Package ‘CovTools’

November 16, 2017

Type Package
Title Statistical Tools for Covariance Analysis
Version 0.2.1
Description Covariance is of universal prevalence across various disciplines within statistics. We provide a rich collection of geometric and inferential tools for convenient analysis of covariance structures, topics including distance measures, mean covariance estimator, covariance hypothesis test for one-sample and two-sample cases, and covariance estimation. For an introduction to covariance in multivariate statistical analysis, see Schervish (1987) <doi:10.1214/ss/1177013111>.
License GPL (>= 3)
Encoding UTF-8
LazyData true
Depends R (>= 2.14.0)
Imports Rcpp, geigen, shapes, expm, mvtnorm, stats, Matrix, doParallel, foreach, parallel, pracma
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 6.0.1
NeedsCompilation yes
Author Kyoungjae Lee [aut], Lizhen Lin [ctb], Kisung You [aut, cre]
Maintainer Kisung You <kyou@nd.edu>
Repository CRAN
Date/Publication 2017-11-15 23:17:32 UTC

R topics documented:

CovTools-package ......................................................... 2
CovDist ................................................................. 2
CovEst ................................................................. 3
CovEst.auto ............................................................ 5
**Description**

Covariance is of universal prevalence across various disciplines within statistics. The **CovTools** aims at providing a rich collection of geometric and inferential tools for convenient analysis of covariance structures. Following is a list of functions,

<table>
<thead>
<tr>
<th>name of a function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CovDist</td>
<td>compute pairwise distance of covariance matrices</td>
</tr>
<tr>
<td>CovEst</td>
<td>estimate covariance matrix</td>
</tr>
<tr>
<td>CovEst.auto</td>
<td>estimate covariance matrix with automatic parameter tuning</td>
</tr>
<tr>
<td>CovMean</td>
<td>compute mean covariance matrix</td>
</tr>
<tr>
<td>CovTest1</td>
<td>1-sample tests for covariance matrix</td>
</tr>
<tr>
<td>CovTest2</td>
<td>2-sample tests for covariance matrices</td>
</tr>
<tr>
<td>PreEst</td>
<td>estimate an inverse covariance matrix</td>
</tr>
<tr>
<td>PreEst.auto</td>
<td>estimate an inverse covariance matrix with automatic parameter tuning</td>
</tr>
</tbody>
</table>

**CovDist**

*Compute pairwise distance for symmetric positive definite matrices.*

**Description**

For a given 3-dimensional array where symmetric positive definite (SPD) matrices are stacked slice by slice, it computes pairwise distance using various popular measures. Some of measures are *metric* as they suffice 3 conditions in mathematical context; nonnegative definiteness, symmetry, and triangle inequalities. Other non-metric measures represent *dissimilarities* between two SPD objects.

**Usage**

```r
```
Arguments

- **A**: a (p-by-\(\cdot\)-by-N) 3d array of N SPD matrices.
- **method**: the type of distance measures to be used; "AIRM" for Affine Invariant Riemannian Metric, "Bhattacharyya" for Bhattacharyya distance based on normal model, "Cholesky" for Cholesky difference in Frobenius norm, "Euclidean" for naive Frobenius norm as distance, "Hellinger" for Hellinger distance based on normal model, "JBLD" for Jensen-Bregman Log Determinant Distance, "KLDM" for symmetrized Kullback-Leibler Distance Measure, "LERM" for Log Euclidean Riemannian Metric, "Procrustes_SS" for Procrustes Size and Shape measure, "Procrustes_Full" for Procrustes analysis with scale, "PowerEuclidean" for weighted eigenvalues by some exponent, and "RootEuclidean" for matrix square root.
- **power**: a non-zero number for PowerEuclidean distance.

Value

an (N-by-N) symmetric matrix of pairwise distances.

