Package ‘matrixNormal’

July 9, 2019

Version 0.0.1

Date 2019-07-09

Type Package

Title The Matrix Normal Distribution

Depends R (>= 3.5.0)

Imports utils (>= 3.5.0), mvtnorm (>= 1.0.8)

Suggests stats (>= 3.5.0), formatR, knitr, LaplacesDemon (>= 16.1.1), matrixcalc (>= 1.0.3), matrixsampling (>= 1.1.0), MBSP (>= 1.0), rmarkdown, roxygen2, spelling, testthat

Description Computes densities, probabilities, and random deviates of the Matrix Normal (Iranmanesh et al. (2010) <doi:10.7508/ijmsi.2010.02.004>). Also includes simple but useful matrix functions. See the vignette for more information.

License GPL-3

Encoding UTF-8

LazyData true

BugReports `https://github.com/phargarten2/matrixNormal/issues`

RoxygenNote 6.1.1

Language en-US

VignetteBuilder knitr

NeedsCompilation no

Author Paul Hargarten [aut, cre]

Maintainer Paul Hargarten <`hargartenp@vcu.edu`>

Repository CRAN

Date/Publication 2019-07-09 14:00:02 UTC

R topics documented:

`is.symmetric.matrix` *Is a matrix symmetric or positive-definite?*

Description

Determine if a matrix is square, symmetric, positive-definite, or positive-semi-definite.

Usage

```
is.square.matrix(A)

is.symmetric.matrix(A, tol = .Machine$double.eps^0.5)

is.positive.semi.definite(A, tol = .Machine$double.eps^0.5)

is.positive.definite(A, tol = .Machine$double.eps^0.5)
```

Arguments

`A`	A numeric matrix.
`tol`	A numeric tolerance level used to check if a matrix is symmetric; that is if the difference between the matrix and its transpose is between `-tol` and `tol`.

Details

A tolerance is added to indicate if a matrix is approximately symmetric. If the matrix is not symmetric, a message as well as the top of the matrix is printed.

- `is.symmetric.matrix` returns TRUE if A is a symmetric square numeric matrix and FALSE otherwise. A matrix is symmetric if the difference between A and its transpose is less than *tol*. If A has any missing values, `is.symmetric.matrix` returns NA.
- `is.positive.semi.definite` returns TRUE if a square symmetric real matrix A is positive semi-definite. A matrix is positive semi-definite if its smallest eigenvalue is greater than or equal to zero. If A has any missing values, `is.symmetric.matrix` returns NA.
- `is.positive.definite` returns TRUE if a square symmetric real matrix A is positive-definite. A matrix is positive-definite if its smallest eigenvalue is greater than zero. If A has any missing values, `is.symmetric.matrix` returns NA.

Note

Functions adapted from Frederick Novomestky's **matrixcalc** package in order to implement `rmatnorm` function. I changed argument x to A to reflect usual matrix notation. For `is.symmetric`, I added a tolerance so that A is symmetric even provided small differences between A and its transpose. Useful for rmatnorm function, which was used repeatedly to generate matrixNormal random variates in a Markov chain. For `is.positive.semi.definite` and `is.positive.definite`, I also saved time by avoiding a $for-loop$ and instead calculating the minimum of eigenvalues.

