariance matrix corresponding to <code>a</code> .
<code>which</code>	names or indices of elements whose marginal distribution is of interest.
<code>which_given</code>	names or indices of elements to condition on.
<code>given</code>	matrix of realisations to condition on (number of rows is equal to <code>length(which)</code> ), the number of columns corresponds to the number of matrices in <code>chol</code> or <code>invchol</code> .
<code>lower</code>	matrix of lower limits (one column for each observation, $J$ rows).
<code>upper</code>	matrix of upper limits (one column for each observation, $J$ rows).
<code>obs</code>	matrix of exact observations (one column for each observation, $J$ rows).
<code>mean</code>	matrix of means (one column for each observation, length is recycled to length of <code>obs</code> , <code>lower</code> and <code>upper</code> ).
<code>invcholmean</code>	matrix of scaled means, that is, <code>invchol %*% mean</code> , one column for each observation, length is recycled to length of <code>obs</code> , <code>lower</code> and <code>upper</code> .
<code>seed</code>	an object specifying if and how the random number generator should be initialized, see <a href="#">simulate</a> .
<code>standardize</code>	logical, should the Cholesky factor (or its inverse) undergo standardization (ensuring the covariance matrix is a correlation matrix) before computing the likelihood.
<code>nsim</code>	number of samples to draw.
<code>as.data.frame</code>	logical, convert the $J \times N$ matrix result to a classical $N \times J$ data frame.
<code>...</code>	Additional arguments to <code>ldpmvnorm</code> and <code>sldpmvnorm</code>

## Details

The constructor `mvnorm` can be used to specify (multiple) multivariate normal distributions. `margDist` derives marginal and `condDist` conditional distributions from such objects. A `simulate` method exists for drawn samples from multivariate normals.

The continuous (data in `obs`), discrete (intervals in `lower` and `upper`), and mixed continuous-discrete log-likelihood is implemented in `logLik`. The corresponding gradients with respect to all model parameters and with respect to the data arguments is available from `lLgrad`.

Rationals and examples are given in Chapter 7 of the package vignette linked to below.

## Value

`mvnorm`, `margDist`, and `condDist` return objects of class `mvnorm`. `logLik` returns the log-likelihood and `lLgrad` a list with gradients.

## See Also

`vignette("lmvnorm_src", package = "mvtnorm")`

lpmvnorm

*Multivariate Normal Log-likelihood and Score Functions***Description**

Computes the log-likelihood (contributions) of multiple exact or interval-censored observations (or a mix thereof) from multivariate normal distributions and evaluates corresponding score functions.

**Usage**

```
lpmvnorm(lower, upper, mean, invcholmean, center = NULL, chol, invchol, logLik = TRUE,
          M = NULL, w = NULL, seed = NULL, tol = .Machine$double.eps, fast = FALSE)
slpmvnorm(lower, upper, mean, invcholmean, center = NULL, chol, invchol, logLik = TRUE,
           M = NULL, w = NULL, seed = NULL, tol = .Machine$double.eps, fast = FALSE)
ldmvnorm(obs, mean, invcholmean, chol, invchol, logLik = TRUE)
sldmvnorm(obs, mean, invcholmean, chol, invchol, logLik = TRUE)
ldpmvnorm(obs, lower, upper, mean, invcholmean, chol, invchol, logLik = TRUE, ...)
sldpmvnorm(obs, lower, upper, mean, invcholmean, chol, invchol, logLik = TRUE, ...)
```

**Arguments**

lower	matrix of lower limits (one column for each observation, $J$ rows).
upper	matrix of upper limits (one column for each observation, $J$ rows).
obs	matrix of exact observations (one column for each observation, $J$ rows).
mean	matrix of means (one column for each observation, length is recycled to length of obs, lower and upper).
invcholmean	matrix of means left-multiplied with inverse Cholesky factor ( <code>invchol %*% mean</code> , one column for each observation, length is recycled to length of obs, lower and upper).
center	matrix of negative rescaled means (one column for each observation, length is recycled to length of lower and upper) as returned by <code>cond_mvnorm(..., center = TRUE)</code> .
chol	Cholesky factors of covariance matrices as <code>ltMatrices</code> object, length is recycled to length of obs, lower and upper.
invchol	Cholesky factors of precision matrices as <code>ltMatrices</code> object, length is recycled to length of lower and upper. Either <code>chol</code> or <code>invchol</code> must be given.
logLik	logical, if <code>TRUE</code> , the log-likelihood is returned, otherwise the individual contributions to the sum are returned.
M	number of iterations, early stopping based on estimated errors is NOT implemented.
w	an optional matrix of weights with $J - 1$ rows. This allows to replace the default Monte-Carlo procedure (Genz 1992) with a quasi-Monte-Carlo approach (Genz and Bretz 2002). Note that the same weights for evaluating the multivariate normal probability are used for all observations when <code>ncol(w) == M</code> is specified.

	If <code>ncol(w) == ncol(lower) * M</code> , each likelihood contribution is evaluated on the corresponding sub-matrix. If <code>w</code> is <code>NULL</code> , different uniform numbers are drawn for each observation.
<code>seed</code>	an object specifying if and how the random number generator should be initialized, see <a href="#">simulate</a> . Only applied when <code>w</code> is <code>NULL</code> .
<code>tol</code>	tolerance limit, values smaller than <code>tol</code> are interpreted as zero.
<code>fast</code>	logical, if <code>TRUE</code> , a faster but less accurate version of <code>pnorm</code> is used internally.
<code>...</code>	additional arguments to <code>lpmvnorm</code> .

### Details

Evaluates the multivariate normal log-likelihood defined by means and `chol` over boxes defined by `lower` and `upper` or for exact observations `obs`.

Monte-Carlo (Genz 1992, the default) and quasi-Monte-Carlo (Genz and Bretz 2002) integration is implemented, the latter with weights obtained, for example, from packages **qrng** or **randtoolbox**. It is the responsibility of the user to ensure a meaningful lattice is used. In case of doubt, use plain Monte-Carlo (`w = NULL`) or [pmvnorm](#).

`slpmvnorm` computes both the individual log-likelihood contributions and the corresponding score matrix (of dimension  $J \times (J + 1)/2 \times N$ ) if `chol` contains diagonal elements. Otherwise, the dimension is  $J \times (J - 1)/2 \times N$ . The scores for exact or mixed exact-interval observations are computed by `sldmvnorm` and `sldpmvnorm`, respectively.

More details can be found in the `lmvnorm_src` package vignette.

### Value

The log-likelihood (`logLik = TRUE`) or the individual contributions to the log-likelihood. `slpmvnorm`, `sldmvnorm`, and `sldpmvnorm` return the score matrices and, optionally (`logLik = TRUE`), the individual log-likelihood contributions as well as scores for `obs`, `lower`, `upper`, and `mean`.

