Package 'fastJT'

July 18, 2019

Type Package

Title Efficient Jonckheere-Terpstra Test Statistics for Robust Machine Learning and Genome-Wide Association Studies

Version 1.0.5

Date 2019-07-18

Author Jiaxing Lin, Alexander Sibley, Ivo Shterev, and Kouros Owzar

Maintainer Jiaxing Lin <jiaxing.lin@duke.edu>

Description This 'Rcpp'-based package implements highly efficient functions for the calculation of the Jonckheere-Terpstra statistic. It can be used for a variety of applications, including feature selection in machine learning problems, or to conduct genome-wide association studies (GWAS) with multiple quantitative phenotypes. The code leverages 'OpenMP' directives for multi-core computing to reduce overall processing time.

License GPL (>= 2)

Imports Rcpp (>= 0.12.3)

LinkingTo Rcpp

Suggests knitr

VignetteBuilder knitr

BuildVignettes yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-07-18 16:40:08 UTC

R topics documented:

`fastJT-package` *Efficient Jonckheere-Terpstra Test Statistics for Robust Machine Learning and Genome-wide Association Studies*

Description

This `Rcpp`-based package implements highly efficient functions for the calculation of the Jonkheere-Terpstra statistic. It can be used for a variety of applications, including feature selection in machine learning problems, or to conduct genome-wide association studies (GWAS) with multiple quantitative phenotypes. The code leverages `OpenMP` directives for multi-core computing to reduce overall processing time.

Details

Package:	fastJT
Type:	Package
Version:	1.0.5
Date:	2019-07-18
License:	GPL-3

Please see the example function calls below, or refer to the individual function documentation or the included vignette for more information. The package vignette serves as a tutorial for using this package. The technical details are provided in the reference cited below. Specifically, the calculation of the standardized test statistics employs the null variance equation as defined by Hollander and Wolfe (1999, eq. 6.19) to account for ties in the data.

Author(s)

Jiaxing Lin, Alexander Sibley, Ivo Shterev, and Kouros Owzar

Maintainer: Jiaxing Lin <jiaxing.lin@duke.edu>

References

Hollander, M. and Wolfe, D. A. (1999) *Nonparametric Statistical Methods*. New York: Wiley, 2nd edition.

See Also

`Rcpp`

Examples

```
# Generate dummy data
num_sample  <- 100
num_marker  <- 10
```

```
num_feature <- 500
set.seed(12345)
Data <- matrix(rnorm(num_sample*num_marker), num_sample, num_marker)
Features <- matrix(rbinom(num_sample*num_feature, 2, 0.5), num_sample, num_feature)
colnames(Data) <- paste0("Var:",1:num_marker)
colnames(Features) <- paste0("Ftr:",1:num_feature)

res <- fastJT(Y=Data, X=Features, outTopN=15)
res
res <- fastJT.select(Y=Data, X=Features, cvMesh=NULL, kFold=5,
                     selCrit=NULL, outTopN=5, numThreads=1)
res
```

`fastJT` *Compute the Jonckheere-Terpstra Test Statistics*

Description

A method to compute the Jonckheere-Terpstra test statistics for large numbers of dependent and independent variables, with optional multi-threaded execution. The calculation of the standardized test statistics employs the null variance equation as defined by Hollander and Wolfe (1999, eq. 6.19) to account for ties in the data.

Usage

```
fastJT(Y, X, outTopN=15L, numThreads=1L, standardized=TRUE)
```

Arguments

`Y`	A matrix of continuous values, representing dependent variables, e.g. marker levels or other observed values. Row names should be sample IDs, and column names should be variable names. Required.
`X`	A matrix of integer values, representing independent variables, e.g. SNP counts or other classification features. Row names should be sample IDs, and column names should be feature IDs. Required.
`outTopN`	An integer to indicate the number of top independent variables to be reported for each dependent variable, based on the standardized Jonckheere-Terpstra test statistics. Optional. The default value is `15L`. If set to `NA`, all results are returned.
`numThreads`	A integer to indicate the number of threads used in the computation. Optional. The default value is `1L` (sequential computation).
`standardized`	A boolean to specify whether to return standardized Jonckheere-Terpstra statistics (`TRUE`) or non-standardized statistics (`FALSE`). Optional. The default value is `TRUE`.

