Package 'EFM'

May 7, 2026

Version 1.3.0

Title Elliptical Factor Models

Description The elliptical factor model, as an extension of the traditional factor model, effectively overcomes the limitations of the traditional model when dealing with heavy-tailed characteristic data. This package implements sparse principal component methods (SPC) and bi-sparse online principal component estimation (SPOC) for parameter estimation. Includes functionality for calculating mean squared error, relative error, and loading matrix sparsity.The philosophy of the package is described in Guo G. (2023) <doi:10.1007/s00180-022-01270-z>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports MASS, stats, SOPC, matrixcalc

Suggests testthat (>= 3.0.0), ggplot2, pracma, psych, sn

Depends R (>= 3.5.0)

NeedsCompilation no

Language en-US

Maintainer Guangbao Guo <ggb11111111@163.com>

Author Guangbao Guo [aut, cre],
Yuanyuan Zhou [aut]

Repository CRAN

Date/Publication 2025-11-14 00:00:02 UTC

Contents

EFM *The EFM function is to generate Elliptical Factor Models data.*

Description

The function supports various distribution types for generating the data, including: Elliptical-Normal Distribution, Elliptical-t Distribution.

Usage

```
EFM(n, p, m, nu, distribution_type)
```

Arguments

n	Sample size.
p	Sample dimensionality.
m	Number of factors.
nu	A numerical parameter used exclusively in the "Elliptical-t" distribution, representing the degrees of freedom.
distribution_type	The type of distribution ("Elliptical-Normal Distribution" or "Elliptical-t Distribution").

Value

A list containing:

data	A matrix of generated data (n x p).
A	A matrix representing the factor loadings (p x m).
D	A diagonal matrix representing the unique variances (p x p).

Examples

```
library(MASS)
library(pracma)

n <- 2000
p <- 10
m <- 5
nu <- 5
distribution_type <- "Elliptical-Normal Distribution"
X <- EFM(n, p, m, nu, distribution_type)
```

`elliptical_errors` *Calculate errors for elliptical distributions*

Description

This function calculates errors for elliptical distributions, including Elliptical-Normal Distribution and Elliptical-t Distribution.

Usage

```
elliptical_errors(data, distribution = "normal", df = NULL)
```

Arguments

`data`	Matrix of data following an elliptical distribution.
`distribution`	Type of elliptical distribution ("normal" or "t").
`df`	Degrees of freedom for Elliptical-t Distribution (required if distribution is "t").

Details

Calculate errors for elliptical distributions

Value

A list containing error metrics for the specified elliptical distribution.

Examples

```
set.seed(123)
n <- 100; p <- 5
data_normal <- MASS::mvrnorm(n, rep(0, p), diag(p))
errors_normal <- elliptical_errors(data_normal, distribution = "normal")

data_t <- matrix(stats::rt(n * p, df = 5), nrow = n, ncol = p)
errors_t <- elliptical_errors(data_t, distribution = "t", df = 5)
print(errors_normal)
print(errors_t)
```

FanPC.EFM *Apply the FanPC method to the Elliptical Factor Model*

Description

This function performs Factor Analysis via Principal Component (FanPC) on a given data set. It calculates the estimated factor loading matrix (AF), specific variance matrix (DF), and the mean squared errors.

Usage

```
FanPC.EFM(data, m, A, D, p)
```

Arguments

data	A matrix of input data.
m	The number of principal components.
A	The true factor loadings matrix.
D	The true uniquenesses matrix.
p	The number of variables.

Value

A list containing:

AF	Estimated factor loadings.
DF	Estimated uniquenesses.
MSESigmaA	Mean squared error for factor loadings.
MSESigmaD	Mean squared error for uniquenesses.
LSigmaA	Loss metric for factor loadings.
LSigmaD	Loss metric for uniquenesses.