### References

Genz A (1992). "Numerical Computation of Multivariate Normal Probabilities." *Journal of Computational and Graphical Statistics*, **1**(2), 141–149. doi:[10.1080/10618600.1992.10477010](https://doi.org/10.1080/10618600.1992.10477010).

Genz A, Bretz F (2002). "Methods for the Computation of Multivariate *t* Probabilities." *Journal of Computational and Graphical Statistics*, **11**(4), 950–971. doi:[10.1198/106186002394](https://doi.org/10.1198/106186002394).

### See Also

[dmvnorm](#), `vignette("lmvnorm_src", package = "mvtnorm")`

### Examples

```
### five observations
N <- 5L
### dimension
J <- 4L

### lower and upper bounds, ie interval-censoring
```

```

lwr <- matrix(-runif(N * J), nrow = J)
upr <- matrix(runif(N * J), nrow = J)

### Cholesky factor
(C <- ltMatrices(runif(J * (J + 1) / 2), diag = TRUE))
### corresponding covariance matrix
(S <- as.array(Tcrossprod(C))[, , 1])

### plain Monte-Carlo (Genz, 1992)
w <- NULL
M <- 25000
### quasi-Monte-Carlo (Genz & Bretz, 2002, but with different weights)
if (require("qrng")) w <- t(ghalton(M * N, J - 1))

### log-likelihood
lpmvnorm(lower = lwr, upper = upr, chol = C, w = w, M = M)

### compare with pmvnorm
exp(lpmvnorm(lower = lwr, upper = upr, chol = C, logLik = FALSE, w = w, M = M))
sapply(1:N, function(i) pmvnorm(lower = lwr[,i], upper = upr[,i], sigma = S))

### log-lik contributions and score matrix
slpmvnorm(lower = lwr, upper = upr, chol = C, w = w, M = M, logLik = TRUE)

```

lpRR

*Multivariate Normal Log-likelihood and Score Functions for Reduced Rank Covariances*

### Description

Computes the log-likelihood (contributions) of interval-censored observations from multivariate normal distributions with reduced rank structure and evaluates corresponding score functions.

### Usage

```

lpRR(lower, upper, mean = 0, B, D = rep(1, nrow(B)),
      Z, weights = 1 / ncol(Z), log.p = TRUE)
slpRR(lower, upper, mean = 0, B, D = rep(1, nrow(B)),
       Z, weights = 1 / ncol(Z), log.p = TRUE)

```

### Arguments

lower	vector of lower limits (one element for each dimension, $J$ elements).
upper	vector of upper limits (one element for each dimension, $J$ elements).
mean	vector of means (one element for each dimension, length is recycled to length of lower and upper).
B	matrix of dimension $J \times K$ .

D	vector of $J$ diagonal elements.
Z	matrix of standard normal random variables, with $K$ nrow.
weights	optional weights.
log.p	logical. By default, log-probabilities are returned.

### Details

Evaluates the multivariate normal log-likelihood defined by mean, B and D when the covariance is  $\Sigma = BB^T + D$  over boxes defined by lower and upper. Details are given in Genz and Bretz (2009), Chapter 2.3.1.

s1pRR computes the corresponding score functions with respect to lower, upper, mean, B and D.

More details can be found in the lmvnorm\_src package vignette.

### Value

The log-likelihood (log.p = TRUE) or corresponding probability. s1pRR return the scores.

### References

Genz A, Bretz F (2009). *Computation of Multivariate Normal and t Probabilities*, series Lecture Notes in Statistics. Springer-Verlag, Heidelberg, Germany. ISBN 978-3-642-01688-2.

### See Also

vignette("lmvnorm\_src", package = "mvtnorm")

### Examples

```
J <- 6
K <- 3
B <- matrix(rnorm(J * K), nrow = J)
D <- runif(J)
S <- tcrossprod(B) + diag(D)
a <- -(2 + runif(J))
b <- 2 + runif(J)
M <- 1e4
Z <- matrix(rnorm(K * M), nrow = K)
## log-likelihood
lpRR(lower = a, upper = b, B = B, D = D, Z = Z)
## score wrt all arguments
s1pRR(lower = a, upper = b, B = B, D = D, Z = Z)
```

**Description**

A class representing multiple lower triangular or symmetric matrices and some methods.

**Usage**

```
ltMatrices(object, diag = FALSE, byrow = FALSE, names = TRUE)
syMatrices(object, diag = FALSE, byrow = FALSE, names = TRUE)
## S3 method for class 'ltMatrices'
as.array(x, symmetric = FALSE, ...)
## S3 method for class 'syMatrices'
as.array(x, ...)
## S3 method for class 'ltMatrices'
diagonals(x, ...)
## S3 method for class 'syMatrices'
diagonals(x, ...)
## S3 method for class 'matrix'
diagonals(x, ...)
## S3 method for class 'integer'
diagonals(x, ...)
diagonals(x) <- value
## S3 replacement method for class 'ltMatrices'
diagonals(x) <- value
## S3 replacement method for class 'syMatrices'
diagonals(x) <- value
## S3 method for class 'ltMatrices'
solve(a, b, transpose = FALSE, ...)
## S3 method for class 'syMatrices'
chol(x, ...)
## S3 method for class 'chol'
aperm(a, perm, ...)
## S3 method for class 'invchol'
aperm(a, perm, ...)
## S3 method for class 'ltMatrices'
aperm(a, perm, ...)
## S3 method for class 'syMatrices'
aperm(a, perm, ...)
deperma(chol = solve(invchol), permuted_chol = solve(permuted_invchol),
        invchol, permuted_invchol, perm, score_schol)
## S3 method for class 'ltMatrices'
Mult(x, y, transpose = FALSE, ...)
## S3 method for class 'syMatrices'
Mult(x, y, ...)
Tcrossprod(x, diag_only = FALSE)
```

```

Crossprod(x, diag_only = FALSE)
## S3 method for class 'ltMatrices'
tcrossprod(x, y = NULL, ...)
## S3 method for class 'syMatrices'
tcrossprod(x, y = NULL, ...)
## S3 method for class 'ltMatrices'
crossprod(x, y = NULL, ...)
## S3 method for class 'syMatrices'
crossprod(x, y = NULL, ...)
logdet(x)
Lower_tri(x, diag = FALSE, byrow = attr(x, "byrow"))
is.ltMatrices(x)
is.syMatrices(x)
as.ltMatrices(x)
## S3 method for class 'ltMatrices'
as.ltMatrices(x)
## S3 method for class 'syMatrices'
as.ltMatrices(x)
as.syMatrices(x)
is.chol(x)
is.invchol(x)
as.chol(x)
as.invchol(x)
chol2cov(x)
invchol2chol(x)
chol2invchol(x)
invchol2cov(x)
invchol2pre(x)
chol2pre(x)
Dchol(x, D = 1 / sqrt(Tcrossprod(x, diag_only = TRUE)))
invcholD(x, D = sqrt(Tcrossprod(solve(x), diag_only = TRUE)))
chol2cor(x)
invchol2cor(x)
chol2pc(x)
invchol2pc(x)
vectrick(C, S, A, transpose = c(TRUE, TRUE))
standardize(chol, invchol)
destandardize(chol = solve(invchol), invchol, score_schol)
as.ltMatrices(x)

