fastJT

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

2017-02-06

Outline

Introduction

- This document provides an example for using the `fastJT` package to calculate the Jonckheere-Terpstra test statistics for large data sets (multiple dependent and independent variables) commonly encountered in machine learning or GWAS. The functionality is also included to perform k-fold cross validation or feature sets.
- The calculation of the standardized test statistic employs the null variance equation as defined by Hollander and Wolfe (1999, eq. 6.19) to account for ties in the data.
- The major algorithm in this package is written in C++, which is ported to `R` by `Rcpp`, to facilitate fast computation.
- Features of this package include:
  1. $O(N \times \log(N))$ computational complexity (where N is the number of the samples)
  2. `OpenMP` supported parallelization
  3. Customized output of top m significant independent variables for each dependent variable

fastJT

```r
res <- fastJT(Y, X, outTopN=15, numThreads=1, standardized=TRUE)
```

Function arguments:

`Y`:	Matrix of continuous values, representing dependent variables, with sample IDs as row names and variable names as column names.
`X`:	A matrix of integer values, representing independent variables, with sample IDs as row names and feature IDs as column names.
`outTopN`:	Number of top statistics to return (i.e., the largest standardized statistics). The default value is 15. If `outTopN` is set to `NA`, all results will be returned.
`numThreads`:	Number of threads to use for parallel computation. The default value is 1 (sequential computation).
`standardized`:	A boolean to specify whether to return standardized statistics or non-standardized statistics. The default value is `TRUE`, returning standardized statistics.

Users may wish to consider the `dplyr::recode()` function for converting non-numeric group indices into ordinal values for argument `X`.

Returned Values

Function returns:

- `J`: A matrix of standardized/non-standardized Jonckheere-Terpstra test statistics, depending on option `standardized`, with column names from input `Y`. If `outTopN` is not `NA`, results are sorted within each column.
- `XIDs`: If `outTopN` is not `NA`, this is a matrix of column names from `X` associated with top standardized Jonckheere-Terpstra test statistics from `J`. Otherwise this is an unsorted vector of column names from input `X`.

fastJT.select

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

Function arguments:

- `Y`: Matrix of continuous values, representing dependent variables, with sample IDs as row names and variable names as column names.
- `X`: Matrix of integer values, representing independent variables, with sample IDs as row names and feature IDs as column names.
- `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 of 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.

fastJT.select

Function arguments continued:

`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 include `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 IDs for each trait of interest. Optional. The default value is `NULL`, and if no function is specified, the features 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 15.
`numThreads`:	A integer to indicate the number of threads to be used in the computation. Optional. The default value is 1 (sequential computation).

The function withholds one subset while the remaining `kFold-1` subsets are used to test the features. The process is repeated `kFold` times, with each of the subsets withheld exactly once as the validation data.

Returned Values

Function returns: 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.

Simulate Data

1 Define the number of markers, samples, and features:

```
num_sample      <- 100
num_marker      <- 4
num_feature     <- 50
```

2 Create two matrices containing marker levels and feature information.

   a. `Data` contains the samples' marker levels.
   b. `Feature` contains the samples' feature values.

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

Load Package

Load `fastJT` (after installing its dependent packages):

```r
library(fastJT)
```

Example Execution

```
JTStat <- fastJT(Y=Data, X=Feature, outTopN=10)
summary(JTStat, Y2Print=1:4, X2Print=1:5)
```

```
##
##
##                   Johckheere-Terpstra Test for Large Matrices
##                     P-values for Top Standardized Statistics
## ================================================================================
##
##               Mrk:1|              Mrk:2|              Mrk:3|              Mrk:4|
## --------------------------------------------------------------------------------
##       SNPID P-value|      SNPID P-value|      SNPID P-value|      SNPID P-value|
## --------------------------------------------------------------------------------
##      Ftr:35 1.7e-02|     Ftr:35 2.0e-02|     Ftr:20 1.9e-02|     Ftr:46 1.2e-02|
##      Ftr:17 1.7e-02|      Ftr:7 5.7e-02|     Ftr:49 3.5e-02|     Ftr:34 1.7e-02|
##      Ftr:27 7.0e-02|     Ftr:46 9.1e-02|     Ftr:26 3.8e-02|     Ftr:47 3.7e-02|
##      Ftr:28 7.0e-02|     Ftr:40 9.4e-02|     Ftr:47 5.6e-02|     Ftr:23 4.5e-02|
##      Ftr:14 8.8e-02|     Ftr:29 1.5e-01|     Ftr:30 9.7e-02|     Ftr:16 5.1e-02|
```

