Package ‘MixMatrix’

July 21, 2025

Type Package

Title Classification with Matrix Variate Normal and t Distributions

Version 0.2.8

Description Provides sampling and density functions for matrix variate normal, t, and inverted t distributions; ML estimation for matrix variate normal and t distributions using the EM algorithm, including some restrictions on the parameters; and classification by linear and quadratic discriminant analysis for matrix variate normal and t distributions described in Thompson et al. (2019) <doi:10.1080/10618600.2019.1696208>. Performs clustering with matrix variate normal and t mixture models.

Depends R (>= 3.5.0)

Imports stats, CholWishart, Rcpp, glue

Suggests knitr, rmarkdown, testthat, covr, ggplot2, dplyr, magrittr, spelling

VignetteBuilder knitr

URL `https://github.com/gzt/MixMatrix/`, `https://gzt.github.io/MixMatrix/`

BugReports `https://github.com/gzt/MixMatrix/issues`

Language en-us

License GPL-3

RoxygenNote 7.3.2

Encoding UTF-8

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Author Geoffrey Thompson [aut, cre] (ORCID: <`https://orcid.org/0000-0003-2436-8822`>), B. D. Ripley [ctb, cph] (author of original lda and qda functions), W. N. Venables [ctb, cph] (author of original lda and qda functions)

Maintainer Geoffrey Thompson <`gzthompson@gmail.com`>

Repository CRAN

Date/Publication 2024-09-30 20:40:02 UTC

Contents

`ARgenerate` *Generate a unit AR(1) covariance matrix*

Description

generate AR(1) correlation matrices

Usage

```
ARgenerate(n, rho)
```

Arguments

`n`	number of columns/rows
`rho`	correlation parameter

Value

Toeplitz $n \times n$ matrix with 1 on the diagonal and rho^k on the other diagonals, where k is distance from the main diagonal. Used internally but it is useful for generating your own random matrices.

See Also

`stats::toeplitz()`

Examples

```
ARgenerate(6, .9)
```

`CSgenerate` *Generate a compound symmetric correlation matrix*

Description

Generate a compound symmetric correlation matrix

Usage

```
CSgenerate(n, rho)
```

Arguments

`n`	number of dimensions
`rho`	off-diagonal element - a correlation between -1 and 1. Will warn if less than 0.

Value

returns an $n \times n$ matrix with 1 on the diagonal and `rho` on the off-diagonal.

Examples

```
# generates a covariance matrix with 1 on the main diagonal
# and 0.5 on the off-diagonal elements.
CSgenerate(3, .5)
```

`init_matrixmixture` *Initializing settings for Matrix Mixture Models*

Description

Providing this will generate a list suitable for use as the `init` argument in the `matrixmixture` function. Either provide data and it will select centers and variance matrices to initialize or provide initial values and it will format them as expected for the function.

Usage

```
init_matrixmixture(
  data,
  prior = NULL,
  K = length(prior),
  centers = NULL,
  U = NULL,
  V = NULL,
  centermethod = "kmeans",
  varmethod = "identity",
```

```
  model = "normal",
  init = NULL,
  ...
)
```

Arguments

`data`	data, $p \times q \times n$ array
`prior`	prior probability. One of `prior` and `K` must be provided. They must be consistent if both provided.
`K`	number of groups
`centers`	(optional) either a matrix or an array of $p \times p$ matrices for use as the `centers` argument. If fewer than `K` are provided, the remainder are chosen by `centermethod`.
`U`	(optional) either a matrix or an array of $p \times p$ matrices for use as the `U` argument. If a matrix is provided, it is duplicated to provide an array. If an array is provided, it should have `K` slices.
`V`	(optional) either a matrix or an array of matrices for use as the `V` argument. If a matrix is provided, it is duplicated to provide an array. If an array is provided, it should have `K` slices.
`centermethod`	what method to use to generate initial centers. Currently support random start (`random`) or performing k-means (`kmeans`) on the vectorized version for a small number of iterations and then converting back. By default, if `K` centers are provided, nothing will be done.
`varmethod`	what method to use to choose initial variance matrices. Currently only identity matrices are created. By default, if `U` and `V` matrices are provided, nothing will be done.
`model`	whether to use a normal distribution or a t-distribution, not relevant for more initialization methods.
`init`	(optional) a (possibly partially-formed) list with some of the components `centers`, `U`, and `V`. The function will complete the list and fill out missing entries.
`...`	Additional arguments to pass to `kmeans()` if that is `centermethod`.

Value

a list suitable to use as the `init` argument in `matrixmixture`:

`centers` the group means, a $p \times q \times K$ array.

