Package ‘oppar’

April 8, 2026

Type Package

Title Outlier profile and pathway analysis in R

Version 1.39.0

Date 2016

Description The R implementation of mCOPA package published by Wang et al. (2012). Oppar provides methods for Cancer Outlier profile Analysis. Although initially developed to detect outlier genes in cancer studies, methods presented in oppar can be used for outlier profile analysis in general. In addition, tools are provided for gene set enrichment and pathway analysis.

biocViews Pathways, GeneSetEnrichment, SystemsBiology, GeneExpression, Software

Depends R (>= 3.3)

Imports Biobase, methods, GSEABase, GSVA

License GPL-2

LazyData true

Collate 'class_OPPARList.R' 'class_nuchar.R' 'getSampleOutlier.R' 'getSubtypeProbes.R' 'gsva.R' 'opa-generic-methods.R' 'oppar.R' 'utils.R' 'zzz.R'

RoxygenNote 5.0.1

Suggests knitr, rmarkdown, limma, org.Hs.eg.db, GO.db, snow, parallel

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/oppar

git_branch devel

git_last_commit 168fc6d

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2026-04-07

Author Chenwei Wang [aut],
Alperen Taciroglu [aut],
Stefan R Maetschke [aut],

Colleen C Nelson [aut],
Mark Ragan [aut],
Melissa Davis [aut],
Soroor Hediyeh zadeh [cre],
Momeneh Foroutan [ctr]

Maintainer Soroor Hediyeh zadeh <hediyehzadeh.s@wehi.edu.au>

Contents

`bcm` *Breast cancer metastases from different anatomical sites*

Description

An ExpressionSet object containing trimmed GSE46141 data. The object contains gene expression measurements on local breast tumours and liver, lymph node, skin local- regional etc metastatic tumours. Contains 9503 features and 80 samples (ascite, bone, lung and skin samples were removed).

Usage

```
data(GSE46141)
```

Format

Contains gene expression matrix, phenotype (pData) and feature (fData) data:

ID In fData(e) – probe IDs

EntrezGeneID In fData(e) – Entrez Ids

GeneSymbol In fData(e) – Gene Symbols

geo_accession In pData(e) – GEO accession IDs

source_name_ch1 In pData(e) – Sample information

Value

An ExpressionSet object

Source

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE46141

`eset` *Tomlins et al. Prostate Cancer data (GEO: GSE6099)*

Description

An ExpressionSet object containing microarray gene expression measurements on normal tissue and metastatic prostate cancer tumoures, and the corresponsing feature and phynotypic meta data.

Usage

```
data(tomlins)
```

Format

An ExpressionSet object with 10945 features and 86 samples.

title In pData(eset) – Sample names

geo_accession In pData(eset) –GEO accession numbers

characteristics_ch1 In pData(eset) – sample description

ID In fData(eset) – probes Ids

Gene.title ...

Gene.symbol In fData(eset) – Gene Symbol

Gene.ID In fData(eset) – Entrez Gene IDs

Value

An ExpressionSet object

Source

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE6099 getSampleOutlier *Retrieving outlier genes in samples*

Description

Returns a list of outlier genes for each given sample

Usage

```
getSampleOutlier(profileMatrix, sample.name)

## S4 method for signature 'matrix,nuchar'
getSampleOutlier(profileMatrix, sample.name)

## S4 method for signature 'OPPARList,nuchar'
getSampleOutlier(profileMatrix, sample.name)
```

Arguments

profileMatrix	A matrix of 0, -1 and 1 representing outlier genes in samples. Also an object of type OPPARList.
sample.name	A character vector containing one or more sample names, or a numeric vector containing sample indices.

Value

A list of lists. For each given sample, the fuction return up-regulated and down-regulated outlier genes.

Methods (by class)

- profileMatrix = matrix,sample.name = nuchar: A method for getSampleOutlier
- profileMatrix = OPPARList,sample.name = nuchar: A method for getSampleOutlier

Examples

```
data(GSE46141)
library(Biobase)
group <- sapply(pData(bcm)$source_name_ch1, function(x){ ifelse(x == "breast",0,1)})
group <- factor(group)
bcm.opa <- opa(bcm,group=group)
# Outlier profile for sample "GSM1124929"
getSampleOutlier(bcm.opa, "GSM1124929")

# Also can use sample index, instead of sample name
getSampleOutlier(bcm.opa, 11)

# A vector of sample names/indices are accepted as well
```

```
getSampleOutlier(bcm.opa, c(1,2))
getSampleOutlier(bcm.opa, c("GSM1124929","GSM1124941"))
```

---

getSubtypeProbes *Retrieving outlier genes from a group of related samples*

Description

Returns a list of genes that are outlier in a group of samples, such as samples from the same subtype.

