Package ‘RTopper’

May 27, 2024

Type Package
Title This package is designed to perform Gene Set Analysis across multiple genomic platforms
Version 1.50.0
Date 2011-09-14
Author Luigi Marchionni <marchion@jhu.edu>, Svitlana Tyekucheva <svitlana@jimmy.harvard.edu>
Maintainer Luigi Marchionni <marchion@jhu.edu>
Depends R (>= 2.12.0), Biobase
Imports limma, multtest
Suggests org.Hs.eg.db, KEGGREST, GO.db
Description the RTopper package is designed to perform and integrate gene set enrichment results across multiple genomic platforms.
License GPL (>= 3) + file LICENSE
biocViews Microarray
LazyLoad yes
git_url https://git.bioconductor.org/packages/RTopper
git_branch RELEASE_3_19
git_last_commit d6e6de3
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-05-26

Contents

RTopper-package .................................................. 2
adjustPvalGSE ..................................................... 3
combineGSE ........................................................ 4
computeDrStat ...................................................... 5
convertToDr ........................................................ 6
**RTopper-package**

Description

Gene sets analysis considers whether genes sharing a biological property also behave in a related way in experimental data. This technique is commonly used in high throughput genomic analyses to assist results interpretation, and has been successfully applied in cancer genome projects for integrating information from multiple genome-wide assays. The RTopper package uses gene sets analysis to overcome the diversity of genomic data providing the statistical framework for integration across data types.

Details

- **Package:** RTopper
- **Type:** Package
- **Version:** 0.1
- **Date:** 2010-11-12
- **License:** GPL version 3 or newer

**RTopper package features**

Rtopper enables two gene set-based data integration approaches:

1. Integration+GSA: computing integrated gene-to-phenotype association scores, followed by conventional gene sets analysis;
2. GSA+Integration: computing consensus significance score after all data types are analyzed individually;
3. Use of alternative enrichment test: RTopper uses the Wilcoxon rank-sum test for enrichment testing, however alternative tests can be defined and used;
4. Multiple testing correction: RTopper enables adjustment of p-values obtained from enrichment analysis;
adjustPvalGSE

Author(s)
Luigi Marchionni <marchion@jhu.edu>

References

---

Description
This function is an interface to the `mt.rawp2adjp` function contained in the `multtest` package. `adjustPvalGSE` works on outputs from `runBatchGSE` and `combineGSE` returning adjusted p-values.

Usage
adjustPvalGSE(gseOut, proc = "BH", alpha = 0.05, na.rm = FALSE)

Arguments
gseOut list of lists, either the output from `runBatchGSE`, or the output from `combineGSE`
proc character, the method to be used for p-values adjusting. This parameter will be passed to the `mt.rawp2adjp` function from the `multtest` package. The available options include: "Bonferroni", "Holm", "Hochberg", "SidakSS", code"SidakSD", "BH" (the default), "BY", "ABH", and "TSBH"
alpha numeric, the nominal type I error rate
na.rm logical, the option for handling NA values in the list of raw p-values

Details
The `adjustPvalGSE` function performs p-value adjusting for multiple testing correction on the list of lists resulting from enrichment analysis obtained using the `runBatchGSE` and `combineGSE` functions. This function is based on the `mt.rawp2adjp` function contained in the `multtest` package.

Value
For each vector of p-value contained in the `gseOut` input object a data.frame is returned, containing original p-value and corrected p-values

Author(s)
Luigi Marchionni <marchion@jh.edu>
**References**


**Examples**

```r
### load gse analysis results for separate gene-to-phenotype score
data(gseResultsSep)

### adjust for multiple testing using the Benjamini and Hochberg method
gseABS.int.BH <- adjustPvalGSE(gseResultsSep)

### adjust for multiple testing using the Holm method
gseABS.int.holm <- adjustPvalGSE(gseResultsSep, proc = "Holm")
```