`U` the between-row covariance matrices, a $p \times p \times K$ array

`V` the between-column covariance matrix, a $q \times q \times K$ array

See Also

`matrixmixture()`

Examples

```
 set.seed(20180221)
A <- rmatrixt(30,mean=matrix(0,nrow=3,ncol=4), df = 10)
# 3x4 matrices with mean 0
B <- rmatrixt(30,mean=matrix(2,nrow=3,ncol=4), df = 10)
# 3x4 matrices with mean 2
C <- array(c(A,B), dim=c(3,4,60)) # combine into one array
prior <- c(.5,.5) # equal probability prior
init = init_matrixmixture(C, prior = prior)
# will find two centers using the "kmeans" method on the vectorized matrices
```

`matrixlda` *LDA for matrix variate distributions*

Description

Performs linear discriminant analysis on matrix variate data. This works slightly differently from the LDA function in MASS: it does not sphere the data or otherwise normalize it. It presumes equal variance matrices and probabilities are given as if the data are from a matrix variate normal distribution. The estimated variance matrices are weighted by the prior. However, if there are not enough members of a class to estimate a variance, this may be a problem. The function does not take the formula interface. If `method = 't'` is selected, this performs discrimination using the matrix variate t distribution, presuming equal covariances between classes.

Usage

```
matrixlda(
  x,
  grouping,
  prior,
  tol = 1e-04,
  method = "normal",
  nu = 10,
  ...,
  subset
)
```

Arguments

`x`	3-D array of matrix data indexed by the third dimension
`grouping`	vector
`prior`	a vector of prior probabilities of the same length as the number of classes
`tol`	by default, `1e-4`. Tolerance parameter checks for 0 variance.
`method`	whether to use the normal distribution (`normal`) or the t distribution (`t`). By default, normal.
`nu`	If using the t-distribution, the degrees of freedom parameter. By default, 10.

`...`	Arguments passed to or from other methods, such as additional parameters to pass to `MLmatrixnorm` (e.g., `row.mean`)
`subset`	An index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)

Value

Returns a list of class `matrixlda` containing the following components:

`prior` the prior probabilities used.

`counts` the counts of group membership

`means` the group means.

`scaling` the scalar variance parameter

`U` the between-row covariance matrix

`V` the between-column covariance matrix

`lev` levels of the grouping factor

`N` The number of observations used.

`method` The method used.

`nu` The degrees of freedom parameter if the t distribution was used.

`call` The (matched) function call.

References

G Z Thompson, R Maitra, W Q Meeker, A Bastawros (2019),
"Classification with the matrix-variate-t distribution", arXiv
e-prints arXiv:1907.09565 <https://arxiv.org/abs/1907.09565>

Ming Li, Baozong Yuan, "2D-LDA: A statistical linear discriminant
analysis for image matrix", Pattern Recognition Letters, Volume 26,
Issue 5, 2005, Pages 527-532, ISSN 0167-8655.

Aaron Molstad & Adam J. Rothman (2019), "A Penalized Likelihood Method for Classification With Matrix-Valued Predictors", Journal of Computational and Graphical Statistics, 28:1, 11-22, doi:10.1080/10618600.2018.1476249 **MatrixLDA**

Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0 **MASS**

See Also

`predict.matrixlda()`, `MASS::lda()`, `MLmatrixnorm()` and `MLmatrixt()` `matrixqda()`, and `matrixmixture()`

Examples

```
set.seed(20180221)
# construct two populations of 3x4 random matrices with different means
A <- rmatrixnorm(30, mean = matrix(0, nrow = 3, ncol = 4))
B <- rmatrixnorm(30, mean = matrix(1, nrow = 3, ncol = 4))
```

```
C <- array(c(A, B), dim = c(3, 4, 60)) # combine together
groups <- c(rep(1, 30), rep(2, 30)) # define groups
prior <- c(.5, .5) # set prior
D <- matrixlda(C, groups, prior) # fit model
logLik(D)
print(D)
```

`matrixmixture` *Fit a matrix variate mixture model*

Description

Clustering by fitting a mixture model using EM with `K` groups and unconstrained covariance matrices for a matrix variate normal or matrix variate t distribution (with specified degrees of freedom `nu`).

Usage

```
matrixmixture(
  x,
  init = NULL,
  prior = NULL,
  K = length(prior),
  iter = 1000,
  model = "normal",
  method = NULL,
  row.mean = FALSE,
  col.mean = FALSE,
  tolerance = 0.1,
  nu = NULL,
  ...,
  verbose = 0,
  miniter = 5,
  convergence = TRUE
)
```

Arguments

`x`	data, $p \times q \times n$ array
`init`	a list containing an array of `K` of $p \times q$ means labeled `centers`, and optionally $p \times p$ and $q \times q$ positive definite variance matrices labeled `U` and `V`. By default, those are presumed to be identity if not provided. If `init` is missing, it will be provided using the `prior` or `K` by `init_matrixmix`.
`prior`	prior for the `K` classes, a vector that adds to unity
`K`	number of classes - provide either this or the prior. If this is provided, the prior will be of uniform distribution among the classes.
`iter`	maximum number of iterations.