```

### Arguments

object	a matrix representing the lower triangular elements of $N$ lower triangular matrix, each of dimension $J \times J$ . Dimensions of object depend on <code>diag</code> : With diagonal elements, object is a $J(J + 1)/2 \times N$ matrix, otherwise, the number of rows is $J(J - 1)/2$ .
diag	logical, object contains diagonal elements if TRUE, otherwise unit diagonal elements are assumed.

byrow	logical, object represents matrices in row-major order if TRUE or, otherwise, in column-major order.
names	logical or character vector of length $J$ .
symmetric	logical, object is interpreted as a symmetric matrix if TRUE.
diag_only	logical, compute diagonal elements of crossproduct only if TRUE.
x, chol, invchol, permuted_chol, permuted_invchol	object of class <code>ltMatrices</code> or <code>syMatrices</code> (for <code>chol</code> ).
value	a matrix of diagonal elements to be assigned (of dimension $J \times N$ ).
a	object of class <code>ltMatrices</code> .
perm	a permutation of the covariance matrix corresponding to a.
D	a matrix (of dimension $J \times N$ ) of diagonal elements to be multiplied with.
y	matrix with $J$ rows.
b	matrix with $J$ rows.
C	an object of class <code>ltMatrices</code> .
S	an object of class <code>ltMatrices</code> or a matrix with $J^2$ rows representing multiple $J \times J$ matrices (columns of vec operators).
A	an object of class <code>ltMatrices</code> .
transpose	a logical of length two indicating if A or B shall be transposed in <code>vectrick</code> . For <code>solve</code> , this argument being true computes <code>solve(t(a), b)</code> (in absence of a <code>t()</code> method for <code>ltMatrices</code> objects).
score_schol	score matrix for a standardized <code>chol</code> object.
...	additional arguments, currently ignored.

### Details

`ltMatrices` interprets a matrix as lower triangular elements of multiple lower triangular matrices. The corresponding class can be used to store such matrices efficiently. Matrix multiplications, solutions to linear systems, explicit inverses, and crossproducts can be computed based on such objects. Details can be found in the `lmvnorm_src` package vignette.

`syMatrices` only store the lower triangular parts of multiple symmetric matrices.

### Value

The constructor `ltMatrices` returns objects of class `ltMatrices` with corresponding methods. The constructor `syMatrices` returns objects of class `syMatrices` with a reduced set of methods.

### See Also

`vignette("lmvnorm_src", package = "mvtnorm")`

**Examples**

```

J <- 4L
N <- 2L
dm <- paste0("d", 1:J)
xm <- paste0("x", 1:N)
(C <- ltMatrices(matrix(runif(N * J * (J + 1) / 2),
                        ncol = N, dimnames = list(NULL, xm)),
                  diag = TRUE, names = dm))

## dimensions and names
dim(C)
dimnames(C)
names(C)

## subset
C[,2:3]

## multiplication
y <- matrix(runif(N * J), nrow = J)
Mult(C, y)
C

## solve
solve(C)
solve(C, y)

## tcrossprod
Tcrossprod(C)
tcrossprod(C)

## convert to matrix
as.array(solve(C[1,]))[,1]

```

---

margcond

---

*Marginal and Conditional Multivariate Normal Distributions*


---

**Description**

Computes means and Cholesky factors of covariance or precision matrices of multiple multivariate normal distributions.

**Usage**

```

marg_mvnorm(chol, invchol, which = 1L)
cond_mvnorm(chol, invchol, which_given = 1L, given, center = FALSE)

```

**Arguments**

<code>chol</code>	Cholesky factors of covariance matrices as <code>ltMatrices</code> object, length is recycled to length of lower and upper.
<code>invchol</code>	Cholesky factors of precision matrices as <code>ltMatrices</code> object, length is recycled to length of lower and upper. Either <code>chol</code> or <code>invchol</code> must be given.
<code>which</code>	names or indices of elements those marginal distribution is of interest.
<code>which_given</code>	names or indices of elements to condition on.
<code>given</code>	matrix of realisations to condition on (number of rows is equal to <code>length(which)</code> ), the number of columns corresponds to the number of matrices in <code>chol</code> or <code>invchol</code> .
<code>center</code>	logical, if TRUE, the negative rescaled conditional mean is returned (such that it can be specified as <code>center</code> argument to <code>slpmvnorm</code> ). By default, the conditional mean is returned.

**Details**

Derives parameters of the requested marginal or conditional distributions, defined by `chol` (Cholesky factor of covariance) or `invchol` (Cholesky factor of precision matrix) and, for conditional distributions, the mean.

More details can be found in the `lmvnorm_src` package vignette.

**Value**

A named list.

**See Also**

`vignette("lmvnorm_src", package = "mvtnorm")`

---

Mvnorm

*Multivariate Normal Density and Random Deviates*


---

**Description**

These functions provide the density function and a random number generator for the multivariate normal distribution with mean equal to `mean` and covariance matrix `sigma`.

**Usage**

```
dmvnorm(x, mean = rep(0, p), sigma = diag(p), log = FALSE, checkSymmetry = TRUE)
rmvnorm(n, mean = rep(0, nrow(sigma)), sigma = diag(length(mean)),
  method=c("eigen", "svd", "chol"), pre0.9_9994 = FALSE,
  checkSymmetry = TRUE, rnorm = stats::rnorm)
```

**Arguments**

<code>x</code>	vector or matrix of quantiles. When <code>x</code> is a matrix, each row is taken to be a quantile and columns correspond to the number of dimensions, <code>p</code> .
<code>n</code>	number of observations.
<code>mean</code>	mean vector, default is <code>rep(0, length = ncol(x))</code> . In <code>ldmvnorm</code> or <code>sldmvnorm</code> , <code>mean</code> is a matrix with observation-specific means arranged in columns.
<code>sigma</code>	covariance matrix, default is <code>diag(ncol(x))</code> .
<code>log</code>	logical; if TRUE, densities <code>d</code> are given as <code>log(d)</code> .
<code>method</code>	string specifying the matrix decomposition used to determine the matrix root of <code>sigma</code> . Possible methods are eigenvalue decomposition (" <code>eigen</code> ", default), singular value decomposition (" <code>svd</code> "), and Cholesky decomposition (" <code>chol</code> "). The Cholesky is typically fastest, not by much though.
<code>pre0.9_9994</code>	logical; if FALSE, the output produced in <code>mvtnorm</code> versions up to 0.9-9993 is reproduced. In 0.9-9994, the output is organized such that <code>rmvnorm(10, ...)</code> has the same first ten rows as <code>rmvnorm(100, ...)</code> when called with the same seed.
<code>checkSymmetry</code>	logical; if FALSE, skip checking whether the covariance matrix is symmetric or not. This will speed up the computation but may cause unexpected outputs when ill-behaved <code>sigma</code> is provided. The default value is TRUE.
<code>rnorm</code>	a function with the same interface as <code>rnorm</code> . This allows switching to other generators of standard normal variables.

