Package: matrixNormal (via r-universe)

August 25, 2024

Version 0.0.5

Type Package

Title The Matrix Normal Distribution

Depends R (>= 3.5.0), stats, utils

Imports mvtnorm (>= 1.0.8)

Suggests formatR, knitr, LaplacesDemon (>= 16.1.1), matrixcalc (>= 1.0-3), matrixsampling (>= 1.1.0), MBSP (>= 1.0), rmarkdown, roxygen2, sessioninfo, 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 false

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

RoxygenNote 7.1.1

Language en-US

VignetteBuilder knitr

Repository https://phargarten2.r-universe.dev

RemoteUrl https://github.com/phargarten2/matrixnormal

RemoteRef HEAD

RemoteSha ac1fd4e31c961c2d1cd4e59a328afeac3562302b

Contents

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

Description

Determines 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, a matrix is symmetric if the difference between the matrix and its transpose is between -`tol` and `tol`.

Details

A tolerance is added to indicate if a matrix *A* is approximately symmetric. If *A* is not symmetric, a message and first few rows of the matrix is printed. If *A* has any missing values, NA is returned.

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

Note

Functions are adapted from Frederick Novomestky's **matrixcalc** package in order to implement the `rmatnorm` function. The following changes are made:

- 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. This is 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. The functions are same.
# 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

Computes the density (`dmatnorm`), calculates the cumulative distribution function (CDF, `pmatnorm`), and generates 1 random number (`rmatnorm`) from the matrix normal:

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

.

Usage

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

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

rmatnorm(s = 1, M, U, V, tol = .Machine$double.eps^0.5, method = "chol")
```

Arguments

`A`	The numeric n x p matrix that follows the matrix-normal. Value used to calculate the density.
`M`	The mean n x p matrix that is numeric and real. Must contain non-missing values. Parameter of matrix Normal.

U	The individual scale n x n real positive-definite matrix (rows). Must contain non-missing values. Parameter of matrix Normal.
V	The parameter scale p x p real positive-definite matrix (columns). Must contain non-missing values. Parameter of matrix Normal.
tol	A numeric tolerance level used to check if a matrix is symmetric. That is, a matrix is symmetric if the difference between the matrix and its transpose is between -tol and tol.
log	Logical; if TRUE, the logarithm of the density is returned.
Lower	The n x p matrix of lower limits for CDF.
Upper	The n x p matrix of upper limits for CDF.
keepAttr	logical indicating if attributes such as error and msg should be attached to the return value. The default, TRUE is back compatible.
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).
s	The number of observations desired to simulate from the matrix normal. Defaults to 1. Currently has no effect but acts as a placeholder in future releases.
method	String specifying the matrix decomposition used to determine the matrix root of the Kronecker product of U and V in rmatnorm. Possible methods are eigenvalue decomposition ("eigen"), singular value decomposition ("svd"), and Cholesky decomposition ("chol"). The Cholesky (the default) is typically fastest, but not by much though. Passed to rmvnorm.

Details

These functions rely heavily on this following property of matrix normal distribution. Let koch() refer to the Kronecker product of a matrix. For a n x p matrix A, if

$$A \sim MatNorm(M, U, V),$$

then

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

Thus, the probability of Lower < A < Upper in the matrix normal can be found by using the CDF of vec(A), which is given by pmvnorm function in **mvtnorm**. See algorithms and pmvnorm for more information.

Also, we can simulate a random matrix A from a matrix normal by sampling *vec(A)* from rmvnorm function in **mvtnorm**. This matrix A takes the rownames from U and the colnames from V.

Calculating Matrix Normal Probabilities

From the mvtnorm package, three algorithms are available for evaluating normal probabilities:

- The default is the randomized Quasi-Monte-Carlo procedure by Genz (1992, 1993) and Genz and Bretz (2002) applicable to arbitrary covariance structures and dimensions up to 1000.

- For smaller dimensions (up to 20) and non-singular covariance matrices, the algorithm by Miwa et al. (2003) can be used as well.
- For two- and three-dimensional problems and semi-infinite integration region, TVPACK implements an interface to the methods described by Genz (2004).

The ... arguments define the hyper-parameters for GenzBertz algorithm:

maxpts maximum number of function values as integer. The internal FORTRAN code always uses a minimum number depending on the dimension.Default 25000.

abseps absolute error tolerance.

releps relative error tolerance as double.

Note

Ideally, both scale matrices are positive-definite. If they do not appear to be symmetric, the tolerance should be increased. Since symmetry is checked, the 'checkSymmetry' arguments in 'mvtnorm::rmvnorm()' are set to FALSE.

References

Pocuca, N., Gallaugher, M.P., Clark, K.M., & McNicholas, P.D. (2019). Assessing and Visualizing Matrix Variate Normality. Methodology. <https://arxiv.org/abs/1910.02859>

Gupta, A. K. and D. K. Nagar (1999). Matrix Variate Distributions. Boca Raton: Chapman & Hall/CRC Press.

Examples

```
# Data Used
# if( !requireNamespace("datasets", quietly = TRUE)) { install.packages("datasets")} #part of baseR.
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, 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(1, M, U, V)

# M has a different sample size than U; will return an error.
## Not run:
M <- cbind(rnorm(4, 435, 296), rnorm(4, 27, 11))
rmatnorm(M, U, V)

## End(Not run)
```

`special.matrix` *Generating Special Matrices*

Description

Creates an Identity Matrix *I* and a Matrix of Ones *J*.

- `I()`: Creates 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*. Names of rows and columns (`dimnames`) are included.
- `J()`: Creates a matrix of ones with any number of rows and columns. Names of rows and columns (`dimnames`) are included.

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.

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*, a m x n matrix.

Usage

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

Arguments

A	A matrix with m rows and n columns.
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

1. Unlike other 'vec()' functions on CRAN, matrixNormal versions inherit names from matrices to their vectorized forms.
2. vec() was adapted from Frederick Novomestky's **matrixcalc**. This function is edited so that it can take dimension names and return the matrix as a vector.
3. These functions were used as accessories used in matrixNormal functions.

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.
```

vech *Half-Vectorization of a matrix*

Description

Stacks elements of the lower triangle of a numeric symmetric matrix *A*.

Usage

```
vech(A, use.Names = TRUE, tol = .Machine$double.eps^0.5)
```

Arguments

A	A matrix with m rows and n columns.
use.Names	Logical. If TRUE, the names of A are taken to be names of the stacked matrix. Default: TRUE.
tol	A numeric tolerance level used to check if a matrix is symmetric. That is, a matrix is symmetric if the difference between the matrix and its transpose is between -tol and tol.

Details

For a symmetric matrix *A*, the vectorization of *A* contains more information than necessary. The half-vectorization, denoted vech(), of a symmetric square n by n matrix *A* is the vectorization of the lower triangular portion.

Value

A vector with n(n+1)/2 elements.

Note

Unlike other vech() functions available on CRAN, matrixNormal versions inherit names from matrices to their vectorized forms.

Examples

```
x <- matrix(c(1, 2, 2, 4),
  nrow = 2, byrow = TRUE,
  dimnames = list(1:2, c("Sex", "Smoker"))
)
print(x)

# Example 1
vech(x)
# If you just want the vectorized form
vech(x, use.Names = FALSE)
```

```
# Example 2: If one has NA's
x[1, 2] <- x[2, 1] <- NA
vech(x)
```

Index