

Package ‘mnormt’

April 2, 2015

Version 1.5-2

Date 2015-04-02

Title The Multivariate Normal and t Distributions

Author Fortran code by Alan Genz, R code by Adelchi Azzalini

Maintainer Adelchi Azzalini <adelchi.azzalini@unipd.it>

Depends R (>= 2.2.0)

Description Functions are provided for computing the density and the distribution function of multivariate normal and “t” random variables, and for generating random vectors sampled from these distributions. Probabilities are computed via a non-Monte Carlo method. Different routines are used for the case $d=1$, $d=2$, $d>2$, if d denotes the number of dimensions.

License GPL-2 | GPL-3

URL <http://azzalini.stat.unipd.it/SW/Pkg-mnormt>

NeedsCompilation yes

R topics documented:

mnormt-package	2
dmnorm	2
dmt	5
pd.solve	7

Index	9
--------------	----------

 mnormt-package

The 'mnormt' package: summary information

Description

This package provides functions for computing the density and the distribution function of multivariate normal and multivariate Student's t variates and for generating random vectors sampled from these distributions.

Details

Probabilities are computed via a non-Monte Carlo method. Different routines are used in the three cases $d=1$, $d=2$, $d>2$, if d denotes the number of dimensions.

Licence

This package and its documentation are usable under the terms of the "GNU General Public License" version 3 or version 2, as you prefer; a copy of them is available from <http://www.R-project.org/Licenses/>.

Author(s)

Adelchi Azzalini (R code and package creation) and Alan Genz (Fortran code, see references below; this includes routines of other authors)

References

Genz, A. (1992). Numerical Computation of Multivariate Normal Probabilities. *J. Computational and Graphical Statist.*, **1**, 141-149.

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

Genz, A.: Fortran code available at <http://www.math.wsu.edu/math/faculty/genz/software/fort77/mvn.f>

 dmnorm

Multivariate normal distribution

Description

The probability density function, the distribution function and random number generation for the multivariate normal (Gaussian) distribution

Usage

```
dmnorm(x, mean = rep(0, d), varcov, log = FALSE)
pmnorm(x, mean = rep(0, d), varcov, ...)
rmnorm(n = 1, mean = rep(0, d), varcov, sqrt=NULL)
sadmvn(lower, upper, mean, varcov, maxpts = 2000*d, abseps = 1e-06, releps = 0)
```

Arguments

<code>x</code>	either a vector of length <code>d</code> or a matrix with <code>d</code> columns, where <code>d=ncol(varcov)</code> , representing the coordinates of the point(s) where the density must be evaluated; for <code>pmnorm</code> , <code>d</code> cannot exceed 20
<code>mean</code>	either a vector of length <code>d</code> , representing the mean value, or (except for <code>rmnorm</code>) a matrix whose rows represent different mean vectors; if it is a matrix, its dimensions must match those of <code>x</code>
<code>varcov</code>	a symmetric positive-definite matrix representing the variance-covariance matrix of the distribution; a vector of length 1 is also allowed (in this case, <code>d=1</code> is set)
<code>sqrt</code>	if not <code>NULL</code> (default value is <code>NULL</code>), a square root of the intended <code>varcov</code> matrix; see ‘Details’ for a detailed description
<code>log</code>	a logical value (default value is <code>FALSE</code>); if <code>TRUE</code> , the logarithm of the density is computed
<code>...</code>	parameters passed to <code>sadmvn</code> , among <code>maxpts</code> , <code>abseps</code> , <code>releps</code>
<code>n</code>	the number of random vectors to be generated
<code>lower</code>	a numeric vector of lower integration limits of the density function; must be of maximal length 20; <code>+Inf</code> and <code>-Inf</code> entries are allowed
<code>upper</code>	a numeric vector of upper integration limits of the density function; must be of maximal length 20; <code>+Inf</code> and <code>-Inf</code> entries are allowed
<code>maxpts</code>	the maximum number of function evaluations (default value: <code>2000*d</code>)
<code>abseps</code>	absolute error tolerance (default value: <code>1e-6</code>)
<code>releps</code>	relative error tolerance (default value: 0)

Details

The function `pmnorm` works by making a suitable call to `sadmvn` if `d>2`, or to `biv.nt.prob` if `d=2`, or to `pnorm` if `d=1`. Function `sadmvn` is an interface to a Fortran-77 routine with the same name written by Alan Genz, and available from his web page; this makes use of some auxiliary functions whose authors are documented in the Fortran code. The routine uses an adaptive integration method.

