Package: tmvtnorm (via r-universe)

October 28, 2026

Version 1.7

Date 2025-09-01

Title Truncated Multivariate Normal and Student t Distribution

Imports stats, methods

Depends R (>= 1.9.0), mvtnorm, utils, Matrix, stats4, gmm

Encoding UTF-8

Suggests lattice, rgl

Description Random number generation for the truncated multivariate normal and Student t distribution. Computes probabilities, quantiles and densities, including one-dimensional and bivariate marginal densities. Computes first and second moments (i.e. mean and covariance matrix) for the double-truncated multinormal case.

License GPL (>= 2)

URL https://www.r-project.org

NeedsCompilation yes

Author Stefan Wilhelm [aut, cre], Manjunath B G [aut]

Maintainer Stefan Wilhelm <wilhelm@financial.com>

Repository https://cran.r-universe.dev

Date/Publication 2025-09-01 20:32:54 UTC

RemoteUrl https://github.com/cran/tmvtnorm

RemoteRef HEAD

RemoteSha b72ab6f285af910cbc08be2b1b76f78ceeb608c5

Contents

dtmvnorm.marginal 2
dtmvnorm.marginal2 5
dtmvt 7
gmm.tmvnorm 9

mle.tmvnorm 12
mtmvnorm 14
ptmvnorm 16
ptmvt 17
ptmvtnorm.marginal 18
qtmvnorm-marginal 20
rtmvnorm 22
rtmvnorm2 28
rtmvt 30
tmvnorm 32

Index **36**

`dtmvnorm.marginal` *One-dimensional marginal density functions from a Truncated Multivariate Normal distribution*

Description

This function computes the one-dimensional marginal density function from a Truncated Multivariate Normal density function using the algorithm given in Cartinhour (1990).

Usage

```
dtmvnorm.marginal(xn, n=1,
  mean= rep(0, nrow(sigma)),
  sigma=diag(length(mean)),
  lower=rep(-Inf, length = length(mean)),
  upper=rep( Inf, length = length(mean)),
  log=FALSE)
```

Arguments

`xn`	Vector of quantiles to calculate the marginal density for.
`n`	Index position (1..k) within the random vector x to calculate the one-dimensional marginal density for.
`mean`	Mean vector, default is `rep(0, length = nrow(sigma))`.
`sigma`	Covariance matrix, default is `diag(length(mean))`.
`lower`	Vector of lower truncation points,\ default is `rep(-Inf, length = length(mean))`.
`upper`	Vector of upper truncation points,\ default is `rep( Inf, length = length(mean))`.
`log`	Logical; if `TRUE`, densities d are given as log(d).

Details

The one-dimensional marginal density $f_i(x_i)$ of x_i is

$$f_i(x_i) = \int_{a_1}^{b_1} \dots \int_{a_{i-1}}^{b_{i-1}} \int_{a_{i+1}}^{b_{i+1}} \dots \int_{a_k}^{b_k} f(x) dx_{-i}$$

Note that the one-dimensional marginal density is not truncated normal, but only conditional densities are truncated normal.

Author(s)

Stefan Wilhelm <Stefan.Wilhelm@financial.com>

References

Cartinhour, J. (1990). One-dimensional marginal density functions of a truncated multivariate normal density function. *Communications in Statistics - Theory and Methods*, **19**, 197–203

Arnold et al. (1993). The Nontruncated Marginal of a Truncated Bivariate Normal Distribution. *Psychometrika*, **58**, 471–488

Examples

```
##############################################
#
# Example 1: truncated bivariate normal
#
##############################################

# parameters of the bivariate normal distribution
sigma = matrix(c(1   , 0.95,
                 0.95, 1    ), 2, 2)
mu = c(0,0)

# sample from multivariate normal distribution
X = rmvnorm(5000, mu, sigma)

# tuncation in x2 with x2 <= 0
X.trunc = X[X[,2]<0,]

# plot the realisations before and after truncation
par(mfrow=c(2,2))
plot(X, col="gray", xlab=expression(x[1]), ylab=expression(x[2]),
  main="realisations from a\n truncated bivariate normal distribution")
points(X.trunc)
abline(h=0, lty=2, col="gray")
#legend("topleft", col=c("gray", "black")

# marginal density for x1 from realisations
plot(density(X.trunc[,1]), main=expression("marginal density for "*x[1]))

# one-dimensional marginal density for x1 using the formula
```

```
x <- seq(-5, 5, by=0.01)
fx <- dtmvnorm.marginal(x, n=1, mean=mu, sigma=sigma,
  lower=c(-Inf,-Inf), upper=c(Inf,0))
lines(x, fx, lwd=2, col="red")

# marginal density for x2
plot(density(X.trunc[,2]), main=expression("marginal density for "*x[2]))

# one-dimensional marginal density for x2 using the formula
x <- seq(-5, 5, by=0.01)
fx <- dtmvnorm.marginal(x, n=2, mean=mu, sigma=sigma,
  lower=c(-Inf,-Inf), upper=c(Inf,0))
lines(x, fx, lwd=2, col="blue")

##############################################
#
# Example 2 : truncated trivariate normal
#
##############################################

# parameters of the trivariate normal distribution
sigma = outer(1:3,1:3,pmin)
mu    = c(0,0,0)

# sample from multivariate normal distribution
X     = rmvnorm(2000, mu, sigma)

# truncation in x2 and x3 : x2 <= 0, x3 <= 0
X.trunc = X[X[,2]<=0 & X[,3]<=0,]

par(mfrow=c(2,3))
plot(X, col="gray", xlab=expression(x[1]), ylab=expression(x[2]),
  main="realisations from a\n truncated trivariate normal distribution")
points(X.trunc, col="black")
abline(h=0, lty=2, col="gray")

plot(X[,2:3], col="gray", xlab=expression(x[2]), ylab=expression(x[3]),
  main="realisations from a\n truncated trivariate normal distribution")
points(X.trunc[,2:3], col="black")
abline(h=0, lty=2, col="gray")
abline(v=0, lty=2, col="gray")

plot(X[,c(1,3)], col="gray", xlab=expression(x[1]), ylab=expression(x[3]),
  main="realisations from a\n truncated trivariate normal distribution")
points(X.trunc[,c(1,3)], col="black")
abline(h=0, lty=2, col="gray")

# one-dimensional marginal density for x1 from realisations and formula
plot(density(X.trunc[,1]), main=expression("marginal density for "*x[1]))
x <- seq(-5, 5, by=0.01)
fx <- dtmvnorm.marginal(x, n=1, mean=mu, sigma=sigma,
  lower=c(-Inf,-Inf,-Inf), upper=c(Inf,0,0))
lines(x, fx, lwd=2, col="red")
```

```
# one-dimensional marginal density for x2 from realisations and formula
plot(density(X.trunc[,2]), main=expression("marginal density for "*x[2]))
x <- seq(-5, 5, by=0.01)
fx <- dtmvnorm.marginal(x, n=2, mean=mu, sigma=sigma,
  lower=c(-Inf,-Inf,-Inf), upper=c(Inf,0,0))
lines(x, fx, lwd=2, col="red")

# one-dimensional marginal density for x3 from realisations and formula
plot(density(X.trunc[,3]), main=expression("marginal density for "*x[3]))
x <- seq(-5, 5, by=0.01)
fx <- dtmvnorm.marginal(x, n=3, mean=mu, sigma=sigma,
  lower=c(-Inf,-Inf,-Inf), upper=c(Inf,0,0))
lines(x, fx, lwd=2, col="red")
```

`dtmvnorm.marginal2` *Bivariate marginal density functions from a Truncated Multivariate Normal distribution*

Description

This function computes the bivariate marginal density function $f(x_q, x_r)$ from a k-dimensional Truncated Multivariate Normal density function (k>=2). The bivariate marginal density is obtained by integrating out (k-2) dimensions as proposed by Tallis (1961). This function is basically an extraction of the Leppard and Tallis (1989) Fortran code for moments calculation, but extended to the double truncated case.

