Package ‘tmvtnorm’

February 20, 2015

Version 1.4-9
Date 2014-02-26
Title Truncated Multivariate Normal and Student t Distribution
Author Stefan Wilhelm <wilhelm@financial.com> with contributions from Manjunath B G <bgmanjunath@gmail.com>
Maintainer Stefan Wilhelm <wilhelm@financial.com>
Imports stats
Depends R (>= 1.9.0), mvtnorm, utils, Matrix, stats4, gmm
Suggests lattice
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 http://www.r-project.org
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-03-04 01:18:57

R topics documented:

dtmvnorm.marginal .................................................. 2
dtmvnorm.marginal2 ............................................... 5
dtmv ................................................................. 7
gmm.tmvnorm ....................................................... 9
mle.tmvnorm ....................................................... 11
mtmvnorm .......................................................... 14
ptmvnorm .......................................................... 15
ptmv ................................................................. 17
ptmvnorm.marginal ................................................. 18
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

```r
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, 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

\[
f_i(x_i) = \int_{a_i}^{b_i} \cdots \int_{a_{i-1}}^{b_{i-1}} \cdots \int_{a_k}^{b_k} f(x) dx_{-i}
\]

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


Examples

# Example 1: truncated bivariate normal

# parameters of the bivariate normal distribution
sigma = matrix(c(1, 0.9, 0.9, 1), nrow = 2, ncol = 2)
mu = c(0, 0)

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

# truncation 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 truncated bivariate normal distribution")
points(X.trunc)
abline(h=0, lty=2, col="gray")
legend("topleft", col=c("gray", "black"),
     main="marginal density for x1 from realisations")

# 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="marginal density for x[2]

# Example 2: truncated univariate normal

# parameters of the univariate normal distribution
mu = 0
sigma = 1

# sample from univariate normal distribution
X = rnorm(5000)

# truncation in x with x <= 0
X.trunc = X[X <= 0,]

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

# one-dimensional marginal density for x using the formula
x <- seq(-5, 5, by=0.01)
fx <- dtmvnorm.marginal(x, n=1, mean=mu, sigma=sigma,
                         lower=c(-Inf), upper=c(Inf,0))
lines(x, fx, lwd=2, col="red")

# marginal density for x
plot(density(X.trunc), main="marginal density for x")
example R: truncated trivariate normal

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

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

c truncation in x2 and x3 : x2 <= 0, x3 <= 0

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

plot(X[,c(1,3)], col="gray", xlab=expression(x[1]), ylab=expression(x[3]),
main="realisations from a 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]")))
**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\(\geq\)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**

```r
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

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

Author(s)

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

References


Examples

```r
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),
```
dtmvt

Truncated Multivariate Student t Density

Description

This function provides the joint density function for the truncated multivariate Student t distribution with mean vector \( \mu \), covariance matrix \( \Sigma \), degrees of freedom parameter \( \nu \) and lower and upper truncation points \( a \leq x \leq b \).

Usage

\[
dtmvt(x, \text{mean} = \text{rep}(0, nrow(\Sigma)), \text{sigma} = \text{diag}(\text{length}(\text{mean})), \text{df} = 1, \\
\quad \text{lower} = \text{rep}(-\text{Inf}, \text{length} = \text{length}(\text{mean})), \\
\quad \text{upper} = \text{rep}(\text{Inf}, \text{length} = \text{length}(\text{mean})), \text{log} = \text{FALSE})
\]

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 \( \text{rep}(0, \text{nrow}(\Sigma)) \).
- **sigma**: Covariance matrix, default is \( \text{diag}(\text{length}(\text{mean})) \).
- **df**: degrees of freedom parameter.
- **lower**: Vector of lower truncation points, default is \( \text{rep}(-\text{Inf}, \text{length} = \text{length}(\text{mean})) \).
- **upper**: Vector of upper truncation points, default is \( \text{rep}(\text{Inf}, \text{length} = \text{length}(\text{mean})) \).
- **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 \( a \leq x \leq b \).