If `sqrt=NULL` (default value), the working of `rmnorm` involves computation of a square root of `varcov` via the Cholesky decomposition. If a non-`NULL` value of `sqrt` is supplied, it is assumed that it represents a matrix, R say, such that $R'R$ represents the required variance-covariance matrix of the distribution; in this case, the argument `varcov` is ignored. This mechanism is intended primarily for use in a sequence of calls to `rmnorm`, all sampling from a distribution with fixed variance matrix; a suitable matrix `sqrt` can then be computed only once beforehand, avoiding that the same operation is repeated multiple times along the sequence of calls; see the examples below. Another use of `sqrt` is to supply a different form of square root of the variance-covariance matrix, in place of the Cholesky factor.

For efficiency reasons, `rmnorm` performs limited check on the supplied arguments.

Value

`dmnorm` returns a vector of density values (possibly log-transformed); `pmnorm` and `sadmvn` return a single probability with attributes giving details on the achieved accuracy, provided `x` of `pmnorm` is a vector; `rmnorm` returns a matrix of `n` rows of random vectors

Note

The attributes `error` and `status` of the probability returned by `pmnorm` and `sadmvn` indicate whether the function had a normal termination, achieving the required accuracy. If this is not the case, re-run the function with an higher value of `maxpts`

Author(s)

Fortran code of `SADMVN` and most auxiliary functions by Alan Genz, some additional auxiliary functions by people referred to within his program. Interface to R and additional R code by Adelchi Azzalini

References

Genz, A. (1992). Numerical Computation of multivariate normal probabilities. *J. Computational and Graphical Statist.*, **1**, 141-149.

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

Genz, A.: Fortran code available at <http://www.math.wsu.edu/math/faculty/genz/software/fort77/mvn.f>

See Also

[dnorm](#), [dmt](#), [biv.nt.prob](#)

Examples

```
x <- seq(-2, 4, length=21)
y <- cos(2*x) + 10
z <- x + sin(3*y)
mu <- c(1,12,2)
Sigma <- matrix(c(1,2,0,2,5,0.5,0,0.5,3), 3, 3)
f <- dmnorm(cbind(x,y,z), mu, Sigma)
f0 <- dmnorm(mu, mu, Sigma)
p1 <- pmnorm(c(2,11,3), mu, Sigma)
p2 <- pmnorm(c(2,11,3), mu, Sigma, maxpts=10000, abseps=1e-10)
p <- pmnorm(cbind(x,y,z), mu, Sigma)
#
set.seed(123)
x1 <- rmnorm(5, mu, Sigma)
set.seed(123)
x2 <- rmnorm(5, mu, sqrt=chol(Sigma)) # x1=x2
eig <- eigen(Sigma, symmetric = TRUE)
R <- t(eig$vectors %*% diag(sqrt(eig$values)))
for(i in 1:50) x <- rmnorm(5, mu, sqrt=R)
#
p <- sadmvn(lower=c(2,11,3), upper=rep(Inf,3), mu, Sigma) # upper tail
#
p0 <- pmnorm(c(2,11), mu[1:2], Sigma[1:2,1:2])
p1 <- biv.nt.prob(0, lower=rep(-Inf,2), upper=c(2, 11), mu[1:2], Sigma[1:2,1:2])
p2 <- sadmvn(lower=rep(-Inf,2), upper=c(2, 11), mu[1:2], Sigma[1:2,1:2])
c(p0, p1, p2, p0-p1, p0-p2)
#
p1 <- pnorm(0, 1, 3)
p2 <- pmnorm(0, 1, 3^2)
```

dmt

*Multivariate t distribution***Description**

The probability density function, the distribution function and random number generation for the multivariate Student's *t* distribution

Usage

```
dmt(x, mean = rep(0, d), S, df=Inf, log = FALSE)
pmt(x, mean = rep(0, d), S, df=Inf, ...)
rmt(n = 1, mean = rep(0, d), S, df=Inf, sqrt=NULL)
sadmvt(df, lower, upper, mean, S, maxpts = 2000*d, abseps = 1e-06, releps = 0)
biv.nt.prob(df, lower, upper, mean, S)
```