References


Examples

```r
## generate 100 SPD matrices of size (5-by-5)
samples = samplecvs(100,5)

## get pairwise distance for "AIRM"
distAIRM = CovDist(samples, method="AIRM")

## dimension reduction using MDS
ss = cmdscale(distAIRM)
plot(ss[,1],ss[,2],main="2d projection")
```
Description

It compiles several popular methods of inferring covariance structure from observed data. Several principles or assumed structures and corresponding estimation techniques are included, such as sparse covariance estimation ([Bickel08], [Cai11]), soft thresholding ([Donoho95]), near positive definiteness ([Qi06]), and so on.

Usage

\[
\text{CovEst}(X, \text{method} = \text{c("Bickel08", "Cai11", "Donoho95", "Fan13", "Qi06")),}\n\text{param = 1})
\]

Arguments

- **X**: an \((n\times p)\) matrix where each row is an observation from the first dataset.
- **method**: a name of estimation method.
- **param**: a parameter value \(>= 0\).

Value

a named list containing:

- **S**: a \((p\times p)\) covariance matrix estimate.

References


Examples

```r
# generate data from multivariate normal with Identity covariance.
data = mvtnorm::rmvnorm(100, sigma=diag(10))

# run estimation
out1 = CovEst(data, method="Bickel08")
out2 = CovEst(data, method="Cai11")
out3 = CovEst(data, method="Donoho95")
out4 = CovEst(data, method="Fan13")
out5 = CovEst(data, method="Qi06")
```
CovEst.auto

Estimating Covariance Matrix with automatic tuning

Description

In this variant of CovEst, we implemented automatic parameter tuning scheme applying 2-fold cross validation repeatedly and choosing the minimal one with the least discrepancy.

Usage

CovEst.auto(X, method = c("Bickel08", "Cai11", "Donoho95", "Fan13", "Qi06"),
          opt = list(nCV = 10, nsearch = 10), parallel = FALSE)

Arguments

X an (n-by-p) matrix where each row is an observation from the first dataset.
method a name of estimation method.
opt a list of options containing following fields:
    nCV the number for repetitions for 2-fold random cross validation.
    nsearch the number of trials on range of regularization parameters.
    parallel a logical; TRUE to use half of available cores, FALSE to do every computation sequentially.

Value

a named list containing:

S a (p-by-p) covariance matrix estimate.
CV a dataframe containing vector of tested threshold values(thr) and corresponding cross validation scores(CVscore).
References


See Also

CovEst

Examples

```r
## generate data from multivariate normal with Identity covariance.
data = mvtnorm::rmvnorm(100, sigma=diag(5))

## run automatic estimation
sopt = list(nCV=2,nsearch=3) # common option
out1 = CovEst.auto(data, method="Bickel08", opt=sopt)
out2 = CovEst.auto(data, method="Cai11", opt=sopt)

## Visualize
par(mfrow=c(1,3))
image(pracma::flipud(diag(5)),main="Original Covariance")
image(pracma::flipud(out1$S), main="Bickel08")
image(pracma::flipud(out2$S), main="Cai11")
```

---

CogMean

Estimate mean covariance matrix

Description

For a given 3-dimensional array where symmetric positive definite (SPD) matrices are stacked slice
by slice, it estimates Frechet mean on an open cone of SPD matrices under corresponding metric/distance measure.

Usage

```r
CovMean(A, method = c("AIRM", "Cholesky", "Euclidean", "LERM", 
"Procrustes.SS", "Procrustes.Full", "PowerEuclidean", "RootEuclidean"), 
power = 1)
```
Arguments

A  
  a (p-by-p-by-N) 3d array of N SPD matrices.

method  
  the type of distance measures to be used; "AIRM" for Affine Invariant Riemannian Metric, "Cholesky" for Cholesky difference in Frobenius norm, "Euclidean" for naive Frobenius norm as distance, "LERM" for Log Euclidean Riemannian Metric, "Procrustes.SS" for Procrustes Size and Shape measure, "Procrustes.Full" for Procrustes analysis with scale, "PowerEuclidean" for weighted eigenvalues by some exponent, and "RootEuclidean" for matrix square root.

power  
  a non-zero number for PowerEuclidean distance.

Value

a (p-by-p) mean covariance matrix estimated.