**Details**

`dmvnorm` computes the density function of the multivariate normal specified by `mean` and the covariance matrix `sigma`.

`rmvnorm` generates multivariate normal variables.

**See Also**

[pmvnorm](#), [rnorm](#), [qmvnorm](#), `vignette("ldmvnorm_src", package = "mvtnorm")`

**Examples**

```
dmvnorm(x=c(0,0))
dmvnorm(x=c(0,0), mean=c(1,1))

sigma <- matrix(c(4,2,2,3), ncol=2)
x <- rmvnorm(n=500, mean=c(1,2), sigma=sigma)
colMeans(x)
var(x)
dS <- dmvnorm(x, sigma = sigma)

### alternative interface
C <- t(chol(sigma))
(C <- ltMatrices(C[lower.tri(C, diag = TRUE)], diag = TRUE))
dC <- exp(ldmvnorm(obs = t(x), chol = C, logLik = FALSE))
```

```

all.equal(dS, dC)

x <- rmvnorm(n=500, mean=c(1,2), sigma=sigma, method="chol")
colMeans(x)
var(x)

plot(x)

```

Mvt

*The Multivariate t Distribution***Description**

These functions provide information about the multivariate  $t$  distribution with non-centrality parameter (or mode)  $\delta$ , scale matrix  $\sigma$  and degrees of freedom  $df$ . `dmvt` gives the density and `rmvt` generates random deviates.

**Usage**

```

rmvt(n, sigma = diag(2), df = 1, delta = rep(0, nrow(sigma)),
     type = c("shifted", "Kshirsagar"), ...)
dmvt(x, delta = rep(0, p), sigma = diag(p), df = 1, log = TRUE,
     type = "shifted", checkSymmetry = TRUE)

```

**Arguments**

<code>x</code>	vector or matrix of quantiles. If <code>x</code> is a matrix, each row is taken to be a quantile.
<code>n</code>	number of observations.
<code>delta</code>	the vector of noncentrality parameters of length <code>n</code> , for <code>type = "shifted"</code> <code>delta</code> specifies the mode.
<code>sigma</code>	scale matrix, defaults to <code>diag(ncol(x))</code> .
<code>df</code>	degrees of freedom. <code>df = 0</code> or <code>df = Inf</code> corresponds to the multivariate normal distribution.
<code>log</code>	<b>logical</b> indicating whether densities $d$ are given as $\log(d)$ .
<code>type</code>	type of the noncentral multivariate $t$ distribution. The choice <code>type = "Kshirsagar"</code> corresponds to formula (1.4) in Genz and Bretz (2009), see also Chapter 5.1 in Kotz and Nadarajah (2004). This is the noncentral $t$ -distribution needed for calculating the power of multiple contrast tests under a normality assumption. <code>type = "shifted"</code> corresponds to the formula right before formula (1.4) in Genz and Bretz (2009) (see also formula (1.1) in Kotz and Nadarajah 2004). It is a location shifted version of the central $t$ -distribution. This noncentral multivariate $t$ distribution appears for example as the Bayesian posterior distribution for the regression coefficients in a linear regression. In the central case both types coincide. Note that the defaults differ from the default in <code>pmvt()</code> (for reasons of backward compatibility).

checkSymmetry logical; if FALSE, skip checking whether the covariance matrix is symmetric or not. This will speed up the computation but may cause unexpected outputs when ill-behaved sigma is provided. The default value is TRUE.

... additional arguments to `rmvnorm()`, for example method.

### Details

If  $\mathbf{X}$  denotes a random vector following a  $t$  distribution with location vector  $\mathbf{0}$  and scale matrix  $\Sigma$  (written  $X \sim t_\nu(\mathbf{0}, \Sigma)$ ), the scale matrix (the argument `sigma`) is not equal to the covariance matrix  $Cov(\mathbf{X})$  of  $\mathbf{X}$ . If the degrees of freedom  $\nu$  (the argument `df`) is larger than 2, then  $Cov(\mathbf{X}) = \Sigma\nu/(\nu - 2)$ . Furthermore, in this case the correlation matrix  $Cor(\mathbf{X})$  equals the correlation matrix corresponding to the scale matrix  $\Sigma$  (which can be computed with `cov2cor()`). Note that the scale matrix is sometimes referred to as “dispersion matrix”; see McNeil, Frey, and Embrechts (2005), p. 74.

For type = "shifted" the density

$$c(1 + (x - \delta)'S^{-1}(x - \delta)/\nu)^{-(\nu+m)/2}$$

is implemented, where

$$c = \Gamma((\nu + m)/2)/((\pi\nu)^{m/2}\Gamma(\nu/2)|S|^{1/2}),$$

$S$  is a positive definite symmetric matrix (the matrix `sigma` above),  $\delta$  is the non-centrality vector and  $\nu$  are the degrees of freedom.

`df=0` historically leads to the multivariate normal distribution. From a mathematical point of view, rather `df=Inf` corresponds to the multivariate normal distribution. This is (now) also allowed for `rmvt()` and `dmvt()`.

Note that `dmvt()` has default `log = TRUE`, whereas `dmvnorm()` has default `log = FALSE`.

### References

- Genz A, Bretz F (2009). *Computation of Multivariate Normal and t Probabilities*, series Lecture Notes in Statistics. Springer-Verlag, Heidelberg, Germany. ISBN 978-3-642-01688-2.
- Kotz S, Nadarajah S (2004). *Multivariate t Distributions and Their Applications*. Cambridge University Press, Cambridge, UK.
- McNeil AJ, Frey R, Embrechts P (2005). *Quantitative Risk Management: Concepts, Techniques, Tools*. Princeton University Press, Princeton, New Jersey, U.S.A.

### See Also

`pmvt()` and `qmvt()`

### Examples

```
## basic evaluation
dmvt(x = c(0,0), sigma = diag(2))

## check behavior for df=0 and df=Inf
x <- c(1.23, 4.56)
```

```

mu <- 1:2
Sigma <- diag(2)
x0 <- dmvt(x, delta = mu, sigma = Sigma, df = 0) # default log = TRUE!
x8 <- dmvt(x, delta = mu, sigma = Sigma, df = Inf) # default log = TRUE!
xn <- dmvt(x, mean = mu, sigma = Sigma, log = TRUE)
stopifnot(identical(x0, x8), identical(x0, xn))

## X ~ t_3(0, diag(2))
x <- rmvt(100, sigma = diag(2), df = 3) # t_3(0, diag(2)) sample
plot(x)

## X ~ t_3(mu, Sigma)
n <- 1000
mu <- 1:2
Sigma <- matrix(c(4, 2, 2, 3), ncol=2)
set.seed(271)
x <- rep(mu, each=n) + rmvt(n, sigma=Sigma, df=3)
plot(x)

## Note that the call rmvt(n, mean=mu, sigma=Sigma, df=3) does *not*
## give a valid sample from t_3(mu, Sigma)! [and thus throws an error]
try(rmvt(n, mean=mu, sigma=Sigma, df=3))

## df=Inf correctly samples from a multivariate normal distribution
set.seed(271)
x <- rep(mu, each=n) + rmvt(n, sigma=Sigma, df=Inf)
set.seed(271)
x. <- rmvnorm(n, mean=mu, sigma=Sigma)
stopifnot(identical(x, x.))