---

**combineGSE**

*Combines GSE results*

**Description**

Combines GSE results obtained from a separate set of gene-to-phenotypes scores

**Usage**

```r
combineGSE(gseOut, method)
```

**Arguments**

- `gseOut`:
  - a list of lists containing the enrichment results to be combined. This is usually the output of `runBatchGSE` obtained from a set of distinct genes-to-phenotype scores (usually one per genomic platform)

- `method`:
  - character, this argument specifies the method used to combine the enrichment results obtained from distinct genes-to-phenotype scores (usually one per genomic platform). Available options are the computation of the geometric or arithmetic means, the use of the median, the selection of the minimum or the maximum enrichment score, and the random selection of a score (respectively "geometricMean", "mean", "median", "min", "max", and "random")

**Details**

This function summarize enrichment results obtained from distinct gene-to-phenotypes scores (usually one per genomic platform) by one of several alternative methods.

**Value**

The output is a list of lists containing integrated enrichment results for all FGS collections
**computeDrStat**

**Author(s)**

Luigi Marchionni <marchion@jhu.edu>

**References**


**Examples**

```r
###load gse analysis results for separate gene-to-phenotype score
data(gseResultsSep)

###combine enrichment score results using geometric mean
gseABS.sep.geoMean <- combineGSE(gseResultsSep, method="geometricMean")

###combine enrichment score results using maximum value
gseABS.sep.max <- combineGSE(gseResultsSep, method="max")
```

---

**computeDrStat**  Computes gene-to-phenotype associations scores

**Description**

`computeDrStat` Computes gene-to-phenotype associations scores, using as input the output from `convertToDr`.

**Usage**

```r
computeDrStat(data, columns = c(1:(ncol(data)-1)), method = "dev", integrate = TRUE)
```

**Arguments**

- `data` a list of data.frames containing genomic measurements. Each element of `dataIntersection` must account for the same set of patients(columns) and genes (rows)
- `columns` a data.frame indicating patients’ phenotypic class
- `method` character, the number of genomic platforms
- `integrate` logical, wheter to integrate the gene-to-phenotype scores across platform or return separates scores for each platform
Details

This function allows computing gene-to-phenotype association scores, using as input the gene-centered list produced by `convertToDr`. The `computeDrStat` function works separately on each gene-centered data.frame created by the `convertToDr` function, assuming that the phenotype information is stored in the last column named "response". It is possible computing both separate association scores for each platform, as well as an integrated score, as specified by the `integrate` arguments. There are currently three methods available for obtaining the scores (see Tyekucheva et al, manuscript under review), as specified by the methods argument:

- "dev": this approach computes the score as the difference of deviances;
- "aic": this approach computes the score as the Akaike information criterion for model selection;
- "bic": this approach computes the score as the penalized likelihood ratio;

Value

A list of named vectors containing separate or integrated gene-to-phenotype association scores.

Author(s)

Luigi Marchionni <marchion@jhu.edu>

References


Examples

```r
###load data
data(exampleData)

###convert
dataDr <- convertToDr(dat, pheno, 4)

###compute the integrated score
bicStatIntegrated <- computeDrStat(dataDr, columns = c(1:4), method="bic", integrate = TRUE)

###compute separate scores for each genomic platform
bicStatSeparate <- computeDrStat(dataDr, columns = c(1:4), method="bic", integrate = FALSE)
```

`convertToDr` Converts genomic data to a list suitable for computing gene-to-phenotype scores

Description

`convertToDr` converts genomic data into a list further used for computing gene-to-phenotype association scores.
**Usage**

```r
cvtColorDr(dataIntersection, response, nPlatforms = length(data))
```

**Arguments**

- `dataIntersection`  
  a list of data.frames containing genomic measurements. Each element of `dataIntersection` must account for the same set of patients(columns) and genes (rows)

- `response`  
  a data.frame indicating patients’ phenotypic class

- `nPlatforms`  
  numeric, the number of genomic platforms

**Details**

This function coverts a list of data.frames containing distinct genomic measurements performed on the same patients into a gene-centered used in further analyses for computing gene-to-phenotype scores. Data.frame in the input list (dataIntersection) must have the same dimensions, with columns being patients, and rows being genes. Column names identify the patients, while rownames identify the genes. The argument response is used to pass phenotypic information about samples to be analyzed. This is a simple two columns data.frame in which the first column correspond to patients identifiers, and the second column to the phenotypic response encoded as binary class (using the integers 0 and 1). The nPlatforms argument specifies the number of platforms that will be analyzed.