Examples

```
## Example 0: Not square matrix
B <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, byrow = TRUE)
B
is.square.matrix(B)

## Example 1: Not a matrix. should get an error.
df <- as.data.frame(matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, byrow = TRUE))
df
## Not run:
is.square.matrix(df)

## End(Not run)

## Example 2: Not Symmetric & Compare against matrixcalc
F <- matrix(c(1, 2, 3, 4), nrow = 2, byrow = TRUE); F
is.square.matrix(F)
is.symmetric.matrix(F)   # should be FALSE
if (!requireNamespace("matrixcalc", quietly = TRUE)) {
  matrixcalc::is.symmetric.matrix(F)
} else {
  message("you need to install the package matrixcalc to compare this example")
}

## Example 3: Symmetric but negative-definite. same test of functions
##' eigenvalues are  3 -1
G <- matrix(c(1, 2, 2, 1), nrow = 2, byrow = TRUE); G
is.symmetric.matrix(G)
if (!requireNamespace("matrixcalc", quietly = TRUE)) {
  matrixcalc::is.symmetric.matrix(G)
} else {
  message("you need to install the package matrixcalc to compare this example.")
}
isSymmetric.matrix(G)
is.positive.definite(G) # FALSE
is.positive.semi.definite(G) # FALSE
## Example 3b: A missing value in G
G[1, 1] <- NA
is.symmetric.matrix(G) # NA
is.positive.definite(G)  # NA

## Example 4: positive definite matrix
# eigenvalues are 3.4142136 2.0000000 0.585786
```

```
Q <- matrix(c(2, -1, 0, -1, 2, -1, 0, -1, 2), nrow = 3, byrow = TRUE)
is.symmetric.matrix(Q)
is.positive.definite(Q)

## Example 5: identity matrix is always positive definite
I <- diag(1, 3)
is.square.matrix(I) # TRUE
is.symmetric.matrix(I) # TRUE
is.positive.definite(I) # TRUE
```

---

matrixNormal_Distribution

The Matrix Normal Distribution

Description

The density (dmatnorm()), cumulative distribution function (CDF, pmatnorm()), and generation of 1 random number from the matrix normal (rmatnorm()) is produced from

$$A\ MatNorm_{n,p}(M, U, V)$$

Usage

```
dmatnorm(A, M, U, V, tol = .Machine$double.eps^0.5, use.log = TRUE)

pmatnorm(Lower = -Inf, Upper = Inf, M, U, V,
  tol = .Machine$double.eps^0.5, algorithm = mvtnorm::GenzBretz(), ...)

rmatnorm(M, U, V, tol = .Machine$double.eps^0.5, method = "chol")
```

Arguments

A	The numeric n x p matrix that follows the matrix-normal.
M	The mean n x p matrix that is numeric and real. Must contain non-missing values.
U	The individual scale n x n real positive-definite matrix (rows). Must contain non-missing values.
V	The parameter scale p x p real positive-definite matrix (columns). Must contain non-missing values.
tol	A numeric tolerance level used to check if a matrix is symmetric; that is if the difference between the matrix and its transpose is between -tol and tol.
use.log	Logical; if TRUE, densities d are given as log(d).
Lower	The n x p matrix of lower limits for CDF
Upper	The n x p matrix of upper limits for CDF
algorithm	an object of class GenzBretz, Miwa or TVPACK specifying both the algorithm to be used as well as the associated hyper parameters.

`...`	additional parameters (currently given to `GenzBretz` for backward compatibility issues).
`method`	string specifying the matrix decomposition used to determine the matrix root of `sigma`. Possible methods are eigenvalue decomposition (`"eigen"`, default), singular value decomposition (`"svd"`), and Cholesky decomposition (`"chol"`). The Cholesky is typically fastest, not by much though.

Details

Ideally, both scale matrices are positive-definite. However, they may not appear to be symmetric; you may want to increase the tolerance.

These functions rely heavily on this following property of matrix normal distribution. Let function 'koch()' refer to the Kronecker product of a matrix. For a n x p matrix A, if $A \sim MatNorm(M, U, V)$, then

$$vec(A) \sim MVN_{np}(M, Sigma = koch(U, V))$$

.

Thus, we can find the probability that Lower < A < Upper by finding the CDF of vec(A), which is given in `pmvnorm` function in **mvtnorm**. See `algorithms` and `pmvnorm`. Also, we can simulate 1 random matrix A from a matrix normal by sampling vec(A) from **mvtnorm**::`rmvnorm`() function. This matrix A takes the rownames from U and the colnames from V.