```

---

pmvnorm

*Multivariate Normal Distribution*


---

### Description

Computes the distribution function of the multivariate normal distribution for arbitrary limits and correlation matrices.

### Usage

```

pmvnorm(lower=-Inf, upper=Inf, mean=rep(0, length(lower)),
        corr=NULL, sigma=NULL, algorithm = GenzBretz(), keepAttr=TRUE,
        seed = NULL, ...)

```

### Arguments

lower	the vector of lower limits of length n.
upper	the vector of upper limits of length n.
mean	the mean vector of length n.

corr	the correlation matrix of dimension $n$ .
sigma	the covariance matrix of dimension $n$ less than 1000. Either <code>corr</code> or <code>sigma</code> can be specified. If <code>sigma</code> is given, the problem is standardized internally. If <code>corr</code> is given, it is assumed that appropriate standardization was performed by the user. If neither <code>corr</code> nor <code>sigma</code> is given, the identity matrix is used for <code>sigma</code> .
algorithm	an object of class <code>GenzBretz</code> , <code>Miwa</code> or <code>TVPACK</code> specifying both the algorithm to be used as well as the associated hyper parameters.
keepAttr	<code>logical</code> indicating if <code>attributes</code> such as <code>error</code> and <code>msg</code> should be attached to the return value. The default, <code>TRUE</code> is back compatible.
seed	an object specifying if and how the random number generator should be initialized, see <code>simulate</code> .
...	additional parameters (currently given to <code>GenzBretz</code> for backward compatibility issues).

## Details

This program involves the computation of multivariate normal probabilities with arbitrary correlation matrices. It involves both the computation of singular and nonsingular probabilities. The implemented methodology is described in Genz (1992) and Genz (1993) for algorithm `GenzBretz`, in Miwa, Hayter, and Kuriki (2003) for algorithm `Miwa`, useful up to dimension 20, and Genz (2004) for the `TVPACK` algorithm, which covers 2- and 3-dimensional problems for semi-infinite integration regions.

Note the default algorithm `GenzBretz` is randomized and hence slightly depends on `.Random.seed` and that both `-Inf` and `+Inf` may be specified in lower and upper. For more details see `pmvt`.

The multivariate normal case is treated as a special case of `pmvt` with `df=0` and univariate problems are passed to `pnorm`.

The multivariate normal density and random deviates are available using `dmvnorm` and `rmvnorm`.

`pmvnorm` is based on original implementations by Alan Genz, Frank Bretz, and Tetsuhisa Miwa developed for computing accurate approximations to the normal integral. Users interested in computing log-likelihoods involving such normal probabilities should consider function `lpmvnorm`, which is more flexible and efficient for this task and comes with the ability to evaluate score functions.

An overview is available from Genz and Bretz (2009).

## Value

The evaluated distribution function is returned, if `keepAttr` is true, with attributes

<code>error</code>	estimated absolute error
<code>msg</code>	status message(s).
<code>algorithm</code>	a <code>character</code> string with <code>class(algorithm)</code> .

## References

Genz A (1992). "Numerical Computation of Multivariate Normal Probabilities." *Journal of Computational and Graphical Statistics*, **1**(2), 141–149. doi:10.1080/10618600.1992.10477010.

Genz A (1993). “Comparison of Methods for the Computation of Multivariate Normal Probabilities.” *Computing Science and Statistics*, **25**, 400–405.

Genz A (2004). “Numerical Computation of Rectangular Bivariate and Trivariate Normal and  $t$  Probabilities.” *Statistics and Computing*, **14**(3), 251–260. doi:10.1023/B:STCO.0000035304.20635.31.

Genz A, Bretz F (2009). *Computation of Multivariate Normal and  $t$  Probabilities*, series Lecture Notes in Statistics. Springer-Verlag, Heidelberg, Germany. ISBN 978-3-642-01688-2.

Miwa T, Hayter AJ, Kuriki S (2003). “The Evaluation of General Non-centred Orthant Probabilities.” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **65**(1), 223–234. doi:10.1111/14679868.00382.

### See Also

[qmvnorm](#) for quantiles and [lpmvnorm](#) for log-likelihoods.

### Examples

```
n <- 5
mean <- rep(0, 5)
lower <- rep(-1, 5)
upper <- rep(3, 5)
corr <- diag(5)
corr[lower.tri(corr)] <- 0.5
corr[upper.tri(corr)] <- 0.5
prob <- pmvnorm(lower, upper, mean, corr)
print(prob)

stopifnot(pmvnorm(lower=-Inf, upper=3, mean=0, sigma=1) == pnorm(3))

a <- pmvnorm(lower=-Inf, upper=c(.3, .5), mean=c(2, 4), diag(2))

stopifnot(round(a, 16) == round(prod(pnorm(c(.3, .5), c(2, 4))), 16))

a <- pmvnorm(lower=-Inf, upper=c(.3, .5, 1), mean=c(2, 4, 1), diag(3))

stopifnot(round(a, 16) == round(prod(pnorm(c(.3, .5, 1), c(2, 4, 1))), 16))

# Example from R News paper (original by Genz, 1992):

m <- 3
sigma <- diag(3)
sigma[2,1] <- 3/5
sigma[3,1] <- 1/3
sigma[3,2] <- 11/15
pmvnorm(lower=rep(-Inf, m), upper=c(1,4,2), mean=rep(0, m), corr=sigma)

# Correlation and Covariance

a <- pmvnorm(lower=-Inf, upper=c(2,2), sigma = diag(2)*2)
b <- pmvnorm(lower=-Inf, upper=c(2,2)/sqrt(2), corr=diag(2))
stopifnot(all.equal(round(a,5) , round(b, 5)))
```

---

pmvt *Multivariate t Distribution*

---

**Description**

Computes the the distribution function of the multivariate t distribution for arbitrary limits, degrees of freedom and correlation matrices based on algorithms by Genz and Bretz.