`model`	whether to use the `normal` or `t` distribution.
`method`	what method to use to fit the distribution. Currently no options.
`row.mean`	By default, `FALSE`. If `TRUE`, will fit a common mean within each row. If both this and `col.mean` are `TRUE`, there will be a common mean for the entire matrix.
`col.mean`	By default, `FALSE`. If `TRUE`, will fit a common mean within each row. If both this and `row.mean` are `TRUE`, there will be a common mean for the entire matrix.
`tolerance`	convergence criterion, using Aitken acceleration of the log-likelihood by default.
`nu`	degrees of freedom parameter. Can be a vector of length `K`.
`...`	pass additional arguments to `MLmatrixnorm` or `MLmatrixt`
`verbose`	whether to print diagnostic output, by default `0`. Higher numbers output more results.
`miniter`	minimum number of iterations
`convergence`	By default, `TRUE`, using Aitken acceleration to determine convergence. If false, it instead checks if the change in log-likelihood is less than `tolerance`. Aitken acceleration may prematurely end in the first few steps, so you may wish to set `miniter` or select `FALSE` if this is an issue.

Value

A list of class `MixMatrixModel` containing the following components:

`prior` the prior probabilities used.

`init` the initialization used.

`K` the number of groups

`N` the number of observations

`centers` the group means.

`U` the between-row covariance matrices

`V` the between-column covariance matrix

`posterior` the posterior probabilities for each observation

`pi` the final proportions

`nu` The degrees of freedom parameter if the t distribution was used.

`convergence` whether the model converged

`logLik` a vector of the log-likelihoods of each iteration ending in the final log-likelihood of the model

`model` the model used

`method` the method used

`call` The (matched) function call.

References

```
Andrews, Jeffrey L., Paul D. McNicholas, and Sanjeena Subedi. 2011.
  "Model-Based Classification via Mixtures of Multivariate
  T-Distributions." Computational Statistics & Data Analysis 55 (1):
  520–29. \doi{10.1016/j.csda.2010.05.019}.

Fraley, Chris, and Adrian E Raftery. 2002. "Model-Based Clustering,
   Discriminant Analysis, and Density Estimation." Journal of the
   American Statistical Association 97 (458). Taylor & Francis: 611–31.
   \doi{10.1198/016214502760047131}.

McLachlan, Geoffrey J, Sharon X Lee, and Suren I Rathnayake. 2019.
      "Finite Mixture Models." Annual Review of Statistics and Its
      Application 6. Annual Reviews: 355–78.
      \doi{10.1146/annurev-statistics-031017-100325}.

Viroli, Cinzia. 2011. "Finite Mixtures of Matrix Normal Distributions
      for Classifying Three-Way Data." Statistics and Computing 21 (4):
      511–22. \doi{10.1007/s11222-010-9188-x}.
```

See Also

`init_matrixmixture()`

Examples

```
set.seed(20180221)
A <- rmatrixt(20,mean=matrix(0,nrow=3,ncol=4), df = 5)
# 3x4 matrices with mean 0
B <- rmatrixt(20,mean=matrix(1,nrow=3,ncol=4), df = 5)
# 3x4 matrices with mean 1
C <- array(c(A,B), dim=c(3,4,40)) # combine into one array
prior <- c(.5,.5) # equal probability prior
# create an intialization object, starts at the true parameters
init = list(centers = array(c(rep(0,12),rep(1,12)), dim = c(3,4,2)),
              U = array(c(diag(3), diag(3)), dim = c(3,3,2))*20,
              V = array(c(diag(4), diag(4)), dim = c(4,4,2))
 )
# fit model
 res<-matrixmixture(C, init = init, prior = prior, nu = 5,
                    model = "t", tolerance = 1e-3, convergence = FALSE)
print(res$centers) # the final centers
print(res$pi) # the final mixing proportion
plot(res) # the log likelihood by iteration
logLik(res) # log likelihood of final result
BIC(res) # BIC of final result
predict(res, newdata = C[,,c(1,21)]) # predicted class membership
```

`matrixqda` *Quadratic Discriminant Analysis for Matrix Variate Observations*

Description

See `matrixlda`: quadratic discriminant analysis for matrix variate observations.

