Package ‘TFM’

February 26, 2025

Type Package

Title Sparse Online Principal Component for Truncated Factor Model

Version 0.2.0

Description The Truncated Factor Model is a statistical model designed to handle specific data structures in data analysis. This R package focuses on the Sparse Online Principal Component Estimation method, which is used to calculate data such as the loading matrix and specific variance matrix for truncated data, thereby better explaining the relationship between common factors and original variables. Additionally, the R package also provides other equations for comparison with the Sparse Online Principal Component Estimation method.The philosophy of the package is described in thesis. (2023) <doi:10.1007/s00180-022-01270-z>.

License MIT + file LICENSE

Suggests rmarkdown, psych

Depends R (>= 3.0)

RoxygenNote 7.3.2

Encoding UTF-8

Language en-US

Author Beibei Wu [aut],
Guangbao Guo [aut, cre]

NeedsCompilation yes

Maintainer Guangbao Guo <ggb11111111@163.com>

Imports relliptical, SOPC, MASS, mvtnorm, matrixcalc

Repository CRAN

Date/Publication 2025-02-26 00:50:02 UTC

Contents

FanPC_TFM *Apply the FanPC method to the Truncated 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_TFM(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

```
## Not run:
library(SOPC)
library(relliptical)
library(MASS)
n=1000
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(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
trnor <- relliptical(n*p,0,1)
epsilon=matrix(trnor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
results <- FanPC_TFM(data, m, A, D, p)
print(results)

## End(Not run)
```

`GulPC_TFM` *Apply the GulPC method to the Truncated factor model*

Description

This function performs General Unilateral Loading Principal Component (GulPC) analysis on a given data set. It calculates the estimated values for the first layer and second layer loadings, specific variances, and the mean squared errors.

Usage

```
GulPC_TFM(data, m, A, D)
```

Arguments

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

Value

A list containing:

`AU1`	The first layer loading matrix.
`AU2`	The second layer loading matrix.

DU3	The estimated specific variance matrix.
MSESigmaD	Mean squared error for uniquenesses.
LSigmaD	Loss metric for uniquenesses.

Examples

```
## Not run:
library(SOPC)
library(relliptical)
library(MASS)
n=1000
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(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
trnor <- rellipitical(n*p,0,1)
epsilon=matrix(trnor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
results <- GulPC_TFM(data, m, A, D)
print(results)
## End(Not run)
```

IPC_TFM *Incremental Principal Component Analysis*

Description

This function performs Incremental Principal Component Analysis (IPC) on the provided data. It updates the estimated factor loadings and uniquenesses as new data points are processed, calculating mean squared errors and loss metrics for comparison with true values.

Usage

```
IPC_TFM(x, m, A, D, p)
```

Arguments

x	The data used in the IPC analysis.
m	The number of common factors.
A	The true factor loadings matrix.
D	The true uniquenesses matrix.
p	The number of variables.

Value

A list of metrics including:

`Ai`	Estimated factor loadings updated during the IPC analysis, a matrix of estimated factor loadings.
`Di`	Estimated uniquenesses updated during the IPC analysis, a vector of estimated uniquenesses corresponding to each variable.
`MSESigmaA`	Mean squared error of the estimated factor loadings (Ai) compared to the true loadings (A).
`MSESigmaD`	Mean squared error of the estimated uniquenesses (Di) compared to the true uniquenesses (D).
`LSigmaA`	Loss metric for the estimated factor loadings (Ai), indicating the relative error compared to the true loadings (A).
`LSigmaD`	Loss metric for the estimated uniquenesses (Di), indicating the relative error compared to the true uniquenesses (D).

