Package ‘GSA’

April 20, 2024

Title  Gene Set Analysis
Version  1.03.3
Author  Brad Efron and R. Tibshirani
Description  Gene Set Analysis.
Maintainer  Rob Tibshirani <tibs@stat.stanford.edu>
Suggests  impute
License  LGPL

URL  https://tibshirani.su.domains/GSA/
Repository  CRAN
Date/Publication  2024-04-20 09:58:26 UTC
NeedsCompilation  no

R topics documented:

GSA .................................................. 2
GSA.correlate ...................................... 5
GSA.func ......................................... 6
GSA.genescores ................................... 9
GSA.listsets ...................................... 11
GSA.make.features ................................. 13
GSA.plot ........................................ 14
GSA.read.gmt ..................................... 16

Index  18
**GSA**

*Gene set analysis*

**Description**

Determines the significance of pre-defined sets of genes with respect to an outcome variable, such as a group indicator, a quantitative variable or a survival time.

**Usage**

```r
GSA(x, y, genesets, genenames, 
    method=c("maxmean", "mean", "absmean"),
    resp.type=c("Quantitative", "Two class unpaired", "Survival", "Multiclass", 
               "Two class paired", "tCorr", "taCorr"),
    censoring.status=NULL, random.seed=NULL, knn.neighbors=10,
    s0=NULL, s0.perc=NULL, minsize=15, maxsize=500,
    restand=TRUE, restand.basis=c("catalog", "data"),
    nperms=200, 
    xl.mode=c("regular", "firsttime", "next20", "lasttime"),
    xl.time=NULL, xl.prevfit=NULL)
```

**Arguments**

- **x**: Data `x`: p by n matrix of features (expression values), one observation per column (missing values allowed); `y`: n-vector of outcome measurements
- **y**: Vector of response values: 1,2 for two class problem, or 1,2,3 ... for multiclass problem, or real numbers for quantitative or survival problems
- **genesets**: Gene set collection (a list)
- **genenames**: Vector of genenames in expression dataset
- **method**: Method for summarizing a gene set: "maxmean" (default), "mean" or "absmean"
- **resp.type**: Problem type: "quantitative" for a continuous parameter; "Two class unpaired" ; "Survival" for censored survival outcome; "Multiclass" : more than 2 groups, coded 1,2,3...; "Two class paired" for paired outcomes, coded -1,1 (first pair), -2,2 (second pair), etc
- **censoring.status**: Vector of censoring status values for survival problems, 1 mean death or failure, 0 means censored
- **random.seed**: Optional initial seed for random number generator (integer)
- **knn.neighbors**: Number of nearest neighbors to use for imputation of missing features values
- **s0**: Exchangeability factor for denominator of test statistic; Default is automatic choice
- **s0.perc**: Percentile of standard deviation values to use for s0; Default is automatic choice; -1 means s0=0 (different from s0.perc=0, meaning s0=zeroeth percentile of standard deviation values= min of sd values)
minsize  Minimum number of genes in genesets to be considered
maxsize  Maximum number of genes in genesets to be considered
restand  Should restandardization be done? Default TRUE
restand.basis  What should be used to do the restandardization? The set of genes in the gene-
sets ("catalog", the default) or the genes in the data set ("data")
nperms  Number of permutations used to estimate false discovery rates
xl.mode  Used by Excel interface
xl.time  Used by Excel interface
xl.prevfit  Used by Excel interface

details  Carries out a Gene set analysis, as described in the paper by Efron and Tibshirani (2006). It differs
from a Gene Set Enrichment Analysis (Subramanian et al 2006) in its use of the "maxmean" statistic:
this is the mean of the positive or negative part of gene scores in the gene set, whichever is large
in absolute values. Efron and Tibshirani shows that this is often more powerful than the modified
KS statistic used in GSEA. GSA also does "restandardization" of the genes (rows), on top of the
permutation of columns (done in GSEA). Gene set analysis is applicable to microarray data and
other data with a large number of features. This is also the R package that is called by the "official"
SAM Excel package v3.0. The format of the response vector y and the calling sequence is illustrated
in the examples below. A more complete description is given in the SAM manual at http://www-
stat.stanford.edu/~tibs/SAM

