Package ‘permGPU’

February 23, 2016

Type Package
Title Using GPUs in Statistical Genomics
Version 0.14.9
Date 2016-02-22
SystemRequirements Nvidia's CUDA toolkit (>= release 6.0)
OS_type unix
Description Can be used to carry out
permutation resampling inference in the context of RNA
microarray studies.
Depends R (>= 2.15.3), Biobase (>= 2.18.0), RUnit (>= 0.4.26), foreach
(>= 1.4.2), survival(>= 2.37-7), methods
License GPL-3
LazyLoad yes
Repository CRAN
Date/Publication 2016-02-23 08:04:33
NeedsCompilation yes
Author Ivo D. Shterev [aut, cre], Kouros Owzar [aut], Sin-Ho Jung [aut], Stephen L. George [aut], Brian J. Smith [ctb, cph] (wrote original version of configure.ac file), Kenneth Wilder [ctb, cph] (wrote ranker.h), Romain Francois [ctb, cph] (wrote original version of cleanup file), Dirk Eddelbuettel [ctb, cph] (wrote original version of cleanup file)
Maintainer Ivo D. Shterev <i.shterev@duke.edu>

R topics documented:
permGPU-package .......................................................... 2
makeExprSet .............................................................. 3
permgpu ................................................................. 4
permGPU: Using graphics processing units in RNA microarray association studies.

Description

This package can be used to carry out permutation resampling inference using GPUs. Currently the package supports six test statistics: the t and Wilcoxon tests, for two-sample problems, the Pearson and Spearman statistics, for non-censored continuous outcomes, and the Cox score and rank score test (Jung et al, 2005), for right-censored time-to-event outcomes. In addition to the test statistics and the corresponding marginal permutation P-values, the package produces family-wise error adjusted P-values using a single-step procedure (Westfall and Young, 1993).

Details

Package: permGPU
Type: Package
Version: 0.14.6
Date: 2014-07-11
License: GPL-3
LazyLoad: yes

~~ An overview of how to use the package, including the most important ~~ ~~ functions ~~

Note

To build this package, the CUDA SDK (version 2.3 or higher) must be installed on the system. Specifically, the nvcc compiler must be in the path and the CUDA_HOME must be properly defined. For example, if the SDK kit is installed under /usr/local/cuda then the CUDA_HOME variable needs to be set to /usr/local/cuda. The SDK can be obtained from www.nvidia.com The CUDA_HOME variable can also be explicitly defined in permGPU/src/Makefile. The maximum number of patients for the current version is 1000.

The R environment variables R_LIB and R_INCLUDE need to be correctly configured to build the package from source. Alternatively, these can be set in permGPU/src/Makefile.

To build this package, a number of C++ classes and functions for random number generation (available from http://www.agner.org/random/ under a GPL license) and a C++ template for calculating ranks (available from http://sites.google.com/site/jivsoft/Home/compute-ranks-of-elements-in-a-c_ARRAY under a BSD license) are needed. The requisite files are included in the package source code tar ball. In future releases, these functionalities will be replaced by native R functions from R主义思想和RMath.h.
makeExprSet

Author(s)
I. D. Shterev, S.-H. Jung, S. L. George and K. Owzar
Maintainer: Ivo D. Shterev <i.shterev@duke.edu>

References

Examples

### See `permGPU`

```
makeExprSet
Create an expressionSet object for use with permGPU
```

Description
The `permGPU` function expects that the phenotypic and molecular (expression) data are provided as an `ExpressionSet`. This is a simple utility function that creates this object so that permGPU can be used along with data objects created for use with certain Bioconductor packages.

Usage

```
makeExprSet(exprdat, phenodat, anno = "custom")
```

Arguments

- **exprdat**: This should be an $K \times n$, where $K$ denotes the number of markers/features and $n$ denotes the number of patients, expression matrix. It is expected that the $K$ marker names are assigned as row names of this matrix (i.e., could be extracted as `rownames(exprdat)`.
- **phenodat**: This is an $n \times p$ data.frame, $n$ denotes the number of patients and $p$ denotes the number of clinical co-variables. It is assumed that the rows of this data.frame are matched up with the columns of `exprdat`.
- **anno**: This slot can be used to assign a label to the data set.