Examples

```
library(MASS)
library(relliptical)
library(SOPC)

IPC_MSESigmaA = c()
IPC_MSESigmaD = c()
IPC_LSigmaA = c()
IPC_LSigmaD = c()

p = 10
m = 5

n = 2000
mu = t(matrix(rep(runif(p, 0, 1000), n), p, n))
mu0 = as.matrix(runif(m, 0))
sigma0 = diag(runif(m, 1))
F = matrix(mvrnorm(n, mu0, sigma0), nrow = n)
A = matrix(runif(p * m, -1, 1), nrow = p)

lower = c(rep(-0.5, p - 3), -5, -5, -Inf)
upper = c(rep(0.5, p - 3), 5, 5, Inf)
Sigma = as.matrix(diag(rep(runif(p, 0, 1))))
mut = runif(p, 0, 10)
trnor = rtelliptical(n, mut, Sigma, lower, upper, dist = "Normal")
epsilon = matrix(trnor, nrow = n)

D = Sigma
data = mu + F %*% t(A) + epsilon

Z = data.frame(IPC_TFM(data, m = m, A = A, D = D, p = p))[c(3, 4, 5, 6),]
IPC_MSESigmaA = Z[1]
IPC_MSESigmaD = Z[2]
```

```
IPC_LSigmaA = Z[3]
IPC_LSigmaD = Z[4]

data_M = data.frame(n = n, MSEA = IPC_MSESigmaA, MSED = IPC_MSESigmaD,
 LSA = IPC_LSigmaA, LSD = IPC_LSigmaD)
print(data_M)
```

OPC_TFM *Apply the OPC method to the Truncated factor model*

Description

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

Usage

```
OPC_TFM(data, m = 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:

`Ao`	Estimated factor loadings.
`Do`	Estimated uniquenesses.
`MSEA`	Mean squared error for factor loadings.
`MSED`	Mean squared error for uniquenesses.
`tau`	The sparsity.

Examples

```
## Not run:
library(SOPC)
library(relliptical)
library(MASS)
n=1000
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(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
trnor <- relliptical(n*p,0,1)
epsilon=matrix(trnor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
results <- OPC_TFM(data, m, A, D, p)
print(results)
## End(Not run)
```

`PC1_TFM` *Apply the PC method to the Truncated 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_TFM(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

```
## Not run:
library(SOPC)
library(relliptical)
library(MASS)
n=1000
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(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
trnor <- relliptical(n*p,0,1)
epsilon=matrix(trnor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
results <- PC1_TFM(data, m, A, D)
print(results)
## End(Not run)
```

PC2_TFM *Apply the PC method to the Truncated 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

```
PC2_TFM(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:

`A2`	Estimated factor loadings.
`D2`	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

```
## Not run:
library(SOPC)
library(relliptical)
library(MASS)
n=1000
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(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
results <- PC2_TFM(data, m, A, D)
print(results)
## End(Not run)
```

`PPC1_TFM` *Projected Principal Component Analysis*

Description

This function computes Projected 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_TFM(x, m, A, D, p)
```

Arguments

x	A matrix of input data.
m	The number of principal components to extract (integer).
A	The true factor loadings matrix (matrix).
D	The true uniquenesses matrix (matrix).
p	The number of variables (integer).

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(MASS)
library(relliptical)
library(SOPC)

PPC_MSESigmaA <- c()
PPC_MSESigmaD <- c()
PPC_LSigmaA <- c()
PPC_LSigmaD <- c()

p <- 10
m <- 5
n <- 2000

mu <- t(matrix(rep(runif(p, 0, 1000), n), p, n))
mu0 <- as.matrix(runif(m, 0))
sigma0 <- diag(runif(m, 1))
F <- matrix(mvrnorm(n, mu0, sigma0), nrow = n)
A <- matrix(runif(p * m, -1, 1), nrow = p)

lower <- c(rep(-0.5, p - 3), -5, -5, -Inf)
upper <- c(rep(0.5, p - 3), 5, 5, Inf)
Sigma <- diag(runif(p, 0, 1))
mut <- runif(p, 0, 10)

trnor <- rtelliptical(n, mut, Sigma, lower, upper, dist = "Normal")
epsilon <- matrix(trnor, nrow = n)
D <- Sigma
```

```
data <- mu + F %*% t(A) + epsilon

result <- PPC1_TFM(data, m, A, D, p)

data_G <- data.frame(n = n,
                     MSEA = result$MSESigmaA,
                     MSED = result$MSESigmaD,
                     LSA = result$LSigmaA,
                     LSD = result$LSigmaD)

print(data_G)
```