Examples

```
library(matrixcalc)
library(MASS)

n <- 100
p <- 10
m <- 5
mu <- t(matrix(rep(runif(p, 0, 1000), n), p, n))
mu0 <- as.matrix(runif(m, 0))
sigma0 <- diag(runif(m, 1))
F_matrix <- matrix(mvrnorm(n, mu0, sigma0), nrow = n)
A <- matrix(runif(p * m, -1, 1), nrow = p)
r <- rnorm(n * p, 0, 1)
```

```
epsilon <- matrix(r, nrow = n)
D <- diag(as.vector(apply(epsilon, 2, function(x) sum(x^2))))
data <- mu + F_matrix %*% t(A) + epsilon
results <- FanPC.EFM(data, m, A, D, p)
print(results)
```

`incremental_perturb_pca` *Incremental Perturbation PCA*

Description

Batch-updated PCA with residual-projection perturbation.

Usage

```
incremental_perturb_pca(batch_data, m, eta = 0.1)
```

Arguments

`batch_data`	List of matrices.
`m`	Components.
`eta`	Step size.

Value

List with Ap, Dp, V.

Examples

```
## Not run:
set.seed(789)
N <- 50; m.true <- 2; n_batch <- 5
batches <- lapply(1:n_batch, function(i) matrix(rnorm(60 * N), 60, N))
inc <- EFM::incremental_perturb_pca(batches, m = m.true)
print(round(inc$Ap[1:5, 1:2], 3))

## End(Not run)
```

PC1.EFM *Apply the PC method to the Elliptical Factor Model*

Description

This function performs Principal Component Analysis (PCA) on a given data set to reduce dimensionality. It calculates the estimated values for the loadings, specific variances, and the covariance matrix.

Usage

```
PC1.EFM(data, m, A, D)
```

Arguments

data	The total data set to be analyzed.
m	The number of principal components to retain in the analysis.
A	The true factor loadings matrix.
D	The true uniquenesses matrix.

Value

A list containing:

A1	Estimated factor loadings.
D1	Estimated uniquenesses.
MSESigmaA	Mean squared error for factor loadings.
MSESigmaD	Mean squared error for uniquenesses.
LSigmaA	Loss metric for factor loadings.
LSigmaD	Loss metric for uniquenesses.

Examples

```
library(matrixcalc)
library(MASS)

n <- 100
p <- 10
m <- 5
mu <- t(matrix(rep(runif(p, 0, 1000), n), p, n))
mu0 <- as.matrix(runif(m, 0))
sigma0 <- diag(runif(m, 1))
F_matrix <- matrix(mvrnorm(n, mu0, sigma0), nrow = n)
A <- matrix(runif(p * m, -1, 1), nrow = p)
r <- rnorm(n * p, 0, 1)
epsilon <- matrix(r, nrow = n)
```

```
D <- diag(as.vector(apply(epsilon, 2, function(x) sum(x^2))))
data <- mu + F_matrix %*% t(A) + epsilon
results <- PC1.EFM(data, m, A, D)
print(results)
```

PPC *PPC: Principal Projection Components*

Description

Principal component projection method that extracts factor loadings and specific variances through a projection matrix.

Usage

```
PPC(data, m)
```

Arguments

data	T x N data matrix.
m	Number of components.

Value

List with Apro (loadings), Dpro (specific variances), Sigmahatpro.

Examples

```
## Not run:
set.seed(123)
N <- 100; T <- 150; m.true <- 3
dat <- matrix(rnorm(T * N), T, N)
ppc.out <- EFM::PPC(dat, m = m.true)
print(round(ppc.out$Apro[1:5, 1:3], 3))

## End(Not run)
```

PPC1.EFM *Apply the PPC method to the Elliptical Factor Model*

Description

This function computes Perturbation Principal Component Analysis (PPC) for the provided input data, estimating factor loadings and uniquenesses. It calculates mean squared errors and loss metrics for the estimated values compared to true values.

Usage

```
PPC1.EFM(data, m, A, D, p)
```

Arguments

data	A matrix of input data.
m	The number of principal components.
A	The true factor loadings matrix.
D	The true uniquenesses matrix.
p	The number of variables.

