Package 'lrpd'

July 30, 2015

Type Package

Title Low-rank Positive Definite Matrices

Version 1.0

Date 2015-07-29

Author Ye Wang (Eric)

Maintainer <ericwang921198@gmail.com>

Description Efficient matrix inversion and determinant calculation for low-rank positive definite (lrpd) matrices. Efficient multivariate Gaussian density calculation and sampling when its covariance matrix is low-rank.

License GPL-2

R topics documented:

`lrpd-package` *Low-rank Positive Definite Matrices*

Description

Efficient matrix inversion and determinant calculation for *low-rank positive definite (lrpd)* matrices. Efficient multivariate Gaussian density calculation and sampling when its covariance matrix is low-rank.

Details

Package:	lrpd
Type:	Package
Version:	1.0
Date:	2015-07-29
License:	GPL-2

This pacakge provides efficient `solve()` and `determinant()` function for class "lrpd". Moreover, the pacakge also provides efficient density calculation, `dlrmvnorm()`, and sampling from multivariate Gaussian, `rlrmvnorm()`, whose covariance is low-rank.

Author(s)

Ye Wang (Eric)

Maintainer: <ericwang921198@gmail.com>

Examples

```
set.seed(50)
K <- 100
L <- matrix(rnorm(K*floor(K/10)),K,floor(K/10))
Sl <- matrix(rnorm(floor(K/10)*floor(K/10)),floor(K/10),floor(K/10))
S <- Sl%*%t(Sl)+diag(rnorm(floor(K/10))^2)
N <- rnorm(K)^2
mat <- lrpd(N,L,S)

# Matrix operation
res1 <- solve(mat)
res2 <- determinant(mat)
# Mutivariate Gaussian
mu <- rnorm(K)
x <- matrix(rnorm(K*10),10,K)
res3 <- dlrmvnorm(x,mu,mat)
res4 <- rlrmvnorm(100,mu,mat)
```

`determinant.lrpd` *Determinant*

Description

This function efficiently computes the determinant of an lrpd matrix.

Usage

```
## S3 method for class 'lrpd'
determinant(object, logarithm = TRUE, ...)
```

Arguments

object	an object of class "lrpd".
logarithm	logical; If TRUE (default) return the logarithm of the determinant.
...	not used.

Value

scalar	(logarithm of) the determinant.

Author(s)

Ye Wang (Eric)

Maintainer: <ericwang921198@gmail.com>

See Also

solve.lrpd for inverse.

Examples

```
library(lrpd)

set.seed(2)
K <- 1000
L <- matrix(rnorm(K*floor(K/10)),K,floor(K/10))
Sl <- matrix(rnorm(floor(K/10)*floor(K/10)),floor(K/10),floor(K/10))
S <- Sl%*%t(Sl)+diag(rnorm(floor(K/10))^2)
N <- rnorm(K)^2
R <- L%*%S%*%t(L) + diag(N)

mat <- lrpd(N,L,S)

system.time(RI1 <- as.numeric(determinant(R)$modulus))
system.time(RI2 <- determinant(mat))

all.equal(RI1,RI2)
```

---

lrmvnorm *Multivariate Gaussian with Low-rank Covariance*

Description

Efficient density calculation and sampling.

Usage

```
dlrmvnorm(x, mu, Sigma, logarithm = TRUE, ...)
rlrmvnorm(n, mu, Sigma, ...)
```

Arguments

x	vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
n	number of observations.
mu	mean vector.
Sigma	covariance matrix.
logarithm	logical; if TRUE (default), the logarithm of the densities are given.
...	not used.