Usage

```
matrixqda(
  x,
  grouping,
  prior,
  tol = 1e-04,
  method = "normal",
  nu = 10,
  ...,
  subset
)
```

Arguments

`x`	3-D array of matrix data indexed by the third dimension
`grouping`	vector
`prior`	a vector of prior probabilities of the same length as the number of classes
`tol`	by default, `1e-4`. Tolerance parameter checks for 0 variance.
`method`	whether to use the normal distribution (`normal`) or the t distribution (`t`). By default, normal.
`nu`	If using the t-distribution, the degrees of freedom parameter. By default, 10.
`...`	Arguments passed to or from other methods, such as additional parameters to pass to `MLmatrixnorm` (e.g., `row.mean`)
`subset`	An index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)

Details

This uses `MLmatrixnorm` or `MLmatrixt` to find the means and variances for the case when different groups have different variances.

Value

Returns a list of class `matrixqda` containing the following components:

`prior` the prior probabilities used.

`counts` the counts of group membership

`means` the group means.

`U` the between-row covariance matrices

`V` the between-column covariance matrices

`lev` levels of the grouping factor

`N` The number of observations used.

`method` The method used.

`nu` The degrees of freedom parameter if the t-distribution was used.

`call` The (matched) function call.

References

G Z Thompson, R Maitra, W Q Meeker, A Bastawros (2019),
"Classification with the matrix-variate-t distribution", arXiv
e-prints arXiv:1907.09565 <https://arxiv.org/abs/1907.09565>

Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

Pierre Dutilleul. The MLE algorithm for the matrix normal distribution.
Journal of Statistical Computation and Simulation, (64):105-123, 1999.

See Also

`predict.matrixqda()`, `MASS::qda()`, `MLmatrixnorm()`, `MLmatrixt()`, `matrixlda()`, and `matrixmixture()`

Examples

```
set.seed(20180221)
# construct two populations of 3x4 random matrices with different means
A <- rmatrixnorm(30, mean = matrix(0, nrow = 3, ncol = 4))
B <- rmatrixnorm(30, mean = matrix(1, nrow = 3, ncol = 4))
C <- array(c(A, B), dim = c(3, 4, 60)) # combine together
groups <- c(rep(1, 30), rep(2, 30)) # define groups
prior <- c(.5, .5) # set prior
D <- matrixqda(C, groups, prior)
logLik(D)
print(D)
```

`MLmatrixnorm` *Maximum likelihood estimation for matrix normal distributions*

Description

Maximum likelihood estimates exist for $N > max(p/q, q/p)+1$ and are unique for $N > max(p, q)$. This finds the estimate for the mean and then alternates between estimates for the U and V matrices until convergence. An AR(1), compound symmetry, correlation matrix, or independence restriction can be proposed for either or both variance matrices. However, if they are inappropriate for the data, they may fail with a warning.

Usage

```
MLmatrixnorm(
  data,
  row.mean = FALSE,
  col.mean = FALSE,
  row.variance = "none",
  col.variance = "none",
  tol = 10 * .Machine$double.eps^0.5,
  max.iter = 100,
  U,
  V,
  ...
)
```

Arguments

`data`	Either a list of matrices or a 3-D array with matrices in dimensions 1 and 2, indexed by dimension 3.
`row.mean`	By default, `FALSE`. If `TRUE`, will fit a common mean within each row. If both this and `col.mean` are `TRUE`, there will be a common mean for the entire matrix.
`col.mean`	By default, `FALSE`. If `TRUE`, will fit a common mean within each row. If both this and `row.mean` are `TRUE`, there will be a common mean for the entire matrix.
`row.variance`	Imposes a variance structure on the rows. Either 'none', 'AR(1)', 'CS' for 'compound symmetry', 'Correlation' for a correlation matrix, or 'Independence' for independent and identical variance across the rows. Only positive correlations are allowed for AR(1) and CS covariances. Note that while maximum likelihood estimators are available (and used) for the unconstrained variance matrices, `optim` is used for any constraints so it may be considerably slower.
`col.variance`	Imposes a variance structure on the columns. Either 'none', 'AR(1)', 'CS', 'Correlation', or 'Independence'. Only positive correlations are allowed for AR(1) and CS.
`tol`	Convergence criterion. Measured against square deviation between iterations of the two variance-covariance matrices.
`max.iter`	Maximum possible iterations of the algorithm.
`U`	(optional) Can provide a starting point for the `U` matrix. By default, an identity matrix.
`V`	(optional) Can provide a starting point for the `V` matrix. By default, an identity matrix.
`...`	(optional) additional arguments can be passed to `optim` if using restrictions on the variance.