value  A list with components
GSA.scores  Gene set scores for each gene set
GSA.scores.perm  Matrix of Gene set scores from permutations, one column per permutation
fdr.lo  Estimated false discovery rates for negative gene sets (negative means lower ex-
pression correlates with class 2 in two sample problems, lower expression cor-
relates with increased y for quantitative problems, lower expression correlates
with higher risk for survival problems)
fdr.hi  Estimated false discovery rates for positive gene sets; positive is opposite of
negative, as defined above
pvalues.lo  P-values for negative gene sets
pvalues.hi  P-values for positive gene sets
stand.info  Information from restandardization process
stand.info.star  Information from restandardization process in permutations
ngenes  Number of genes in union of gene sets
nperms  Number of permutations used
gene.scores Individual gene scores (e.g., t-statistics for two class problem)
s0 Computed exchangeability factor
s0.perc Computed percentile of standard deviation values. s0 = s0.perc percentile of the
gene standard deviations
call The call to GSA
x For internal use
y For internal use
genesets For internal use
genenames For internal use
r.obs For internal use
r.star For internal use
gs.mat For internal use
gs.ind For internal use
catalog For internal use
catalog.unique For internal use

Author(s)

Robert Tibshirani

References


Examples

######### two class unpaired comparison
# y must take values 1,2

set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)

dd<-sample(1:1000,size=100)

u<-matrix(2*rnorm(100),ncol=10,nrow=100)

y<-c(rep(1,10),rep(2,10))

genenames=paste("g",1:1000,sep="")
#create some random gene sets
genesets=vector("list",50)
for(i in 1:50){
  genesets[[i]]=paste("g",sample(1:1000,size=30),sep="")
}
geneset.names=paste("set",as.character(1:50),sep="")

GSA.obj<-GSA(x,y, genenames=genenames, genesets=genesets,
  resp.type="Two class unpaired", nperms=100)

GSA.listsets(GSA.obj, geneset.names=geneset.names,FDRcut=.5)

#to use "real" gene set collection, we read it in from a gmt file:
#
# geneset.obj<- GSA.read.gmt("file.gmt")
#
# where file.gmt is a gene set collection from GSEA collection or
# or the website http://www-stat.stanford.edu/~tibs/GSA, or one
# that you have created yourself. Then
#
# GSA.obj<-GSA(x,y, genenames=genenames, genesets=geneset.obj$genesets,
#   #
#   #

GSA.correlate

"Correlates" a gene set collection with a given list of gene names

Description

"Correlates" a gene set collection with a given list of gene names. Gives info on the overlap between the collection and the list of genes

Usage

GSA.correlate(GSA.genesets.obj, genenames)

Arguments

GSA.genesets.obj
  Gene set collection, created for example by GSA.read.gmt
genenames
  Vector of gene names in expression dataset
Details

Gives info on the overlap between a gene set collection and the list of gene names. This is for information purposes, to find out, for example, how many genes in the list of genes appear in the gene set collection.

Author(s)

Robert Tibshirani

References


Examples

```
########## two class unpaired comparison
# y must take values 1,2

set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)
dd<-sample(1:1000,size=100)

u<-matrix(2*rnorm(100),ncol=10,nrow=100)
y<-c(rep(1,10),rep(2,10))

genenames=paste("g",1:1000,sep="")

# create some random gene sets
gensets=vector("list",50)
for(i in 1:50){
  genesets[[i]]=paste("g",sample(1:1000,size=30),sep="")
}
genset.names=paste("set",as.character(1:50),sep="")

GSA.correlate(genesets, genenames)
```
**Description**

Determines the significance of pre-defined sets of genes with respect to an outcome variable, such as a group indicator, quantitative variable or survival time. This is the basic function called by GSA.

