Package ‘FarmTest’

April 28, 2020

**Type** Package

**Title** Factor-Adjusted Robust Multiple Testing

**Version** 2.1.0

**Date** 2020-04-27

**Description** Performs robust multiple testing for means in the presence of known and unknown latent factors presented in Fan et al.(2019) "FarmTest: Factor-Adjusted Robust Multiple Testing With Approximate False Discovery Control" <doi:10.1080/01621459.2018.1527700>. Implements a series of adaptive Huber methods combined with fast data-drive tuning schemes proposed in Ke et al.(2019) "User-Friendly Covariance Estimation for Heavy-Tailed Distributions" <doi:10.1214/19-STS711> to estimate model parameters and construct test statistics that are robust against heavy-tailed and/or asymmetric error distributions. Extensions to two-sample simultaneous mean comparison problems are also included. As by-products, this package contains functions that compute adaptive Huber mean, covariance and regression estimators that are of independent interest.

**Depends** R (>= 3.6.0)

**License** GPL-3

**Encoding** UTF-8

**URL** https://github.com/XiaoouPan/FarmTest

**SystemRequirements** C++11

**Imports** Rcpp, graphics

**LinkingTo** Rcpp, RcppArmadillo

**RoxygenNote** 7.1.0

**NeedsCompilation** yes

**Author** Xiaoou Pan [aut, cre],
Yuan Ke [aut],
Wen-Xin Zhou [aut]

**Maintainer** Xiaoou Pan <xip024@ucsd.edu>

**Repository** CRAN

**Date/Publication** 2020-04-28 12:10:06 UTC
Description

FarmTest package performs robust multiple testing for means in the presence of known and unknown latent factors (Fan et al, 2019). It implements a series of adaptive Huber methods combined with fast data-drive tuning schemes (Wang et al, 2018; Ke et al, 2019) to estimate model parameters and construct test statistics that are robust against heavy-tailed and/or asymmetric error distributions. Extensions to two-sample simultaneous mean comparison problems are also included. As by-products, this package also contains functions that compute adaptive Huber mean, covariance and regression estimators that are of independent interest.

Details

See its GitHub page https://github.com/XiaoouPan/FarmTest for details.

References


---

farm.test

Factor-adjusted robust multiple testing

Description

This function conducts factor-adjusted robust multiple testing (FarmTest) for means of multivariate data proposed in Fan et al. (2019) via a tuning-free procedure.

Usage

farm.test(
  X,
  fX = NULL,
  KX = -1,
  Y = NULL,
  fY = NULL,
  KY = -1,
  h0 = NULL,
  alternative = c("two.sided", "less", "greater"),
  alpha = 0.05,
  p.method = c("bootstrap", "normal"),
  nBoot = 500
)

Arguments

X  An n by p data matrix with each row being a sample.

fX  An optional factor matrix with each column being a factor for X. The number of rows of fX and X must be the same.

KX  An optional positive number of factors to be estimated for X when fX is not specified. KX cannot exceed the number of columns of X. If KX is not specified or specified to be negative, it will be estimated internally. If KX is specified to be 0, no factor will be adjusted.

Y  An optional data matrix used for two-sample FarmTest. The number of columns of X and Y must be the same.

fY  An optional factor matrix for two-sample FarmTest with each column being a factor for Y. The number of rows of fY and Y must be the same.
KY
An optional positive number of factors to be estimated for Y for two-sample FarmTest when fY is not specified. KY cannot exceed the number of columns of Y. If KY is not specified or specified to be negative, it will be estimated internally. If KY is specified to be 0, no factor will be adjusted.

h0
An optional p-vector of true means, or difference in means for two-sample FarmTest. The default is a zero vector.

alternative
An optional character string specifying the alternate hypothesis, must be one of "two.sided" (default), "less" or "greater".

alpha
An optional level for controlling the false discovery rate. The value of alpha must be between 0 and 1. The default value is 0.05.

p.method
An optional character string specifying the method to calculate p-values when fX is known or when KX = 0, possible options are multiplier bootstrap or normal approximation. It must be one of "bootstrap" (default) or "normal".

nBoot
An optional positive integer specifying the size of bootstrap sample, only available when p.method = "bootstrap". The default value is 500.