Value

A `list` with two objects

`J`	A matrix of the standardized/non-standardized Jonckheere-Terpstra test statistics, depending on the value of the `standardized` argument.
`XIDs`	If `outTopN` was specified, this object is a matrix of the column IDs of `X` corresponding to the top standardized Jonckheere-Terpstra test statistics for each dependent variable. Otherwise this is a vector of the column IDs of `X`.

Note

Rows (samples) are assumed to be in the same order in `X` and `Y`.

References

Hollander, M. and Wolfe, D. A. (1999) *Nonparametric Statistical Methods*. New York: Wiley, 2nd edition.

Examples

```
# Generate dummy data
num_sample <- 100
num_marker <- 10
num_SNP  <- 500
set.seed(12345)
Mark <- matrix(rnorm(num_sample*num_marker), num_sample, num_marker)
Geno <- matrix(rbinom(num_sample*num_SNP, 2, 0.5), num_sample, num_SNP)
colnames(Mark) <- paste0("Mrk:",1:num_marker)
colnames(Geno) <- paste0("SNP:",1:num_SNP)

res <- fastJT(Y=Mark, X=Geno, outTopN=5)
res
res <- fastJT(Y=Mark, X=Geno, outTopN=NA)
head(res)
```

---

fastJT.select — *Conduct k-Fold Cross-Validation based on Jonckheere-Terpstra Test Statistics*

Description

A method to conduct `k`-fold cross-validation based the Jonckheere-Terpstra test statistics for large numbers of dependent and independent variables, with optional multi-threaded execution. The data are divided into `k` subsets, one of which is withheld while the remaining `k`-1 subsets are used to test the features. The process is repeated `k` times, with each of the subsets withheld exactly once as the validation data. The calculation of the standardized test statistics employs the null variance equation as defined by Hollander and Wolfe (1999, eq. 6.19) to account for ties in the data.

Usage

```
fastJT.select(Y, X, cvMesh=NULL, kFold=10L, selCrit=NULL, outTopN=15L, numThreads=1
```

Arguments

Y	A matrix of continuous values, representing dependent variables, e.g. marker levels or other observed values. Row names should be sample IDs, and column names should be variable names. Required.
X	A matrix of integer values, representing independent variables, e.g. SNP counts or other classification features. Row names should be sample IDs, and column names should be feature IDs. Required.
cvMesh	A user-defined function to specify how to separate the data into training and testing parts. The inputs of this function are a vector representing the sample IDs and kFold, an integer representing the number of folds for cross validation. The output of this function is a list of kFold vectors of sample IDs forming the testing subset for each fold. The default value is NULL, and if no function is specified, the data are partitioned sequentially into kFold equal sized subsets. Optional.
kFold	An integer to indicate the number of folds. Optional. The default value is 10.
selCrit	A user-defined function to specify the criteria for selecting the top features. The inputs of this function are J, the matrix of statistics resulting from fastJT, and P, the matrix of p-values from pvalues(J). The output is a data frame containing the selected feature ID for each trait of interest. Optional. The default value is NULL, and if no function is specified, the features with the largest standardized Jonckheere-Terpstra test statistics are selected.
outTopN	An integer to indicate the number of top hits to be returned when selCrit=NULL. Unused if selCrit!=NULL. Optional. The default value is 15L.
numThreads	A integer to indicate the number of threads to be used in the computation. Optional. The default value is 1L (sequential computation).

Value

Three lists of length kFold

J	A list of matrices of standardized Jonckheere-Terpstra test statistics, one for each cross validation.
Pval	A list of matrices of p-values, one for each cross validation.
XIDs	A list of matrices of the selected feature IDs, one for each cross validation.

Note

Rows (samples) are assumed to be in the same order in X and Y.

References

Hollander, M. and Wolfe, D. A. (1999) *Nonparametric Statistical Methods*. New York: Wiley, 2nd edition.

See Also

fastJT

Examples

```
# Generate dummy data
num_sample  <- 100
num_marker  <- 10
num_feature <- 500
set.seed(12345)
Data <- matrix(rnorm(num_sample*num_marker), num_sample, num_marker)
Features <- matrix(rbinom(num_sample*num_feature, 2, 0.5), num_sample, num_feature)
colnames(Data) <- paste0("Var:",1:num_marker)
colnames(Features) <- paste0("Ftr:",1:num_feature)

res <- fastJT.select(Y=Data, X=Features, cvMesh=NULL, kFold=5,
                     selCrit=NULL, outTopN=5, numThreads=1)
res
```

---

pvalues *Compute P-values Based on Jonckheere-Terpstra Test Statistics*

Description

Method to compute the p-values for results from fastJT. fastJT must be run with standardized=TRUE in order to use this function.