Value

Returns a list with a the following elements:

`mean` the mean matrix

`scaling` the scalar variance parameter (the first entry of the covariances are restricted to unity)

`U` the between-row covariance matrix

`V` the between-column covariance matrix

`iter` the number of iterations

`tol` the squared difference between iterations of the variance matrices at the time of stopping

`logLik` vector of log likelihoods at each iteration.

`convergence` a convergence flag, `TRUE` if converged.

`call` The (matched) function call.

References

Pierre Dutilleul. The MLE algorithm for the matrix normal distribution. Journal of Statistical Computation and Simulation, (64):105–123, 1999.

```
Gupta, Arjun K, and Daya K Nagar. 1999. Matrix Variate Distributions.
Vol. 104. CRC Press. ISBN:978-1584880462
```

See Also

`rmatrixnorm()` and `MLmatrixt()`

Examples

```
set.seed(20180202)
# simulating from a given density
A <- rmatrixnorm(
  n = 100, mean = matrix(c(100, 0, -100, 0, 25, -1000), nrow = 2),
  L = matrix(c(2, 1, 0, .1), nrow = 2), list = TRUE
)
# finding the parameters by ML estimation
results <- MLmatrixnorm(A, tol = 1e-5)
print(results)
```

`MLmatrixt` *Maximum likelihood estimation for matrix variate t distributions*

Description

For the matrix variate normal distribution, maximum likelihood estimates exist for $N > max(p/q, q/p) + 1$ and are unique for $N > max(p, q)$. The number necessary for the matrix variate t has not been worked out but this is a lower bound. This implements an ECME algorithm to estimate the mean, covariance, and degrees of freedom parameters. An AR(1), compound symmetry, or independence restriction can be proposed for either or both variance matrices. However, if they are inappropriate for the data, they may fail with a warning.

Usage

```
MLmatrixt(
  data,
  row.mean = FALSE,
  col.mean = FALSE,
  row.variance = "none",
  col.variance = "none",
  df = 10,
  fixed = TRUE,
  tol = .Machine$double.eps^0.5,
  max.iter = 5000,
  U,
  V,
  ...
)
```

Arguments

`data`	Either a list of matrices or a 3-D array with matrices in dimensions 1 and 2, indexed by dimension 3.
`row.mean`	By default, `FALSE`. If `TRUE`, will fit a common mean within each row. If both this and `col.mean` are `TRUE`, there will be a common mean for the entire matrix.
`col.mean`	By default, `FALSE`. If `TRUE`, will fit a common mean within each row. If both this and `row.mean` are `TRUE`, there will be a common mean for the entire matrix.
`row.variance`	Imposes a variance structure on the rows. Either 'none', 'AR(1)', 'CS' for 'compound symmetry', 'Correlation' for a correlation matrix, or 'Independence' for independent and identical variance across the rows. Only positive correlations are allowed for AR(1) and CS and these restrictions may not be guaranteed to converge. Note that while maximum likelihood estimators are available (and used) for the unconstrained variance matrices, `optim` is used for any constraints so it may be considerably slower.
`col.variance`	Imposes a variance structure on the columns. Either 'none', 'AR(1)', 'CS', 'Correlation', or 'Independence'. Only positive correlations are allowed for AR(1) and CS.
`df`	Starting value for the degrees of freedom. If `fixed = TRUE`, then this is required and not updated. By default, set to 10.
`fixed`	Whether `df` is estimated or fixed. By default, `TRUE`.
`tol`	Convergence criterion. Measured against square deviation between iterations of the two variance-covariance matrices.
`max.iter`	Maximum possible iterations of the algorithm.
`U`	(optional) Can provide a starting point for the U matrix. By default, an identity matrix.
`V`	(optional) Can provide a starting point for the V matrix. By default, an identity matrix.
`...`	(optional) additional arguments can be passed to `optim` if using restrictions on the variance.

Value

Returns a list with the following elements:

`mean`	the mean matrix
`U`	the between-row covariance matrix
`V`	the between-column covariance matrix
`var`	the scalar variance parameter (the first entry of the covariances are restricted to unity)
`nu`	the degrees of freedom parameter
`iter`	the number of iterations
`tol`	the squared difference between iterations of the variance matrices at the time of stopping
`logLik`	log likelihood of result.
`convergence`	a convergence flag, `TRUE` if converged.
`call`	The (matched) function call.