References

Iranmanesh, Anis, M. Arashi, and S. M. M. Tabatabaey On Conditional Applications of Matrix Variate Normal Distribution. *Iranian Journal of Mathematical Sciences and Informatics* 5, no. 2. (November 1, 2010): 33-43. < https://doi.org/10.7508/ijmsi.2010.02.004 >

Examples

```
#Data Used
A <- datasets::CO2[1:10, 4:5]
M <- cbind(stats::rnorm(10, 435, 296), stats::rnorm(10, 27, 11) )
V <- matrix(c(87, 13, 13, 112), nrow = 2, ncol = 2, byrow = TRUE)
V  #Right covariance matrix (2 x 2), say the covariance between parameters.
U <- I(10) #Block of left-covariance matrix ( 84 x 84), say the covariance between subjects.

#PDF
dmatnorm(A, M, U, V )
dmatnorm(A, M, U, V, use.log = FALSE)

#Generating Probability Lower and Upper Bounds (They're matrices )
Lower <- matrix( rep(-1, 20), ncol = 2)
Upper <- matrix( rep(3, 20), ncol = 2)
Lower; Upper
#The probablity that a randomly chosen matrix A is between Lower and Upper
pmatnorm( Lower, Upper, M, U, V)
#CDF
pmatnorm( Lower = -Inf, Upper, M, U, V)
#entire domain = 1
```

```
pmatnorm( Lower = -Inf, Upper = Inf, M, U, V)

#Random generation
set.seed(123)
M <- cbind(rnorm(3, 435, 296), rnorm(3, 27, 11) )
U <- diag(1, 3)
V <- matrix(c(10, 5 ,5, 3), nrow = 2)
rmatnorm(M, U, V)
## Not run:   #M has a different sample size than U; will return an error.
M <- cbind(rnorm(4, 435, 296), rnorm(4, 27, 11) )
rmatnorm(M, U, V)

## End(Not run)
```

special.matrix   *Generating Special Matrices*

Description

Creates Identity Matrix I and Matrix of Ones J.

Usage

```
I(n)

J(n, m = n)
```

Arguments

n	number of rows in I or J.
m	number of columns in J. Default: same as number of rows.

Details

Create an Identity Matrix where the number of columns is n. This is a diagonal matrix with all equal to one (1). An identity matrix is usually written as I. To make an identity matrix with r rows and columns, use `I`.

A J matrix is a general matrix of any number of rows and columns, but in which all elements in the matrix are equal to one (1). `J` will make a n x m J matrix, given the number or rows, n, and number of columns, m. Names of rows and columns (dimnames) are included.

See Also

Other matrix: `tr`, `vec`

Examples

```
#To create an identity matrix of order 12
I(2)
#To make a matrix of 6 rows and 10 columns of all ones
J(6,10)
#To make a matrix of unity, dimensions 6 x 6 .
J(6)
```

`tr` *Matrix Trace*

Description

Computes the trace of a square numeric matrix A.

Usage

```
tr(A)
```

Arguments

`A`	Square matrix.

Note

If the argument is not a square numeric matrix, the function presents an error and terminates.

See Also

Other matrix: `special.matrix`, `vec`

Examples

```
A <- matrix( seq( 1, 16, 1 ), nrow=4, byrow=TRUE )
A
tr( A )
tr( I(3) )
```

vec *Stacks a Matrix using matrix operator "vec"*

Description

Returns a column vector that stacks the columns of A, an m by n matrix.

Usage

```
vec(A, use.Names = TRUE)
```

Arguments

`A`	A m x n matrix.
`use.Names`	logical. If TRUE, the names of A are taken to be names of the stacked matrix. Default: TRUE.

Value

A vector with mn elements.

Note

Adapted from Frederick Novomestky's **matrixcalc**. This function is edited so that it can take dimension names and return the matrix as a vector.

References

Magnus, J. R. and H. Neudecker (1999). *Matrix Differential Calculus with Applications in Statistics and Econometrics.* Second Edition, John Wiley, ed.

See Also

Other matrix: `special.matrix`, `tr`

Examples

```
M <- matrix(c(4,5,6,7,8,9), nrow=3)
M
#Compare vec from \pkg{matrixcalc} and new function.
matrixcalc::vec(M)
vec(M)
#The names are rownames(M):colnames(M) in that order.
#Very similar to matrixcalc but dimension names are different.
```

Index