Details
For two-sample FarmTest, means, stdDev, loadings, eigenVal, eigenRatio, nFactors and n will be lists of items for sample X and Y separately.

alternative = "greater" is the alternative that \( \mu > \mu_0 \) for one-sample test or \( \mu_X > \mu_Y \) for two-sample test.

Setting p.method = "bootstrap" for factor-known model will slow down the program, but it will achieve lower empirical FDP than setting p.method = "normal".

Value
An object with S3 class farm.test containing the following items will be returned:

- means Estimated means, a vector with length p.
- stdDev Estimated standard deviations, a vector with length p. It’s not available for bootstrap method.
- loadings Estimated factor loadings, a matrix with dimension p by K, where K is the number of factors.
- eigenVal Eigenvalues of estimated covariance matrix, a vector with length p. It’s only available when factors fX and fY are not given.
- eigenRatio Ratios of eigenVal to estimate nFactors, a vector with length min(n, p)/2. It’s only available when number of factors KX and KY are not given.
- nFactors Estimated or input number of factors, a positive integer.
- tStat Values of test statistics, a vector with length p. It’s not available for bootstrap method.
- pValues P-values of tests, a vector with length p.
- pAdjust Adjusted p-values of tests, a vector with length p.
- significant Boolean values indicating whether each test is significant, with 1 for significant and 0 for non-significant, a vector with length p.
reject  Indices of tests that are rejected. It will show "no hypotheses rejected" if none of the tests are rejected.

type  Indicator of whether factor is known or unknown.
n  Sample size.
p  Data dimension.
h0  Null hypothesis, a vector with length \( p \).
alpha  \( \alpha \) value.
alternative  Alternative hypothesis.

References


See Also

print.farm.test, summary.farm.test and plot.farm.test.

Examples

```r
n = 20
p = 50
K = 3
muX = rep(0, p)
muX[1:5] = 2
epsilonX = matrix(rnorm(p * n, 0, 1), nrow = n)
BX = matrix(runif(p * K, -2, 2), nrow = p)
fx = matrix(rnorm(K * n, 0, 1), nrow = n)
X = rep(1, n) %*% t(muX) + fx %*% t(BX) + epsilonX
# One-sample FarmTest with two sided alternative
output = farm.test(X)
# One-sample FarmTest with one sided alternative
output = farm.test(X, alternative = "less")
# One-sample FarmTest with known factors
output = farm.test(X, fX = fx)
```
# Two-sample FarmTest
muY = rep(0, p)
muY[1:5] = 4
epsilonY = matrix(rnorm(p * n, 0, 1), nrow = n)
BY = matrix(runif(p * K, -2, 2), nrow = p)
fY = matrix(rnorm(K * n, 0, 1), nrow = n)
Y = rep(1, n) %*% t(muY) + fY %*% t(BY) + epsilonY
output = farm.test(X, Y = Y)

## Description

The function calculates adaptive Huber-type covariance estimator from a data sample, with robustification parameter \( \tau \) determined by a tuning-free principle. For the input matrix \( X \), both low-dimension \( (p < n) \) and high-dimension \( (p > n) \) are allowed.

## Usage

```r
huber.cov(X)
```

## Arguments

- **X**: An \( n \) by \( p \) data matrix.

## Value

A \( p \) by \( p \) Huber-type covariance matrix estimator will be returned.

## References


## See Also

- `huber.mean` for tuning-free Huber mean estimation and `huber.reg` for tuning-free Huber regression.

## Examples

```r
n = 100
d = 50
X = matrix(rt(n * d, df = 3), n, d) / sqrt(3)
Sigma = huber.cov(X)
```
huber.mean

**Tuning-free Huber mean estimation**

**Description**

The function calculates adaptive Huber mean estimator from a data sample, with robustification parameter $\tau$ determined by a tuning-free principle.

**Usage**

```r
huber.mean(X)
```

**Arguments**

- `X` An $n$-dimensional data vector.

**Value**

A Huber mean estimator will be returned.

**References**


**See Also**

- `huber.cov` for tuning-free Huber-type covariance estimation and `huber.reg` for tuning-free Huber regression.