**Value**

A list of data.frames, one for each analyzed gene, summarizing all genomic measurements and phenotipic information across patients and platforms.

**Author(s)**

Luigi Marchionni <marchion@jhu.edu>

**References**

Svitlana Tyekucheva, Luigi Marchionni, Rachel Karchin, and Giovanni Parmigiani "Integrating diverse genomic data using gene sets." Manuscript submitted.

**Examples**

```r
###load data
data(exampleData)

###convert
dataDr <- convertToDr(dat, pheno, 4)
```
A test dataset for the RTopper package

Description

A small subset of pre-processed Glioblasoma Multiforme (GBM) genomic data from The Cancer Genome Atlas (TCGA) project, encompassing Differential Gene Expression (DGE), and Copy Number Variation (CNV). Can be used as input to convertToDr.

Usage

data(exampleData)

Format

This object is a list of 4 data.frames containing genomic measurements obtained across distinct genomic scopes (copy number variation and gene expression), platforms (Affymetrix and Agiles), and laboratories. In particular each data.frame consist of 500 gene measurements (by rows), for 95 distinct patients (by columns) from the following 4 distinct platforms:

"dat.affy": DGE obtained using Affymetrix microarrays;
"dat.agilent": DGE obtained using Agilent microarrays;
"dat.cnvHarvard": CNV data obtained at Harvard;
"dat.cnvMskcc": CNV data obtained at Memorial Sloan Kettering Cancer Center;

Source


References


Examples

data(exampleData)
ls()
class(dat)
names(dat)
sapply(dat,class)
sapply(dat,dim)
**fgsList**

A list of Functional Gene Set (FGS) to be used to run the examples in the RTopper package

**Description**

A list containing distinct types of FGS (i.e. Gene Ontology, KEGG pathways). Each FGS is type is a list of named character vectors, one for each FGS, containing the gene identifiers. Vectors names describe the FGS. Can be used as input to runBatchGSE.

**Usage**

```r
data(fgsList)
```

**Format**

This object is a list of length two:

- "go": this is a list of 5 character vectors, corresponding to 5 distinct Gene Ontology (GO) terms. Genes annotated to each GO term are identified by their gene symbol;
- "kegg": this is a list of 5 character vectors, corresponding to 5 distinct KEGG pathways. Genes annotated to each KEGG pathway are identified by their gene symbol;

**Source**

The FGS were obtained from the org.Hs.eg.db package, (use the org.Hs.eg function to see the content); These FGS were annotated using data from GO.db and KEGGREST packages (use the GO and keggGet functions to see the content).

**Examples**

```r
data(fgsList)
class(fgsList)
names(fgsList)
str(fgsList)
```

---

**gseResultsSep**

A list of separated gene set enrichment p-values to be used to run the examples in the RTopper package

**Description**

A list containing distinct named numeric vectors corresponding to the gene set enrichment p-value separately computed with runBatchGSE for each distinct data set. Note that for each data set there are two set of p-values, one for GO and one for KEGG. These separate p-values can be combined across data sets by the combineGSE function. Can be used as input to adjustPvalGSE.
Usage

data(gseResultsSep)

Format

This object is a list of length four:

"dat.affy": a list of length two: "go" is a numeric vector of length 5, containing the p-values resulting from gene set enrichment analysis of 5 GO terms on Affymetrix gene expression data; "keg" is a numeric vector of length 5, containing the p-values resulting from gene set enrichment analysis of 5 KEGG pathways on Affymetrix gene expression data;