Example Execution: Statistics in the Summary

```
summary(JTStat, Y2Print=1:4, X2Print=1:5, printP=FALSE)
```

```
##
##
##                     Johckheere-Terpstra Test for Large Matrices
##                             Top Standardized Statistics
## ================================================================================
##
##              Mrk:1|              Mrk:2|              Mrk:3|              Mrk:4|
## --------------------------------------------------------------------------------
##       SNPID      J*|      SNPID      J*|      SNPID      J*|      SNPID      J*|
## --------------------------------------------------------------------------------
##      Ftr:35  -2.390|     Ftr:35  -2.331|     Ftr:20   2.350|     Ftr:46   2.505|
##      Ftr:17  -2.388|      Ftr:7   1.905|     Ftr:49  -2.106|     Ftr:34   2.396|
##      Ftr:27  -1.813|     Ftr:46   1.693|     Ftr:26  -2.072|     Ftr:47  -2.089|
##      Ftr:28   1.813|     Ftr:40   1.676|     Ftr:47   1.914|     Ftr:23   2.000|
##      Ftr:14   1.706|     Ftr:29   1.432|     Ftr:30  -1.662|     Ftr:16   1.955|
```

Example Execution: Sorting in the Summary

```r
JTAll <- fastJT(Y=Data, X=Feature, outTopN=NA)
summary(JTAll, Y2Print=1:4, outTopN=3)
```

```
##
##
##                   Johckheere-Terpstra Test for Large Matrices
##                     P-values for Top Standardized Statistics
## ================================================================================
##
##               Mrk:1|              Mrk:2|              Mrk:3|              Mrk:4|
## --------------------------------------------------------------------------------
##       SNPID P-value|      SNPID P-value|      SNPID P-value|      SNPID P-value|
## --------------------------------------------------------------------------------
##      Ftr:35 1.7e-02|     Ftr:35 2.0e-02|     Ftr:20 1.9e-02|     Ftr:46 1.2e-02|
##      Ftr:17 1.7e-02|      Ftr:7 5.7e-02|     Ftr:49 3.5e-02|     Ftr:34 1.7e-02|
##      Ftr:27 7.0e-02|     Ftr:46 9.1e-02|     Ftr:26 3.8e-02|     Ftr:47 3.7e-02|
```

Example Execution: fastJT.select

```r
fastJT.select(Y=Data, X=Feature, cvMesh=NULL, kFold=5,
              selCrit=NULL, outTopN=5, numThreads=1)
```

```
## $J
## $J[[1]]
##          Mrk:1     Mrk:2     Mrk:3     Mrk:4
## [1,] -2.449202 -2.401972  2.402878  2.491444
## [2,] -2.356094  1.863353  2.159547  2.333146
## [3,] -1.798716  1.662280 -2.066804 -2.185883
## [4,]  1.796900  1.662069 -1.902853  2.004838
## [5,]  1.754959  1.399926 -1.893419  1.955443
##
## $J[[2]]
##          Mrk:1     Mrk:2     Mrk:3     Mrk:4
## [1,] -2.449202 -2.401972  2.402878  2.491444
## [2,] -2.356094  1.863353  2.159547  2.333146
## [3,] -1.798716  1.662280 -2.066804 -2.185883
## [4,]  1.796900  1.662069 -1.902853  2.004838
## [5,]  1.754959  1.399926 -1.893419  1.955443
##
## $J[[3]]
##          Mrk:1     Mrk:2     Mrk:3     Mrk:4
## [1,] -2.449202 -2.401972  2.402878  2.491444
## [2,] -2.356094  1.863353  2.159547  2.333146
## [3,] -1.798716  1.662280 -2.066804 -2.185883
## [4,]  1.796900  1.662069 -1.902853  2.004838
## [5,]  1.754959  1.399926 -1.893419  1.955443
##
## $J[[4]]
##          Mrk:1     Mrk:2     Mrk:3     Mrk:4
## [1,] -2.449202 -2.401972  2.402878  2.491444
## [2,] -2.356094  1.863353  2.159547  2.333146
## [3,] -1.798716  1.662280 -2.066804 -2.185883
## [4,]  1.796900  1.662069 -1.902853  2.004838
## [5,]  1.754959  1.399926 -1.893419  1.955443
##
## $J[[5]]
```

Example User-Defined cvMesh Function

```
Mesh <- function(rownamesData, kFold){
  numSamples <- length(rownamesData)
  res <- NULL
  subSampleSize <- floor(numSamples/kFold)
  for (i in 1:kFold){
    start <- (i-1)*subSampleSize + 1
    if(i < kFold)
      end <- i*subSampleSize
    else
      end <- numSamples
    if(i == 1)
      res <- list(c(start:end))
    else
      res[[i]] <- c(start:end)
  }
  res
}
```

