Package: FastBandChol (via r-universe)

September 16, 2024

Type Package

Title Fast Estimation of a Covariance Matrix by Banding the Cholesky Factor

Version 0.1.1

Author Aaron Molstad <molst029@umn.edu>

Maintainer Aaron Molstad <molst029@umn.edu>

Description Fast and numerically stable estimation of a covariance matrix by banding the Cholesky factor using a modified Gram-Schmidt algorithm implemented in RcppArmadilo. See <http://stat.umn.edu/~molst029> for details on the algorithm.

License GPL-2

LazyData TRUE

Imports Rcpp

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-08-26 16:44:04

Contents

FastBandChol-package 2
banded.chol 3
banded.chol.cv 4
banded.sample 5
banded.sample.cv 6

Index **8**

FastBandChol-package *Fast estimation of covariance matrix by banded Cholesky factor*

Description

Fast and numerically stable estimation of covariance matrix by banding the Cholesky factor using a modified Gram-Schmidt algorithm implemented in RcppArmadilo. See <https://stat.umn.edu/~molst029> for details on the algorithm.

Details

Package:	FastBandChol
Type:	Package
Version:	0.1.0
Date:	2015-08-22
License:	GPL-2

Author(s)

Aaron Molstad

References

Rothman, A.J., Levina, E., and Zhu, J. (2010). A new approach to Cholesky-based covariance regularization in high dimensions. Biometrika, 97(3):539-550.

Examples

```
## set sample size and dimension
n = 20
p = 100

## create covariance with AR1 structure
Sigma = matrix(0, nrow=p, ncol=p)
for(l in 1:p){
  for(m in 1:p){
    Sigma[l,m] = .5^(abs(l-m))
  }
}

## simulation Normal data
eo1 = eigen(Sigma)
Sigma.sqrt = eo1$vec%*%diag(eo1$val^.5)%*%t(eo1$vec)
X = t(Sigma.sqrt%*%matrix(rnorm(n*p), nrow=p, ncol=n))
```

```
## compute estimates
est.sample = banded.sample(X, bandwidth=4)$est
est.chol = banded.chol(X, bandwidth=4)$est
```

`banded.chol` *Computes estimate of covariance matrix by banding the Cholesky factor*

Description

Computes estimate of covariance matrix by banding the Cholesky factor using a modified Gram Schmidt algorithm implemented in RcppArmadillo.

Usage

```
banded.chol(X, bandwidth, centered = FALSE)
```

Arguments

`X`	A data matrix with n rows and p columns. Rows are assumed to be independent realizations from a p-variate distribution with covariance Σ.
`bandwidth`	A positive integer. Must be less than $n - 1$ and $p - 1$.
`centered`	Logical. Is data matrix centered? Default is `centered = FALSE`

Value

A list with

`est`	The estimated covariance matrix.

Examples

```
## set sample size and dimension
n=20
p=100

## create covariance with AR1 structure
Sigma = matrix(0, nrow=p, ncol=p)
for(l in 1:p){
  for(m in 1:p){
    Sigma[l,m] = .5^(abs(l-m))
  }
}

## simulation Normal data
eo1 = eigen(Sigma)
Sigma.sqrt = eo1$vec%*%diag(eo1$val^.5)%*%t(eo1$vec)
```

```
X = t(Sigma.sqrt%*%matrix(rnorm(n*p), nrow=p, ncol=n))

## compute estimate
out1 = banded.chol(X, bandwidth=4)
```

`banded.chol.cv` *Selects bandwidth for Cholesky factorization by cross validation*

Description

Selects bandwidth for Cholesky factorization by k-fold cross validation

Usage

```
banded.chol.cv(X, bandwidth, folds = 3, est.eval = TRUE, Frob = TRUE)
```

Arguments

`X`	A data matrix with n rows and p columns. Rows are assumed to be independent realizations from a p-variate distribution with covariance Σ.
`bandwidth`	A vector of candidate bandwidths. Candidate bandwidths can only positive integers such that the maximum is less than the sample size outside of the kth fold.
`folds`	The number of folds used for cross validation. Default is `folds =3`.
`est.eval`	Logical: `est.eval = TRUE` returns a list with both the selected bandwidth and the estimated covariance matrix. `est.eval=FALSE` returns a list with only the selected bandwidth. The default is `est.eval = TRUE`.
`Frob`	Logical: `Frob = TRUE` uses squared Frobenius norm loss for cross-validation. `Frob = FALSE` uses operator norm loss. Default is `Frob = TRUE`.