**Usage**

```r
GSA.func(x, y, genesets, genenames, geneset.names = NULL, 
method = c("maxmean", "mean", "absmean"), 
resp.type = c("Quantitative", "Two class unpaired", "Survival", "Multiclass", 
"Two class paired", "tCorr", "taCorr"), 
censoring.status = NULL, 
first.time = TRUE, return.gene.ind = TRUE, 
ngenes = NULL, gs.mat = NULL, gs.ind = NULL, 
catalog = NULL, catalog.unique = NULL, 
s0 = NULL, s0.perc = NULL, minsize = 15, maxsize = 500, restand = TRUE, 
restand.basis = c("catalog", "data"))
```

**Arguments**

- **x**: Data x: p by n matrix of features, one observation per column (missing values allowed)
- **y**: Vector of response values: 1,2 for two class problem, or 1,2,3 ... for multiclass problem, or real numbers for quantitative or survival problems
- **genesets**: Gene set collection (a list)
- **genenames**: Vector of genenames in expression dataset
- **geneset.names**: Optional vector of gene set names
- **method**: Method for summarizing a gene set: "maxmean" (default), "mean" or "absmean"
- **resp.type**: Problem type: "quantitative" for a continuous parameter; "Two class unpaired"; "Survival" for censored survival outcome; "Multiclass" : more than 2 groups; "Two class paired" for paired outcomes, coded -1,1 (first pair), -2,2 (second pair), etc
- **censoring.status**: Vector of censoring status values for survival problems, 1 mean death or failure, 0 means censored
- **first.time**: internal use
- **return.gene.ind**: internal use
- **ngenes**: internal use
- **gs.mat**: internal use
- **gs.ind**: internal use
- **catalog**: internal use
- **catalog.unique**: internal use
- **s0**: Exchangeability factor for denominator of test statistic; Default is automatic choice
### Details

Carries out a Gene set analysis, computing the gene set scores. This function does not do any permutations for estimation of false discovery rates. GSA calls this function to estimate FDRs.

### Value

A list with components

- **scores**: Gene set scores for each gene set
- **norm.scores**: Gene set scores transformed by the inverse Gaussian cdf
- **mean**: Means of gene expression values for each sample
- **sd**: Standard deviation of gene expression values for each sample
- **gene.ind**: List indicating which genes in each positive gene set had positive individual scores, and similarly for negative gene sets
- **geneset.names**: Names of the gene sets
- **nperms**: Number of permutations used
- **gene.scores**: Individual gene scores (e.g., t-statistics for two-class problem)
- **s0**: Computed exchangeability factor
- **s0.perc**: Computed percentile of standard deviation values
- **stand.info**: Information computed used in the restandardization process
- **method**: Method used (from call to GSA.func)
- **call**: The call to GSA

### Author(s)

Robert Tibshirani

### References


<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s0.perc</td>
<td>Percentile of standard deviation values to use for s0; default is automatic choice; -1 means s0=0 (different from s0.perc=0, meaning s0=zeroeth percentile of standard deviation values= min of sd values)</td>
</tr>
<tr>
<td>minsize</td>
<td>Minimum number of genes in genesets to be considered</td>
</tr>
<tr>
<td>maxsize</td>
<td>Maximum number of genes in genesets to be considered</td>
</tr>
<tr>
<td>restand</td>
<td>Should restandardization be done? Default TRUE</td>
</tr>
<tr>
<td>restand.basis</td>
<td>What should be used to do the restandardization? The set of genes in the genesets (&quot;catalog&quot;, the default) or the genes in the data set (&quot;data&quot;)</td>
</tr>
</tbody>
</table>
Examples

```
# two class unpaired comparison
# y must take values 1,2

set.seed(100)
x <- matrix(rnorm(1000*20), ncol=20)
dd <- sample(1:1000, size=100)

u <- matrix(2*rnorm(100), ncol=10, nrow=100)
y <- c(rep(1, 10), rep(2, 10))

genenames = paste("g", 1:1000, sep="")

genesets = vector("list", 50)
for(i in 1:50){
genomes[i] = paste("g", sample(1:1000, size=30), sep="")
}
geneset.names = paste("set", as.character(1:50), sep="")

GSA.func.obj <- GSA.func(x, y, genenames=genenames, genesets=genesets, resp.type="Two class unpaired")

# to use "real" gene set collection, we read it in from a gmt file:
#
# geneset.obj <- GSA.read.gmt("file.gmt")
#
# where file.gmt is a gene set collection from GSEA collection or
# or the website http://www-stat.stanford.edu/~tibs/GSA, or one
# that you have created yourself. Then

# GSA.func.obj <- GSA.func(x, y, genenames=genenames,
# genesets=geneset.obj$genesets,
# resp.type="Two class unpaired")
```

---

**GSA.genescores**

**Individual gene scores from a gene set analysis**

**Description**

Compute individual gene scores from a gene set analysis
Usage

GSA.genescores(geneset.number, genesets, GSA.obj, genenames, negfirst=FALSE)

Arguments

geneset.number  Number indicating which gene set is to examined
genesets        The gene set collection
GSA.obj         Object returned by function GSA
genenames        Vector of gene names for gene in expression dataset
negfirst         Should negative genes be listed first? Default FALSE

Details

Compute individual gene scores from a gene set analysis. Useful for looking “inside” a gene set that has been called significant by GSA.

