Package ‘mvtnorm’

June 8, 2023

Title Multivariate Normal and t Distributions
Version 1.2-2
Date 2023-06-05
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
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] (<https://orcid.org/0000-0002-6294-8185>),
  Martin Maechler [ctb] (<https://orcid.org/0000-0002-8685-9910>),
  Torsten Hothorn [aut, cre] (<https://orcid.org/0000-0001-8301-0471>)
Maintainer Torsten Hothorn <Torsten.Hothorn@R-project.org>
Repository CRAN
Date/Publication 2023-06-08 14:50:02 UTC

R topics documented:

mvtnorm-package .............................................................................. 2
mvtnorm-package

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 slldmvnorm, which help to speed-up parameter estimation considerably. Users interested in this functionality should consult the lmvnorm_src package vignette.

See Also

vignette("lmvnorm_src", package = "mvtnorm")
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 pmvt() or pmvnorm()).
maxval replacement for Inf when non-orthant probabilities involving Inf 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, 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 et al. (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


---

**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

```r
lpmvnorm(lower, upper, mean = 0, center = NULL, chol, invchol, logLik = TRUE,
         M = NULL, w = NULL, seed = NULL, tol = .Machine$double.eps, fast = FALSE)
slpmvnorm(lower, upper, mean = 0, center = NULL, chol, invchol, logLik = TRUE,
         M = NULL, w = NULL, seed = NULL, tol = .Machine$double.eps, fast = FALSE)
ldmvnorm(obs, mean = 0, chol, invchol, logLik = TRUE)
sldmvnorm(obs, mean = 0, chol, invchol, logLik = TRUE)
ldpmvnorm(obs, lower, upper, mean = 0, chol, invchol, logLik = TRUE, ...)
sldpmvnorm(obs, lower, upper, mean = 0, 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`).
center  matrix of negative rescaled means (one column for each observation, length is recycled to length of lower and upper) as returned by cond_mvnorm(..., center = TRUE).

chol   Cholesky factors of covariance matrices as \texttt{ltMatrices} object, length is recycled to length of obs, lower and upper.

invchol Cholesky factors of precision matrices as \texttt{ltMatrices} object, length is recycled to length of lower and upper. Either chol or invchol must be given.

logLik logical, if TRUE, 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 & Bretz, 2002). Note that the same weights for evaluating the multivariate normal probability are used for all observations when \(\text{ncol}(w) == M\) is specified. If \(\text{ncol}(w) == \text{nrow}(\text{lower}) \times M\), each likelihood contribution is evaluated on the corresponding sub-matrix. If \(w\) is NULL, different uniform numbers are drawn for each observation.

seed an object specifying if and how the random number generator should be initialized, see \texttt{simulate}. Only applied when \(w\) is NULL.

tol   tolerance limit, values smaller than tol are interpreted as zero.

fast  logical, if TRUE, a faster but less accurate version of \texttt{pnorm} is used internally.

...  additional arguments to \texttt{lpmvnorm}.

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 & Bretz, 2002) integration is implemented, the latter with weights obtained, for example, from packages \texttt{qrng} or \texttt{randtoolbox}.
It is the responsibility of the user to ensure a meaningful lattice is used. In case of doubt, use plain Monte-Carlo (\(w = \text{NULL}\)) or \texttt{pmvnorm}.

\texttt{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 \texttt{sldmvnorm} and \texttt{sldpmvnorm}, respectively.

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

Value

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


See Also
dmvnorm, vignette("lmvnorm_src", package = "mvtnorm")

Examples

```r
### 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)
```

---

*ltMatrices*  
*Multiple Lower Triangular Matrices*

Description

A class representing multiple lower triangular matrices and some methods.
ltMatrices

Usage

ltMatrices(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'
solve(a, b, transpose = FALSE, ...)
## S3 method for class 'ltMatrices'
chol(x, ...)
## S3 method for class 'ltMatrices'
aperm(a, perm, is_chol = FALSE, ...)
Lower_tri(x, diag = FALSE, byrow = attr(x, "byrow"))
Mult(x, y, transpose = FALSE)
Tcrossprod(x, diag_only = FALSE)
Crossprod(x, diag_only = FALSE)
chol2cov(x)
invchol2chol(x)
chol2invchol(x)
invchol2cov(x)
invchol2pre(x)
chol2pre(x)
Dchol(x, D = 1 / sqrt(Tcrossprod(x, diag_only = TRUE)))
invchol2cov(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_chol)
**Arguments**

- **object** a matrix representing the lower triangular elements of $N$ lower triangular matrices, each of dimension $J \times J$. Dimensions of object depend on `diag`: 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** object of class `ltMatrices` or `symMatrices` (for `chol`).

- **value** a matrix of diagonal elements to be assigned (of dimension $J \times N$).

- **a** object of class `ltMatrices`.

- **perm** a permutation of the covariance matrix corresponding to `a`.

- **is_chol** a logical indicating if `a` is the Cholesky of the covariance (chol = TRUE) of the precision matrix.

- **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 `ltMatrices`.

- **S** an object of class `ltMatrices` or a matrix with $J^2$ rows representing multiple \(J \times J\) matrices (columns of vec operators).

- **A** an object of class `ltMatrices`.

- **transpose** a logical of length two indicating if `A` or `B` shall be transposed in `vectrick`. For `solve`, this argument being true computes `solve(t(a), b)` (in absence of a `t()` method for `ltMatrices` objects).

- **score_schol** score matrix for a standardized chol 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, explicite inverses, and crossproducts can be computed based on such objects. Details can be found in the `lmvnorm_src` package vignette.

**Value**

The constructor `ltMatrices` returns objects of class `ltMatrices` with corresponding methods.
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

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