Usage

```
dtmvnorm.marginal2(xq, xr, q, r,
 mean = rep(0, nrow(sigma)),
 sigma = diag(length(mean)),
 lower = rep(-Inf, length = length(mean)),
 upper = rep(Inf, length = length(mean)),
 log = FALSE, pmvnorm.algorithm=GenzBretz())
```

Arguments

`xq`	Value x_q
`xr`	Value x_r
`q`	Index position for x_q within mean vector to calculate the bivariate marginal density for.
`r`	Index position for x_r within mean vector to calculate the bivariate marginal density for.
`mean`	Mean vector, default is `rep(0, length = nrow(sigma))`.
`sigma`	Covariance matrix, default is `diag(length(mean))`.
`lower`	Vector of lower truncation points, default is `rep(-Inf, length = length(mean))`.

`upper` Vector of upper truncation points, default is `rep( Inf, length = length(mean))`.

`log` Logical; if `TRUE`, densities d are given as log(d).

`pmvnorm.algorithm` Algorithm used for `pmvnorm`

Details

The bivariate marginal density function $f(x_q, x_r)$ for $x \sim TN(\mu, \Sigma, a, b)$ and $q \neq r$ is defined as

$$F_{q,r}(x_q = c_q, x_r = c_r) = \int_{a_1}^{b_1} ... \int_{a_{q-1}}^{b_{q-1}} \int_{a_{q+1}}^{b_{q+1}} ... \int_{a_{r-1}}^{b_{r-1}} \int_{a_{r+1}}^{b_{r+1}} ... \int_{a_k}^{b_k} \varphi_{\alpha\Sigma}(x_s, c_q, c_r) dx_s$$

Author(s)

Stefan Wilhelm <Stefan.Wilhelm@financial.com>, Manjunath B G <bgmanjunath@gmail.com>

References

Tallis, G. M. (1961). The moment generating function of the truncated multinormal distribution. *Journal of the Royal Statistical Society, Series B*, **23**, 223–229

Leppard, P. and Tallis, G. M. (1989). Evaluation of the Mean and Covariance of the Truncated Multinormal *Applied Statistics*, **38**, 543–553

Manjunath B G and Wilhelm, S. (2009). Moments Calculation For the Double Truncated Multivariate Normal Density. Working Paper. Available at SSRN: https://www.ssrn.com/abstract=1472153

Examples

```
lower = c(-0.5, -1, -1)
upper = c( 2.2,  2,  2)

mean  = c(0,0,0)
sigma = matrix(c(2.0, -0.6,  0.7,
                -0.6,  1.0, -0.2,
                 0.7, -0.2,  1.0), 3, 3)

# generate random samples from untruncated and truncated distribution
Y = rmvnorm(10000, mean=mean, sigma=sigma)
X = rtmvnorm(500,  mean=mean, sigma=sigma, lower=lower, upper=upper,
    algorithm="gibbs")

# compute bivariate marginal density of x1 and x2
xq <- seq(lower[1], upper[1], by=0.1)
xr <- seq(lower[2], upper[2], by=0.1)

grid <- matrix(NA, length(xq), length(xr))
for (i in 1:length(xq))
{
  for (j in 1:length(xr))
  {
```

```
    grid[i,j] = dtmvnorm.marginal2(xq=xq[i], xr=xr[j],
      q=1, r=2, sigma=sigma, lower=lower, upper=upper)
  }
}

plot(Y[,1], Y[,2], xlim=c(-4, 4), ylim=c(-4, 4),
   main=expression("bivariate marginal density ("*x[1]*","*x[2]*")"),
   xlab=expression(x[1]), ylab=expression(x[2]), col="gray80")
points(X[,1], X[,2], col="black")

lines(x=c(lower[1], upper[1], upper[1], lower[1], lower[1]),
      y=c(lower[2],lower[2],upper[2],upper[2],lower[2]),
      lty=2, col="red")
contour(xq, xr, grid, add=TRUE, nlevels = 8, col="red", lwd=2)

# scatterplot matrices for untruncated and truncated points
require(lattice)
splom(Y)
splom(X)
```

---

dtmvt *Truncated Multivariate Student t Density*

Description

This function provides the joint density function for the truncated multivariate Student t distribution with mean vector equal to `mean`, covariance matrix `sigma`, degrees of freedom parameter `df` and lower and upper truncation points `lower` and `upper`.

Usage

```
dtmvt(x, mean = rep(0, nrow(sigma)), sigma = diag(length(mean)), df = 1,
lower = rep(-Inf, length = length(mean)),
upper = rep(Inf, length = length(mean)), log = FALSE)
```

Arguments

`x`	Vector or matrix of quantiles. If `x` is a matrix, each row is taken to be a quantile.
`mean`	Mean vector, default is `rep(0, nrow(sigma))`.
`sigma`	Covariance matrix, default is `diag(length(mean))`.
`df`	degrees of freedom parameter
`lower`	Vector of lower truncation points, default is `rep(-Inf, length = length(mean))`.
`upper`	Vector of upper truncation points, default is `rep( Inf, length = length(mean))`.
`log`	Logical; if `TRUE`, densities d are given as log(d).

Details

The Truncated Multivariate Student t Distribution is a conditional Multivariate Student t distribution subject to (linear) constraints $a \leq \boldsymbol{x} \leq b$.

The density of the p-variate Multivariate Student t distribution with ν degrees of freedom is

$$f(\boldsymbol{x}) = \frac{\Gamma((\nu+p)/2)}{(\pi\nu)^{p/2}\Gamma(\nu/2)\|\Sigma\|^{1/2}}[1+\frac{1}{\nu}(x-\mu)^T\Sigma^{-1}(x-\mu)]^{-(\nu+p)/2}$$

The density of the truncated distribution $f_{a,b}(x)$ with constraints $(a \leq x \leq b)$ is accordingly

$$f_{a,b}(x) = \frac{f(\boldsymbol{x})}{P(a \leq x \leq b)}$$

Value

a numeric vector with density values

Author(s)

Stefan Wilhelm <wilhelm@financial.com>

References

Geweke, J. F. (1991) Efficient simulation from the multivariate normal and Student-t distributions subject to linear constraints and the evaluation of constraint probabilities. https://www.researchgate.net/publication/2335219_Efficient_Simulation_from_the_Multivariate_Normal_and_Student-t_Distributions_Subject_to_Linear_Constraints_and_the_Evaluation_of_Constraint_Probabilities

Samuel Kotz, Saralees Nadarajah (2004). Multivariate t Distributions and Their Applications. *Cambridge University Press*

See Also

ptmvt and rtmvt for probabilities and random number generation in the truncated case, see dmvt, rmvt and pmvt for the untruncated multi-t distribution.

Examples

```
# Example

x1 <- seq(-2, 3, by=0.1)
x2 <- seq(-2, 3, by=0.1)

mean <- c(0,0)
sigma <- matrix(c(1, -0.5, -0.5, 1), 2, 2)
lower <- c(-1,-1)


density <- function(x)
{
```

```
 z=dtmvt(x, mean=mean, sigma=sigma, lower=lower)
 z
}

fgrid <- function(x, y, f)
{
 z <- matrix(nrow=length(x), ncol=length(y))
 for(m in 1:length(x)){
  for(n in 1:length(y)){
   z[m,n] <- f(c(x[m], y[n]))
  }
 }
 z
}

# compute multivariate-t density d for grid
d <- fgrid(x1, x2, function(x) dtmvt(x, mean=mean, sigma=sigma, lower=lower))

# compute multivariate normal density d for grid
d2 <- fgrid(x1, x2, function(x) dtmvnorm(x, mean=mean, sigma=sigma, lower=lower))

# plot density as contourplot
contour(x1, x2, d, nlevels=5, main="Truncated Multivariate t Density",
  xlab=expression(x[1]), ylab=expression(x[2]))

contour(x1, x2, d2, nlevels=5, add=TRUE, col="red")
abline(v=-1, lty=3, lwd=2)
abline(h=-1, lty=3, lwd=2)
```

---

gmm.tmvnorm *GMM Estimation for the Truncated Multivariate Normal Distribution*

Description

Generalized Method of Moments (GMM) Estimation for the Truncated Multivariate Normal Distribution

Usage

```
gmm.tmvnorm(X,
  lower = rep(-Inf, length = ncol(X)),
  upper = rep(+Inf, length = ncol(X)),
  start = list(mu = rep(0, ncol(X)), sigma = diag(ncol(X))),
  fixed = list(),
  method=c("ManjunathWilhelm","Lee"),
  cholesky = FALSE,
  ...)
```

Arguments

X	Matrix of quantiles, each row is taken to be a quantile.
lower	Vector of lower truncation points, default is rep(-Inf, length = ncol(X)).
upper	Vector of upper truncation points, default is rep(Inf, length = ncol(X)).
start	Named list with elements mu (mean vector) and sigma (covariance matrix). Initial values for optimizer.
fixed	Named list. Parameter values to keep fixed during optimization.
method	Which set of moment conditions used, possible methods are "ManjunathWilhelm" (default) and "Lee".
cholesky	if TRUE, we use the Cholesky decomposition of sigma as parametrization
...	Further arguments to pass to gmm

Details

This method performs an estimation of the parameters mean and sigma of a truncated multinormal distribution using the Generalized Method of Moments (GMM), when the truncation points lower and upper are known. gmm.tmvnorm() is a wrapper for the general GMM method gmm, so one does not have to specify the moment conditions.