Arguments

x	either a vector of length <i>d</i> or a matrix with <i>d</i> columns, where $d = \text{ncol}(S)$, giving the coordinates of the point(s) where the density must be evaluated; for <code>pmt</code> , <i>d</i> cannot exceed 20
mean	either a vector of length <i>d</i> , representing the location parameter (equal to the mean vector when $df > 1$), or (except for <code>rmt</code>) a matrix whose rows represent different location vectors; if it is a matrix, its dimensions must match those of <i>x</i>
S	a symmetric positive-definite matrix representing the scale matrix of the distribution, such that $S * df / (df - 2)$ is the variance-covariance matrix when $df > 2$; a vector of length 1 is also allowed (in this case, $d = 1$ is set)
df	degrees of freedom; it must be a positive integer for <code>pmt</code> , <code>sadmvt</code> and <code>biv.nt.prob</code> , otherwise a positive number. If $df = \text{Inf}$ (default value), the corresponding <code>*mnorm</code> function is called, unless $d = 2$; in this case <code>biv.nt.prob</code> is used. If <code>biv.nt.prob</code> is called with $df = \text{Inf}$, it returns the probability of a rectangle assigned by a bivariate normal distribution
log	a logical value (default value is <code>FALSE</code>); if <code>TRUE</code> , the logarithm of the density is computed
sqrt	if not <code>NULL</code> (default value is <code>NULL</code>), a square root of the intended scale matrix <i>S</i> ; see 'Details' for a detailed description
...	parameters passed to <code>sadmvt</code> , among <code>maxpts</code> , <code>absrel</code> , <code>releps</code>
n	the number of random vectors to be generated
lower	a numeric vector of lower integration limits of the density function; must be of maximal length 20; $+\text{Inf}$ and $-\text{Inf}$ entries are allowed
upper	a numeric vector of upper integration limits of the density function; must be of maximal length 20; $+\text{Inf}$ and $-\text{Inf}$ entries are allowed
maxpts	the maximum number of function evaluations (default value: $2000 * d$)
abseps	absolute error tolerance (default value: $1e-6$)
releps	relative error tolerance (default value: 0)

Details

The functions `sadmvt` and `biv.nt.prob` are interfaces to Fortran-77 routines by Alan Genz, and available from his web page; they makes uses of some auxiliary functions whose authors are documented in the Fortran code. The routine `sadmvt` uses an adaptive integration method. The routine `biv.nt.prob` is specific for the bivariate case; if `df<1` or `df=Inf`, it computes the bivariate normal distribution function using a non-iterative method described in a reference given below. If `pmt` is called with `d>2`, this is converted into a suitable call to `sadmvt`; if `d=2`, a call to `biv.nt.prob` is used; if `d=1`, then `pt` is used.

If `sqrt=NULL` (default value), the working of `rmt` involves computation of a square root of S via the Cholesky decomposition. If a non-NULL value of `sqrt` is supplied, it is assumed that it represents a square root of the scale matrix, otherwise represented by S , whose value is ignored in this case. This mechanism is intended primarily for use in a sequence of calls to `rmt`, all sampling from a distribution with fixed scale matrix; a suitable matrix `sqrt` can then be computed only once beforehand, avoiding that the same operation is repeated multiple times along the sequence of calls. Another use of `sqrt` is to supply a different form of square root of the scale matrix, in place of the Cholesky factor.

If `sqrt=NULL` (default value), the working of `rmnorm` involves computation of a square root of S via the Cholesky decomposition. If a non-NULL value of `sqrt` is supplied, it is assumed that it represents a matrix, R say, such that $R \backslash ' R$ represents the required scale matrix of the distribution; in this case, the argument S is ignored. This mechanism is intended primarily for use in a sequence of calls to `rmt`, all sampling from a distribution with fixed scale matrix; a suitable matrix `sqrt` can then be computed only once beforehand, avoiding that the same operation is repeated multiple times along the sequence of calls. Another use of `sqrt` is to supply a different form of square root of the scale matrix, in place of the Cholesky factor.

For efficiency reasons, `rmt` performs limited check on the supplied arguments.