"dat.agilent": a list of length two: "go" is a numeric vector of length 5, containing the p-values resulting from gene set enrichment analysis of 5 GO terms on Agilent gene expression data; "keg" is a numeric vector of length 5, containing the p-values resulting from gene set enrichment analysis of 5 KEGG pathways on Agilent gene expression data;

"dat.cnvHarvard": a list of length two: "go" is a numeric vector of length 5, containing the p-values resulting from gene set enrichment analysis of 5 GO terms on Harvard CNV data; "keg" is a numeric vector of length 5, containing the p-values resulting from gene set enrichment analysis of 5 KEGG pathways on Harvard CNV data;

"dat.cnvMskcc": a list of length two: "go" is a numeric vector of length 5, containing the p-values resulting from gene set enrichment analysis of 5 GO terms on MSKCC CNV data; "keg" is a numeric vector of length 5, containing the p-values resulting from gene set enrichment analysis of 5 KEGG pathways on MSKCC CNV data;

Source

Computed using the runBatchGSE function from the TCGA data contained in sepScores and fgsList.

Examples

data(gseResultsSep)  
class(gseResultsSep)  
names(gseResultsSep)  
str(gseResultsSep)

---

intScores  
A list of genomic scores integrated across distinct data sets to be used to run the examples in the RTopper package

Description

A list containing a named numeric vector corresponding to the genomic score resulting from the integration across distinct data set. These integrated gene-to-phenotype scores are computed by the computeDrStat function. Can be used as input to runBatchGSE.
**Usage**

```r
data(intScores)
```

**Format**

This object is a list of length one:

- "integrated": a numeric vector of length 500, corresponding to the integrated phenotype association scores computed for each of the 500 genes used in the examples;

**Source**

Computed using the `computeDrStat` function from the TCGA data contained in `dat` and `pheno`.

**Examples**

```r
data(intScores)
class(intScores)
names(intScores)
str(intScores)
```

---

**Description**

A data.frame with 2 columns containing the phenotypic class indicator for the 95 patients analyzed and used in the examples. Can be used as input to `convertToDr`.

**Usage**

```r
data(exampleData)
```

**Format**

This object is a data.frame with two columns:

- "Sample": the first column contains the patients identifiers;
- "Class": the second columns contain a numeric indicator (0 or 1) corresponding to the phenotypic class of each patient;

**Source**


**References**

runBatchGSE

To perform GSE analysis over multiple experiments and functional themes

Description

The runBatchGSE function enables performing Gene Set Enrichment analysis over multiple ranking statistics and multiple lists of gene sets. By default this function is an interface to the geneSetTest in the limma package, and most of the arguments passed to runBatchGSE are indeed passed to such lower level function. As an alternative the user can also define and pass to runBatchGSE a custom function, defining the ranking statistics and the gene set membership in the same way done for geneSetTest (see Details below).

Usage

runBatchGSE(dataList, fgsList, ...)

Arguments

dataList: a list containing the gene-to-phenotype scores to be used as ranking statistics in the GSE analysis. This list is usually produced by running computeDrStat

fgsList: a list of FGS collection, in which each element is a list of character vectors, one for each gene set

...: additional arguments to be passed to lower level functions (see details below)

Details

This function performs enrichment analysis for all the gene-to-phenotype scores (argument dataList) passed to it over a list of Functional Gene Set (FGS) (argument fgsList), returning a p-value for each FGS. Additional arguments can be bassed to this function to modify the way the enrichment test is performed, as follows:

absolute: logical, this specifies whether the absolute values of the ranking statistics should be used in the test (the default being TRUE)

gseFunc: a function to perform GSE analysis. If not specified the default is the geneSetTest function from the limma package. If a function is specified by the user, the membership of the analyzed genes to a FGS, and the ranking statistics must be defined in the same way this is done for geneSetTest, and the new function must return an integer (usually a p-value) (see the help for geneSetTest)