Value

A list containing:

Ap	Estimated factor loadings.
Dp	Estimated uniquenesses.
MSESigmaA	Mean squared error for factor loadings.
MSESigmaD	Mean squared error for uniquenesses.
LSigmaA	Loss metric for factor loadings.
LSigmaD	Loss metric for uniquenesses.

Examples

```
library(matrixcalc)
library(MASS)

n <- 100
p <- 10
m <- 5
mu <- t(matrix(rep(runif(p, 0, 1000), n), p, n))
mu0 <- as.matrix(runif(m, 0))
sigma0 <- diag(runif(m, 1))
F_matrix <- matrix(mvrnorm(n, mu0, sigma0), nrow = n)
A <- matrix(runif(p * m, -1, 1), nrow = p)
r <- rnorm(n * p, 0, 1)
```

```
epsilon <- matrix(r, nrow = n)
D <- diag(as.vector(apply(epsilon, 2, function(x) sum(x^2))))
data <- mu + F_matrix %*% t(A) + epsilon
results <- PPC1.EFM(data, m, A, D, p)
print(results)
```

`robust_perturb_pca` *Random Perturbation Robust PCA*

Description

Robust PCA by adding random symmetric perturbation to covariance matrix.

Usage

```
robust_perturb_pca(data, m, epsilon = 0.001, sigma = 0.01)
```

Arguments

`data`	T x N matrix.
`m`	Components.
`epsilon`	Perturbation scale.
`sigma`	Noise sd.

Value

List with Ap, Dp, lambda.

Examples

```
## Not run:
set.seed(456)
N <- 80; T <- 100; m.true <- 2
X <- matrix(rnorm(T * N), T, N)
rpca.out <- EFM::robust_perturb_pca(X, m = m.true)
print(round(rpca.out$Ap[1:5, 1:2], 3))

## End(Not run)
```

SOPC.EFM *SOPC Estimation Function for Elliptical Factor Model*

Description

This function processes Elliptical Factor Model (EFM) data using the Sparse Online Principal Component (SOPC) method.

Usage

```
SOPC.EFM(data, m, p, A, D)
```

Arguments

data	A numeric matrix containing the data used in the SOPC analysis.
m	An integer specifying the number of subsets or common factors.
p	An integer specifying the number of variables in the data.
A	A numeric matrix representing the true factor loadings.
D	A numeric matrix representing the true uniquenesses.

Value

A list containing the following metrics:

Aso	Estimated factor loadings matrix.
Dso	Estimated uniquenesses matrix.
MSEA	Mean squared error of the estimated factor loadings (Aso) compared to the true loadings (A).
MSED	Mean squared error of the estimated uniquenesses (Dso) compared to the true uniquenesses (D).
LSA	Loss metric for the estimated factor loadings (Aso), indicating the relative error compared to the true loadings (A).
LSD	Loss metric for the estimated uniquenesses (Dso), indicating the relative error compared to the true uniquenesses (D).
tauA	Proportion of zero factor loadings in the estimated loadings matrix (Aso), representing the sparsity.

Examples

```
library(matrixcalc)
library(MASS)

n <- 100
p <- 10
m <- 5
```

```
mu <- t(matrix(rep(runif(p, 0, 1000), n), p, n))
mu0 <- as.matrix(runif(m, 0))
sigma0 <- diag(runif(m, 1))
F_matrix <- matrix(mvrnorm(n, mu0, sigma0), nrow = n)
A <- matrix(runif(p * m, -1, 1), nrow = p)
r <- rnorm(n * p, 0, 1)
epsilon <- matrix(r, nrow = n)
D <- diag(as.vector(apply(epsilon, 2, function(x) sum(x^2))))
data <- mu + F_matrix %*% t(A) + epsilon
results <- SOPC.EFM(data, m, p, A, D)
print(results)
```

Index