Value

`dmt` returns a vector of density values (possibly log-transformed); `pmt` and `sadmvt` return a single probability with attributes giving details on the achieved accuracy, provided x of `pmnorm` is a vector; `rmt` returns a matrix of n rows of random vectors

Note

The attributes `error` and `status` of the probability returned by `pmt` and `sadmvt` indicate whether the function had a normal termination, achieving the required accuracy. If this is not the case, re-run the function with an higher value of `maxpts`

Author(s)

Fortran code of `SADMVT` and most auxiliary functions by Alan Genz, some additional auxiliary functions by people referred to within his program. Interface to R and additional R code by Adelchi Azzalini

References

Genz, A.: Fortran code in files `mvt.f` and `mvtdstpack.f` available at <http://www.math.wsu.edu/math/faculty/genz/software/>

Dunnett, C.W. and Sobel, M. (1954). A bivariate generalization of Student's t -distribution with tables for certain special cases. *Biometrika* 41, 153–169.

See Also

[dt](#), [dmnorm](#) for use of argument `sqrt`

Examples

```
x <- seq(-2,4,length=21)
y <- 2*x+10
z <- x+cos(y)
mu <- c(1,12,2)
Sigma <- matrix(c(1,2,0,2,5,0.5,0,0.5,3), 3, 3)
df <- 4
f <- dmt(cbind(x,y,z), mu, Sigma,df)
p1 <- pmt(c(2,11,3), mu, Sigma, df)
p2 <- pmt(c(2,11,3), mu, Sigma, df, maxpts=10000, abseps=1e-8)
x <- rmt(10, mu, Sigma, df)
p <- sadmvt(df, lower=c(2,11,3), upper=rep(Inf,3), mu, Sigma) # upper tail
#
p0 <- pmt(c(2,11), mu[1:2], Sigma[1:2,1:2], df=5)
p1 <- biv.nt.prob(5, lower=rep(-Inf,2), upper=c(2, 11), mu[1:2], Sigma[1:2,1:2])
p2 <- sadmvt(5, lower=rep(-Inf,2), upper=c(2, 11), mu[1:2], Sigma[1:2,1:2])
c(p0, p1, p2, p0-p1, p0-p2)
```

pd.solve

Inverse of a symmetric positive-definite matrix

Description

The inverse of a symmetric positive-definite matrix and its log-determinant

Usage

```
pd.solve(x, silent = FALSE, log.det=FALSE)
```

Arguments

<code>x</code>	a symmetric positive-definite matrix.
<code>silent</code>	a logical value which indicates the action to take in case of an error. If <code>silent==TRUE</code> and an error occurs, the function silently returns a <code>NULL</code> value; if <code>silent==FALSE</code> (default) an error generates a <code>stop</code> with an error message.
<code>log.det</code>	a logical value to indicate whether the log-determinant of <code>x</code> is required (default is <code>FALSE</code>).

Details

The function checks that `x` is a symmetric positive-definite matrix. If an error is detected, an action is taken which depends on the value of the argument `silent`.

Value

the inverse matrix of `x`; if `log.det=TRUE`, this inverse has an attribute which contains the logarithm of the determinant of `x`.

Author(s)

Adelchi Azzalini

Examples

```
x <- toeplitz(rev(1:4))
x.inv <- pd.solve(x)
print(x.inv %*% x)
x.inv <- pd.solve(x, log.det=TRUE)
logDet <- attr(x.inv, "log.det")
print(abs(logDet - determinant(x, logarithm=TRUE)$modulus))
```


Index

***Topic algebra**

pd.solve, 7

***Topic array**

pd.solve, 7

***Topic distribution**

dmnorm, 2

dmt, 5

mnormt-package, 2

***Topic multivariate**

dmnorm, 2

dmt, 5

mnormt-package, 2

***Topic package**

mnormt-package, 2

biv.nt.prob, 4

biv.nt.prob(dmt), 5

dmnorm, 2, 7

dmt, 4, 5

dnorm, 4

dt, 7

mnormt-package, 2

pd.solve, 7

pmnorm(dmnorm), 2

pmt(dmt), 5

rmnorm(dmnorm), 2

rmt(dmt), 5

sadmvn(dmnorm), 2

sadmvt(dmt), 5