

Package ‘DLFM’

December 6, 2025

Type Package
Version 0.1.4
Title Distributed Laplace Factor Model
Description Distributed estimation method is based on a Laplace factor model to solve the estimates of load and specific variance. The philosophy of the package is described in Guangbao Guo. (2022). <[doi:10.1007/s00180-022-01270-z](https://doi.org/10.1007/s00180-022-01270-z)>.
License MIT + file LICENSE
Encoding UTF-8
RoxygenNote 7.3.2
Imports stats, MASS, LaplacesDemon, matrixcalc, rellyptical, LFM
NeedsCompilation no
Language en-US
Author Guangbao Guo [aut, cre],
Siqu Liu [aut]
Depends R (>= 3.5.0)
BuildManual yes
Suggests testthat (>= 3.0.0)
Config/testthat/edition 3
Maintainer Guangbao Guo <ggb11111111@163.com>
Repository CRAN
Date/Publication 2025-12-06 12:20:06 UTC

Contents

Dfactor.tests	2
DGulPC	3
DIPC	4
DPC	5
DPPC	5
DSAPC	6

factor.tests	7
FanPC	8
Ftest	9
GulPC	10
LFM	11
PC	12
PPC	12
SAPC	13

Index	14
--------------	-----------

Dfactor.tests	<i>Distributed Factor Model Testing with Wald, GRS, PY tests and FDR control</i>
---------------	--

Description

Performs comprehensive factor model testing in distributed environment across multiple nodes, including joint tests (Wald, GRS, PY), individual asset t-tests, and False Discovery Rate control.

Usage

```
Dfactor.tests(ret, fac, n1, K, q.fdr = 0.05)
```

Arguments

ret	A $T \times N$ matrix representing the excess returns of N assets at T time points.
fac	A $T \times K$ matrix representing the returns of K factors at T time points.
n1	The number of assets allocated to each node
K	The number of nodes
q.fdr	The significance level for FDR (False Discovery Rate) testing, defaulting to 5%.

Value

A list containing the following components:

alpha_list	List of alpha vectors from each node
tstat_list	List of t-statistics from each node
pval_list	List of p-values from each node
Wald_list	List of Wald test statistics from each node
p_Wald_list	List of p-values for Wald tests from each node
GRS_list	List of GRS test statistics from each node
p_GRS_list	List of p-values for GRS tests from each node
PY_list	List of Pesaran and Yamagata test statistics from each node
p_PY_list	List of p-values for PY tests from each node

```

reject_fdr_list
    List of logical vectors indicating significant assets after FDR correction from
    each node
power_proxy_list
    List of number of significant assets after FDR correction from each node
combined_alpha Combined alpha vector from all nodes
combined_pval  Combined p-value vector from all nodes
combined_reject_fdr
    Combined FDR rejection vector from all nodes
total_power_proxy
    Total number of significant assets across all nodes after FDR correction

```

Examples

```

set.seed(42)
T <- 120
N <- 100 # Larger dataset for distributed testing
K_factors <- 3
fac <- matrix(rnorm(T * K_factors), T, K_factors)
beta <- matrix(rnorm(N * K_factors), N, K_factors)
alpha <- rep(0, N)
alpha[1:10] <- 0.4 / 100 # 10 non-zero alphas
eps <- matrix(rnorm(T * N, sd = 0.02), T, N)
ret <- alpha + fac %*% t(beta) + eps

# Distributed testing with 4 nodes, each handling 25 assets
results <- Dfactor.tests(ret, fac, n1 = 25, K = 4, q.fdr = 0.05)

# View combined results
cat("Total significant assets after FDR:", results$total_power_proxy, "\n")
cat("Combined results across all nodes:\n")
print(summary(results$combined_alpha))

```

 DGulPC

Distributed general unilateral loading principal component

Description

Distributed general unilateral loading principal component

Usage

```
DGulPC(data, m, n1, K)
```

Arguments

<code>data</code>	is a total data set
<code>m</code>	is the number of principal component
<code>n1</code>	is the length of each data subset
<code>K</code>	is the number of nodes

Value

AU1,AU2,DU3,Shat

Examples

```
library(LFM)
data_from_package <- Wine
data_a <- Wine
DGulPC(data_a,m=3,n1=128,K=2)
```

DIPC

Distributed Incremental Principal Component Analysis (DIPC)

Description

Apply IPC in a distributed manner across K nodes.