Value

a list with

`bandwidth.min`	The bandwidth minimizing cross-validation error.
`est`	The estimated covariance matrix computed with `bandwidth=bandwidth.min`.

Examples

```
## set sample size and dimension
n=20
p=100

## create covariance with AR1 structure
Sigma = matrix(0, nrow=p, ncol=p)
for(l in 1:p){
  for(m in 1:p){
```

```
    Sigma[l,m] = .5^(abs(l-m))
  }
}

## simulation Normal data
eo1 = eigen(Sigma)
Sigma.sqrt = eo1$vec%*%diag(eo1$val^.5)%*%t(eo1$vec)
X = t(Sigma.sqrt%*%matrix(rnorm(n*p), nrow=p, ncol=n))

## perform cross validation
k = 4:7
out1.cv = banded.chol.cv(X, bandwidth=k, folds = 5)
```

`banded.sample` *Computes banded sample covariance matrix*

Description

Estimates a covariance matrix by banding the sample covariance matrix

Usage

```
banded.sample(X, bandwidth, centered = FALSE)
```

Arguments

`X`	A data matrix with n rows and p columns. Rows are assumed to be independent realizations from a p-variate distribution with covariance Σ.
`bandwidth`	A positive integer. Must be less than $p - 1$.
.	
`centered`	Logical. Is data matrix centered? Default is `centered = FALSE`

Value

A list with

`est`	The estimated covariance matrix.

Examples

```
## set sample size and dimension
n=20
p=100

## create covariance with AR1 structure
Sigma = matrix(0, nrow=p, ncol=p)
for(l in 1:p){
```

```
  for(m in 1:p){
    Sigma[l,m] = .5^(abs(l-m))
  }
}

## simulation Normal data
eo1 = eigen(Sigma)
Sigma.sqrt = eo1$vec%*%diag(eo1$val^.5)%*%t(eo1$vec)
X = t(Sigma.sqrt%*%matrix(rnorm(n*p), nrow=p, ncol=n))

## compute estimate
out2 = banded.sample(X, bandwidth=4)
```

`banded.sample.cv` *Selects bandwidth for sample covariance matrix by cross validation*

Description

Selects bandwidth for sample covariance matrix by k-fold cross validation

Usage

```
banded.sample.cv(X, bandwidth, folds = 3, est.eval = TRUE, Frob = TRUE)
```

Arguments

`X`	A data matrix with n rows and p columns. Rows are assumed to be independent realizations from a p-variate distribution with covariance Σ.
`bandwidth`	A vector of candidate bandwidths. Candidate bandwidths can only positive integers such that the maximum is less than $p - 1$
.	
`folds`	The number of folds used for cross validation. Default is `folds =3`.
`est.eval`	Logical: `est.eval = TRUE` returns a list with both the selected bandwidth and the estimated covariance matrix. `est.eval=FALSE` returns a list with only the selected bandwidth. The default is `est.eval = TRUE`.
`Frob`	Logical: `Frob = TRUE` uses squared Frobenius norm loss for cross-validation. `Frob = FALSE` uses operator norm loss. Default is `Frob = TRUE`.

Value

A list with

`bandwidth.min`	the bandwidth minimizing cv error
`est`	the sample covariance matrix at bandwidth.min

Examples

```
## set sample size and dimension
n=20
p=100

## create covariance with AR1 structure
Sigma = matrix(0, nrow=p, ncol=p)
for(l in 1:p){
  for(m in 1:p){
    Sigma[l,m] = .5^(abs(l-m))
  }
}

## simulation Normal data
eo1 = eigen(Sigma)
Sigma.sqrt = eo1$vec%*%diag(eo1$val^.5)%*%t(eo1$vec)
X = t(Sigma.sqrt%*%matrix(rnorm(n*p), nrow=p, ncol=n))

## perform cross validation
k = 4:7
out2.cv = banded.sample.cv(X, bandwidth=k, folds=5)
```

Index

banded.chol, 3
banded.chol.cv, 4
banded.sample, 5
banded.sample.cv, 6

FastBandChol *(*FastBandChol-package*)*, 2
FastBandChol-package, 2