References

```
Thompson, G Z.  R Maitra, W Q Meeker, A Bastawros (2019),
"Classification with the matrix-variate-t distribution", arXiv
e-prints arXiv:1907.09565 <https://arxiv.org/abs/1907.09565>

Dickey, James M. 1967. “Matricvariate Generalizations of the
Multivariate t Distribution and the Inverted Multivariate t
Distribution.” Ann. Math. Statist. 38 (2): 511–18.
\doi{10.1214/aoms/1177698967}

Liu, Chuanhai, and Donald B. Rubin. 1994. “The ECME Algorithm:
A Simple Extension of EM and ECM with Faster Monotone Convergence.”
Biometrika 81 (4): 633–48.
      \doi{10.2307/2337067}
```

Meng, Xiao-Li, and Donald B. Rubin. 1993. “Maximum Likelihood Estimation via the ECM Algorithm: A General Framework.” Biometrika 80 (2): 267–78. doi:10.1093/biomet/80.2.267

```
Rubin, D.B. 1983. “Encyclopedia of Statistical Sciences.” In, 4th ed.,
  272–5. John Wiley.
```

See Also

`rmatrixnorm()`, `rmatrixt()`, `MLmatrixnorm()`

Examples

```
set.seed(20180202)
# drawing from a distribution with specified mean and covariance
A <- rmatrixt(
  n = 100, mean = matrix(c(100, 0, -100, 0, 25, -1000), nrow = 2),
  L = matrix(c(2, 1, 0, .1), nrow = 2), list = TRUE, df = 5
```

```
)
# fitting maximum likelihood estimates
results <- MLmatrixt(A, tol = 1e-5, df = 5)
print(results)
```

`predict.matrixlda` *Classify Matrix Variate Observations by Linear Discrimination*

Description

Classify matrix variate observations in conjunction with `matrixlda`.

Usage

```
## S3 method for class 'matrixlda'
predict(object, newdata, prior = object$prior, ...)
```

Arguments

`object`	object of class `matrixlda`
`newdata`	array or list of new observations to be classified. If newdata is missing, an attempt will be made to retrieve the data used to fit the `matrixlda` object.
`prior`	The prior probabilities of the classes, by default the proportions in the training set or what was set in the call to `matrixlda`.
`...`	arguments based from or to other methods

Details

This function is a method for the generic function `predict()` for class "`matrixlda`". It can be invoked by calling `predict(x)` for an object `x` of the appropriate class.

Value

Returns a list containing the following components:

`class`	The MAP classification (a factor)
`posterior`	posterior probabilities for the classes

See Also

`matrixlda()`, `matrixqda()`, and `matrixmixture()`

Examples

```
set.seed(20180221)
# construct two populations of 3x4 random matrices with different means
A <- rmatrixnorm(30, mean = matrix(0, nrow = 3, ncol = 4))
B <- rmatrixnorm(30, mean = matrix(1, nrow = 3, ncol = 4))
C <- array(c(A, B), dim = c(3, 4, 60)) # combine together
groups <- c(rep(1, 30), rep(2, 30)) # define groups
prior <- c(.5, .5) # set prior
D <- matrixlda(C, groups, prior)
predict(D)$posterior[1:10, ]

## S3 method for class 'matrixlda'
```

`predict.matrixqda` *Classify Matrix Variate Observations by Quadratic Discrimination*

Description

Classify matrix variate observations in conjunction with `matrixqda`.

Usage

```
## S3 method for class 'matrixqda'
predict(object, newdata, prior = object$prior, ...)
```

Arguments

`object`	object of class `matrixqda`
`newdata`	array or list of new observations to be classified. If newdata is missing, an attempt will be made to retrieve the data used to fit the `matrixqda` object.
`prior`	The prior probabilities of the classes, by default the proportions in the training set or what was set in the call to `matrixqda`.
`...`	arguments based from or to other methods

Details

This function is a method for the generic function `predict()` for class "`matrixqda`". It can be invoked by calling `predict(x)` for an object `x` of the appropriate class.

Value

Returns a list containing the following components:

`class` The MAP classification (a factor)

`posterior` posterior probabilities for the classes

See Also

`matrixlda()`, `matrixqda()`, and `matrixmixture()`

Examples

```
set.seed(20180221)
# construct two populations of 3x4 random matrices with different means
A <- rmatrixnorm(30, mean = matrix(0, nrow = 3, ncol = 4))
B <- rmatrixnorm(30, mean = matrix(1, nrow = 3, ncol = 4))
C <- array(c(A, B), dim = c(3, 4, 60)) # combine together
groups <- c(rep(1, 30), rep(2, 30)) # define groups
prior <- c(.5, .5) # set prior
D <- matrixqda(C, groups, prior) # fit model
predict(D)$posterior[1:10, ] # predict, show results of first 10
## S3 method for class "matrixqda"
```

`rmatrixinvt` *Distribution functions for matrix variate inverted t distributions*

Description

Generate random samples from the inverted matrix variate t distribution or compute densities.