Manjunath/Wilhelm moment conditions
Because the first and second moments can be computed thanks to the mtmvnorm function, we can set up a method-of-moments estimator by equating the sample moments to their population counterparts. This way we have an exactly identified case.

Lee (1979,1983) moment conditions
The recursive moment conditions presented by Lee (1979,1983) are defined for $l = 0, 1, 2, \ldots$ as

$$\sigma^{iT} E(x_i^l \mathbf{x}) = \sigma^{iT} \mu E(x_i^l) + l E(x_i^{l-1}) + \frac{a_i^l F_i(a_i)}{F} - \frac{b_i^l F_i(b_i)}{F}$$

where $E(x_i^l)$ and $E(x_i^l \mathbf{x})$ are the moments of x_i^l and $x_i^l \mathbf{x}$ respectively and $F_i(c)/F$ is the one-dimensional marginal density in variable i as calculated by dtmvnorm.marginal. σ^{iT} is the i-th column of the inverse covariance matrix Σ^{-1}.

This method returns an object of class gmm, for which various diagnostic methods are available, like profile(), confint() etc. See examples.

Value

An object of class gmm

Author(s)

Stefan Wilhelm <wilhelm@financial.com>

References

Tallis, G. M. (1961). The moment generating function of the truncated multinormal distribution. *Journal of the Royal Statistical Society, Series B*, **23**, 223–229

Lee, L.-F. (1979). On the first and second moments of the truncated multi-normal distribution and a simple estimator. *Economics Letters*, **3**, 165–169

Lee, L.-F. (1983). The determination of moments of the doubly truncated multivariate normal Tobit model. *Economics Letters*, **11**, 245–250

Manjunath B G and Wilhelm, S. (2009). Moments Calculation For the Double Truncated Multivariate Normal Density. Working Paper. Available at SSRN: https://www.ssrn.com/abstract=1472153

See Also

gmm

Examples

```
## Not run:
set.seed(1.234)

# the actual parameters
lower <- c(-1, -2)
upper <- c(3, Inf)
mu    <- c(0, 0)
sigma <- matrix(c(1, 0.8,
                  0.8, 2), 2, 2)

# generate random samples
X <- rtmvnorm(n=500, mu, sigma, lower, upper)

# estimate mean vector and covariance matrix sigma from random samples X
# with default start values
gmm.fit1 <- gmm.tmvnorm(X, lower=lower, upper=upper)

# diagnostic output of the estimated parameters
summary(gmm.fit1)
vcov(gmm.fit1)

# confidence intervals
confint(gmm.fit1)

# choosing a different start value
gmm.fit2 <- gmm.tmvnorm(X, lower=lower, upper=upper,
  start=list(mu=c(0.1, 0.1),
  sigma=matrix(c(1, 0.4, 0.4, 1.8),2,2)))
summary(gmm.fit2)

# GMM estimation with Lee (1983) moment conditions
gmm.fit3 <- gmm.tmvnorm(X, lower=lower, upper=upper, method="Lee")
summary(gmm.fit3)
```

```
confint(gmm.fit3)

# MLE estimation for comparison
mle.fit1 <- mle.tmvnorm(X, lower=lower, upper=upper)
confint(mle.fit1)

## End(Not run)
```

`mle.tmvnorm` *Maximum Likelihood Estimation for the Truncated Multivariate Normal Distribution*

Description

Maximum Likelihood Estimation for the Truncated Multivariate Normal Distribution

Usage

```
mle.tmvnorm(X,
  lower = rep(-Inf, length = ncol(X)),
  upper = rep(+Inf, length = ncol(X)),
  start = list(mu = rep(0, ncol(X)), sigma = diag(ncol(X))),
  fixed = list(), method = "BFGS",
  cholesky = FALSE,
  lower.bounds = -Inf,
  upper.bounds = +Inf,
  ...)
```

Arguments

`X`	Matrix of quantiles, each row is taken to be a quantile.
`lower`	Vector of lower truncation points, default is `rep(-Inf, length = ncol(X))`.
`upper`	Vector of upper truncation points, default is `rep( Inf, length = ncol(X))`.
`start`	Named list with elements `mu` (mean vector) and `sigma` (covariance matrix). Initial values for optimizer.
`fixed`	Named list. Parameter values to keep fixed during optimization.
`method`	Optimization method to use. See `optim`
`cholesky`	if TRUE, we use the Cholesky decomposition of `sigma` as parametrization
`lower.bounds`	lower bounds/box constraints for method "L-BFGS-B"
`upper.bounds`	upper bounds/box constraints for method "L-BFGS-B"
`...`	Further arguments to pass to `optim`

Details

This method performs a maximum likelihood estimation of the parameters `mean` and `sigma` of a truncated multinormal distribution, when the truncation points `lower` and `upper` are known. `mle.tmvnorm()` is a wrapper for the general maximum likelihood method `mle`, so one does not have to specify the negative log-likelihood function.

The log-likelihood function for a data matrix X (T x n) can be established straightforward as

$$\log L(X|\mu,\Sigma) = -T \log \alpha(\mu,\Sigma) + -T/2 \log \|\Sigma\| - \frac{1}{2} \sum_{t=1}^{T} (x_t - \mu)' \Sigma^{-1} (x_t - \mu)$$

As `mle`, this method returns an object of class `mle`, for which various diagnostic methods are available, like `profile()`, `confint()` etc. See examples.

In order to adapt the estimation problem to `mle`, the named parameters for mean vector elements are "mu_i" and the elements of the covariance matrix are "sigma_ij" for the lower triangular matrix elements, i.e. (j <= i).

Value

An object of class `mle-class`

Author(s)

Stefan Wilhelm <wilhelm@financial.com>

See Also

`mle` and `mle-class`

Examples

```
## Not run:
set.seed(1.2345)

# the actual parameters
lower <- c(-1,-1)
upper <- c(1, 2)
mu    <- c(0, 0)
sigma <- matrix(c(1, 0.7,
                  0.7, 2), 2, 2)

# generate random samples
X <- rtmvnorm(n=500, mu, sigma, lower, upper)
method <- "BFGS"

# estimate mean vector and covariance matrix sigma from random samples X
# with default start values
mle.fit1 <- mle.tmvnorm(X, lower=lower, upper=upper)

# diagnostic output of the estimated parameters
```

```
summary(mle.fit1)
logLik(mle.fit1)
vcov(mle.fit1)

# profiling the log likelihood and confidence intervals
mle.profile1 <- profile(mle.fit1, X, method="BFGS", trace=TRUE)
confint(mle.profile1)

par(mfrow=c(3,2))
plot(mle.profile1)

# choosing a different start value
mle.fit2 <- mle.tmvnorm(X, lower=lower, upper=upper,
  start=list(mu=c(0.1, 0.1),
  sigma=matrix(c(1, 0.4, 0.4, 1.8),2,2)))
summary(mle.fit2)

## End(Not run)
```

`mtmvnorm` *Computation of Mean Vector and Covariance Matrix For Truncated Multivariate Normal Distribution*

Description

Computation of the first two moments, i.e. mean vector and covariance matrix for the Truncated Multivariate Normal Distribution based on the works of Tallis (1961), Lee (1979) and Leppard and Tallis (1989), but extended to the double-truncated case with general mean and general covariance matrix.