PPC2_TFM — *Apply the PPC method to the Truncated factor model*

Description

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

Usage

```
PPC2_TFM(data, m, A, D)
```

Arguments

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

Value

A list containing:

`Ap2`	Estimated factor loadings.
`Dp2`	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

```
## Not run:
library(SOPC)
library(relliptical)
library(MASS)
n=1000
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(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
trnor <- relliptical(n*p,0,1)
epsilon=matrix(trnor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
results <- PPC2_TFM(data, m, A, D)
print(results)

## End(Not run)
```

SAPC_TFM *Stochastic Approximation Principal Component Analysis*

Description

This function calculates several metrics for the SAPC method, including the estimated factor loadings and uniquenesses, and various error metrics comparing the estimated matrices with the true matrices.

Usage

```
SAPC_TFM(x, m, A, D, p)
```

Arguments

x	The data used in the SAPC analysis.
m	The number of common factors.
A	The true factor loadings matrix.
D	The true uniquenesses matrix.
p	The number of variables.

Value

A list of metrics including:

`Asa`	Estimated factor loadings matrix obtained from the SAPC analysis.
`Dsa`	Estimated uniquenesses vector obtained from the SAPC analysis.
`MSESigmaA`	Mean squared error of the estimated factor loadings (Asa) compared to the true loadings (A).
`MSESigmaD`	Mean squared error of the estimated uniquenesses (Dsa) compared to the true uniquenesses (D).
`LSigmaA`	Loss metric for the estimated factor loadings (Asa), indicating the relative error compared to the true loadings (A).
`LSigmaD`	Loss metric for the estimated uniquenesses (Dsa), indicating the relative error compared to the true uniquenesses (D).

Examples

```
library(MASS)
library(relliptical)
library(SOPC)

SAPC_MSESigmaA <- c()
SAPC_MSESigmaD <- c()
SAPC_LSigmaA <- c()
SAPC_LSigmaD <- c()

p <- 10
m <- 5
n <- 2000

mu <- t(matrix(rep(runif(p, 0, 1000), n), p, n))
mu0 <- as.matrix(runif(m, 0))
sigma0 <- diag(runif(m, 1))
F <- matrix(mvrnorm(n, mu0, sigma0), nrow = n)
A <- matrix(runif(p * m, -1, 1), nrow = p)

lower <- c(rep(-0.5, p - 3), -5, -5, -Inf)
upper <- c(rep(0.5, p - 3), 5, 5, Inf)
Sigma <- diag(runif(p, 0, 1))
mut <- runif(p, 0, 10)
trnor <- rtelliptical(n, mut, Sigma, lower, upper, dist = "Normal")
epsilon <- matrix(trnor, nrow = n)
D <- Sigma

data <- mu + F %*% t(A) + epsilon

result <- SAPC_TFM(data, m = m, A = A, D = D, p = p)

SAPC_MSESigmaA <- result$MSESigmaA
SAPC_MSESigmaD <- result$MSESigmaD
SAPC_LSigmaA <- result$LSigmaA
```

```
SAPC_LSigmaD <- result$LSigmaD

data_K <- data.frame(
  n = n,
  MSEA = SAPC_MSESigmaA,
  MSED = SAPC_MSESigmaD,
  LSA = SAPC_LSigmaA,
  LSD = SAPC_LSigmaD
)

print(data_K)
```

SOPC_TFM *Sparse Online Principal Component Analysis*

Description

This function calculates various metrics for the Sparse Online Principal Component Analysis (SOPC) method. It estimates the factor loadings and uniquenesses while calculating mean squared errors and loss metrics for comparison with true values. Additionally, it computes the proportion of zero factor loadings in the estimated loadings matrix.

Usage

```
SOPC_TFM(data, m, p, gamma, eta, A, D)
```

Arguments

`data`	The data used in the SOPC analysis.
`m`	the number of common factors
`p`	the number of variables
`gamma`	Tuning parameter for the sparseness of the loadings matrix.
`eta`	Tuning parameter for the sparseness of the uniquenesses matrix.
`A`	The true A matrix.
`D`	The true D matrix.