References


Examples

## generate 100 sample covariances of size (5-by-5).
samples = samplecovs(100,5)

## Compute mean of first 50 sample covariances from data under Normal(0,Identity).
mLERM = CovMean(samples[,1:50],method="LERM")
mAIRM = CovMean(samples[,1:50],method="AIRM")

---

CovTest1  
One-Sample Tests for Covariance Matrices

Description

Given data, CovTest1 performs 1-sample test for Covariance where the null hypothesis is

\[ H_0 : \Sigma_n = \Sigma_0 \]

where \( \Sigma_n \) is the covariance of data model and \( \Sigma_0 \) is a hypothesized covariance.

Usage

CovTest1(data, Sigma0 = diag(ncol(data)), alpha = 0.05,  
method = c("Cai13"))
Arguments

data an (n-by-p) data matrix where each row is an observation.
Sigma0 a (p-by-p) given covariance matrix.
alpha level of significance.
method a name of test.

Value

a named list containing

statistic a test statistic value.
threshold rejection criterion to be compared against test statistic.
reject a logical; TRUE to reject null hypothesis, FALSE otherwise.

References


Examples

## generate data from multivariate normal with trivial covariance.
data = mvtnorm::rmvnorm(100, sigma=diag(5))

## run test
covtest1(data, Sigma0=diag(5))

Description

Given two sets of data, CovTest2 performs 2-sample test for Covariance where the null hypothesis is

\[ H_0 : \Sigma_1 = \Sigma_2 \]

where \( \Sigma_1 \) and \( \Sigma_2 \) represent true (unknown) covariance for each dataset.

Usage

CovTest2(X, Y, alpha = 0.05, method = c(“Cai13”))
PreEst

Arguments

- **X**: an \((m \times p)\) matrix where each row is an observation from the first dataset.
- **Y**: an \((n \times p)\) matrix where each row is an observation from the second dataset.
- **alpha**: level of significance.
- **method**: a name of test.

Value

- a named list containing
  - **statistic**: a test statistic value.
  - **threshold**: rejection criterion to be compared against test statistic.
  - **reject**: a logical; TRUE to reject null hypothesis, FALSE otherwise.

References


Examples

```r
## generate 2 datasets from multivariate normal with identical covariance.
data1 = mvtnorm::rmvnorm(100, sigma=diag(5))
data2 = mvtnorm::rmvnorm(200, sigma=diag(5))

## run test
CovTest2(data1, data2)
```

PreEst  

*Estimating Precision Matrix*

Description

This code compiles several estimation methods for precision matrix, which is an inverse of covariance matrix, including penalized likelihood method with L1 penalty(Yuan07, Banerjee06) or Bayesian approaches incorporating banded structure assumptions(Banerjee14, Lee17).

Usage

```r
PreEst(X, method = c("Banerjee06", "Banerjee14", "Lee17", "Yuan07"),
    opt = list(Banerjee06.confidence = 0.95, Banerjee14.upperK =
        floor(ncol(X)/2), Banerjee14.delta = 10, Banerjee14.logpi = function(k) {
            -k^4 }, Banerjee14.loss = c("Stein", "Squared"), Lee17.upperK =
        floor(ncol(X)/2), Lee17.logpi = function(k) { -k^4 }, Yuan07.lambda = 1))
```
Arguments

X

an (n-by-p) matrix where each row is an observation from the first dataset.
method

a name of estimation method.
opt

a list containing following parameters,
Banerjee06.confidence level of confidence in (0, 1).
Banerjee14.upperK upper bound for bandwidth.
Banerjee14.delta a number larger than 2.
Banerjee14.logpi log of prior distribution for bandwidth k.
Banerjee14.loss loss type; either "Stein" or "Squared" type.
Lee17.upperK upper bound for bandwidth.
Lee17.logpi log of prior distribution for bandwidth k.
Yuan07.lambda a regularization parameter.

Value

a (p-by-p) estimated precision matrix.