Usage

```
rmatrixinvt(
  n,
  df,
  mean,
  L = diag(dim(as.matrix(mean))[1]),
  R = diag(dim(as.matrix(mean))[2]),
  U = L %*% t(L),
  V = t(R) %*% R,
  list = FALSE,
  array = NULL
)

dmatrixinvt(
  x,
  df,
  mean = matrix(0, p, n),
  L = diag(p),
  R = diag(n),
  U = L %*% t(L),
  V = t(R) %*% R,
  log = FALSE
)
```

Arguments

`n`	number of observations for generation
`df`	degrees of freedom (> 0, may be non-integer), `df = 0, Inf` is allowed and will return a normal distribution.
`mean`	$p \times q$ This is really a 'shift' rather than a mean, though the expected value will be equal to this if $df > 2$
`L`	$p \times p$ matrix specifying relations among the rows. By default, an identity matrix.
`R`	$q \times q$ matrix specifying relations among the columns. By default, an identity matrix.
`U`	LL^T - $p \times p$ positive definite matrix for rows, computed from L if not specified.
`V`	$R^T R$ - $q \times q$ positive definite matrix for columns, computed from R if not specified.
`list`	Defaults to `FALSE` . If this is `TRUE` , then the output will be a list of matrices.
`array`	If $n = 1$ and this is not specified and `list` is `FALSE` , the function will return a matrix containing the one observation. If $n > 1$, should be the opposite of `list` . If `list` is `TRUE` , this will be ignored.
`x`	quantile for density
`log`	logical; in `dmatrixt`, if `TRUE`, probabilities `p` are given as `log(p)`.

Value

`rmatrixinvt` returns either a list of n $p \times q$ matrices or a $p \times q \times n$ array.

`dmatrixinvt` returns the density at `x`.

References

Gupta, Arjun K, and Daya K Nagar. 1999. Matrix Variate Distributions. Vol. 104. CRC Press. ISBN:978-1584880462

Dickey, James M. 1967. “Matricvariate Generalizations of the Multivariate t Distribution and the Inverted Multivariate t Distribution.” Ann. Math. Statist. 38 (2): 511–18. doi:10.1214/aoms/1177698967

See Also

`rmatrixnorm()`, `rmatrixt()`, and `stats::Distributions()`.

Examples

```
# an example of drawing from the distribution and computing the density.
A <- rmatrixinvt(n = 2, df = 10, diag(4))
dmatrixinvt(A[, , 1], df = 10, mean = diag(4))
```

`rmatrixnorm` *Matrix variate Normal distribution functions*

Description

Density and random generation for the matrix variate normal distribution

Usage

```
rmatrixnorm(
  n,
  mean,
  L = diag(dim(as.matrix(mean))[1]),
  R = diag(dim(as.matrix(mean))[2]),
  U = L %*% t(L),
  V = t(R) %*% R,
  list = FALSE,
  array = NULL,
  force = FALSE
)

dmatrixnorm(
  x,
  mean = matrix(0, p, n),
  L = diag(p),
  R = diag(n),
  U = L %*% t(L),
  V = t(R) %*% R,
  log = FALSE
)
```

Arguments

`n`	number of observations to generate - must be a positive integer.
`mean`	$p \times q$ matrix of means
`L`	$p \times p$ matrix specifying relations among the rows. By default, an identity matrix.
`R`	$q \times q$ matrix specifying relations among the columns. By default, an identity matrix.
`U`	LL^T - $p \times p$ positive definite variance-covariance matrix for rows, computed from L if not specified.
`V`	$R^T R$ - $q \times q$ positive definite variance-covariance matrix for columns, computed from R if not specified.
`list`	Defaults to `FALSE` . If this is `TRUE` , then the output will be a list of matrices.
`array`	If $n = 1$ and this is not specified and `list` is `FALSE` , the function will return a matrix containing the one observation. If $n > 1$, should be the opposite of `list` . If `list` is `TRUE` , this will be ignored.

`force`	If TRUE, will take the input of `L` and/or `R` directly - otherwise computes `U` and `V` and uses Cholesky decompositions. Useful for generating degenerate normal distributions. Will also override concerns about potentially singular matrices unless they are not, in fact, invertible.
`x`	quantile for density
`log`	logical; if TRUE, probabilities p are given as log(p).

Value

`rmatrixnorm` returns either a list of n $p \times q$ matrices or a $p \times q \times n$ array.

`dmatrixnorm` returns the density at `x`.