**Examples**

```r
n = 10000
X = rt(n, 2) + 2
mu = huber.mean(X)
```
Tuning-free Huber regression

Description

The function calculates adaptive Huber regression estimator from a data sample, with robustification parameter \( \tau \) determined by a tuning-free principle.

Usage

```
huber.reg(X, Y)
```

Arguments

- **X**: An \( n \) by \( p \) design matrix, where \( p < n \).
- **Y**: A continuous response with length \( n \).

Value

A coefficients estimator with length \( p + 1 \) will be returned.

References


See Also

- \texttt{huber.mean} for tuning-free Huber mean estimation and \texttt{huber.cov} for tuning-free Huber-type covariance estimation.

Examples

```
n = 200
d = 10
beta = rep(1, d)
X = matrix(rnorm(n * d), n, d)
err = rnorm(n)
Y = 1 + X %%*%% beta + err
beta.hat = huber.reg(X, Y)
```
plot.farm.test

Description

This is the plot function of S3 objects with class "farm.test". It produces the histogram of estimated means.

Usage

```r
## S3 method for class 'farm.test'
plot(x, ...)
```

Arguments

- `x`: A `farm.test` object.
- `...`: Further arguments passed to or from other methods.

Details

For two-sample FarmTest, the histogram is based on the difference: estimated means of sample X - estimated means of sample Y.

Value

No variable will be returned, but a histogram of estimated means will be presented.

See Also

`farm.test`, `print.farm.test` and `summary.farm.test`.

Examples

```r
n = 50
p = 100
K = 3
muX = rep(0, p)
muX[1:5] = 2
epsilonX = matrix(rnorm(p * n, 0, 1), nrow = n)
BX = matrix(runif(p * K, -2, 2), nrow = p)
fx = matrix(rnorm(K * n, 0, 1), nrow = n)
X = rep(1, n) %*% t(muX) + fx %*% t(BX) + epsilonX
output = farm.test(X)
plot(output)
```
print.farm.test  

*Print function of FarmTest*

**Description**

This is the print function of S3 objects with class "farm.test".

**Usage**

```
## S3 method for class 'farm.test'
print(x, ...)  
```

**Arguments**

- **x**: A `farm.test` object.
- **...**: Further arguments passed to or from other methods.

**Value**

No variable will be returned, but a brief summary of FarmTest will be displayed.

**See Also**

- `farm.test`, `summary.farm.test` and `plot.farm.test`.

**Examples**

```r
n = 50
p = 100
K = 3
muX = rep(0, p)
muX[1:5] = 2
epsilonX = matrix(rnorm(p * n, 0, 1), nrow = n)
BX = matrix(runif(p * K, -2, 2), nrow = p)
fX = matrix(rnorm(K * n, 0, 1), nrow = n)
X = rep(1, n) %*% t(muX) + fX %*% t(BX) + epsilonX
output = farm.test(X)
print(output)
```
Summary function of FarmTest

Description

This is the summary function of S3 objects with class "farm.test".

Usage

```r
## S3 method for class 'farm.test'
summary(object, ...)  
```

Arguments

- `object` A `farm.test` object.
- `...` Further arguments passed to or from other methods.

Details

For two-sample FarmTest, the first column is the difference: estimated means of sample X - estimated means of sample Y.

Value

A data frame including the estimated means, p-values, adjusted p-values and significance for all the features will be presented.

See Also

`farm.test`, `print.farm.test` and `plot.farm.test`.

Examples

```r
n = 50
p = 100
K = 3
muX = rep(0, p)
muX[1:5] = 2
epsilonX = matrix(rnorm(p * n, 0, 1), nrow = n)
BX = matrix(runif(p * K, -2, 2), nrow = p)
fX = matrix(rnorm(K * n, 0, 1), nrow = n)
X = rep(1, n) %*% t(muX) + fX %*% t(BX) + epsilonX
output = farm.test(X)
summary(output)
```
Index

farm.test, 3, 9–11
FarmTest-package, 2

huber.cov, 6, 7, 8
huber.mean, 6, 7, 8
huber.reg, 6, 7, 8

plot.farm.test, 5, 9, 10, 11
print.farm.test, 5, 9, 10, 11

summary.farm.test, 5, 9, 10, 11