Usage

```
mtmvnorm(mean = rep(0, nrow(sigma)),
  sigma = diag(length(mean)),
  lower = rep(-Inf, length = length(mean)),
  upper = rep(Inf, length = length(mean)),
  doComputeVariance=TRUE,
  pmvnorm.algorithm=GenzBretz())
```

Arguments

`mean`	Mean vector, default is `rep(0, length = ncol(x))`.
`sigma`	Covariance matrix, default is `diag(ncol(x))`.
`lower`	Vector of lower truncation points,\ default is `rep(-Inf, length = length(mean))`.
`upper`	Vector of upper truncation points,\ default is `rep( Inf, length = length(mean))`.
`doComputeVariance`	flag whether to compute the variance for users who are interested only in the mean. Defaults to `TRUE` for backward compatibility.

`pmvnorm.algorithm`
: Algorithm used for `pmvnorm`

Details

Details for the moment calculation under double truncation and the derivation of the formula can be found in the Manjunath/Wilhelm (2009) working paper. If only a subset of variables are truncated, we calculate the truncated moments only for these and use the Johnson/Kotz formula for the remaining untruncated variables.

Value

`tmean`	Mean vector of truncated variables
`tvar`	Covariance matrix of truncated variables

Author(s)

Stefan Wilhelm <Stefan.Wilhelm@financial.com>, Manjunath B G <bgmanjunath@gmail.com>

References

Tallis, G. M. (1961). The moment generating function of the truncated multinormal distribution. *Journal of the Royal Statistical Society, Series B*, **23**, 223–229

Johnson, N./Kotz, S. (1970). Distributions in Statistics: Continuous Multivariate Distributions *Wiley & Sons*, pp. 70–73

Lee, L.-F. (1979). On the first and second moments of the truncated multi-normal distribution and a simple estimator. *Economics Letters*, **3**, 165–169

Leppard, P. and Tallis, G. M. (1989). Evaluation of the Mean and Covariance of the Truncated Multinormal. *Applied Statistics*, **38**, 543–553

Manjunath B G and Wilhelm, S. (2009). Moments Calculation For the Double Truncated Multivariate Normal Density. Working Paper. Available at SSRN: https://www.ssrn.com/abstract=1472153

Examples

```
mu    <- c(0.5, 0.5, 0.5)
sigma <- matrix(c(  1,  0.6, 0.3,
                  0.6,    1, 0.2,
                  0.3,  0.2,   2), 3, 3)

a  <- c(-Inf, -Inf, -Inf)
b  <- c(1, 1, 1)

# compute first and second moments
mtmvnorm(mu, sigma, lower=a, upper=b)

# compare with simulated results
X <- rtmvnorm(n=1000, mean=mu, sigma=sigma, lower=a, upper=b)
colMeans(X)
cov(X)
```

ptmvnorm *Truncated Multivariate Normal Distribution*

Description

Computes the distribution function of the truncated multivariate normal distribution for arbitrary limits and correlation matrices based on the pmvnorm() implementation of the algorithms by Genz and Bretz.

Usage

```
ptmvnorm(lowerx, upperx, mean=rep(0, length(lowerx)), sigma,
  lower = rep(-Inf, length = length(mean)),
  upper = rep( Inf, length = length(mean)),
  maxpts = 25000, abseps = 0.001, releps = 0)
```

Arguments

lowerx	the vector of lower limits of length n.
upperx	the vector of upper limits of length n.
mean	the mean vector of length n.
sigma	the covariance matrix of dimension n. Either corr or sigma can be specified. If sigma is given, the problem is standardized. If neither corr nor sigma is given, the identity matrix is used for sigma.
lower	Vector of lower truncation points,\ default is rep(-Inf, length = length(mean)).
upper	Vector of upper truncation points,\ default is rep(Inf, length = length(mean)).
maxpts	maximum number of function values as integer.
abseps	absolute error tolerance as double.
releps	relative error tolerance as double.

Details

The computation of truncated multivariate normal probabilities and densities is done using conditional probabilities from the standard/untruncated multivariate normal distribution. So we refer to the documentation of the mvtnorm package and the methodology is described in Genz (1992, 1993) and Genz/Bretz (2009).

For properties of the truncated multivariate normal distribution see for example Johnson/Kotz (1970) and Horrace (2005).

Value

The evaluated distribution function is returned with attributes

error	estimated absolute error and
msg	status messages.

References

Genz, A. (1992). Numerical computation of multivariate normal probabilities. *Journal of Computational and Graphical Statistics*, **1**, 141–150

Genz, A. (1993). Comparison of methods for the computation of multivariate normal probabilities. *Computing Science and Statistics*, **25**, 400–405

Genz, A. and Bretz, F. (2009). Computation of Multivariate Normal and t Probabilities. *Lecture Notes in Statistics*, Vol. **195**, Springer-Verlag, Heidelberg.

Johnson, N./Kotz, S. (1970). Distributions in Statistics: Continuous Multivariate Distributions *Wiley & Sons*, pp. 70–73

Horrace, W. (2005). Some Results on the Multivariate Truncated Normal Distribution. *Journal of Multivariate Analysis*, **94**, 209–221

Examples

```
sigma <- matrix(c(5, 0.8, 0.8, 1), 2, 2)
Fx <- ptmvnorm(lowerx=c(-1,-1), upperx=c(0.5,0), mean=c(0,0),
  sigma=sigma, lower=c(-1,-1), upper=c(1,1))
```

---

ptmvt *Truncated Multivariate Student t Distribution*

Description

Computes the distribution function of the truncated multivariate t distribution

Usage

```
ptmvt(lowerx, upperx, mean = rep(0, length(lowerx)), sigma, df = 1,
  lower = rep(-Inf, length = length(mean)),
  upper = rep(Inf, length = length(mean)), maxpts = 25000, abseps = 0.001,
  releps = 0)
```

Arguments

lowerx	the vector of lower limits of length n.
upperx	the vector of upper limits of length n.
mean	the mean vector of length n.
sigma	the covariance matrix of dimension n. Either corr or sigma can be specified. If sigma is given, the problem is standardized. If neither corr nor sigma is given, the identity matrix is used for sigma.
df	Degrees of freedom parameter
lower	Vector of lower truncation points, default is rep(-Inf, length = length(mean)).
upper	Vector of upper truncation points, default is rep(Inf, length = length(mean)).
maxpts	maximum number of function values as integer.
abseps	absolute error tolerance as double.
releps	relative error tolerance as double.

Value

The evaluated distribution function is returned with attributes

`error`	estimated absolute error and
`msg`	status messages.

Author(s)

Stefan Wilhelm <Stefan.Wilhelm@financial.com>

References

Geweke, J. F. (1991) Efficient simulation from the multivariate normal and Student-t distributions subject to linear constraints and the evaluation of constraint probabilities. https://www.researchgate.net/publication/2335219_Efficient_Simulation_from_the_Multivariate_Normal_and_Student-t_Distributions_Subject_to_Linear_Constraints_and_the_Evaluation_of_Constraint_Probabilities

Samuel Kotz, Saralees Nadarajah (2004). Multivariate t Distributions and Their Applications. *Cambridge University Press*

Examples

```
sigma <- matrix(c(5, 0.8, 0.8, 1), 2, 2)
Fx <- ptmvt(lowerx=c(-1,-1), upperx=c(0.5,0), mean=c(0,0), sigma=sigma, df=3,
  lower=c(-1,-1), upper=c(1,1))
```

---

ptmvtnorm.marginal *One-dimensional marginal CDF function for a Truncated Multivariate Normal and Student t distribution*

Description

This function computes the one-dimensional marginal probability function from a Truncated Multivariate Normal and Student t density function using integration in `pmvnorm()` and `pmvt()`.

Usage

```
ptmvnorm.marginal(xn,
    n = 1,
    mean = rep(0, nrow(sigma)),
    sigma = diag(length(mean)),
    lower = rep(-Inf, length = length(mean)),
    upper = rep(Inf, length = length(mean)))
ptmvt.marginal(xn,
    n = 1,
    mean = rep(0, nrow(sigma)),
    sigma = diag(length(mean)),
```

```
    df = 1,
    lower = rep(-Inf, length = length(mean)),
    upper = rep(Inf, length = length(mean)))
```

Arguments

xn	Vector of quantiles to calculate the marginal probability for.
n	Index position (1..k) within the random vector xn to calculate the one-dimensional marginal probability for.
mean	the mean vector of length k.
sigma	the covariance matrix of dimension k. Either corr or sigma can be specified. If sigma is given, the problem is standardized. If neither corr nor sigma is given, the identity matrix is used for sigma.
df	degrees of freedom parameter
lower	Vector of lower truncation points, default is rep(-Inf, length = length(mean)).
upper	Vector of upper truncation points, default is rep(Inf, length = length(mean)).