Value

A list with components

res             Matrix of gene names and gene scores (eg t-statistics) for each gene in the gene set

Author(s)

Robert Tibshirani

References


Examples

########### two class unpaired comparison
# y must take values 1,2
set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)
dd<-sample(1:1000,size=100)

u<-matrix(2*rnorm(100),ncol=10,nrow=100)


y<-c(rep(1,10),rep(2,10))

genenames=paste("g",1:1000,sep="")
# create some random gene sets
genesets=vector("list",50)
for(i in 1:50){
  genesets[[i]]=paste("g",sample(1:1000,size=30),sep="")
}
geneset.names=paste("set",as.character(1:50),sep="")

GSA.obj<-GSA(x,y, genenames=genenames, genesets=genesets, resp.type="Two class unpaired", nperms=100)

# look at 10th gene set
GSA.genescores(10, genesets, GSA.obj, genenames)

GSA.listsets

List the results from a Gene set analysis

Description
List the results from a call to GSA (Gene set analysis)

Usage
GSA.listsets(GSA.obj, geneset.names = NULL, maxchar = 20, FDRcut = 0.2)

Arguments
GSA.obj Object returned by GSA function
geneset.names Optional vector of names for the gene sets
maxchar Maximum number of characters in printed output
FDRcut False discovery rate cutpoint for listed sets. A value of 1 will cause all sets to be listed

Details
This function list the significant gene sets, based on a call to the GSA (Gene set analysis) function.
Value

A list with components

- **FDRcut**: The false discovery rate threshold used.
- **negative**: A table of the negative gene sets. "Negative" means that lower expression of most genes in the gene set correlates with higher values of the phenotype y. Eg for two classes coded 1,2, lower expression correlates with class 2. For survival data, lower expression correlates with higher risk, i.e shorter survival (Be careful, this can be confusing!)
- **positive**: A table of the positive gene sets. "Positive" means that higher expression of most genes in the gene set correlates with higher values of the phenotype y. See "negative" above for more info.
- **nsets.neg**: Number of negative gene sets
- **nsets.pos**: Number of positive gene sets

Author(s)

Robert Tibshirani

References


Examples

```
######### two class unpaired comparison
# y must take values 1,2

set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)
dd<-sample(1:1000,size=100)
u<-matrix(2*rnorm(100),ncol=10,nrow=100)
y<-c(rep(1,10),rep(2,10))
genenames=paste("g",1:1000,sep="")

# create some random gene sets
genesets=vector("list",50)
for(i in 1:50){
genetiesets[[i]]=paste("g",sample(1:1000,size=30),sep="")
}
genenet.names=paste("set",as.character(1:50),sep="")

GSA.obj<-GSA(x,y, genenames=genenames, genesets=genesets,
resp.type="Two class unpaired", nperms=100)
```
GSA.listsets(GSA.obj, geneset.names=geneset.names,FDRcut=.5)

GSA.make.features

Description

Creates features from a GSA analysis that can be used in other procedures, for example, sample classification.

Usage

GSA.make.features(GSA.func.obj, x, genesets, genenames)

Arguments

GSA.func.obj Object returned by GSA.func
x Expression dataset from which the features are to be created
genesets Gene set collection
genenames Vector of gene names in expression dataset

Details

Creates features from a GSA analysis that can be used in other procedures, for example, sample classification. For example, suppose the GSA analysis computes a maxmean score for gene set 1 that is positive, based on the mean of the positive part of the scores in that gene set. Call the subset of genes with positive scores "A". Then we compute a new feature for this geneset, for each sample, by computing the mean of the scores for genes in A, setting other gene scores to zero.

