Package 'sparseMVN'

October 25, 2021

Type Package

Title Multivariate Normal Functions for Sparse Covariance and Precision Matrices

Version 0.2.2

Date 2021-10-19

Maintainer Michael Braun <braunm@smu.edu>

URL https://braunm.github.io/sparseMVN/,
https://github.com/braunm/sparseMVN/

BugReports https://github.com/braunm/sparseMVN/issues/

Description Computes multivariate normal (MVN) densities, and samples from MVN distributions, when the covariance or precision matrix is sparse.

License MPL (>= 2.0)

Depends R (>= 3.4.0)

Imports Matrix (>= 1.3), methods

Suggests dplyr (>= 1.0), tidyr (>= 1.1), ggplot2 (>= 3.3), forcats (>= 0.5), mvtnorm (>= 1.0.6) , knitr, bookdown, kableExtra, testthat, scales, trustOptim (>= 0.8.5)

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.1.2

NeedsCompilation no

Author Michael Braun [aut, cre, cph] (<https://orcid.org/0000-0003-4774-2119>)

Repository CRAN

Date/Publication 2021-10-25 12:40:02 UTC

R topics documented:

`sparseMVN-package` *Multivariate Normal Functions for Sparse Covariate and Precision Matrices*

Description

MVN functions for sparse covariance and precision matrices.

Details

Computes multivariate normal (MVN) densities, and samples from MVN distributions, when either the covariance or precision matrix is stored as a sparse Matrix (a dsCMatrix object, as defined in the Matrix package. The user can provide the precision matrix directly, rather than convert it to a covariance via matrix inversion.

Author(s)

Maintainer: Michael Braun <braunm@smu.edu> (ORCID) [copyright holder]

See Also

Useful links:

- https://braunm.github.io/sparseMVN/
- https://github.com/braunm/sparseMVN/
- Report bugs at https://github.com/braunm/sparseMVN/issues/

`binary` *Binary choice example*

Description

Functions for binary choice example in the vignette.

Usage

```
binary.f(P, data, priors, order.row = FALSE)

binary.grad(P, data, priors, order.row = FALSE)

binary.hess(P, data, priors, order.row = FALSE)

binary.sim(N, k, T)
```

Arguments

`P`	Numeric vector of length $(N+1)k$. First Nk elements are heterogeneous coefficients. The remaining k elements are population parameters.
`data`	Named list of data matrices Y and X, and choice count integer T
`priors`	Named list of matrices inv.Omega and inv.A.
`order.row`	Determines order of heterogeneous coefficients in parameter vector. If TRUE, heterogeneous coefficients are ordered by unit. If FALSE, they are ordered by covariate.
`N`	Number of heterogeneous units
`k`	Number of heterogeneous parameters
`T`	Observations per household

Details

These functions are used by the heterogeneous binary choice example in the vignette. There are N heterogeneous units, each making T binary choices. The choice probabilities depend on k covariates. binary.sim simulates a dataset suitable for running the example.

Value

For binary.f, binary.df and binary.hess, the log posterior density, gradient and Hessian, respectively. The Hessian is a dgCMatrix object. binary.sim returns a list with simulated Y and X, and the input T.

dmvn.sparse *Compute density from multivariate normal distribution*

Description

Compute density from multivariate normal distribution

Usage

```
dmvn.sparse(x, mu, CH, prec = TRUE, log = TRUE)
```

Arguments

x	numeric matrix, where each row is an MVN sample.
mu	mean (numeric vector)
CH	An object of class dCHMsimpl or dCHMsuper that represents the Cholesky factorization of either the precision (default) or covariance matrix. See details.
prec	If TRUE, CH is the Cholesky decomposition of the precision matrix. If false, it is the decomposition for the covariance matrix.
log	If TRUE (default), returns the log density, else returns density.

Value

A density or log density for each row of x

Details

This function use sparse matrix operations to compute the log density of a multivariate normal distribution. The user must compute the Cholesky decomposition first, using the Cholesky function in the Matrix package. This function operates on a sparse symmetric matrix, and returns an object of class dCHMsimpl or dCHMsuper (this depends on the algorithm that was used for the decomposition). This object contains information about any fill-reducing permutations that were used to preserve sparsity. The rmvn.sparse and dmvn.sparse functions use this permutation information, even if pivoting was turned off.

Examples

```
   require(Matrix)
   m <- 20
   p <- 2
   k <- 4

## build sample sparse covariance matrix
   Q1 <- tril(kronecker(Matrix(seq(0.1,p,length=p*p),p,p),diag(m)))
   Q2 <- cbind(Q1,Matrix(0,m*p,k))
   Q3 <- rbind(Q2,cbind(Matrix(rnorm(k*m*p),k,m*p),Diagonal(k)))
   V <- tcrossprod(Q3)
```

```
CH <- Cholesky(V)

x <- rmvn.sparse(10,rep(0,p*m+k),CH, FALSE)
y <- dmvn.sparse(x[1,],rep(0,p*m+k), CH, FALSE)
```

`rmvn.sparse` *Sample from multivariate normal distribution*

Description

Efficient sampling and density calculation from a multivariate normal, when the covariance or precision matrix is sparse. These functions are designed for MVN samples of very large dimension.

Usage

```
rmvn.sparse(n, mu, CH, prec = TRUE)
```

Arguments

`n`	number of samples
`mu`	mean (numeric vector)
`CH`	An object of class dCHMsimpl or dCHMsuper that represents the Cholesky factorization of either the precision (default) or covariance matrix. See details.
`prec`	If TRUE, CH is the Cholesky decomposition of the precision matrix. If false, it is the decomposition for the covariance matrix.

Value

A matrix of samples from an MVN distribution (one in each row)

Details

This function uses sparse matrix operations to sample from a multivariate normal distribution. The user must compute the Cholesky decomposition first, using the Cholesky function in the Matrix package. This function operates on a sparse symmetric matrix, and returns an object of class dCHMsimpl or dCHMsuper (this depends on the algorithm that was used for the decomposition). This object contains information about any fill-reducing permutations that were used to preserve sparsity. The rmvn.sparse and dmvn.sparse functions use this permutation information, even if pivoting was turned off.

Examples

```
require(Matrix)
m <- 20
p <- 2
k <- 4

## build sample sparse covariance matrix
Q1 <- tril(kronecker(Matrix(seq(0.1,p,length=p*p),p,p),diag(m)))
Q2 <- cbind(Q1,Matrix(0,m*p,k))
Q3 <- rbind(Q2,cbind(Matrix(rnorm(k*m*p),k,m*p),Diagonal(k)))
V <- tcrossprod(Q3)
CH <- Cholesky(V)

x <- rmvn.sparse(10,rep(0,p*m+k),CH, FALSE)
y <- dmvn.sparse(x[1,],rep(0,p*m+k), CH, FALSE)
```

Index