**Usage**

```
pmvt(lower=-Inf, upper=Inf, delta=rep(0, length(lower)),
      df=1, corr=NULL, sigma=NULL, algorithm = GenzBretz(),
      type = c("Kshirsagar", "shifted"), keepAttr=TRUE, seed = NULL, ...)
```

**Arguments**

lower	the vector of lower limits of length n.
upper	the vector of upper limits of length n.
delta	the vector of noncentrality parameters of length n, for type = "shifted" delta specifies the mode.
df	degree of freedom as integer. Normal probabilities are computed for df=0.
corr	the correlation matrix of dimension n.
sigma	the scale matrix of dimension n. Either corr or sigma can be specified. If sigma is given, the problem is standardized internally. If corr is given, it is assumed that appropriate standardization was performed by the user. If neither corr nor sigma is given, the identity matrix is used for sigma.
algorithm	an object of class <a href="#">GenzBretz</a> or <a href="#">TVPACK</a> defining the hyper parameters of this algorithm.
type	type of the noncentral multivariate t distribution to be computed. The choice type = "Kshirsagar" corresponds to formula (1.4) in Genz and Bretz (2009), see also Chapter 5.1 in Kotz and Nadarajah (2004). This is the noncentral t-distribution needed for calculating the power of multiple contrast tests under a normality assumption. type = "shifted" corresponds to the formula right before formula (1.4) in Genz and Bretz (2009) (see also formula (1.1) in Kotz and Nadarajah 2004). It is a location shifted version of the central t-distribution. This noncentral multivariate t distribution appears for example as the Bayesian posterior distribution for the regression coefficients in a linear regression. In the central case both types coincide.
keepAttr	<a href="#">logical</a> indicating if <a href="#">attributes</a> such as error and msg should be attached to the return value. The default, TRUE is back compatible.
seed	an object specifying if and how the random number generator should be initialized, see <a href="#">simulate</a> .
...	additional parameters (currently given to GenzBretz for backward compatibility issues).

## Details

This function involves the computation of central and noncentral multivariate t-probabilities with arbitrary correlation matrices. It involves both the computation of singular and nonsingular probabilities. The methodology (for default `algorithm = GenzBretz()`) is based on randomized quasi Monte Carlo methods and described in Genz and Bretz (1999); Genz and Bretz (2002).

Because of the randomization, the result for this algorithm (slightly) depends on `.Random.seed`.

For 2- and 3-dimensional problems one can also use the `TVPACK` routines described by Genz (2004), which only handles semi-infinite integration regions (and for `type = "Kshirsagar"` only central problems).

For `type = "Kshirsagar"` and a given correlation matrix `corr`, for short  $A$ , say, (which has to be positive semi-definite) and degrees of freedom  $\nu$  the following values are numerically evaluated

$$I = 2^{1-\nu/2}/\Gamma(\nu/2) \int_0^\infty s^{\nu-1} \exp(-s^2/2) \Phi(s \cdot \text{lower}/\sqrt{\nu} - \delta, s \cdot \text{upper}/\sqrt{\nu} - \delta) ds$$

where

$$\Phi(a, b) = (\det(A)(2\pi)^m)^{-1/2} \int_a^b \exp(-x'Ax/2) dx$$

is the multivariate normal distribution and  $m$  is the number of rows of  $A$ .

For `type = "shifted"`, a positive definite symmetric matrix  $S$  (which might be the correlation or the scale matrix), mode (vector)  $\delta$  and degrees of freedom  $\nu$  the following integral is evaluated:

$$c \int_{\text{lower}_1}^{\text{upper}_1} \dots \int_{\text{lower}_m}^{\text{upper}_m} (1 + (x - \delta)'S^{-1}(x - \delta)/\nu)^{-(\nu+m)/2} dx_1 \dots dx_m,$$

where

$$c = \Gamma((\nu + m)/2)/((\pi\nu)^{m/2}\Gamma(\nu/2)|S|^{1/2}),$$

and  $m$  is the number of rows of  $S$ .

Note that both `-Inf` and `+Inf` may be specified in the lower and upper integral limits in order to compute one-sided probabilities.

Univariate problems are passed to `pt`. If `df = 0`, normal probabilities are returned.

## Value

The evaluated distribution function is returned, if `keepAttr` is true, with attributes

<code>error</code>	estimated absolute error and
<code>msg</code>	status message (a <code>character</code> string).
<code>algorithm</code>	a <code>character</code> string with <code>class(algorithm)</code> .

## References

- Genz A (2004). “Numerical Computation of Rectangular Bivariate and Trivariate Normal and  $t$  Probabilities.” *Statistics and Computing*, **14**(3), 251–260. doi:10.1023/B:STCO.0000035304.20635.31.
- Genz A, Bretz F (2009). *Computation of Multivariate Normal and  $t$  Probabilities*, series Lecture Notes in Statistics. Springer-Verlag, Heidelberg, Germany. ISBN 978-3-642-01688-2.
- Genz A, Bretz F (1999). “Numerical Computation of Multivariate  $t$ -probabilities with Application to Power Calculation of Multiple Contrasts.” *Journal of Statistical Computation and Simulation*, **63**(4), 103–117. doi:10.1080/00949659908811962.
- Genz A, Bretz F (2002). “Methods for the Computation of Multivariate  $t$  Probabilities.” *Journal of Computational and Graphical Statistics*, **11**(4), 950–971. doi:10.1198/106186002394.
- Kotz S, Nadarajah S (2004). *Multivariate  $t$  Distributions and Their Applications*. Cambridge University Press, Cambridge, UK.

## See Also

[qmvmt](#)

## Examples

```
n <- 5
lower <- -1
upper <- 3
df <- 4
corr <- diag(5)
corr[lower.tri(corr)] <- 0.5
delta <- rep(0, 5)
prob <- pmvt(lower=lower, upper=upper, delta=delta, df=df, corr=corr)
print(prob)

pmvt(lower=-Inf, upper=3, df = 3, sigma = 1) == pt(3, 3)

# Example from R News paper (original by Edwards and Berry, 1987)

n <- c(26, 24, 20, 33, 32)
V <- diag(1/n)
df <- 130
C <- c(1,1,1,0,0,-1,0,0,1,0,0,-1,0,0,1,0,0,0,-1,-1,0,0,-1,0,0)
C <- matrix(C, ncol=5)
### scale matrix
cv <- C %%% tcrossprod(V, C)
### correlation matrix
cr <- cov2cor(cv)
delta <- rep(0,5)

myfct <- function(q, alpha) {
  lower <- rep(-q, ncol(cv))
  upper <- rep(q, ncol(cv))
  pmvt(lower=lower, upper=upper, delta=delta, df=df,
        corr=cr, abseps=0.0001) - alpha
}
```

```

### uniroot for this simple problem
round(uniroot(myfct, lower=1, upper=5, alpha=0.95)$root, 3)

# compare pmvt and qmvnorm for large df:

a <- qmvnorm(lower=-Inf, upper=1, mean=rep(0, 5), corr=diag(5))
b <- pmvt(lower=-Inf, upper=1, delta=rep(0, 5), df=300,
          corr=diag(5))
a
b

stopifnot(round(a, 2) == round(b, 2))

# correlation and scale matrix

a <- pmvt(lower=-Inf, upper=2, delta=rep(0,5), df=3,
          sigma = diag(5)*2)
b <- pmvt(lower=-Inf, upper=2/sqrt(2), delta=rep(0,5),
          df=3, corr=diag(5))
attributes(a) <- NULL
attributes(b) <- NULL
a
b
stopifnot(all.equal(round(a,3) , round(b, 3)))

a <- pmvt(0, 1, df=10)
attributes(a) <- NULL
b <- pt(1, df=10) - pt(0, df=10)
stopifnot(all.equal(round(a,10) , round(b, 10)))

```

---

qmvnorm

*Quantiles of the Multivariate Normal Distribution*


---

## Description

Computes the equicoordinate quantile function of the multivariate normal distribution for arbitrary correlation matrices based on inversion of [qmvnorm](#), using a stochastic root finding algorithm described in Bornkamp (2018).