Author(s)

Robert Tibshirani

References

Examples

```
# two class unpaired comparison
# y must take values 1,2

set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)
dd<-sample(1:1000,size=100)

u<-matrix(2*rnorm(100),ncol=10,nrow=100)
y<-c(rep(1,10),rep(2,10))

genenames=paste("g",1:1000,sep="")

# create some random gene sets
genesets=vector("list",50)
for(i in 1:50){
  genesets[[i]]=paste("g",sample(1:1000,size=30),sep="")
}
geneset.names=paste("set",as.character(1:50),sep="")

GSA.func.obj<GSA.func(x,y, genenames=genenames, genesets=genesets, resp.type="Two class unpaired")

GSA.func.obj$GSA.make.features(GSA.func.obj, x, genesets, genenames)
```

---

**GSA.plot**

*Plot the results from a Gene set analysis*

**Description**

Plots the results from a call to GSA (Gene set analysis)

**Usage**

```
GSA.plot(GSA.obj, fac=1, FDRcut = 1)
```

**Arguments**

- `GSA.obj` Object returned by GSA function
fac  
value for jittering points in plot ("factor" in called to jitter())

FDRcut  
False discovery rate cutpoint for sets to be plotted. A value of 1 (the default) will cause all sets to be plotted

Details
This function makes a plot of the significant gene sets, based on a call to the GSA (Gene set analysis) function.

Author(s)
Robert Tibshirani

References

Examples

########## two class unpaired comparison
# y must take values 1,2

set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)
dd<-sample(1:1000,size=100)

u<-matrix(2*rnorm(100),ncol=10,nrow=100)
y<-c(rep(1,10),rep(2,10))

# create some random gene sets
genenames=paste("g",1:1000,sep="")

genomesets=vector("list",50)
for(i in 1:50){
genomesets[[i]]=paste("g",sample(1:1000,size=30),sep="")
}
genet.names=paste("set",as.character(1:50),sep="")

GSA.obj<-GSA(x,y, genenames=genenames, genomesets=genomesets, resp.type="Two class unpaired", nperms=100)

GSA.listsets(GSA.obj, genenet.names=genenet.names,FDRcut=.5)

GSA.plot(GSA.obj)
**GSA.read.gmt**

*Read in a gene set collection from a .gmt file*

**Description**

Read in a gene set collection from a .gmt file

**Usage**

GSA.read.gmt(filename)

**Arguments**

- **filename** The name of a file to read data values from. Should be a tab-separated text file, with one row per gene set. Column 1 has gene set names (identifiers), column 2 has gene set descriptions, remaining columns are gene ids for genes in that geneset.

**Details**

This function reads in a geneset collection from a .gmt text file, and creates an R object that can be used as input into GSA. We use UniGene symbols for our gene set names in our .gmt files and expression datasets, to match the two. However the user is free to use other identifiers, as long as the same ones are used in the gene set collections and expression datasets.

**Value**

A list with components

- **genesets** List of gene names (identifiers) in each gene set

- **geneset.names** Vector of gene set names (identifiers)

- **geneset.descriptions** Vector of gene set descriptions

**Author(s)**

Robert Tibshirani

**References**

Examples

# read in functional pathways gene set file from Broad institute GSEA website
# http://www.broad.mit.edu/gsea/msigdb/msigdb_index.html
# You have to register first and then download the file C2.gmt from
# their site

#GSA.read.gmt(C2.gmt)
Index

* **nonparametric**
  GSA, 2
  GSA.correlate, 5
  GSA.func, 6
  GSA.genescores, 9
  GSA.listsets, 11
  GSA.plot, 14
  GSA.read.gmt, 16

* **survival**
  GSA, 2
  GSA.correlate, 5
  GSA.func, 6
  GSA.genescores, 9
  GSA.listsets, 11
  GSA.make.features, 13
  GSA.plot, 14
  GSA.read.gmt, 16

* **ts**
  GSA, 2
  GSA.correlate, 5
  GSA.func, 6
  GSA.genescores, 9
  GSA.listsets, 11
  GSA.make.features, 13
  GSA.plot, 14
  GSA.read.gmt, 16

* **univar**
  GSA, 2
  GSA.correlate, 5
  GSA.func, 6
  GSA.genescores, 9
  GSA.listsets, 11
  GSA.make.features, 13
  GSA.plot, 14
  GSA.read.gmt, 16

GSA, 2
GSA.correlate, 5
GSA.func, 6
GSA.genescores, 9