Usage

```
getSubtypeProbes(profileMatrix, sample.names)

## S4 method for signature 'matrix,nuchar'
getSubtypeProbes(profileMatrix, sample.names)

## S4 method for signature 'OPPARList,nuchar'
getSubtypeProbes(profileMatrix, sample.names)
```

Arguments

profileMatrix	A matrix of 0,1 and -1, representing outlier genes in samples. Also an object of type OPPARList.
sample.names	A character vector containing sample names, or a numeric vector containing the indices of the samples.

Value

A list of lists. The sub-lists are up-regulated outlier genes, and down-regulated outlier genes.

Methods (by class)

- profileMatrix = matrix,sample.names = nuchar: A method for getSubtypeProbes with signature profileMatrix = matrix and sample.names = nuchar
- profileMatrix = OPPARList,sample.names = nuchar: A method for getSubtypeProbes with signature profileMatrix = OPPARList and sample.names = nuchar

Examples

```
data(GSE46141)
library(Biobase)
group <- sapply(pData(bcm)$source_name_ch1, function(x){ ifelse(x == "breast",0,1)})
group <- factor(group)
bcm.opa <- opa(bcm,group=group)
# extracting liver samples
index <- which(pData(bcm)$source_name_ch1 == "liver")
samples <- rownames(pData(bcm)[index,])
```

```
samples <- match(samples, colnames(bcm.opa$profileMatrix))
samples <- Reduce(c,samples)
# liver subtype outlier profile
liver.subtype.outlier <- getSubtypeProbes(bcm.opa, samples)
```

---

gsva *gsva*

Description

Gene Set Variation Analysis

Usage

```
gsva(expr, gset.idx.list, ...)

## S4 method for signature 'ExpressionSet,list'
gsva(expr, gset.idx.list, annotation,
  method = c("gsva", "ssgsea", "zscore", "plage"), rnaseq = FALSE,
  abs.ranking = FALSE, min.sz = 1, max.sz = Inf, no.bootstraps = 0,
  bootstrap.percent = 0.632, parallel.sz = 0, parallel.type = "SOCK",
  mx.diff = TRUE, tau = switch(method, gsva = 1, ssgsea = 0.25, NA),
  kernel = TRUE, ssgsea.norm = TRUE, verbose = TRUE,
  is.gset.list.up.down = FALSE)

## S4 method for signature 'ExpressionSet,GeneSetCollection'
gsva(expr, gset.idx.list,
  annotation, method = c("gsva", "ssgsea", "zscore", "plage"),
  rnaseq = FALSE, abs.ranking = FALSE, min.sz = 1, max.sz = Inf,
  no.bootstraps = 0, bootstrap.percent = 0.632, parallel.sz = 0,
  parallel.type = "SOCK", mx.diff = TRUE, tau = switch(method, gsva = 1,
  ssgsea = 0.25, NA), kernel = TRUE, ssgsea.norm = TRUE, verbose = TRUE,
  is.gset.list.up.down = FALSE)

## S4 method for signature 'matrix,GeneSetCollection'
gsva(expr, gset.idx.list, annotation,
  method = c("gsva", "ssgsea", "zscore", "plage"), rnaseq = FALSE,
  abs.ranking = FALSE, min.sz = 1, max.sz = Inf, no.bootstraps = 0,
  bootstrap.percent = 0.632, parallel.sz = 0, parallel.type = "SOCK",
  mx.diff = TRUE, tau = switch(method, gsva = 1, ssgsea = 0.25, NA),
  kernel = TRUE, ssgsea.norm = TRUE, verbose = TRUE,
  is.gset.list.up.down = FALSE)

## S4 method for signature 'matrix,list'
gsva(expr, gset.idx.list, annotation,
  method = c("gsva", "ssgsea", "zscore", "plage"), rnaseq = FALSE,
  abs.ranking = FALSE, min.sz = 1, max.sz = Inf, no.bootstraps = 0,
```

```
bootstrap.percent = 0.632, parallel.sz = 0, parallel.type = "SOCK",
mx.diff = TRUE, tau = switch(method, gsva = 1, ssgsea = 0.25, NA),
kernel = TRUE, ssgsea.norm = TRUE, verbose = TRUE,
is.gset.list.up.down = FALSE)
```