Details

The one-dimensional marginal probability for index i is $F_i(x_i) = P(X_i \leq x_i)$

$$F_i(x_i) = \int_{a_1}^{b_1} \dots \int_{a_{i-1}}^{b_{i-1}} \int_{a_i}^{x_i} \int_{a_{i+1}}^{b_{i+1}} \dots \int_{a_k}^{b_k} f(x)dx = \alpha^{-1}\Phi_k(a, u, \mu, \Sigma)$$

where $u = (b_1, \dots, b_{i-1}, x_i, b_{i+1}, \dots, b_k)'$ is the upper integration bound and Φ_k is the k-dimensional normal probability (i.e. functions pmvnorm() and pmvt() in R package mvtnorm).

Value

Returns a vector of the same length as xn with probabilities.

Author(s)

Stefan Wilhelm <Stefan.Wilhelm@financial.com>

Examples

```
## Example 1: Truncated multi-normal
lower <- c(-1,-1,-1)
upper <- c(1,1,1)
mean <- c(0,0,0)
sigma <- matrix(c(  1, 0.8, 0.2,
                  0.8,   1, 0.1,
                  0.2, 0.1,   1), 3, 3)

X <- rtmvnorm(n=1000, mean=c(0,0,0), sigma=sigma, lower=lower, upper=upper)

x <- seq(-1, 1, by=0.01)
Fx <- ptmvnorm.marginal(xn=x, n=1, mean=c(0,0,0), sigma=sigma, lower=lower, upper=upper)
```

```
plot(ecdf(X[,1]), main="marginal CDF for truncated multi-normal")
lines(x, Fx, type="l", col="blue")

## Example 2: Truncated multi-t
X <- rtmvt(n=1000, mean=c(0,0,0), sigma=sigma, df=2, lower=lower, upper=upper)

x <- seq(-1, 1, by=0.01)
Fx <- ptmvt.marginal(xn=x, n=1, mean=c(0,0,0), sigma=sigma, lower=lower, upper=upper)

plot(ecdf(X[,1]), main="marginal CDF for truncated multi-t")
lines(x, Fx, type="l", col="blue")
```

`qtmvnorm-marginal` *Quantiles of the Truncated Multivariate Normal Distribution in one dimension*

Description

Computes the equicoordinate quantile function of the truncated multivariate normal distribution for arbitrary correlation matrices based on an inversion of the algorithms by Genz and Bretz.

Usage

```
qtmvnorm.marginal(p,
        interval = c(-10, 10),
        tail = c("lower.tail","upper.tail","both.tails"),
        n=1,
  mean=rep(0, nrow(sigma)),
  sigma=diag(length(mean)),
  lower=rep(-Inf, length = length(mean)),
  upper=rep( Inf, length = length(mean)),
  ...)
```

Arguments

`p`	probability.
`interval`	a vector containing the end-points of the interval to be searched by `uniroot`.
`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 $P[-x \leq X \leq x] = p$
`n`	index (1..n) to calculate marginal quantile for
`mean`	the mean vector of length n.
`sigma`	the covariance matrix of dimension n. Either `corr` or `sigma` can be specified. If `sigma` is given, the problem is standardized. If neither `corr` nor `sigma` is given, the identity matrix is used for `sigma`.

lower	Vector of lower truncation points,\ default is rep(-Inf, length = length(mean)).
upper	Vector of upper truncation points,\ default is rep(Inf, length = length(mean)).
...	additional parameters to be passed to uniroot.

Details

Only equicoordinate quantiles are computed, i.e., the quantiles in each dimension coincide. Currently, the distribution function is inverted by using the uniroot function which may result in limited accuracy of the quantiles.

Value

A list with four components: quantile and f.quantile give the location of the quantile and the value of the function evaluated at that point. iter and estim.prec give the number of iterations used and an approximate estimated precision from uniroot.

See Also

ptmvnorm, pmvnorm

Examples

```
# finite dimensional distribution of the Geometric Brownian Motion log-returns
# with truncation

# volatility p.a.
sigma=0.4

# risk free rate
r = 0.05

# n=3 points in time
T <- c(0.5, 0.7, 1)

# covariance matrix of Geometric Brownian Motion returns
Sigma = sigma^2*outer(T,T,pmin)

# mean vector of the Geometric Brownian Motion returns
mu    = (r - sigma^2/2) * T

# lower truncation vector a (a<=x<=b)
a = rep(-Inf, 3)

# upper truncation vector b (a<=x<=b)
b = c(0, 0, Inf)

# quantile of the t_1 returns
qtmvnorm.marginal(p=0.95, interval = c(-10, 10), tail = "lower.tail", n=1,
  mean  = mu, sigma = Sigma, lower=a, upper=b)
```

`rtmvnorm` *Sampling Random Numbers From The Truncated Multivariate Normal Distribution*

Description

This function generates random numbers from the truncated multivariate normal distribution with mean equal to `mean` and covariance matrix `sigma` (or alternatively precision matrix `H`), lower and upper truncation points `lower` and `upper` with either rejection sampling or Gibbs sampling.

Usage

```
rtmvnorm(n, mean = rep(0, nrow(sigma)),
  sigma = diag(length(mean)),
  lower=rep(-Inf, length = length(mean)),
  upper=rep( Inf, length = length(mean)),
  D = diag(length(mean)),
  H = NULL,
  algorithm=c("rejection", "gibbs", "gibbsR"),
  ...)

rtmvnorm.sparseMatrix(n, mean = rep(0, nrow(H)),
    H = sparseMatrix(i=1:length(mean), j=1:length(mean), x=1),
    lower = rep(-Inf, length = length(mean)),
    upper = rep( Inf, length = length(mean)),
    ...)
```

Arguments

`n`	Number of random points to be sampled. Must be an integer ≥ 1.
`mean`	Mean vector, default is `rep(0, length = ncol(x))`.
`sigma`	Covariance matrix, default is `diag(ncol(x))`.
`lower`	Vector of lower truncation points, default is `rep(-Inf, length = length(mean))`.
`upper`	Vector of upper truncation points, default is `rep( Inf, length = length(mean))`.
`D`	Matrix for linear constraints, defaults to diagonal matrix.
`H`	Precision matrix, default is `NULL`.
`algorithm`	Method used, possible methods are rejection sampling ("rejection", default), the Fortan Gibbs sampler ("gibbs") and the old Gibbs sampler implementation in R ("gibbsR").
`...`	additional parameters for Gibbs sampling, given to the internal method `rtmvnorm.gibbs()`, such as `burn.in.samples`, `start.value` and `thinning`, see details

Details

The generation of random numbers from a truncated multivariate normal distribution is done using either rejection sampling or Gibbs sampling.

Rejection sampling
Rejection sampling is done from the standard multivariate normal distribution. So we use the function `rmvnorm` of the **mvtnorm** package to generate proposals which are either accepted if they are inside the support region or rejected. In order to speed up the generation of N samples from the truncated distribution, we first calculate the acceptance rate alpha from the truncation points and then generate N/alpha samples iteratively until we have got N samples. This typically does not take more than 2-3 iterations. Rejection sampling may be very inefficient when the support region is small (i.e. in higher dimensions) which results in very low acceptance rates alpha. In this case the Gibbs sampler is preferable.

Gibbs sampling
The Gibbs sampler samples from univariate conditional distributions, so all samples can be accepted except for a burn-in period. The number of burn-in samples to be discarded can be specified, as well as a start value of the chain. If no start value is given, we determine a start value from the support region using either lower bound or upper bound if they are finite, or 0 otherwise.

The Gibbs sampler has been reimplemented in Fortran 90 for performance reasons (`algorithm="gibbs"`). The old R implementation is still accessible through `algorithm="gibbsR"`.

The arguments to be passed along with `algorithm="gibbs"` or `algorithm="gibbsR"` are:

`burn.in.samples` number of samples in Gibbs sampling to be discarded as burn-in phase, must be non-negative.

`start.value` Start value (vector of length `length(mean)`) for the MCMC chain. If one is specified, it must lie inside the support region ($lower <= start.value <= upper$). If none is specified, the start value is taken componentwise as the finite lower or upper boundaries respectively, or zero if both boundaries are infinite. Defaults to NULL.