Value

A list of metrics including:

`Aso`	Estimated factor loadings matrix obtained from the SOPC analysis.
`Dso`	Estimated uniquenesses vector obtained from the SOPC analysis.
`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), indicating the sparsity of the loadings.

Examples

```
library(MASS)
library(relliptical)
library(SOPC)

SOPC_MSEA <- c()
SOPC_MSED <- c()
SOPC_LSA <- c()
SOPC_LSD <- c()
SOPC_TAUA <- c()

p = 10; m = 5
n = 2000  # Set n to 2000
mu = t(matrix(rep(runif(p, 0, 1000), n), p, n))
mu0 = as.matrix(runif(m, 0))
sigma0 = diag(runif(m, 1))
F = matrix(mvrnorm(n, mu0, sigma0), nrow = n)
A = matrix(runif(p * m, -1, 1), nrow = p)

# Sampling from the Truncated Normal distribution
lower = c(rep(-0.5, p - 3), -5, -5, -Inf)
upper = c(rep(0.5, p - 3), 5, 5, Inf)
Sigma = as.matrix(diag(rep(runif(p, 0, 1))))
mut = runif(p, 0, 10)
trnor = rtelliptical(n, mut, Sigma, lower, upper, dist = "Normal")
epsilon = matrix(trnor, nrow = n)
D = Sigma

data = mu + F %*% t(A) + epsilon

Z = data.frame(SOPC_TFM(data, m = m, p = p, gamma = 0.1, eta = 0.8, A = A, D = D))
SOPC_MSEA = c(SOPC_MSEA, Z$MSEA)
SOPC_MSED = c(SOPC_MSED, Z$MSED)
SOPC_LSA = c(SOPC_LSA, Z$LSA)
SOPC_LSD = c(SOPC_LSD, Z$LSD)
SOPC_TAUA = c(SOPC_TAUA, Z$tauA)

# Ensure the data frame has the correct column structure, even with one value
data_F = data.frame(n = rep(n, length(SOPC_MSEA)), MSEA = SOPC_MSEA, MSED = SOPC_MSED,
 LSA = SOPC_LSA, LSD = SOPC_LSD, tauA = SOPC_TAUA)
data_F
```

SPC_TFM *Sparse Principal Component Analysis*

Description

This function performs Sparse Principal Component Analysis (SPC) on the input data. It estimates factor loadings and uniquenesses while calculating mean squared errors and loss metrics for comparison with true values. Additionally, it computes the proportion of zero factor loadings.

Usage

```
SPC_TFM(data, A, D, m, p)
```

Arguments

`data`	The data used in the SPC analysis.
`A`	The true factor loadings matrix.
`D`	The true uniquenesses matrix.
`m`	The number of common factors.
`p`	The number of variables.

Value

A list containing:

`As`	Estimated factor loadings, a matrix of estimated factor loadings from the SPC analysis.
`Ds`	Estimated uniquenesses, a vector of estimated uniquenesses corresponding to each variable.
`MSESigmaA`	Mean squared error of the estimated factor loadings (As) compared to the true loadings (A).
`MSESigmaD`	Mean squared error of the estimated uniquenesses (Ds) compared to the true uniquenesses (D).
`LSigmaA`	Loss metric for the estimated factor loadings (As), indicating the relative error compared to the true loadings (A).
`LSigmaD`	Loss metric for the estimated uniquenesses (Ds), indicating the relative error compared to the true uniquenesses (D).
`tau`	Proportion of zero factor loadings in the estimated loadings matrix (As).