References

Gupta, Arjun K, and Daya K Nagar. 1999. Matrix Variate Distributions. Vol. 104. CRC Press. ISBN:978-1584880462

See Also

`rmatrixt()`, `rmatrixinvt()`, `rnorm()` and `stats::Distributions()`

Examples

```
set.seed(20180202)
# a draw from a matrix variate normal with a certain mean
# and row-wise covariance
rmatrixnorm(
  n = 1, mean = matrix(c(100, 0, -100, 0, 25, -1000), nrow = 2),
  L = matrix(c(2, 1, 0, .1), nrow = 2), list = FALSE
)
set.seed(20180202)
# another way of specifying this - note the output is equivalent
A <- rmatrixnorm(
  n = 10, mean = matrix(c(100, 0, -100, 0, 25, -1000), nrow = 2),
  L = matrix(c(2, 1, 0, .1), nrow = 2), list = TRUE
)
A[[1]]
# demonstrating the dmatrixnorm function
dmatrixnorm(A[[1]],
  mean = matrix(c(100, 0, -100, 0, 25, -1000), nrow = 2),
  L = matrix(c(2, 1, 0, .1), nrow = 2), log = TRUE
)
```

`rmatrixt` *Distribution functions for the matrix variate t distribution.*

Description

Density and random generation for the matrix variate t distribution.

Usage

```
rmatrixt(
  n,
  df,
  mean,
  L = diag(dim(as.matrix(mean))[1]),
  R = diag(dim(as.matrix(mean))[2]),
  U = L %*% t(L),
  V = t(R) %*% R,
  list = FALSE,
  array = NULL,
  force = FALSE
)

dmatrixt(
  x,
  df,
  mean = matrix(0, p, n),
  L = diag(p),
  R = diag(n),
  U = L %*% t(L),
  V = t(R) %*% R,
  log = FALSE
)
```

Arguments

`n`	number of observations for generation
`df`	degrees of freedom (> 0, may be non-integer), `df = 0, Inf` is allowed and will return a normal distribution.
`mean`	$p \times q$ This is really a 'shift' rather than a mean, though the expected value will be equal to this if $df > 2$
`L`	$p \times p$ matrix specifying relations among the rows. By default, an identity matrix.
`R`	$q \times q$ matrix specifying relations among the columns. By default, an identity matrix.
`U`	LL^T - $p \times p$ positive definite matrix for rows, computed from L if not specified.
`V`	$R^T R$ - $q \times q$ positive definite matrix for columns, computed from R if not specified.

`list`	Defaults to `FALSE` . If this is `TRUE` , then the output will be a list of matrices.
`array`	If $n = 1$ and this is not specified and `list` is `FALSE` , the function will return a matrix containing the one observation. If $n > 1$, should be the opposite of `list` . If `list` is `TRUE` , this will be ignored.
`force`	In `rmatrix`: if `TRUE`, will take the input of `R` directly - otherwise uses `V` and uses Cholesky decompositions. Useful for generating degenerate t-distributions. Will also override concerns about potentially singular matrices unless they are not, in fact, invertible.
`x`	quantile for density
`log`	logical; in `dmatrixt`, if `TRUE`, probabilities `p` are given as `log(p)`.

Details

The matrix t-distribution is parameterized slightly differently from the univariate and multivariate t-distributions

- the variance is scaled by a factor of `1/df`. In this parameterization, the variance for a 1×1 matrix variate t-distributed random variable with identity variance matrices is $1/(df - 2)$ instead of $df/(df - 2)$. A Central Limit Theorem for the matrix variate T is then that as `df` goes to infinity, $MVT(0, df, I_p, df * I_q)$ converges to $MVN(0, I_p, I_q)$.

Value

`rmatrixt` returns either a list of n $p \times q$ matrices or a $p \times q \times n$ array.

`dmatrixt` returns the density at `x`.

References

Gupta, Arjun K, and Daya K Nagar. 1999. Matrix Variate Distributions. Vol. 104. CRC Press. ISBN:978-1584880462

Dickey, James M. 1967. “Matricvariate Generalizations of the Multivariate t Distribution and the Inverted Multivariate t Distribution.” Ann. Math. Statist. 38 (2): 511–18. doi:10.1214/aoms/1177698967

See Also

`rmatrixnorm()`, `rmatrixinvt()`,`rt()` and `stats::Distributions()`.

Examples

```
set.seed(20180202)
# random matrix with df = 10 and the given mean and L matrix
rmatrixt(
  n = 1, df = 10, mean = matrix(c(100, 0, -100, 0, 25, -1000), nrow = 2),
  L = matrix(c(2, 1, 0, .1), nrow = 2), list = FALSE
)
# comparing 1-D distribution of t to matrix
summary(rt(n = 100, df = 10))
summary(rmatrixt(n = 100, df = 10, matrix(0)))
```

```
# demonstrating equivalence of 1x1 matrix t to usual t
set.seed(20180204)
x <- rmatrixt(n = 1, mean = matrix(0), df = 1)
dt(x, 1)
dmatrixt(x, df = 1)
```

Index