```r
# Density function
dmvnorm(x, mean = rep(0, p), sigma = diag(p), log = FALSE, checkSymmetry = TRUE)
# Random number generator
rmvnorm(n, mean = rep(0, nrow(sigma)), sigma = diag(length(mean)),
        method=c("eigen", "svd", "chol"), pre0.9_9994 = FALSE,
        checkSymmetry = TRUE)
```
Arguments

- **x**: vector or matrix of quantiles. When x is a matrix, each row is taken to be a quantile and columns correspond to the number of dimensions, p.
- **n**: number of observations.
- **mean**: mean vector, default is rep(0, length = ncol(x)). In ldmvnorm or sldmvnorm, mean is a matrix with observation-specific means arranged in columns.
- **sigma**: covariance matrix, default is diag(ncol(x)).
- **log**: logical; if TRUE, densities d are given as log(d).
- **method**: string specifying the matrix decomposition used to determine the matrix root of sigma. Possible methods are eigenvalue decomposition ("eigen", default), singular value decomposition ("svd"), and Cholesky decomposition ("chol"). The Cholesky is typically fastest, not by much though.
- **pre0.9_9994**: logical; if FALSE, the output produced in mvtnorm versions up to 0.9-9993 is reproduced. In 0.9-9994, the output is organized such that rmvnorm(10,...) has the same first ten rows as rmvnorm(100,...) when called with the same seed.
- **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.

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("lmvnorm_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) \( \text{delta} \), scale matrix \( \text{sigma} \) and degrees of freedom \( \text{df} \). \( \text{dmvt} \) gives the density and \( \text{rmvt} \) generates random deviates.

**Usage**

```r
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**

- **x** vector or matrix of quantiles. If \( x \) is a matrix, each row is taken to be a quantile.
- **n** number of observations.
- **delta** the vector of noncentrality parameters of length \( n \), for \( \text{type} = \text{"shifted"} \) \( \text{delta} \) specifies the mode.
- **sigma** scale matrix, defaults to \( \text{diag(ncol(x))} \).
- **df** degrees of freedom. \( \text{df = 0} \) or \( \text{df = Inf} \) corresponds to the multivariate normal distribution.
- **log** logical indicating whether densities \( d \) are given as \( \log(d) \).
- **type** type of the noncentral multivariate t distribution. \( \text{type} = \text{"Kshirsagar"} \) 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. \( \text{type} = \text{"shifted"} \) 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. Note that the defaults differ from the default in \( \text{pmvt()} \) (for reasons of backward compatibility).
- **checkSymmetry** logical; if \( \text{FALSE} \), skip checking whether the covariance matrix is symmetric or not. This will speed up the computation but may cause unexpected outputs when ill-behaved \( \text{sigma} \) is provided. The default value is \( \text{TRUE} \).
- **...** additional arguments to \( \text{rmvnorm()} \), for example \( \text{method} \).
Details

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

Note that \( \text{dmvt}() \) has default log = TRUE, whereas \( \text{dmvnorm}() \) has default log = FALSE.

References


See Also

\( \text{pmvt}() \) and \( \text{qmvt}() \)

Examples

```r
## 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 <- dmvnorm(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 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 GenzBretz, Miwa or TVPACK specifying both the algorithm to be used as well as the associated hyper parameters.
- **keepAttr**
  - logical indicating if attributes such as error and msg should be attached to the return value. The default, TRUE is back compatible.
pmvnorm

seed an object specifying if and how the random number generator should be initialized, see simulate.

additional parameters (currently given to GenzBretz 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, 1993) (for algorithm GenzBretz), in Miwa et al. (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.

Value

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

<table>
<thead>
<tr>
<th>error</th>
<th>estimated absolute error</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>status message(s).</td>
</tr>
<tr>
<td>algorithm</td>
<td>a character string with class(algorithm).</td>
</tr>
</tbody>
</table>

References


See Also

qmvnorm for quantiles and lpmvnorm for log-likelihoods.
Examples

```r
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 GenzBretz or TVPACK 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: logical indicating if attributes 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 simulate.
- ...: 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, 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

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

\[
c \int_{lower_1}^{upper_1} \cdots \int_{lower_m}^{upper_m} (1 + ((x - \delta)'S^{-1}(x - \delta)/\nu)^{-(\nu+m)/2} dx_1...dx_m,
\]

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 \(-\infty\) and \(+\infty\) 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

- error estimated absolute error and
- msg status message (a character string).
- algorithm a character string with class(algorithm).