## Usage

```

qmvnorm(p, interval = NULL, tail = c("lower.tail", "upper.tail", "both.tails"),
        mean = 0, corr = NULL, sigma = NULL, algorithm = GenzBretz(),
        ptol = 0.001, maxiter = 500, trace = FALSE, seed = NULL, ...)

```

**Arguments**

p	probability.
interval	optional, a vector containing the end-points of the interval to be searched. Does not need to contain the true quantile, just used as starting values by the root-finder. If equal to NULL a guess is used.
tail	specifies which quantiles should be computed. <code>lower.tail</code> gives the quantile $x$ for which $P[X \leq x] = p$ , <code>upper.tail</code> gives $x$ with $P[X > x] = p$ and <code>both.tails</code> leads to $x$ with $P[-x \leq X \leq x] = p$ .
mean	the mean vector of length n.
corr	the correlation matrix of dimension n.
sigma	the covariance matrix of dimension n. Either <code>corr</code> or <code>sigma</code> can be specified. If <code>sigma</code> is given, the problem is standardized internally. If <code>corr</code> is given, it is assumed that appropriate standardization was performed by the user. If neither <code>corr</code> nor <code>sigma</code> is given, the identity matrix is used for <code>sigma</code> .
algorithm	an object of class <a href="#">GenzBretz</a> , <a href="#">Miwa</a> or <a href="#">TVPACK</a> specifying both the algorithm to be used as well as the associated hyper parameters.
ptol, maxiter, trace	Parameters passed to the stochastic root-finding algorithm. Iteration stops when the 95% confidence interval for the predicted quantile is inside $[p-ptol, p+ptol]$ . <code>maxiter</code> is the maximum number of iterations for the root finding algorithm. <code>trace</code> prints the iterations of the root finder.
seed	an object specifying if and how the random number generator should be initialized, see <a href="#">simulate</a> .
...	additional parameters to be passed to <a href="#">GenzBretz</a> .

**Details**

Only equicoordinate quantiles are computed, i.e., the quantiles in each dimension coincide. The result is seed dependent. The concept is explained in Bornkamp (2018).

**Value**

A list with two components: `quantile` and `f.quantile` give the location of the quantile and the difference between the distribution function evaluated at the quantile and  $p$ .

**References**

Bornkamp B (2018). "Calculating Quantiles of Noisy Distribution Functions Using Local Linear Regressions." *Computational Statistics*, **33**(1), 487–501. doi:10.1007/s0018001707360.

**See Also**

[pmvnorm](#), [qmv](#)

**Examples**

```
qmvnorm(0.95, sigma = diag(2), tail = "both")
```

qmvt

*Quantiles of the Multivariate t Distribution***Description**

Computes the equicoordinate quantile function of the multivariate t distribution for arbitrary correlation matrices based on inversion of [pmvt](#), using a stochastic root finding algorithm described in Bornkamp (2018).

**Usage**

```
qmvt(p, interval = NULL, tail = c("lower.tail", "upper.tail", "both.tails"),
     df = 1, delta = 0, corr = NULL, sigma = NULL, algorithm = GenzBretz(),
     type = c("Kshirsagar", "shifted"), ptol = 0.001, maxiter = 500,
     trace = FALSE, seed = NULL, ...)
```

**Arguments**

p	probability.
interval	optional, a vector containing the end-points of the interval to be searched. Does not need to contain the true quantile, just used as starting values by the root-finder. If equal to NULL a guess is used.
tail	specifies which quantiles should be computed. <code>lower.tail</code> gives the quantile $x$ for which $P[X \leq x] = p$ , <code>upper.tail</code> gives $x$ with $P[X > x] = p$ and <code>both.tails</code> leads to $x$ with $P[-x \leq X \leq x] = p$ .
delta	the vector of noncentrality parameters of length $n$ , for <code>type = "shifted"</code> delta specifies the mode.
df	degree of freedom as integer. Normal quantiles are computed for $df = 0$ or $df = \text{Inf}$ .
corr	the correlation matrix of dimension $n$ .
sigma	the covariance matrix of dimension $n$ . Either <code>corr</code> or <code>sigma</code> can be specified. If <code>sigma</code> is given, the problem is standardized internally. If <code>corr</code> is given, it is assumed that appropriate standardization was performed by the user. If neither <code>corr</code> nor <code>sigma</code> is given, the identity matrix in the univariate case (so <code>corr = 1</code> ) is used for <code>corr</code> .
algorithm	an object of class <a href="#">GenzBretz</a> or <a href="#">TVPACK</a> defining the hyper parameters of this algorithm.
type	type of the noncentral multivariate t distribution to be computed. The choice <code>type = "Kshirsagar"</code> corresponds to formula (1.4) in Genz and Bretz (2009), see also Chapter 5.1 in Kotz and Nadarajah (2004). This is the noncentral t-distribution needed for calculating the power of multiple contrast tests under a normality assumption. <code>type = "shifted"</code> corresponds to the formula right before formula (1.4) in Genz and Bretz (2009) (see also formula (1.1) in Kotz and Nadarajah 2004).

ptol, maxiter, trace	Parameters passed to the stochastic root-finding algorithm. Iteration stops when the 95% confidence interval for the predicted quantile is inside [p-ptol, p+ptol]. maxiter is the maximum number of iterations for the root finding algorithm. trace prints the iterations of the root finder.
seed	an object specifying if and how the random number generator should be initialized, see <a href="#">simulate</a> .
...	additional parameters to be passed to <a href="#">GenzBretz</a> .

### Details

Only equicoordinate quantiles are computed, i.e., the quantiles in each dimension coincide. The result is seed dependent. The concept is explained in Bornkamp (2018).

### Value

A list with two components: `quantile` and `f.quantile` give the location of the quantile and the difference between the distribution function evaluated at the quantile and `p`.

### References

- Bornkamp B (2018). "Calculating Quantiles of Noisy Distribution Functions Using Local Linear Regressions." *Computational Statistics*, **33**(1), 487–501. doi:10.1007/s0018001707360.
- Genz A, Bretz F (2009). *Computation of Multivariate Normal and t Probabilities*, series Lecture Notes in Statistics. Springer-Verlag, Heidelberg, Germany. ISBN 978-3-642-01688-2.
- Kotz S, Nadarajah S (2004). *Multivariate t Distributions and Their Applications*. Cambridge University Press, Cambridge, UK.