Value
An object of class `ExpressionSet`. 
Note

This function may be deprecated in future releases if similar functionality is found in the base Bioconductor extension packages.

See Also

ExpressionSet

Examples

library(Biobase)
set.seed(123)

## Generate toy phenotype and expression data sets
## This example consists of 4 markers and ten patients
n<-10
K<-4
pdat=data.frame(grp=rep(1:0,each=n/2),bp=rnorm(n),ostime=rexp(n),event=rbinom(n,1,0.8))
expdat=matrix(rnorm(K*n),K,n)

## Assign marker names g1,...,gK to the expression data set
## (as row names) and patient ids id1,...,idn to the expression
## data set (as column names) and phenotype data (as row names)
rownames(expdat)=paste("g",1:K,sep="")
patid=paste("id",1:n,sep="")
rownames(pdat)=patid
colnames(expdat)=patid

## Create the ExprSet object
testdat=makeExprSet(expdat,pdat)
class(testdat)

## Check the dimensions of the expression and phenotype data sets
dim(exprs(testdat))
dim(pData(testdat))

## Get sample and marker ids
sampleNames(testdat)
featureNames(testdat)

---

### permgpu

**Conduct permutation resampling analysis using permGPU**

Description

This function can be used to carry our permutation resampling inference with GPUs. Currently the function supports six test statistics: the t and Wilcoxon tests, for two-sample problems, the Pearson
and Spearman statistics, for non-censored continuous outcomes, and the Cox score and rank score tests (Jung et al, 2005), for right-censored time-to-event outcomes.

Usage

```r
permgpu(datobj, y, event = NULL, test, B, diag = FALSE, scale = FALSE)
```

Arguments

datobj an `ExpressionSet` object containing the expression and phenotype data.
y The name of the outcome variable. Note that `y` must be an element of `names(pData(datobj))`. See example.
event In the case of survival analysis, `event` is the name of the event indicator. Note that `event` must be an element of `names(pData(datobj))`. See example.
test specifies the test to be performed. Possible options are `ttest` (two-sample t-test), `wilcoxon` (two-sample Wilcoxon test), `pearson` (Pearson correlation test), `spearman` (Spearman rank correlation test), `cox` (Cox score test) and `npcox` (Cox rank score test).
B specifies the number of random permutations to be performed.
diag This flag can be set to TRUE if specifies the type of object returned.
scale If TRUE, markers are centered.

Value

This function returns a data frame. The first column contains the gene names. The second, third and fourth columns contain the marginal test statistics, marginal unadjusted permutation P-values and FWER adjusted P-values respectively. If `diag=TRUE`, this function returns a list consisting of the following elements:

- `RESULTS` The results data frame as described above
- `EXPR` The gene expression data
- y The outcome data
- event event indicator(s) for survival analysis
- n The number of patients
- k The number of genes
- B The number of permutations
- test The test used in the permutation analysis

Note

The maximum number of patients for the current version is 1000.
References


Examples

```r
library(Biobase)
set.seed(123)

## Generate toy phenotype and expression data sets
## This example consists of 4 markers and 100 patients
## grp is a binary trait (e.g., case vs control)
## bp is a continuous trait (e.g., blood pressure)
## ostime is a right-censored time-to-event trait (e.g., observed
## time of death)
## event is the event indicator (1=dead or 0=censored) for ostime

n<-100
K<-4
grp=rep(1:0,each=n/2)
bp=rnorm(n)
atime=exp(n)
ctime=runif(n,0,1)
otime=pmin(atime,ctime)
event=as.integer(atime<=ctime)
pdat=data.frame(grp,bp,otime,event)
rm(grp,atime,ctime,otime,event)
expdat=matrix(rnorm(K+n),K,n)

## Assign marker names g1,...,gK to the expression data set and
## patient ids id1,...,idn to the expression and phenotype data
rownames(expdat)=paste("g",1:K,sep="")
patid=paste("id",1:n,sep="")
rownames(pdat)=patid
colnames(expdat)=patid

## Create the ExprSet object
testdat=makeExprSet(expdat,pdat)
class(testdat)
```
## Carry out permutation analysis with grp as the outcome
## using the two-sample t-test with B=100 random permutations
permgpu(testdat,"grp",B=100,test="ttest")