Usage

```
DIPC(data, m, eta, K)
```

Arguments

<code>data</code>	Matrix of input data ($n \times p$).
<code>m</code>	Number of principal components.
<code>eta</code>	Proportion of initial batch to total data within each node.
<code>K</code>	Number of nodes (distributed splits).

Value

List with per-node results and aggregated averages.

Examples

```
library(LFM)
data_from_package <- Wine
data_a <- Wine
m=5
eta=0.8
K=2
results <- DIPC(data_a, m, eta, K)
```

DPC	<i>Distributed principal component</i>
-----	--

Description

Distributed principal component

Usage

```
DPC(data, m, n1, K)
```

Arguments

data	is a total data set
m	is the number of principal component
n1	is the length of each data subset
K	is the number of nodes

Value

Ahat,Dhat,Sigmahathat

Examples

```
library(LFM)
data_from_package <- Wine
data_a <- Wine
DPC(data_a,m=3,n1=128,K=2)
```

DPPC	<i>Distributed projection principal component</i>
------	---

Description

Distributed projection principal component

Usage

```
DPPC(data, m, n1, K)
```

Arguments

data	is a total data set
m	is the number of principal component
n1	is the length of each data subset
K	is the number of nodes

Value

Apro,pro,Sigmahathatpro

Examples

```
library(LFM)
data_from_package <- Wine
data_a <- Wine
DPPC(data_a,m=3,n1=128,K=2)
```

DSAPC

The distributed stochastic approximation principal component for handling online data sets with highly correlated data across multiple nodes.

Description

The distributed stochastic approximation principal component for handling online data sets with highly correlated data across multiple nodes.

Usage

```
DSAPC(data, m, eta, n1, K)
```

Arguments

data	is a highly correlated online data set
m	is the number of principal component
eta	is the proportion of online data to total data
n1	is the length of each data subset
K	is the number of nodes

Value

Asa, Dsa (lists containing results from each node)

Examples

```
library(LFM)
data_from_package <- Wine
data_a <- Wine
DSAPC(data=data_a, m=3, eta=0.8, n1=128, K=2)
```

factor.tests

*Factor Model Testing with Wald, GRS, PY tests and FDR control***Description**

Performs comprehensive factor model testing including joint tests (Wald, GRS, PY), individual asset t-tests, and False Discovery Rate control.

Usage

```
factor.tests(ret, fac, q.fdr = 0.05)
```

Arguments

ret	A $T \times N$ matrix representing the excess returns of N assets at T time points.
fac	A $T \times K$ matrix representing the returns of K factors at T time points.
q.fdr	The significance level for FDR (False Discovery Rate) testing, defaulting to 5%.

Value

A list containing the following components:

alpha	N-vector of estimated alphas for each asset
tstat	N-vector of t-statistics for testing individual alphas
pval	N-vector of p-values for individual alpha tests
Wald	Wald test statistic for joint alpha significance
p_Wald	p-value for Wald test
GRS	GRS test statistic (finite-sample F-test)
p_GRS	p-value for GRS test
PY	Pesaran and Yamagata test statistic
p_PY	p-value for PY test
reject_fdr	Logical vector indicating which assets have significant alphas after FDR correction
fdr_p	Adjusted p-values using Benjamini-Hochberg procedure
power_proxy	Number of significant assets after FDR correction

Examples

```
set.seed(42)
T <- 120
N <- 25
K <- 3
fac <- matrix(rnorm(T * K), T, K)
beta <- matrix(rnorm(N * K), N, K)
```

```

alpha <- rep(0, N)
alpha[1:3] <- 0.4 / 100 # 3 non-zero alphas
eps <- matrix(rnorm(T * N, sd = 0.02), T, N)
ret <- alpha + fac %*% t(beta) + eps
results <- factor.tests(ret, fac, q.fdr = 0.05)

# View results
cat("Wald test p-value:", results$p_Wald, "\n")
cat("GRS test p-value:", results$p_GRS, "\n")
cat("PY test p-value:", results$p_PY, "\n")
cat("Significant assets after FDR:", results$power_proxy, "\n")

```

FanPC

Apply the FanPC method to the Laplace 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(data, m)
```

Arguments

<code>data</code>	A matrix of input data.
<code>m</code>	is the number of principal component

Value

AF,DF,SigmahatF

Examples