Example User-Defined `selCrit` Function

```r
whichpart <- function(x, n=30) {
  nx <- length(x)
  p <- nx-n
  xp <- sort(x, partial=p)[p]
  which(x > xp)
}

selectCrit <- function(J, P){
  pcut <- 0.95
  hit <- NULL
  for(i in 1:ncol(P)){
    if(i == 1)
      hit <- list(rownames(P)[whichpart(P[,i], 4)])
    else
      hit[[i]] = rownames(P)[whichpart(P[,i], 4)]
  }
  res <- do.call(cbind, hit)
  colnames(res) <- colnames(P)
  res
}
```

Example Execution with User-Defined Functions

```r
fastJT.select(Data, Feature, cvMesh=Mesh, kFold=5,
              selCrit=selectCrit, outTopN=5, numThreads=1)
```

```
## $J
## $J[[1]]
##              Mrk:1       Mrk:2       Mrk:3       Mrk:4
## Ftr:1   1.044871948  0.30810327 -0.52243597 -1.41325629
## Ftr:2   0.064765493 -0.57607202  0.48744556  0.91694304
## Ftr:3   0.708932518 -0.93211498  0.38072302 -0.99775688
## Ftr:4   0.177698249 -0.10530267  1.24388774  0.13820975
## Ftr:5  -0.135648047  0.29164330 -1.58029974  1.23439722
## Ftr:6   0.626045363  0.20007635 -1.20691219 -0.61313721
## Ftr:7  -1.007934807  1.86335316  0.53712315  1.32623001
## Ftr:8   0.382152784 -1.08715878 -1.30459054  0.17789871
## Ftr:9  -0.627298931 -1.04772268  0.28695589  1.08776304
## Ftr:10 -0.552417489 -1.03827865  0.62562945 -0.54576186
## Ftr:11  0.718808299  0.69218577  0.79867589 -0.06655632
## Ftr:12  0.214931358 -0.73882654 -1.25600512 -1.24928852
## Ftr:13 -0.637403711 -1.34635274 -0.27317302 -0.61138723
## Ftr:14  1.754958762 -0.17582763 -0.27535270  0.53411788
## Ftr:15 -0.987651750  0.20792668  0.01299542 -1.13709905
## Ftr:16  1.487980337  0.09875976  0.72423822  1.95544319
## Ftr:17 -2.356093867 -0.36605978  0.00000000 -1.92347776
## Ftr:18  0.846622444  1.31915590  0.22970376  0.01312593
## Ftr:19  0.006991013  0.34955065  0.78998447 -1.37023855
## Ftr:20 -0.055126760 -0.45074233  2.40287819  0.49614084
## Ftr:21 -0.201986904  0.05212565 -0.19547120 -0.81446332
## Ftr:22 -0.242183804 -0.22254728  0.88364361 -0.96873521
## Ftr:23 -0.359172943  0.58773754  0.34611211  2.00483806
## Ftr:24  0.661565280 -0.17050652 -1.35041160  1.05032014
## Ftr:25  0.892047636  0.20079301  1.25413339 -0.41146109
## Ftr:26  0.275352698 -0.93221817 -2.06680399  0.82605809
## Ftr:27 -1.798716484 -0.26774787 -0.46684245  0.51489976
## Ftr:28  1.796899533  1.09120808  0.31364065  1.17615242
## Ftr:29  0.364631969  1.39992631  0.20184984 -0.71624137
## Ftr:30  0.451127576  0.81993260 -1.89341866  1.28752469
## Ftr:31 -0.868212309 -0.42431429  0.41125846  0.15014198
```

References

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

Session Information

- R Under development (unstable) (2017-02-20 r72220), `x86_64-pc-linux-gnu`
- Running under: `Debian GNU/Linux 8 (jessie)`
- Matrix products: default
- BLAS: `/usr/lib/openblas-base/libblas.so.3`
- LAPACK: `/usr/lib/openblas-base/liblapack.so.3`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: fastJT 1.0.2, knitr 1.11
- Loaded via a namespace (and not attached): Rcpp 0.12.9, compiler 3.4.0, evaluate 0.8, formatR 1.2.1, highr 0.5.1, magrittr 1.5, stringi 1.0-1, stringr 1.0.0, tools 3.4.0

```
## [1] "Start Time Mon Feb 20 16:24:54 2017"
## [1] "End Time   Mon Feb 20 16:24:56 2017"
```