Arguments

`expr`	Gene expression data which can be given either as an `ExpressionSet` object or as a matrix of expression values where rows correspond to genes and columns correspond to samples.
`gset.idx.list`	Gene sets provided either as a `list` object or as a `GeneSetCollection` object.
`...`	other optional arguments.
`annotation`	In the case of calling `gsva()` with expression data in a `matrix` and gene sets as a `GeneSetCollection` object, the `annotation` argument can be used to supply the name of the Bioconductor package that contains annotations for the class of gene identifiers occurring in the row names of the expression data matrix. By default `gsva()` will try to match the identifiers in `expr` to the identifiers in `gset.idx.list` just as they are, unless the `annotation` argument is set.
`method`	Method to employ in the estimation of gene-set enrichment scores per sample. By default this is set to `gsva` (Hanzelmann et al, 2013) and other options are `ssgsea` (Barbie et al, 2009), `zscore` (Lee et al, 2008) or `plage` (Tomfohr et al, 2005). The latter two standardize first expression profiles into z-scores over the samples and, in the case of `zscore`, it combines them together as their sum divided by the square-root of the size of the gene set, while in the case of `plage` they are used to calculate the singular value decomposition (SVD) over the genes in the gene set and use the coefficients of the first right-singular vector as pathway activity profile.
`rnaseq`	Flag to inform whether the input gene expression data comes from microarray (`rnaseq=FALSE`, default) or RNA-Seq (`rnaseq=TRUE`) experiments.
`abs.ranking`	Flag to determine whether genes should be ranked according to their sign (`abs.ranking=FALSE`) or by absolute value (`abs.ranking=TRUE`). In the latter, pathways with genes enriched on either extreme (high or low) will be regarded as 'highly' activated.
`min.sz`	Minimum size of the resulting gene sets.
`max.sz`	Maximum size of the resulting gene sets.
`no.bootstraps`	Number of bootstrap iterations to perform.
`bootstrap.percent`	.632 is the ideal percent samples bootstrapped.
`parallel.sz`	Number of processors to use when doing the calculations in parallel. This requires to previously load either the `parallel` or the `snow` library. If `parallel` is loaded and this argument is left with its default value (`parallel.sz=0`) then it will use all available core processors unless we set this argument with a smaller number. If `snow` is loaded then we must set this argument to a positive integer number that specifies the number of processors to employ in the parallel calculation.
`parallel.type`	Type of cluster architecture when using `snow`.

`mx.diff` Offers two approaches to calculate the enrichment statistic (ES) from the KS random walk statistic. `mx.diff=FALSE`: ES is calculated as the maximum distance of the random walk from 0. `mx.diff=TRUE` (default): ES is calculated as the magnitude difference between the largest positive and negative random walk deviations.

`tau` Exponent defining the weight of the tail in the random walk performed by both the `gsva` (Hanzelmann et al., 2013) and the `ssgsea` (Barbie et al., 2009) methods. By default, this `tau=1` when `method="gsva"` and `tau=0.25` when `method="ssgsea"` just as specified by Barbie et al. (2009) where this parameter is called `alpha`.

`kernel` Logical, set to `TRUE` when the GSVA method employs a kernel non-parametric estimation of the empirical cumulative distribution function (default) and `FALSE` when this function is directly estimated from the observed data. This last option is justified in the limit of the size of the sample by the so-called Glivenko-Cantelli theorem.

`ssgsea.norm` Logical, set to `TRUE` (default) with `method="ssgsea"` runs the SSGSEA method from Barbie et al. (2009) normalizing the scores by the absolute difference between the minimum and the maximum, as described in their paper. When `ssgsea.norm=FALSE` this last normalization step is skipped.

`verbose` Gives information about each calculation step. Default: `FALSE`.

`is.gset.list.up.down`
logical. Is the gene list divided into up/down sublists? Please note that it is important to name the up-regulated gene set list 'up', and the down-regulated gene set list to 'down', if this argument is used (e.g gset = list(up = up_gset, down = down_gset))

Value

returns gene set enrichment scores for each sample and gene set

Methods (by class)

- `expr = ExpressionSet,gset.idx.list = list`: Method for ExpressionSet and list
- `expr = ExpressionSet,gset.idx.list = GeneSetCollection`: Method for ExpressionSet and GeneSetCollection
- `expr = matrix,gset.idx.list = GeneSetCollection`: Method for matrix and GeneSetCollection
- `expr = matrix,gset.idx.list = list`: Method for matrix and list

See Also

Hanzelmann, S., Castelo, R., & Guinney, J. (2013). GSVA: gene set variation analysis for microarray and RNA-Seq data. BMC Bioinformatics, 14, 7. http://doi.org/10.1186/1471-2105-14-7

Examples

```
data("Maupin")
names(maupin)
geneSet<- maupin$sig$EntrezID    #Symbol  ##EntrezID # both up and down genes:
up_sig<- maupin$sig[maupin$sig$upDown == "up",]
d_sig<- maupin$sig[maupin$sig$upDown == "down",]
u_geneSet<- up_sig$EntrezID   #Symbol   # up_sig$Symbol  ## EntrezID
d_geneSet<- d_sig$EntrezID
es.dif <- gsva(maupin$data, list(up = u_geneSet, down= d_geneSet), mx.diff=1,
    verbose=TRUE, abs.ranking=FALSE, is.gset.list.up.down=TRUE, parallel.sz = 1 )$es.obs
```

---

maupin *Maupin's TGFb data and a TGFb gene signature*

Description

A list consisting of two components: data: A matrix consisting of gene expression values on 3 control and 3 TGFb treated samples. sig: A TGFb gene signature. A dataframe containing a gene signature and information on whether genes in the signature are up or dow regulated.