Examples

```
library(MASS)
library(relliptical)
library(SOPC)

SPC_MSESigmaA <- c()
SPC_MSESigmaD <- c()
SPC_LSigmaA <- c()
SPC_LSigmaD <- c()
SPC_tau <- c()

p <- 10
m <- 5
n <- 2000

mu <- t(matrix(rep(runif(p, 0, 1000), n), p, n))
mu0 <- as.matrix(runif(m, 0))
sigma0 <- diag(runif(m, 1))
F <- matrix(mvrnorm(n, mu0, sigma0), nrow = n)
A <- matrix(runif(p * m, -1, 1), nrow = p)

lower <- c(rep(-0.5, p - 3), -5, -5, -Inf)
upper <- c(rep(0.5, p - 3), 5, 5, Inf)
Sigma <- diag(runif(p, 0, 1))
mut <- runif(p, 0, 10)

trnor <- rtelliptical(n, mut, Sigma, lower, upper, dist = "Normal")
epsilon <- matrix(trnor, nrow = n)
D <- Sigma

data <- mu + F %*% t(A) + epsilon

result <- SPC_TFM(data, A, D, m, p)

SPC_MSESigmaA <- c(SPC_MSESigmaA, result$MSESigmaA)
SPC_MSESigmaD <- c(SPC_MSESigmaD, result$MSESigmaD)
SPC_LSigmaA <- c(SPC_LSigmaA, result$LSigmaA)
SPC_LSigmaD <- c(SPC_LSigmaD, result$LSigmaD)
SPC_tau <- c(SPC_tau, result$tau)

data_G <- data.frame(n = n,
                     MSEA = SPC_MSESigmaA,
                     MSED = SPC_MSESigmaD,
                     LSA = SPC_LSigmaA,
                     LSD = SPC_LSigmaD,
                     tau = SPC_tau)

print(data_G)
```

`TFM` *The TFM function is to generate Truncated factor model data.*

Description

The TFM function generates truncated factor model data supporting various distribution types for related analyses using multiple methods.

Usage

```
TFM(n, mu, sigma, lower, upper, distribution_type)
```

Arguments

`n`	Total number of observations.
`mu`	The mean of the distribution.
`sigma`	The parameter of the distribution.
`lower`	The lower bound of the interval.
`upper`	The upper bound of the interval.
`distribution_type`	String specifying the distribution type to use.

Value

A list containing:

`X`	A matrix of generated truncated factor model data based on the specified distribution type. Each row corresponds to an observation, and each column corresponds to a variable.

Examples

```
library(relliptical)
set.seed(123)
mu <- c(0, 1)
n <- 100
sigma <- matrix(c(1, 0.70, 0.70, 3), 2, 2)
lower <- c(-2, -3)
upper <- c(3, 3)
distribution_type <- "truncated_normal"
X <- TFM(n, mu, sigma, lower, upper, distribution_type)
```

`ttest.TFM` *T-test for Truncated Factor Model*

Description

This function performs a simple t-test for each variable in the dataset of a truncated factor model and calculates the False Discovery Rate (FDR) and power.

Usage

```
ttest.TFM(X, p, alpha = 0.05)
```

Arguments

`X`	A matrix or data frame of simulated or observed data from a truncated factor model.
`p`	The number of variables (columns) in the dataset.
`alpha`	The significance level for the t-test.

Value

A list containing:

`FDR`	The False Discovery Rate calculated from the rejected hypotheses.
`Power`	The power of the test, representing the proportion of true positives among the non-zero hypotheses.
`pValues`	A numeric vector of p-values obtained from the t-tests for each variable.
`RejectedHypotheses`	A logical vector indicating which hypotheses were rejected based on the specified significance level.

Examples

```
# Load necessary libraries
library(MASS)
library(mvtnorm)

set.seed(100)
# Set parameters for the simulation
p <- 400  # Number of features
n <- 120  # Number of samples
K <- 5    # Number of latent factors
true_non_zero <- 100  # Assume 100 features have non-zero means

# Simulate factor loadings matrix B (p x K)
B <- matrix(rnorm(p * K), nrow = p, ncol = K)

# Simulate factor scores (n x K)
FX <- MASS::mvrnorm(n, rep(0, K), diag(K))

# Simulate noise U (n x p), assuming Student's t-distribution with 3 degrees of freedom
U <- mvtnorm::rmvt(n, df = 3, sigma = diag(p))

# Create the data matrix X based on the truncated factor model
# Non-zero means for the first 100 features
mu <- c(rep(1, true_non_zero), rep(0, p - true_non_zero))
X <- rep(1, n) %*% t(mu) + FX %*% t(B) + U  # The observed data

# Apply the t-test function on the data
```

```
results <- ttest.TFM(X, p, alpha = 0.05)

# Print the results
print(results)
```

Index