The density of the \( p \)-variate Multivariate Student t distribution with \( \nu \) degrees of freedom is

\[
f(x) = \frac{\Gamma((\nu + p)/2)}{(\pi\nu)^{p/2}\Gamma(\nu/2)^{|\Sigma|^{1/2}}} \left[ 1 + \frac{1}{\nu}(x - \mu)^T\Sigma^{-1}(x - \mu) \right]^{-(\nu+p)/2}
\]
The density of the truncated distribution $f_{a,b}(x)$ with constraints ($a \leq x \leq b$) is accordingly

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

**Value**

a numeric vector with density values

**Author(s)**

Stefan Wilhelm <wilhelm@financial.com>

**References**


**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**

```r
# 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]))
    }
  }
}
```
\begin{verbatim}

z

# compute multivariate-t density d for grid
d <- fgrid(x1, x2, function(x) dtmv(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)

\end{verbatim}

\textbf{gmm.tmvnorm} \hspace{1cm} \textit{GMM Estimation for the Truncated Multivariate Normal Distribution}

\textbf{Description}

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

\textbf{Usage}

\begin{verbatim}
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,
    ...
)
\end{verbatim}

\textbf{Arguments}

- \textbf{X} \hspace{.5cm} Matrix of quantiles, each row is taken to be a quantile.
- \textbf{lower} \hspace{.5cm} Vector of lower truncation points, default is \text{rep}(\text{-Inf}, \text{length} = \text{ncol}(X)).
- \textbf{upper} \hspace{.5cm} Vector of upper truncation points, default is \text{rep}(\text{Inf}, \text{length} = \text{ncol}(X)).
- \textbf{start} \hspace{.5cm} Named list with elements \text{mu} (mean vector) and \text{sigma} (covariance matrix). Initial values for optimizer.
- \textbf{fixed} \hspace{.5cm} Named list. Parameter values to keep fixed during optimization.
- \textbf{method} \hspace{.5cm} Which set of moment conditions used, possible methods are "ManjunathWilhelm" (default) and "Lee".
- \textbf{cholesky} \hspace{.5cm} if \text{TRUE}, we use the Cholesky decomposition of \text{sigma} as parametrization
- \textbf{...} \hspace{.5cm} Further arguments to pass to \text{gmm}

\end{verbatim}
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. \texttt{gmm.tmvnorm()} is a wrapper for the general GMM method \texttt{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 \texttt{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

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

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

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

Value

An object of class \texttt{gmm}

Author(s)

Stefan Wilhelm &lt;wilhelm@financial.com&gt;

References


See Also

\texttt{gmm}
### Examples

```r
## 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

\texttt{mle.tmvnorm(X,}
\begin{itemize}
  \item \texttt{lower = rep(-Inf, length = ncol(X)),}
  \item \texttt{upper = rep(+Inf, length = ncol(X)),}
  \item \texttt{start = list(mu = rep(0, ncol(X)), sigma = diag(ncol(X))),}
  \item \texttt{fixed = list(), method = "BFGS",}
  \item \texttt{cholesky = FALSE,}
  \item \texttt{lower.bounds = -Inf,}
  \item \texttt{upper.bounds = +Inf,}
\end{itemize}
\texttt{...)}

Arguments

\begin{itemize}
  \item \texttt{X} \hspace{1cm} Matrix of quantiles, each row is taken to be a quantile.
  \item \texttt{lower} \hspace{1cm} Vector of lower truncation points, default is \texttt{rep(-Inf, length = ncol(X))}.
  \item \texttt{upper} \hspace{1cm} Vector of upper truncation points, default is \texttt{rep(Inf, length = ncol(X))}.
  \item \texttt{start} \hspace{1cm} Named list with elements \texttt{mu} (mean vector) and \texttt{sigma} (covariance matrix). Initial values for optimizer.
  \item \texttt{fixed} \hspace{1cm} Named list. Parameter values to keep fixed during optimization.
  \item \texttt{method} \hspace{1cm} Optimization method to use. See \texttt{optim}
  \item \texttt{cholesky} \hspace{1cm} if \texttt{TRUE}, we use the Cholesky decomposition of \texttt{sigma} as parametrization
  \item \texttt{lower.bounds} \hspace{1cm} lower bounds/box constraints for method "L-BFGS-B"
  \item \texttt{upper.bounds} \hspace{1cm} upper bounds/box constraints for method "L-BFGS-B"
  \item \texttt{...} \hspace{1cm} Further arguments to pass to \texttt{optim}
\end{itemize}

Details

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

The log-likelihood function for a data matrix \texttt{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 \texttt{mle}, this method returns an object of class \texttt{mle}, for which various diagnostic methods are available, like \texttt{profile()}, \texttt{confint()} etc. See examples.

In order to adapt the estimation problem to \texttt{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 \texttt{mle-class}
Author(s)

Stefan Wilhelm <wilhelm@financial.com>

See Also

mle and mle-class

Examples

## Not run:

```r
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)
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

```r
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, 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>
ptmvnorm

References


Examples

```r
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

```r
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)
```
ptmvnorm

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 \( \text{corr} \) or \( \text{sigma} \) can be specified. If \( \text{sigma} \) is given, the problem is standardized. If neither \( \text{corr} \) nor \( \text{sigma} \) is given, the identity matrix is used for \( \text{sigma} \).
lower Vector of lower truncation points, default is \( \text{rep}(\text{Inf}, \text{length} = \text{length}(\text{mean})) \).
upper Vector of upper truncation points, default is \( \text{rep}(-\text{Inf}, \text{length} = \text{length}(\text{mean})) \).
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 \texttt{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

\texttt{error} estimated absolute error and \n\texttt{msg} status messages.