`thinning` Thinning factor for reducing autocorrelation of random points in Gibbs sampling. Must be an integer >= 1. We create a Markov chain of length `(n*thinning)` and take only those samples `j=1:(n*thinning)` where `j %% thinning == 0` Defaults to 1 (no thinning of the chain).

Sampling with linear constraints
We extended the method to also simulate from a multivariate normal distribution subject to general linear constraints $lower <= Dx <= upper$. For general D, both rejection sampling or Gibbs sampling according to Geweke (1991) are available.

Gibbs sampler and the use of the precision matrix H
Why is it important to have a random sampler that works with the precision matrix? Especially in Bayesian and spatial statistics, there are a number of high-dimensional applications where the precision matrix `H` is readily available, but is sometimes nearly singular and cannot be easily inverted to sigma. Additionally, it turns out that the Gibbs sampler formulas are much simpler in terms of the precision matrix than in terms of the covariance matrix. See the details of the Gibbs sampler implementation in the package vignette or for example Geweke (2005), pp.171-172. (Thanks to Miguel Godinho de Matos from Carnegie Mellon University for pointing me to this.) Therefore, we now provide an interface for the direct use of the precision matrix `H` in `rtmvnorm()`.

Gibbs sampler with sparse precision matrix H
The size of the covariance matrix `sigma` or precision matrix `H` - if expressed as a dense `matrix` -

grows quadratic with the number of dimensions d. For high-dimensional problems (such as d > 5000), it is no longer efficient and appropriate to work with dense matrix representations, as one quickly runs into memory problems.
It is interesting to note that in many applications the precision matrix, which holds the conditional dependencies, will be sparse, whereas the covariance matrix will be dense. Hence, expressing H as a sparse matrix will significantly reduce the amount of memory to store this matrix and allows much larger problems to be handled. In the current version of the package, the precision matrix (not `sigma` since it will be dense in most cases) can be passed to `rtmvnorm.sparseMatrix()` as a `sparseMatrix` from the `Matrix` package. See the examples section below for a usage example.

Warning

A word of caution is needed for useRs that are not familiar with Markov Chain Monte Carlo methods like Gibbs sampling:

Rejection sampling is exact in the sense that we are sampling directly from the target distribution and the random samples generated are independent. So it is clearly the default method.

Markov Chain Monte Carlo methods are only approximate methods, which may suffer from several problems:

- Poor mixing
- Convergence problems
- Correlation among samples

Diagnostic checks for Markov Chain Monte Carlo include trace plots, CUSUM plots and autocorrelation plots like `acf`. For a survey see for instance Cowles (1996).

That is, consecutive samples generated from `rtmvnorm(..., algorithm=c("gibbs", "gibbsR"))` are correlated (see also example 3 below). One way of reducing the autocorrelation among the random samples is "thinning" the Markov chain, that is recording only a subset/subsequence of the chain. For example, one could record only every 100th sample, which clearly reduces the autocorrelation and "increases the independence". But thinning comes at the cost of higher computation times, since the chain has to run much longer. We refer to autocorrelation plots in order to determine optimal thinning.

Author(s)

Stefan Wilhelm <Stefan.Wilhelm@financial.com>, Manjunath B G <bgmanjunath@gmail.com>

References

Alan Genz, Frank Bretz, Tetsuhisa Miwa, Xuefei Mi, Friedrich Leisch, Fabian Scheipl, Torsten Hothorn (2009). mvtnorm: Multivariate Normal and t Distributions. R package version 0.9-7. URL https://CRAN.R-project.org/package=mvtnorm

Johnson, N./Kotz, S. (1970). Distributions in Statistics: Continuous Multivariate Distributions *Wiley & Sons*, pp. 70–73

Horrace, W. (2005). Some Results on the Multivariate Truncated Normal Distribution. *Journal of Multivariate Analysis*, **94**, 209–221

Jayesh H. Kotecha and Petar M. Djuric (1999). Gibbs Sampling Approach For Generation of Truncated Multivariate Gaussian Random Variables *IEEE Computer Society*, 1757–1760

Cowles, M. and Carlin, B. (1996). Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review *Journal of the American Statistical Association*, **91**, 883–904

Geweke, J. F. (1991). Effcient Simulation from the Multivariate Normal and Student-t Distributions Subject to Linear Constraints *Computer Science and Statistics. Proceedings of the 23rd Symposium on the Interface. Seattle Washington, April 21-24, 1991*, 571–578

Geweke, J. F. (2005). Contemporary Bayesian Econometrics and Statistics, *Wiley & Sons*, pp.171–172

See Also

ptmvnorm, pmvnorm, rmvnorm, dmvnorm

Examples

```
################################################################################
#
# Example 1:
# rejection sampling in 2 dimensions
#
################################################################################

sigma <- matrix(c(4,2,2,3), ncol=2)
x <- rtmvnorm(n=500, mean=c(1,2), sigma=sigma, upper=c(1,0))
plot(x, main="samples from truncated bivariate normal distribution",
  xlim=c(-6,6), ylim=c(-6,6),
  xlab=expression(x[1]), ylab=expression(x[2]))
abline(v=1, lty=3, lwd=2, col="gray")
abline(h=0, lty=3, lwd=2, col="gray")

################################################################################
#
# Example 2:
# Gibbs sampler for 4 dimensions
#
################################################################################

C <- matrix(0.8, 4, 4)
diag(C) <- rep(1, 4)
lower <- rep(-4, 4)
upper <- rep(-1, 4)

# acceptance rate alpha
alpha <- pmvnorm(lower=lower, upper=upper, mean=rep(0,4), sigma=C)
alpha

# Gibbs sampler
X1 <- rtmvnorm(n=20000, mean = rep(0,4), sigma=C, lower=lower, upper=upper,
  algorithm="gibbs", burn.in.samples=100)
# Rejection sampling
X2 <- rtmvnorm(n=5000, mean = rep(0,4), sigma=C, lower=lower, upper=upper)
```

```
colMeans(X1)
colMeans(X2)

plot(density(X1[,1], from=lower[1], to=upper[1]), col="red", lwd=2,
     main="Kernel density estimates from random samples
     generated by Gibbs vs. Rejection sampling")
lines(density(X2[,1], from=lower[1], to=upper[1]), col="blue", lwd=2)
legend("topleft",legend=c("Gibbs Sampling","Rejection Sampling"),
  col=c("red","blue"), lwd=2, bty="n")

###############################################################################
#
# Example 3:
# Autocorrelation plot for Gibbs sampler
# with and without thinning
#
###############################################################################

sigma <- matrix(c(4,2,2,3), ncol=2)
X1 <- rtmvnorm(n=10000, mean=c(1,2), sigma=sigma, upper=c(1,0),
  algorithm="rejection")
acf(X1)
# no autocorrelation among random points

X2 <- rtmvnorm(n=10000, mean=c(1,2), sigma=sigma, upper=c(1,0),
  algorithm="gibbs")
acf(X2)
# exhibits autocorrelation among random points

X3 <- rtmvnorm(n=10000, mean=c(1,2), sigma=sigma, upper=c(1,0),
  algorithm="gibbs", thinning=2)
acf(X3)
# reduced autocorrelation among random points

plot(density(X1[,1], to=1))
lines(density(X2[,1], to=1), col="blue")
lines(density(X3[,1], to=1), col="red")

###############################################################################
#
# Example 4: Univariate case
#
###############################################################################

X <- rtmvnorm(100, mean=0, sigma=1, lower=-1, upper=1)

###############################################################################
#
# Example 5: Linear Constraints
#
###############################################################################

mean  <- c(0, 0)
```

```
sigma <- matrix(c(10, 0,
                  0, 1), 2, 2)

# Linear Constraints
#
# a1 <= x1 + x2 <= b2
# a2 <= x1 - x2 <= b2
#
# [ a1 ] <= [ 1   1 ] [ x1 ] <= [b1]
# [ a2 ]    [ 1  -1 ] [ x2 ]    [b2]
a     <- c(-2, -2)
b     <- c( 2,  2)
D     <- matrix(c(1, 1,
                  1, -1), 2, 2)

X <- rtmvnorm(n=10000, mean, sigma, lower=a, upper=b, D=D, algorithm="gibbsR")
plot(X, main="Gibbs sampling for multivariate normal
              with linear constraints according to Geweke (1991)")

# mark linear constraints as lines
for (i in 1:nrow(D)) {
  abline(a=a[i]/D[i, 2], b=-D[i,1]/D[i, 2], col="red")
  abline(a=b[i]/D[i, 2], b=-D[i,1]/D[i, 2], col="red")
}

###############################################################################
#
# Example 6: Using precision matrix H rather than sigma
#
###############################################################################

lower <- c(-1, -1)
upper <- c(1, 1)
mean <- c(0.5, 0.5)
sigma <- matrix(c(1, 0.8, 0.8, 1), 2, 2)
H <- solve(sigma)
D <- matrix(c(1, 1, 1, -1), 2, 2)
X <- rtmvnorm(n=1000, mean=mean, H=H, lower=lower, upper=upper, D=D, algorithm="gibbs")
plot(X, main="Gibbs sampling with precision matrix and linear constraints")

###############################################################################
#
# Example 7: Using sparse precision matrix H in high dimensions
#
###############################################################################

## Not run:
d <- 1000
I_d <- sparseMatrix(i=1:d, j=1:d, x=1)
W <- sparseMatrix(i=c(1:d, 1:(d-1)), j=c(1:d, (2:d)), x=0.5)
H <- t(I_d - 0.5 * W)
lower <- rep(0, d)
upper <- rep(2, d)
```

```
# Gibbs sampler generates n=100 draws in d=1000 dimensions
X <- rtmvnorm.sparseMatrix(n=100, mean = rep(0,d), H=H, lower=lower, upper=upper,
  burn.in.samples=100)
colMeans(X)
cov(X)

## End(Not run)
```