### See Also

[pmvnorm](#), [qmvnorm](#)

### Examples

```
## basic evaluation
qmvt(0.95, df = 16, tail = "both")

## check behavior for df=0 and df=Inf
Sigma <- diag(2)
set.seed(29)
q0 <- qmvt(0.95, sigma = Sigma, df = 0, tail = "both")$quantile
set.seed(29)
q8 <- qmvt(0.95, sigma = Sigma, df = Inf, tail = "both")$quantile
set.seed(29)
qn <- qmvnorm(0.95, sigma = Sigma, tail = "both")$quantile
stopifnot(identical(q0, q8),
           isTRUE(all.equal(q0, qn, tol = (.Machine$double.eps)^(1/3))))

## if neither sigma nor corr are provided, corr = 1 is used internally
df <- 0
```

```
set.seed(29)
qt95 <- qmvt(0.95, df = df, tail = "both")$quantile
set.seed(29)
qt95.c <- qmvt(0.95, df = df, corr = 1, tail = "both")$quantile
set.seed(29)
qt95.s <- qmvt(0.95, df = df, sigma = 1, tail = "both")$quantile
stopifnot(identical(qt95, qt95.c),
           identical(qt95, qt95.s))

df <- 4
set.seed(29)
qt95 <- qmvt(0.95, df = df, tail = "both")$quantile
set.seed(29)
qt95.c <- qmvt(0.95, df = df, corr = 1, tail = "both")$quantile
set.seed(29)
qt95.s <- qmvt(0.95, df = df, sigma = 1, tail = "both")$quantile
stopifnot(identical(qt95, qt95.c),
           identical(qt95, qt95.s))
```

# Index

- \* **distribution**
  - algorithms, 3
  - interface, 4
  - lpmvnorm, 6
  - lpRR, 8
  - margcond, 13
  - Mvnorm, 14
  - Mvt, 16
  - pmvnorm, 18
  - pmvt, 21
  - qmvnorm, 24
  - qmvt, 26
- \* **matrix**
  - ltMatrices, 10
- \* **multivariate**
  - Mvnorm, 14
  - Mvt, 16
- \* **package**
  - mvtnorm-package, 2
  - .Random.seed, 19, 22
- adddiag (ltMatrices), 10
- algorithms, 3
- aperm.chol (ltMatrices), 10
- aperm.invchol (ltMatrices), 10
- aperm.ltMatrices (ltMatrices), 10
- aperm.mvnorm (interface), 4
- aperm.syMatrices (ltMatrices), 10
- as.array.ltMatrices (ltMatrices), 10
- as.array.syMatrices (ltMatrices), 10
- as.chol (ltMatrices), 10
- as.invchol (ltMatrices), 10
- as.ltMatrices (ltMatrices), 10
- as.syMatrices (ltMatrices), 10
- attributes, 19, 21
- character, 19, 22
- chol.syMatrices (ltMatrices), 10
- chol2cor (ltMatrices), 10
- chol2cov (ltMatrices), 10
- chol2invchol (ltMatrices), 10
- chol2pc (ltMatrices), 10
- chol2pre (ltMatrices), 10
- cond\_mvnorm (margcond), 13
- condDist (interface), 4
- cov2cor, 17
- Crossprod (ltMatrices), 10
- crossprod.ltMatrices (ltMatrices), 10
- crossprod.syMatrices (ltMatrices), 10
- Dchol (ltMatrices), 10
- deperma (ltMatrices), 10
- destandardize (ltMatrices), 10
- diagonals (ltMatrices), 10
- diagonals<- (ltMatrices), 10
- dmvnorm, 7, 17, 19
- dmvnorm (Mvnorm), 14
- dmt (Mvt), 16
- GenzBretz, 19, 21, 25–27
- GenzBretz (algorithms), 3
- interface, 4
- invchol2chol (ltMatrices), 10
- invchol2cor (ltMatrices), 10
- invchol2cov (ltMatrices), 10
- invchol2pc (ltMatrices), 10
- invchol2pre (ltMatrices), 10
- invcholD (ltMatrices), 10
- is.chol (ltMatrices), 10
- is.invchol (ltMatrices), 10
- is.ltMatrices (ltMatrices), 10
- is.syMatrices (ltMatrices), 10
- ldmvnorm, 2
- ldmvnorm (lpmvnorm), 6
- ldpmvnorm, 2, 5
- ldpmvnorm (lpmvnorm), 6
- llgrad (interface), 4
- logdet (ltMatrices), 10

logical, [16](#), [19](#), [21](#)  
logLik.mvnorm (interface), [4](#)  
Lower\_tri (ltMatrices), [10](#)  
lpmvnorm, [2](#), [6](#), [19](#), [20](#)  
lpRR, [8](#)  
ltMatrices, [6](#), [10](#), [14](#)

marg\_mvnorm (margcond), [13](#)  
margcond, [13](#)  
margDist (interface), [4](#)  
Miwa, [19](#), [25](#)  
Miwa (algorithms), [3](#)  
Mult (ltMatrices), [10](#)  
Mvnorm, [14](#)  
mvnorm (interface), [4](#)  
Mvt, [16](#)  
mvtnorm (mvtnorm-package), [2](#)  
mvtnorm-package, [2](#)

pmvnorm, [2](#), [7](#), [15](#), [18](#), [24](#), [25](#), [27](#)  
pmvt, [2](#), [16](#), [17](#), [19](#), [21](#), [26](#)  
pnorm, [19](#)  
pt, [22](#)

qmvnorm, [2](#), [15](#), [20](#), [24](#), [27](#)  
qmvt, [2](#), [17](#), [23](#), [25](#), [26](#)

rmvnorm, [17](#), [19](#)  
rmvnorm (Mvnorm), [14](#)  
rmvt (Mvt), [16](#)  
rnorm, [15](#)

simulate, [5](#), [7](#), [19](#), [21](#), [25](#), [27](#)  
simulate.mvnorm (interface), [4](#)  
sldmvnorm, [2](#)  
sldmvnorm (lpmvnorm), [6](#)  
sldpmvnorm, [2](#), [5](#)  
sldpmvnorm (lpmvnorm), [6](#)  
slpmvnorm, [2](#), [14](#)  
slpmvnorm (lpmvnorm), [6](#)  
slpRR (lpRR), [8](#)  
solve.ltMatrices (ltMatrices), [10](#)  
standardize (ltMatrices), [10](#)  
syMatrices (ltMatrices), [10](#)

Tcrossprod (ltMatrices), [10](#)  
tcrossprod.ltMatrices (ltMatrices), [10](#)  
tcrossprod.syMatrices (ltMatrices), [10](#)  
TVPACK, [19](#), [21](#), [22](#), [25](#), [26](#)  
TVPACK (algorithms), [3](#)

vctrick (ltMatrices), [10](#)