References


Examples

```r
sigma <- matrix(c(5, 0.8, 0.8, 1), 2, 2)
Fx <- ptmvt(lower=x=-1, upper=x=1, mean=x=0, sigma=sigma, lower=x=-1, upper=x=1)
```

---

Truncated Multivariate Student t Distribution

Description

Computes the distribution function of the truncated multivariate t distribution

Usage

```r
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>
ptmvtnorm.marginal

References


Examples

```r
sigma <- matrix(c(5, 0.8, 0.8, 1), 2, 2)
Fx <- ptmvt(lower=c(-1,-1), upper=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

```r
ptmvtnorm.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)))
```

ptmvvt.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  
\[ F_i(x_i) = P(X_i \leq x_i) \]

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

where \( u = (b_1, \ldots, b_{i-1}, x_i, b_{i+1}, \ldots, b_k)' \) is the upper integration bound and \( \Phi_k \) 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 \( x_n \) with probabilities.

## Author(s)

Stefan Wilhelm <Stefan.Wilhelm@financial.com>

## Examples

```r
## 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=mean, sigma=sigma, lower=lower, upper=upper)
x <- seq(-1, 1, by=0.01)
Fx <- pmvnorm.marginal(xn=x, n=1, mean=mean, 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 <- rtmt(n=1000, mean=c(0,0,0), sigma=sigma, df=2, lower=lower, upper=upper)
x <- seq(-1, 1, by=0.01)
Fx <- pmvt.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")
```
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
```
qtmvnormNmarginal(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 $P[-x \leq X \leq x] = p$. $P[-x \leq X \leq x] = p$.
- `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.
rtmvnorm

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

```r
# 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)
```

### 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.
rtmvnorm

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)),
 D = diag(nrow(x)),
 ...)

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.
algorithmm Method used, possible methods are rejection sampling ("rejection", default), the Fortran Gibbs sampler ("gibbs") and the old Gibbs sampler implementation in R ("gibbsR").
... additional parameters for Gibbs sampling, given to the internal method rtmvnorngibbs(), 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


See Also

`ptmvnorm`, `pmvnorm`, `rmvnorm`, `dmvnorm`
Examples

```r
# 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")
```

```r
# 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")
```

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

# sigma <- matrix(c(4,2,2), ncol=2)
#
# Example T: univariate case
#
# 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)

# Example U: linear constraints
#
# linear constraints
#
a1 \leq x1 + x2 \leq b2
# a2 \leq x1 - x2 \leq b2
# \{ a1 \} \leq \{ 1, 1 \} \{ x1 \} \leq \{ b1 \}
# \{ a2 \} \{ 1, -1 \} \{ x2 \} \leq \{ b2 \}
a <- c(-2, -2)
b <- c( 2, 2)
D <- matrix(c(1, 1, 
             1, -1), 2, 2)

plot(density(x1[,1], to=1))
lines(density(x2[,1], to=1), col="blue")
lines(density(x3[,1], to=1), col="red")
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")
}

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)

### 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

\[ \text{lower} \leq D x \leq \text{upper} \]

with either rejection sampling or Gibbs sampling.

Usage

\[
\text{rtmvnorm2}(n, \text{mean} = \text{rep}(\emptyset, \text{nrow}(\text{sigma})), \\
\text{sigma} = \text{diag}(\text{length}(\text{mean})), \\
\text{lower} = \text{rep}(\text{-Inf}, \text{length} = \text{length}(\text{mean})), \\
\text{upper} = \text{rep}(\text{Inf}, \text{length} = \text{length}(\text{mean})), \\
D = \text{diag}(\text{length}(\text{mean})), \\
\text{algorithm} = c("\text{gibbs", "gibbsR", "rejection"}, \ldots)
\]

Arguments

n
Number of random points to be sampled. Must be an integer \( \geq 1 \).

mean
Mean vector (d x 1), default is \( \text{rep}(\emptyset, \text{length} = \text{ncol}(x)) \).

sigma
Covariance matrix (d x d), default is \( \text{diag}(\text{ncol}(x)) \).

lower
Vector of lower truncation points (r x 1), default is \( \text{rep}(\text{Inf}, \text{length} = \text{length}(\text{mean})) \).

upper
Vector of upper truncation points (r x 1), default is \( \text{rep}(\text{Inf}, \text{length} = \text{length}(\text{mean})) \).

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 \text{rtmvnorm.gibbs()}, such as \text{burn.in.samples}, \text{start.value} and \text{thinning}, see details in \text{rtmvnorm}

Details

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

Warning

This method will be merged with \text{rtmvnorm} in one of the next releases.

Author(s)

Stefan Wilhelm
See Also

rtmvnorm

Examples