References


Examples

## generate data from multivariate normal with Identity precision.
data = mvtnorm::rmvnorm(100, sigma=diag(10))

## run estimation
out1 = PreEst(data, method="Banerjee06")
out2 = PreEst(data, method="Banerjee14")
out3 = PreEst(data, method="Lee17")
out4 = PreEst(data, method="Yuan07")

## Visualize
par(mfrow=c(2,2))
image(pracma::flipud(out1), main="Banerjee06")
image(pracma::flipud(out2), main="Banerjee14")
image(pracma::flipud(out3), main="Lee17")
image(pracma::flipud(out4), main="Yuan07")
PreEst.auto

Estimating Precision Matrix with automatic parameter tuning

Description

This code compiles several estimation methods for precision matrix, which is an inverse of covariance matrix, including penalized likelihood method with L1 penalty (Yuan07, Banerjee06) or Bayesian approaches incorporating banded structure assumptions (Banerjee14, Lee17). At this moment, the only automation is necessary for the method "Yuan07" for multiple trials with different λ values.

Usage

PreEst.auto(X, method = c("Banerjee06", "Banerjee14", "Lee17", "Yuan07"),
            opt = list(Banerjee06.confidence = 0.95, Banerjee14.upperK =
                        floor(ncol(X)/2), Banerjee14.delta = 10, Banerjee14.logpi = function(k) {
                        -k^4 }, Banerjee14.loss = c("Stein", "Squared"), Lee17.upperK =
                        floor(ncol(X)/2), Lee17.logpi = function(k) { -k^4 }, Yuan07.lambdagrid =
                        seq(from = 0.01, to = 2, length.out = 10)), parallel = FALSE)

Arguments

X: an (n-by-p) matrix where each row is an observation from the first dataset.

method: a name of estimation method.

opt: a list containing following parameters,

- **Banerjee06.confidence**: level of confidence in (0, 1).
- **Banerjee14.upperK**: upper bound for bandwidth.
- **Banerjee14.delta**: a number larger than 2.
- **Banerjee14.logpi**: log of prior distribution for bandwidth k.
- **Banerjee14.loss**: loss type; either "Stein" or "Squared" type.
- **Lee17.upperK**: upper bound for bandwidth.
- **Lee17.logpi**: log of prior distribution for bandwidth k.
- **Yuan07.lambdagrid**: a vector of regularization parameters.

parallel: a logical; TRUE for using half the cores available, FALSE otherwise.

Value

a named list containing:

- **C**: a (p-by-p) estimated precision matrix.
- **BIC**: a vector of BIC scores for the method Yuan07, only.
References


Examples

```r
## generate data from multivariate normal with Identity precision.
data = mvtnorm::rmvnorm(100, sigma=diag(10))

## run estimation
out1 = PreEst.auto(data, method="Banerjee06")
out2 = PreEst.auto(data, method="Lee17")

## Visualize
par(mfrow=c(1,3))
image(pracma::flipud(diag(10)), main="Original Covariance")
image(pracma::flipud(out1$C), main="Banerjee06")
image(pracma::flipud(out2$C), main="Lee17")
```

---

**samplecovs**

Generate Sample Covariances of 2 groups

**Description**

For visualization purpose, samplecovs generates a 3d array of stacked sample covariances where - in 3rd dimension, the first half are sample covariances of samples generated independently from normal distribution with identity covariance, where the latter half consists of samples covariances from dense random population covariance.

**Usage**

`samplecovs(ncopy, size)`

**Arguments**

- `ncopy` : the total number of sample covariances to be generated.
- `size` : dimension p
Value

a (p-by-p-by-ncopy) array of strictly positive definite sample covariances.

Examples

```r
## generate total of 20 samples covariances of size 5-by-5.
samples <- samplecovs(20, 5)
```
Index

CovDist, 2, 2
CovEst, 2, 3, 5, 6
CovEst.auto, 2, 5
CovMean, 2, 6
CovTest1, 2, 7
CovTest2, 2, 8
CovTools-package, 2

PreEst, 2, 9
PreEst.auto, 2, 11

samplecoves, 12