Usage

```
pvalues(object)
```

Arguments

object	A fastJT object that is the return of method fastJT. Required.

Value

A matrix of p-values with the same dimensions as the standardized statistics from fastJT.

See Also

fastJT

Examples

```
# Generate dummy data
num_patient <- 100
num_marker  <- 10
num_SNP     <- 500
set.seed(12345)
Mark <- matrix(rnorm(num_patient*num_marker), num_patient, num_marker)
Geno <- matrix(rbinom(num_patient*num_SNP, 2, 0.5), num_patient, num_SNP)
colnames(Mark) <- paste0("Mrk:",1:num_marker)
colnames(Geno) <- paste0("SNP:",1:num_SNP)

res <- fastJT(Y=Mark, X=Geno, outTopN=5)
pvalues(res)
res <- fastJT(Y=Mark, X=Geno, outTopN=NA)
pvalues(res)
```

---

summary.fastJT *Summarize Jonckheere-Terpstra Test Statistics and P-Values*

Description

Summary method for fastJT results.

Usage

```
## S3 method for class 'fastJT'
summary(object, Y2Print=1:10, X2Print=1:10, printP=TRUE,
                outTopN=NA, subObj=FALSE, ...)
```

Arguments

`object`	A `fastJT` object that is the return of method `fastJT`. Required.
`Y2Print`	Either a numeric or character vector that indicates the desired dependent variables to print. The default is 1:10. Set to `NA` to print full results. Optional.
`X2Print`	Either a numeric or character vector that indicates the desired independent variables to print. If `outTopN=NA` in function `fastJT`, i.e., the results are not sorted, both numeric and character vectors can be used to set the print range. The default range is 1:10. Set to `NA` to print full results. If `outTopN!=NA` in function `fastJT`, the range of `X2Print` refers to the range of top normalized statistics computed in `fastJT`, and only numeric vectors can be used. Optional.
`printP`	A boolean indicating whether to print the p-values (`TRUE`) or the standardized statistics (`FALSE`). The default value is `TRUE`. Optional.
`outTopN`	An integer specifying the number of top hits to print in the summary, if the statistics were not sorted during the evaluation of the statistics (i.e., if `outTopN=NA` in function `fastJT`). Optional. The default value is `NA`.

subObj	A boolean indicating whether to return a fastJT object subset per the requested summary (TRUE). Optional. The default value is FALSE (nothing is returned).
...	Additional arguments affecting the summary produced.

Value

If subObj=TRUE, this method returns a fastJT object matching the subset of statistics from object that are being printed. For example, if object is not sorted by top hits, summary(object,outTopN=10,sub will print the summary and return a subset of object that contains only the top 10 independent variables for each dependent variable. If subObj=FALSE, the summary is printed but no values are returned.

Note

This function prints a matrix or paired columns of independent variable IDs and statistics/p-values to the log.

See Also

fastJT, pvalues

Examples

```
# Generate dummy data
num_patient <- 100
num_marker  <- 10
num_SNP     <- 500
set.seed(12345);
Mark <- matrix(rnorm(num_patient*num_marker), num_patient, num_marker)
Geno <- matrix(rbinom(num_patient*num_SNP, 2, 0.5), num_patient, num_SNP)
colnames(Mark) <- paste0("Mrk:",1:num_marker)
colnames(Geno) <- paste0("SNP:",1:num_SNP)

res <- fastJT(Y=Mark, X=Geno, outTopN=5)
summary(res, Y2Print=c("Mrk:1","Mrk:2"), X2Print=1:5, printP=FALSE)
summary(res, Y2Print=NA, X2Print=1:5, printP=FALSE)

res <- fastJT(Y=Mark, X=Geno, outTopN=NA)
summary(res, Y2Print=1:10, X2Print=1:10, printP=TRUE)
summary(res, Y2Print=c("Mrk:1","Mrk:2"), X2Print=c("SNP:1","SNP:2"), printP=TRUE)

res <- fastJT(Y=Mark, X=Geno, outTopN=NA, standardized=FALSE)
summary(res, outTopN=10)
```