```r
## 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 = 0.5*x1 - x2 <= b3
#
# [ a1 ] <= [ 1 1 ] [ x1 ] <= [b1]
# [ a2 ] [ 1 -1 ] [ x2 ] [b2]
# [ a3 ] [ 0.5 -1 ]

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

```r
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 \( \geq 1 \).
- **mean**: Mean vector, default is \( \text{rep}(0, \text{ncol}(x)) \).
- **sigma**: Covariance matrix, default is \( \text{diag}(\text{ncol}(x)) \).
- **df**: Degrees of freedom parameter (positive, may be non-integer)
- **lower**: Vector of lower truncation points, default is \( \text{rep}(-\text{Inf}, \text{length} = \text{length}(\text{mean})) \).
- **upper**: Vector of upper truncation points, default is \( \text{rep}(\text{Inf}, \text{length} = \text{length}(\text{mean})) \).
- **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 \( \text{lower} \leq x \leq \text{upper} \). 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 \((\text{lower} \leq \text{start.value} \leq \text{upper})\). 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 \(\geq 1\). We create a Markov chain of length \((n*\text{thinning})\) and take only those samples \(j=1:(n*\text{thinning})\) where \(j \equiv \text{thinning} \equiv 0\). Defaults to 1 (no thinning of the chain).

**Warning**

The same warnings for the Gibbs sampler apply as for the method \texttt{rtmvnorm}.

**Author(s)**

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

**References**


**Examples**

```r
# 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))
```

```r
# Example 2
#
```

```r
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)).
tmvnorm

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 re-
turned, if MARGIN=c(q,r) then the bivariate marginal density in variates q and
r for q,r = 1..length(mean) and q ≠ r is returned.

Details

The computation of truncated multivariate normal probabilities and densities is done using con-
ditional 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

tational and Graphical Statistics*, 1, 141–150


See Also

`ptmvnorm`, `pmvnorm`, `rmvnorm`, `dmvnorm`, `dtmvnorm` and `dtmvnorm.marginal` 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)
```r
z <- dtmvnorm(x, mean=c(0,0), sigma=sigma, lower=c(-1,-1))
}

def grid <- 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(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=100, mean=c(1,2), sigma=sigma, upper=c(1,0))
plot(x, main="samples from truncated bivariate normal distribution",
  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

*Topic distribution
  dtmvnorm.marginal, 2
  dtmvnorm.marginal2, 5
dtmvt, 7
  mtmvnorm, 14
  ptmvnorm, 15
  ptmvtnorm.marginal, 18
  qtmvnorm-marginal, 20
  rtmvnorm, 21
  rtmvnorm2, 27
  rtmvt, 30
  tmvnorm, 32

*Topic math
  ptmvt, 17

*Topic multivariate
  dtmvnorm.marginal, 2
  dtmvnorm.marginal2, 5
  dtmvnorm, 7
  mtmvnorm, 14
  ptmvnorm, 15
  ptmvtnorm.marginal, 18
  qtmvnorm-marginal, 20
  rtmvnorm, 21
  rtmvnorm2, 27
  rtmvt, 30
  tmvnorm, 32

acf, 24
dmvnorm, 24, 33
dmvt, 8
dtmvnorm(tmvnorm), 32
dtmvnorm.marginal, 2, 10, 33
dtmvnorm.marginal2, 5, 33
dtmvt, 7
gmm, 9, 10
gmm.tmvnorm, 9
matrix, 23

mle, 12, 13
mle.tmvnorm, 11
moments (mtmvnorm), 14
mtmvnorm, 10, 14
optim, 12
pmvnorm, 5, 14, 21, 24, 33
pmvt, 8, 30
ptmvnorm, 15, 21, 24, 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, 22, 24, 33
rmvt, 8, 30
rtmvnorm, 21, 28, 29, 31
rtmvnorm2, 27
rtmvt, 8, 30
sparseMatrix, 23
tmvnorm, 32
uniroot, 20, 21