Author(s)

Ye Wang (Eric)

Maintainer: <ericwang921198@gmail.com>

Examples

```
## Not run:
library(mvtnorm)
library(lrpd)

set.seed(2)
K <- 1000
L <- matrix(rnorm(K*floor(K/10)),K,floor(K/10))
Sl <- matrix(rnorm(floor(K/10)*floor(K/10)),floor(K/10),floor(K/10))
S <- Sl%*%t(Sl)+diag(rnorm(floor(K/10))^2)
N <- rnorm(K)^2
R <- L%*%S%*%t(L) + diag(N)
# Gaussian mean vectors and synthethic data
mu <- rnorm(K)
x <- matrix(rnorm(K*10),10,K)

mat <- lrpd(N,L,S)

# logarithm of densities
system.time(RI1 <- dmvnorm(x,mu,R,log=TRUE))
system.time(RI2 <- dlrmvnorm(x,mu,mat))

all.equal(RI1,RI2)

# random samples
system.time(RI1 <- rmvnorm(100,mu,R))
system.time(RI2 <- rlrmvnorm(100,mu,mat))

## End(Not run)
```

lrpd *Low-rank Positive Definite Matrices*

Description

Run this function to create a "lrpd" object.

Usage

```
lrpd(N, L, S = diag(rep(1, ncol(L))), ...)

## S3 method for class 'lrpd'
print(object,...)
## S3 method for class 'lrpd'
summary(object,...)
```

Arguments

N	vector containing the diagonal elements.
L	matrix with each column being a factor loading vector.
S	score matrix.
object	object of class "lrpd".
...	not used.

Details

Low-rank positive definite matrix M has the following form

$$M = N + LSL^T,$$

where N is a positive diagonal matrix and S is a positive definite matrix.

Author(s)

Ye Wang (Eric)

Examples

```
set.seed(50)
K <- 100
L <- matrix(rnorm(K*floor(K/10)),K,floor(K/10))
Sl <- matrix(rnorm(floor(K/10)*floor(K/10)),floor(K/10),floor(K/10))
S <- Sl%*%t(Sl)+diag(rnorm(floor(K/10))^2)
N <- rnorm(K)^2

mat <- lrpd(N,L,S)
print(mat)
summary(mat)
```

mult_diag *Efficient Matrix Multiplication with a Diagonal Matrix*

Description

This functoin efficiently computes the matrix multiplication between an arbitrary matrix and a diagonal matrix.

Usage

```
mult_diag(a, b, ...)
```

Arguments

a	matrix to be multiplied with a diagonal matrix.
b	vector containing the diagonal elements of the diagonal matrix; if length(b)==ncol(a), the function returns a
...	not used.

Author(s)

Ye Wang (Eric)

Maintainer: <ericwang921198@gmail.com>

Examples

```
## Not run:
n1 <- 1000; n2 <- 100
A <- matrix(rnorm(n1*n2),n1,n2)
d <- rnorm(n1)
system.time(R1 <- diag(d)
system.time(R2 <- mult_diag(A,d))

all.equal(R1, R2)

## End(Not run)
```

solve.lrpd *Matrix Inverse*

Description

This function efficiently invert an lrpd matrix.

Usage

```
## S3 method for class 'lrpd'
solve(a, b, ...)

invlrpd(object,...)
```

Arguments

a	a square numeric matrix containing the coefficients of the linear system.
b	a numeric vector or matrix giving the right-hand side(s) of the linear system. If missing, b is taken to be an identity matrix and solve will return the inverse of a.
object	an object of class "lrpd".
...	not used.

Value

matrix	the solution of the linear system.

Author(s)

Ye Wang (Eric)

Maintainer: <ericwang921198@gmail.com>

See Also

determinant.lrpd for determinant.

Examples

```
## Not run:
library(MASS)
library(lrpd)

set.seed(2)
K <- 1000
L <- matrix(rnorm(K*floor(K/10)),K,floor(K/10))
Sl <- matrix(rnorm(floor(K/10)*floor(K/10)),floor(K/10),floor(K/10))
S <- Sl%*%t(Sl)+diag(rnorm(floor(K/10))^2)
N <- rnorm(K)^2
R <- L%*%S%*%t(L) + diag(N)

mat <- lrpd(N,L,S)

# Test matrix inverting
system.time(RI1 <- solve(R))
system.time(RI2 <- chol2inv(chol(R)))
system.time(RI3 <- qr.solve(R))
system.time(RI4 <- solve(mat))

all.equal(RI1, RI2)
```

```
all.equal(RI1, RI3)
all.equal(RI1, RI4)

## End(Not run)
```

Index