```

library(LaplacesDemon)
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 <- FanPC(data, m)
print(results)
```

Ftest

*Apply the Farmtest method to the Laplace factor model***Description**

This function simulates data from a Laplace factor model and applies the FarmTest for multiple hypothesis testing. It calculates the false discovery rate (FDR) and power of the test.

Usage

```
Ftest(
  data,
  p1,
  alpha = 0.05,
  K = -1,
  alternative = c("two.sided", "less", "greater")
)
```

Arguments

data	A matrix or data frame of simulated or observed data from a Laplace factor model.
p1	The number or proportion of non-zero hypotheses.
alpha	The significance level for controlling the false discovery rate (default: 0.05).
K	The number of factors to estimate (default: -1, meaning auto-detect).
alternative	The alternative hypothesis: "two.sided", "less", or "greater" (default: "two.sided").

Value

A list containing the following elements:

FDR	The false discovery rate, which is the proportion of false positives among all discoveries (rejected hypotheses).
Power	The statistical power of the test, which is the probability of correctly rejecting a false null hypothesis.
PValues	A vector of p-values associated with each hypothesis test.
RejectedHypotheses	The total number of hypotheses that were rejected by the FarmTest.
reject	Indices of rejected hypotheses.
means	Estimated means.

Examples

```

library(LaplacesDemon)
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
p1=40
results <- Ftest(data, p1)
print(results$FDR)
print(results$Power)

```

GulPC

*General unilateral loading principal component***Description**

General unilateral loading principal component

Usage

```
GulPC(data, m)
```

Arguments

data	is a total data set
m	is the number of first layer principal component

Value

AU1,AU2,DU3,SigmaUhat

Examples

```

library(LFM)
data_from_package <- Wine
data_a <- Wine
GulPC(data=data_a,m=5)

```

Description

The function is to generate Laplace factor model data. The function supports various distribution types for generating the data, including: - 'truncated_laplace': Truncated Laplace distribution - 'log_laplace': Univariate Symmetric Log-Laplace distribution - 'Asymmetric Log_Laplace': Log-Laplace distribution - 'Skew-Laplace': Skew-Laplace distribution

Usage

```
LFM(n, p, m, distribution_type)
```

Arguments

n	An integer specifying the sample size.
p	An integer specifying the sample dimensionality or the number of variables.
m	An integer specifying the number of factors in the model.
distribution_type	A character string indicating the type of distribution to use for generating the data.

Value

A list containing the following elements:

data	A numeric matrix of the generated data.
A	A numeric matrix representing the factor loadings.
D	A numeric matrix representing the uniquenesses, which is a diagonal matrix.

Examples

```
library(MASS)
library(matrixcalc)
library(relliptical)
n <- 1000
p <- 10
m <- 5
sigma1 <- 1
sigma2 <- matrix(c(1,0.7,0.7,1), 2, 2)
distribution_type <- "truncated_laplace"
results <- LFM(n, p, m, distribution_type)
print(results)
```

PC	<i>Principal component</i>
----	----------------------------

Description

Principal component

Usage

```
PC(data, m)
```

Arguments

data	is a total data set
m	is the number of principal component

Value

Ahat, Dhat, Sigmahat

Examples

```
library(LFM)
data_from_package <- Wine
data_a <- Wine
PC(data_a,m=5)
```

PPC	<i>Projection principal component</i>
-----	---------------------------------------

Description

Projection principal component

Usage

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

Arguments

data	is a total data set
m	is the number of principal component

Value

Apro, Dpro, Sigmahatpro

Examples

```
library(LFM)
data_from_package <- Wine
data_a <- Wine
PPC(data=data_a,m=5)
```

SAPC

The stochastic approximation principal component can handle online data sets with highly correlated.

Description

The stochastic approximation principal component can handle online data sets with highly correlated.

Usage

```
SAPC(data, m, eta)
```

Arguments

data	is a highly correlated online data set
m	is the number of principal component
eta	is the proportion of online data to total data

Value

Asa,Dsa

Examples

```
library(LFM)
data_from_package <- Wine
data_a <- Wine
SAPC(data=data_a,m=3,eta=0.8)
```

Index

Dfactor.tests, [2](#)

DGuIPC, [3](#)

DIPC, [4](#)

DPC, [5](#)

DPPC, [5](#)

DSAPC, [6](#)

factor.tests, [7](#)

FanPC, [8](#)

Ftest, [9](#)

GuIPC, [10](#)

LFM, [11](#)

PC, [12](#)

PPC, [12](#)

SAPC, [13](#)