The following main arguments are used by geneSetTest:
runBatchGSE

type character, specifies the type of statistics used to rank the genes by geneSetTest: 'f' for F-like statistics (default), 't' for t-like statistics, or 'auto' for an educated guess
alternative character, defines the alternative with the following possible options: 'mixed' (default), 'either', 'up' or 'down', 'two.sided', 'greater', or 'less'
ranks.only logical, if TRUE (default) only ranks will be used by geneSetTest
nsim numeric, the number of randomly selected sets of genes to be used in simulations to compute the p-value

Value

The output is a list of lists containing the set of enrichment results for all gene-to-phenotype scores and FGS collections used as input.

Author(s)

Luigi Marchionni <marchion@jhu.edu>

References


Examples

###require limma to run the example
require(limma)

###load integrated gene-to-phenotype scores
data(intScores)

###load separate gene-to-phenotype scores
data(sepScores)

###load list of functional gene sets
data(fgsList)

###run GSE analysis in batch with default parameters
gseABS.int <- runBatchGSE(dataList=intScores, fgsList=fgsList)

###run GSE analysis in batch with alternative parameters
gseABS.sep <- runBatchGSE(dataList=sepScores, fgsList=fgsList, absolute=FALSE, type="t", alternative="up")

###run GSE analysis in batch passing an enrichment function
gseUP.int.2 <- runBatchGSE(dataList=intScores, fgsList=fgsList, absolute=FALSE, gseFunc=wilcoxGST, alternative="up")

###define and use a new enrichment function
gseFunc <- function (selected, statistics, threshold) {
  diffExpGenes <- statistics > threshold
tab <- table(diffExpGenes, selected)
```r
pVal <- fisher.test(tab)["p.value"]
}
gseUP.sep.2 <- runBatchGSE(dataList=sepScores, fgsList=fgsList,
  absolute=FALSE, gseFunc=gseFunc, threshold=7.5)
```

---

**sepScores**

A list of separate gene-to-phenotype association scores, obtained independently for each distinct data set to be used to run the examples in the RTopper package.

---

**Description**

A list containing distinct named numeric vectors corresponding to the gene-to-phenotype association scores resulting from the separate analysis of each data set. These separate gene-to-phenotype scores are computed by `computeDrStat` function. Can be used as input to `runBatchGSE`.

**Usage**

data(sepScores)

**Format**

This object is a list of length four, one element for data set:

- "dat.affy": a numeric vector of length 500, corresponding to the separate phenotype association scores computed for Affymetrix gene expression data;
- "dat.agilent": a numeric vector of length 500, corresponding to the separate phenotype association scores computed for Agilent gene expression data;
- "dat.cnvHarvard": a numeric vector of length 500, corresponding to the separate phenotype association scores computed for Harvard CNV data;
- "dat.cnvMskcc": a numeric vector of length 500, corresponding to the separate phenotype association scores computed for the MSKCC CNV data;

**Source**

Computed using the `computeDrStat` function from the TCGA data contained in `data`.

**Examples**

data(sepScores)
class(sepScores)
names(sepScores)
str(sepScores)
Index

* datasets
  dat, 8
  fgsList, 9
  gseResultsSep, 9
  intScores, 10
  pheno, 11
  sepScores, 14
* manip
  adjustPvalGSE, 3
  combineGSE, 4
  computeDrStat, 5
  convertToDr, 6
  runBatchGSE, 12
* package
  RTopper-package, 2
adjustPvalGSE, 3, 9
combineGSE, 3, 4, 9
calculateDrStat, 5, 10, 11, 14
convertToDr, 6, 8, 11
dat, 8, 11
data, 14
fgsList, 9, 10
geneSetTest, 12
GO, 9
gseResultsSep, 9
intScores, 10
keggGet, 9
mt.raw2adjp, 3
org.Hs.eg, 9
pheno, 11, 11
RTopper (RTopper-package), 2
runBatchGSE, 3, 4, 9, 10, 12, 14
sepScores, 10, 14