---

rtmvnorm2 *Sampling Random Numbers From The Truncated Multivariate Normal Distribution With Linear Constraints*

Description

This function generates random numbers from the truncated multivariate normal distribution with mean equal to mean and covariance matrix sigma and general linear constraints

$$lower \leq Dx \leq upper$$

with either rejection sampling or Gibbs sampling.

Usage

```
rtmvnorm2(n, mean = rep(0, nrow(sigma)),
  sigma = diag(length(mean)),
  lower = rep(-Inf, length = length(mean)),
  upper = rep(Inf, length = length(mean)),
  D = diag(length(mean)),
  algorithm = c("gibbs", "gibbsR", "rejection"), ...)
```

Arguments

n	Number of random points to be sampled. Must be an integer ≥ 1.
mean	Mean vector (d x 1), default is rep(0, length = ncol(x)).
sigma	Covariance matrix (d x d), default is diag(ncol(x)).
lower	Vector of lower truncation points (r x 1), default is rep(Inf, length = length(mean)).
upper	Vector of upper truncation points (r x 1), default is rep(Inf, length = length(mean)).
D	Matrix for linear constraints (r x d), defaults to diagonal matrix (d x d), i.e. r = d.
algorithm	Method used, possible methods are the Fortan Gibbs sampler ("gibbs", default), the Gibbs sampler implementation in R ("gibbsR") and rejection sampling ("rejection")
...	additional parameters for Gibbs sampling, given to the internal method rtmvnorm.gibbs(), such as burn.in.samples, start.value and thinning, see details in rtmvnorm

Details

This method allows for $r > d$ linear constraints, whereas `rtmvnorm` requires a full-rank matrix D $(d \times d)$ and can only handle $r \leq d$ constraints at the moment. The lower and upper bounds `lower` and `upper` are $(r \times 1)$, the matrix `D` is $(r \times d)$ and `x` is $(d \times 1)$. The default case is $r = d$ and $D = I_d$.

Warning

This method will be merged with `rtmvnorm` in one of the next releases.

Author(s)

Stefan Wilhelm

See Also

`rtmvnorm`

Examples

```
## Not run:
##############################################################################
#
# Example 5a: Number of linear constraints r > dimension d
#
##############################################################################

# general linear restrictions a <= Dx <= b with x (d x 1); D (r x d); a,b (r x 1)

# Dimension d=2, r=3 linear constraints
#
# a1 <=    x1 + x2 <= b2
# a2 <=    x1 - x2 <= b2
# a3 <= 0.5x1 - x2 <= b3
#
# [ a1 ] <= [ 1     1 ] [ x1 ] <= [b1]
# [ a2 ]    [ 1    -1 ] [ x2 ]    [b2]
# [ a3 ]    [ 0.5  -1 ]           [b3]

D <- matrix(
      c(  1,  1,
          1, -1,
        0.5, -1), 3, 2, byrow=TRUE)
a <- c(0, 0, 0)
b <- c(1, 1, 1)

# mark linear constraints as lines
plot(NA, xlim=c(-0.5, 1.5), ylim=c(-1,1))
for (i in 1:3) {
  abline(a=a[i]/D[i, 2], b=-D[i,1]/D[i, 2], col="red")
  abline(a=b[i]/D[i, 2], b=-D[i,1]/D[i, 2], col="red")
```

```
}

### Gibbs sampling for general linear constraints a <= Dx <= b
mean <- c(0, 0)
sigma <- matrix(c(1.0, 0.2,
                  0.2, 1.0), 2, 2)
x0 <- c(0.5, 0.2) # Gibbs sampler start value
X <- rtmvnorm2(n=1000, mean, sigma, lower=a, upper=b, D, start.value=x0)

# show random points within simplex
points(X, pch=20, col="black")

## End(Not run)
```

`rtmvt` *Sampling Random Numbers From The Truncated Multivariate Student t Distribution*

Description

This function generates random numbers from the truncated multivariate Student-t distribution with mean equal to `mean` and covariance matrix `sigma`, lower and upper truncation points `lower` and `upper` with either rejection sampling or Gibbs sampling.

Usage

```
rtmvt(n, mean = rep(0, nrow(sigma)), sigma = diag(length(mean)), df = 1,
  lower = rep(-Inf, length = length(mean)),
  upper = rep(Inf, length = length(mean)),
  algorithm=c("rejection", "gibbs"), ...)
```

Arguments

`n`	Number of random points to be sampled. Must be an integer >= 1.
`mean`	Mean vector, default is `rep(0, length = ncol(x))`.
`sigma`	Covariance matrix, default is `diag(ncol(x))`.
`df`	Degrees of freedom parameter (positive, may be non-integer)
`lower`	Vector of lower truncation points,\ default is `rep(-Inf, length = length(mean))`.
`upper`	Vector of upper truncation points,\ default is `rep( Inf, length = length(mean))`.
`algorithm`	Method used, possible methods are rejection sampling ("rejection", default) and the R Gibbs sampler ("gibbs").
`...`	additional parameters for Gibbs sampling, given to the internal method `rtmvt.gibbs()`, such as `burn.in.samples`, `start.value` and `thinning`, see details

Details

We sample $x \sim T(\mu, \Sigma, df)$ subject to the rectangular truncation $lower \leq x \leq upper$. Currently, two random number generation methods are implemented: rejection sampling and the Gibbs Sampler.

For rejection sampling `algorithm="rejection"`, we sample from `rmvt` and retain only samples inside the support region. The acceptance probability will be calculated with `pmvt`. `pmvt` does only accept integer degrees of freedom `df`. For non-integer `df`, `algorithm="rejection"` will throw an error, so please use `algorithm="gibbs"` instead.

The arguments to be passed along with `algorithm="gibbs"` are:

`burn.in.samples` number of samples in Gibbs sampling to be discarded as burn-in phase, must be non-negative.

`start.value` Start value (vector of length `length(mean)`) for the MCMC chain. If one is specified, it must lie inside the support region ($lower \leq start.value \leq upper$). If none is specified, the start value is taken componentwise as the finite lower or upper boundaries respectively, or zero if both boundaries are infinite. Defaults to NULL.

`thinning` Thinning factor for reducing autocorrelation of random points in Gibbs sampling. Must be an integer ≥ 1. We create a Markov chain of length `(n*thinning)` and take only those samples `j=1:(n*thinning)` where `j %% thinning == 0` Defaults to 1 (no thinning of the chain).

Warning

The same warnings for the Gibbs sampler apply as for the method `rtmvnorm`.