References


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(1,1,0,0,-1,0,0,1,0,0,-1,0,0,1,0,0,-1,0,0,-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 pmvnorm for large df:
a <- pmvnorm(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
```r
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 `pmvnorm`, using a stochastic root finding algorithm described in Bornkamp (2018).

**Usage**

```r
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. lower.tail gives the quantile $x$ for which $P[X \leq x] = p$, upper.tail gives $x$ with $P[X > x] = p$ and both.tails 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 `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 GenzBretz, Miwa or TVPACK 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]. 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 simulate.

... additional parameters to be passed to GenzBretz.

Details
Only equicoordinate quantiles are computed, i.e., the quantiles in each dimension coincide. The result is seed dependent.

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

See Also
pmvnorm, qmvt

Examples

qmnorm(0.95, sigma = diag(2), tail = "both")

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. lower.tail gives the quantile \( x \) for which \( P[X \leq x] = p \), upper.tail gives \( x \) with \( P[X > x] = p \) and both.tails leads to \( x \) with \( P[-x \leq X \leq x] = p \).

delta

- the vector of noncentrality parameters of length \( n \), for type = "shifted" delta specifies the mode.

df

- degree of freedom as integer. Normal quantiles are computed for \( df = 0 \) or \( df = \infty \).

corr

- the correlation matrix of dimension \( n \).

sigma

- the covariance 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 in the univariate case (so corr = 1) is used for corr.

algorithm

- an object of class GenzBretz or TVPACK 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)) and type = "shifted" corresponds to the formula 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 \([p-\text{ptol}, p+\text{ptol}]\). 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 simulate.

... additional parameters to be passed to GenzBretz.

Details

Only equicoordinate quantiles are computed, i.e., the quantiles in each dimension coincide. The result is seed dependend.

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 re-

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
  lpmvnorm, 4
  margcond, 9
  Mvnorm, 10
  Mvt, 12
  pmvnorm, 14
  pmvt, 16
  qmvnorm, 20
  qmvt, 21
* matrix
  ltMatrices, 6
* multivariate
  Mvnorm, 10
  Mvt, 12
* package
  mvtnorm-package, 2
  .Random.seed, 15, 17
adddiag (ltMatrices), 6
algorithms, 3
aperm.ltMatrices (ltMatrices), 6
as.array.ltMatrices (ltMatrices), 6
as.array.syMatrices (ltMatrices), 6
attributes, 14, 17
character, 15, 18
chol.syMatrices (ltMatrices), 6
chol2cor (ltMatrices), 6
chol2cov (ltMatrices), 6
chol2invchol (ltMatrices), 6
chol2pc (ltMatrices), 6
chol2pre (ltMatrices), 6
cond_mvnorm (margcond), 9
cov2cor, 13
Crossprod (ltMatrices), 6
Dchol (ltMatrices), 6
destandardize (ltMatrices), 6
diagonals (ltMatrices), 6
diagonals<- (ltMatrices), 6
dmvnorm, 6, 13, 15
dmvnorm (Mvnorm), 10
dmvt (Mvt), 12
GenzBretz, 14, 15, 17, 21, 22
GenzBretz (algorithms), 3
invchol2chol (ltMatrices), 6
invchol2cor (ltMatrices), 6
invchol2cov (ltMatrices), 6
invchol2pc (ltMatrices), 6
invchol2pre (ltMatrices), 6
invcholD (ltMatrices), 6
ldmvnorm, 2
ldmvnorm (lpmvnorm), 4
lpmvnorm, 6
1pmvnorm, 2
lpmvnorm (lpmvnorm), 4
logical, 12, 14, 17
Lower_tri (ltMatrices), 6
1pmvnorm, 2, 4, 15
ltMatrices, 5, 6, 10
marg_mvnorm (margcond), 9
margcond, 9
Miwa, 14, 21
Miwa (algorithms), 3
Mult (ltMatrices), 6
Mvnorm, 10
Mvt, 12
mvtnorm (mvtnorm-package), 2
mvtnorm-package, 2
pmvnorm, 2, 5, 11, 14, 20, 21, 23
pmvt, 2, 12, 13, 15, 16, 21
pnorm, 15
pt, 18
qmvmnorm, 2, 11, 15, 20, 23
qmvt, 2, 13, 19, 21, 21
INDEX

rmvnorm, 12, 15
rmvnorm (Mvnorm), 10
rmvt (Mvt), 12
rnorm, 11

simulate, 5, 15, 17, 21, 22
sldmvnorm, 2
sldmvnorm (lpmvnorm), 4
sldpmvnorm, 2
sldpmvnorm (lpmvnorm), 4
slpmvnorm, 2, 10
slpmvnorm (lpmvnorm), 4
solve.ltMatrices (ltMatrices), 6
standardize (ltMatrices), 6

tcrossprod (ltMatrices), 6
TVPACK, 14, 17, 21, 22
TVPACK (algorithms), 3

vectrick (ltMatrices), 6