## Carry out permutation analysis with grp as the outcome
## using the two-sample Wilcoxon with B=100 random permutations
permgpu(testdat,"grp",B=100,test="wilcoxon")

## Carry out permutation analysis with bp as the outcome
## using the Pearson test with B=100 random permutations
permgpu(testdat,"bp",B=100,test="pearson")

## Carry out permutation analysis with bp as the outcome
## using the Spearman test with B=100 random permutations
permgpu(testdat,"bp",B=100,test="spearman")

## Carry out permutation analysis with ostime as the outcome
## using the covariance test (Jung et al, 2005) with B=100
## random permutations.
permgpu(testdat,"otime",event="event",B=100,test="cox")

## Carry out permutation analysis with ostime as the outcome
## using the rank-covariance test (Jung et al, 2005) with B=100
## random permutations.
permgpu(testdat,"otime",event="event",B=100,test="npcox")

## To carry out the analyses for the Director's Challenge
## Consortium Lung Cancer data, download the RMA pre-processed
## expressionSet object from the project webpage
## http://code.google.com/p/permgpu/
## After attaching it, check the md5sum signature
## attach("RMADAT-DCHALL.RData")
## md5sum("RMADAT-DCHALL.RData")
## 404fc27fe0c6d11c844e06139912f7ca
## A Sweave file outlining the steps carried out to pre-process
## the data is available from the project page.
##
## To carry out association testing using the Cox score test
## permgpu(RMADAT,"otime",event="event",B=10000,test="cox")
## To carry out association testing using the Cox rank score test
## permgpu(RMADAT,"otime",event="event",B=10000,test="npcox")
Description
This function can be used to carry out score test inference with GPUs. Currently the function supports two test statistics: the Cox score and rank score tests (Jung et al, 2005), for right-censored time-to-event outcomes.

Usage
scoregpu(y, event, markers, test, B=0, stand=TRUE, pval=TRUE, index=FALSE, scale=FALSE)

Arguments
y The name of the outcome variable.
event In the case of survival analysis, event is the name of the event indicator.
markers Expression matrix. Probes are along rows.
test specifies the test to be performed. Possible options are cox (Cox score test) and npcox (Cox rank score test).
B number of permutations.
stand If TRUE, the squared test statistic is returned.
pval If TRUE, the p-value is returned (stand has to be FALSE).
index If TRUE, the sample indexes at each permutation are returned.
scale If TRUE, markers are centered.

Value
This function returns a data frame with one column of test statistics.

Note
The maximum number of patients for the current version is 1000.

References

Examples
set.seed(123)
n<-100
K<-3
x1<-matrix(rnorm(n*K),K,n)
x2<-matrix(rnorm(n*K),K,n)
x3<-matrix(rnorm(n*K),K,n)
otime<-rexp(n)
describe the content of the image
test.scoregpu  

Conduct permutation resampling analysis using permGPU

Description
This function can be used to carry our permutation resampling inference with GPUs. Currently the function supports six test statistics: the t and Wilcoxon tests, for two-sample problems, the Pearson and Spearman statistics, for non-censored continuous outcomes, and the Cox score and rank score tests (Jung et al, 2005), for right-censored time-to-event outcomes.

Usage

test.scoregpu()

Examples

test.scoregpu()
Index

*Topic **datagen**
  makeExprSet, 3
*Topic **manip**
  makeExprSet, 3
*Topic **nonparametric**
  permgpu, 4
  scoregpu, 7
  test.permgpu, 9
  test.scoregpu, 10
*Topic **package**
  permGPU-package, 2
*Topic **survival**
  permgpu, 4
  scoregpu, 7
  test.permgpu, 9
  test.scoregpu, 10

ExpressionSet, 3–5
makeExprSet, 3

permGPU, 3
permGPU (permGPU-package), 2
permgpu, 4
permGPU-package, 2

scoregpu, 7

test.permgpu, 9
test.scoregpu, 10