Author(s)

Stefan Wilhelm <Stefan.Wilhelm@financial.com>, Manjunath B G <bgmanjunath@gmail.com>

References

Geweke, John F. (1991) Efficient Simulation from the Multivariate Normal and Student-t Distributions Subject to Linear Constraints. *Computer Science and Statistics. Proceedings of the 23rd Symposium on the Interface. Seattle Washington, April 21-24, 1991*, pp. 571–578 An earlier version of this paper is available at https://www.researchgate.net/publication/2335219_Efficient_Simulation_from_the_Multivariate_Normal_and_Student-t_Distributions_Subject_to_Linear_Constraints_and_the_Evaluation_of_Constraint_Probabilities

Examples

```
############################################################
#
# Example 1
#
############################################################

# Draw from multi-t distribution without truncation
X1 <- rtmvt(n=10000, mean=rep(0, 2), df=2)
X2 <- rtmvt(n=10000, mean=rep(0, 2), df=2, lower=c(-1,-1), upper=c(1,1))
```

```
############################################################
#
# Example 2
#
############################################################

df = 2
mu = c(1,1,1)
sigma = matrix(c(  1, 0.5, 0.5,
                 0.5,   1, 0.5,
                 0.5, 0.5,   1), 3, 3)
lower = c(-2,-2,-2)
upper = c(2, 2, 2)

# Rejection sampling
X1 <- rtmvt(n=10000, mu, sigma, df, lower, upper)

# Gibbs sampling without thinning
X2 <- rtmvt(n=10000, mu, sigma, df, lower, upper,
  algorithm="gibbs")

# Gibbs sampling with thinning
X3 <- rtmvt(n=10000, mu, sigma, df, lower, upper,
  algorithm="gibbs", thinning=2)

plot(density(X1[,1], from=lower[1], to=upper[1]), col="red", lwd=2,
  main="Gibbs vs. Rejection")
lines(density(X2[,1], from=lower[1], to=upper[1]), col="blue", lwd=2)
legend("topleft",legend=c("Rejection Sampling","Gibbs Sampling"),
  col=c("red","blue"), lwd=2)

acf(X1)  # no autocorrelation in Rejection sampling
acf(X2)  # strong autocorrelation of Gibbs samples
acf(X3)  # reduced autocorrelation of Gibbs samples after thinning
```

`tmvnorm` *Truncated Multivariate Normal Density*

Description

This function provides the joint density function for the truncated multivariate normal distribution with mean equal to `mean` and covariance matrix `sigma`, lower and upper truncation points `lower` and `upper`. For convenience, it furthermore serves as a wrapper function for the one-dimensional and bivariate marginal densities `dtmvnorm.marginal()` and `dtmvnorm.marginal2()` respectively when invoked with the `margin` argument.

Usage

```
dtmvnorm(x, mean = rep(0, nrow(sigma)),
  sigma = diag(length(mean)),
```

```
  lower=rep(-Inf, length = length(mean)),
  upper=rep( Inf, length = length(mean)),
  log=FALSE,
  margin=NULL)
```

Arguments

x	Vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
mean	Mean vector, default is rep(0, nrow(sigma)).
sigma	Covariance matrix, default is diag(length(mean)).
lower	Vector of lower truncation points, default is rep(-Inf, length = length(mean)).
upper	Vector of upper truncation points, default is rep(Inf, length = length(mean)).
log	Logical; if TRUE, densities d are given as log(d).
margin	if NULL then the joint density is computed (the default), if MARGIN=1 then the one-dimensional marginal density in variate q (q = 1..length(mean)) is returned, if MARGIN=c(q,r) then the bivariate marginal density in variates q and r for q,r = 1..length(mean) and $q \neq r$ is returned.

Details

The computation of truncated multivariate normal probabilities and densities is done using conditional probabilities from the standard/untruncated multivariate normal distribution. So we refer to the documentation of the mvtnorm package and the methodology is described in Genz (1992, 1993).

Author(s)

Stefan Wilhelm <Stefan.Wilhelm@financial.com>

References

Genz, A. (1992). Numerical computation of multivariate normal probabilities. *Journal of Computational and Graphical Statistics*, **1**, 141–150

Genz, A. (1993). Comparison of methods for the computation of multivariate normal probabilities. *Computing Science and Statistics*, **25**, 400–405

Johnson, N./Kotz, S. (1970). Distributions in Statistics: Continuous Multivariate Distributions *Wiley & Sons*, pp. 70–73

Horrace, W. (2005). Some Results on the Multivariate Truncated Normal Distribution. *Journal of Multivariate Analysis*, **94**, 209–221

See Also

ptmvnorm, pmvnorm, rmvnorm, dmvnorm, dtmvnorm.marginal and dtmvnorm.marginal2 for marginal density functions

Examples

```
dtmvnorm(x=c(0,0), mean=c(1,1), upper=c(0,0))

##########################################
#
# Example 1:
# truncated multivariate normal density
#
##########################################

x1<-seq(-2, 3, by=0.1)
x2<-seq(-2, 3, by=0.1)

density<-function(x)
{
  sigma=matrix(c(1, -0.5, -0.5, 1), 2, 2)
  z=dtmvnorm(x, mean=c(0,0), sigma=sigma, lower=c(-1,-1))
  z
}

fgrid <- function(x, y, f)
{
    z <- matrix(nrow=length(x), ncol=length(y))
    for(m in 1:length(x)){
        for(n in 1:length(y)){
            z[m,n] <- f(c(x[m], y[n]))
        }
    }
    z
}

# compute density d for grid
d=fgrid(x1, x2, density)

# plot density as contourplot
contour(x1, x2, d, nlevels=5, main="Truncated Multivariate Normal Density",
  xlab=expression(x[1]), ylab=expression(x[2]))
abline(v=-1, lty=3, lwd=2)
abline(h=-1, lty=3, lwd=2)

##########################################
#
# Example 2:
# generation of random numbers
# from a truncated multivariate normal distribution
#
##########################################

sigma <- matrix(c(4,2,2,3), ncol=2)
x <- rtmvnorm(n=500, mean=c(1,2), sigma=sigma, upper=c(1,0))
plot(x, main="samples from truncated bivariate normal distribution",
  xlim=c(-6,6), ylim=c(-6,6),
```

```
  xlab=expression(x[1]), ylab=expression(x[2]))
abline(v=1, lty=3, lwd=2, col="gray")
abline(h=0, lty=3, lwd=2, col="gray")
```

Index

∗ **distribution**
dtmvnorm.marginal, 2
dtmvnorm.marginal2, 5
dtmvt, 7
mtmvnorm, 14
ptmvnorm, 16
ptmvtnorm.marginal, 18
qtmvnorm-marginal, 20
rtmvnorm, 22
rtmvnorm2, 28
rtmvt, 30
tmvnorm, 32
∗ **math**
ptmvt, 17
∗ **multivariate**
dtmvnorm.marginal, 2
dtmvnorm.marginal2, 5
dtmvt, 7
mtmvnorm, 14
ptmvnorm, 16
ptmvt, 17
ptmvtnorm.marginal, 18
qtmvnorm-marginal, 20
rtmvnorm, 22
rtmvnorm2, 28
rtmvt, 30
tmvnorm, 32

acf, *24*

dmvnorm, *25*, *33*
dmvt, *8*
dtmvnorm *(*tmvnorm*)*, 32
dtmvnorm.marginal, 2, *10*, *33*
dtmvnorm.marginal2, 5, *33*
dtmvt, 7

gmm, *10, 11*
gmm.tmvnorm, 9

matrix, *23*
mle, *13*
mle.tmvnorm, 12
moments *(*mtmvnorm*)*, 14
mtmvnorm, *10*, 14

optim, *12*

pmvnorm, *6*, *15*, *21*, *25*, *33*
pmvt, *8*, *31*
ptmvnorm, 16, *21*, *25*, *33*
ptmvnorm.marginal *(*ptmvtnorm.marginal*)*, 18
ptmvt, *8*, 17
ptmvt.marginal *(*ptmvtnorm.marginal*)*, 18
ptmvtnorm.marginal, 18

qtmvnorm-marginal, 20
qtmvnorm.marginal *(*qtmvnorm-marginal*)*, 20

rmvnorm, *23*, *25*, *33*
rmvt, *8*, *31*
rtmvnorm, 22, *28, 29*, *31*
rtmvnorm2, 28
rtmvt, *8*, 30

sparseMatrix, *24*

tmvnorm, 32

uniroot, *20, 21*