Usage

```
data(Maupin)
```

Format

A list of two components:

M_Ctrl_R1 In data; Control Sample - Replicate 1

M_Ctrl_R2 In data; Control Sample - Replicate 2

M_Ctrl_R3 In data; Control Sample - Replicate 3

M_TGFb_R1 In data; Control Sample - Replicate 1

M_TGFb_R2 In data; Control Sample - Replicate 2

M_TGFb_R3 In data; Control Sample - Replicate 3

EntrezID In sig; The Entrez IDs

Symbol In sig; Gene symbols

upDown In sig; Direction of regulation e.g. up, down

Value

A list of 2

Source

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE23952 opa *Outlier profile Analysis*

Description

Returns a matrix with 0, -1 and 1 entries that describe outlier profiles in samples. The rows reperesent genes and the columns represent samples. -1 implies that the gene is a down-regulated outlier, 1 indicates an up-regulate outlier and 0 means that the gene is not an outlier in a sample.

Usage

```
opa(exprs.matrix, ...)

## S4 method for signature 'matrix'
opa(exprs.matrix, group, upper.quantile = 0.95,
  lower.quantile = 0.05)

## S4 method for signature 'ExpressionSet'
opa(exprs.matrix, group, upper.quantile = 0.95,
  lower.quantile = 0.05)
```

Arguments

`exprs.matrix`	Gene expression data. Can be either a matrix or an object of type ExpressionSet.
`...`	Numeric. To supply values for upper.quantile and lower.quantile arguments if default values are going to be override.
`group`	A vector of factors representing the groups to which each sample belong. This can be either a vector of 0s and 1s, or normal and cases.
`upper.quantile`	Numeric. The cut-off for upper quantile when determining outliers. Default to 0.95
`lower.quantile`	Numeric. The cut-off for lower quantile when determining outliers. Default to 0.05

Value

`opa` returns an object of type `OPPARList`. The outlier profiles are stored in `profileMatrix` and can be accessed using $. It it also possible to retrieve parameters used to run the outlier profile analysis, such as `upper.quantile`, `lower.quantile` via the $ operator.

Methods (by class)

- `matrix`: opa(exprs.matrix, group, lower.quantile = 0.05, upper.quantile = 0.95)
- `ExpressionSet`: opa(eset, group, lower.quantile = 0.05, upper.quantile = 0.95)

See Also

Wang, C., Taciroglu, A., Maetschke, S. R., Nelson, C. C., Ragan, M. A., & Davis, M. J. (2012). mCOPA: analysis of heterogeneous features in cancer expression data. Journal of Clinical Bioinformatics, 2, 22. http://doi.org/10.1186/2043-9113-2-22

Examples

```
# loading bcm object from GSE46141 dataset
data(GSE46141)
library(Biobase)
# defining the group variable. local breast tumors are the controls
# and the rest of the samples are the diseased samples
group <- sapply(pData(bcm)$source_name_ch1, function(x){ ifelse(x == "breast",0,1)})
group <- factor(group)
# running opa with default values (i.e upper.quantile = 0.95, lower.quantile = 0.05)
# the result is an object of type OPPARList
opa(bcm,group = group)
```

---

oppar — *oppar: A package for outlier profile and pathway analysis.*

Description

The oppar package provides 3 main function for outlier profile analysis: opa, getSampleOutlier and getSubtypeProbes, and 1 function for pathway and gene set enrichment analysis, based on gsva function implemented in the GSVA package

opa

calculates the outlier profile matrix, using the method proposed in Wang et al. (2012) paper

getSampleOutlier

extracts outlier profile for individual samples

getSubtypeProbes

extracts outlier profile for a group of related samples, such as subtypes

OPPARList-class *A S4 class for the output of OPPAR main function, opa.*

Description

A S4 class for the output of OPPAR main function, opa.

Usage

```
## S4 method for signature 'OPPARList'
show(object)

## S4 method for signature 'OPPARList'
x$name
```

Arguments

`object`	An object of type OPPARList
`x`	Object of type OPPARList.
`name`	Name of the slot to access.

Value

returns the number of outlier features detected, the number of samples retained, and the parameters used to run the `opa` function

extracts slots from an object of type `OPPARList`.

Methods (by generic)

- `show`: A show method for objects of class OPPARList
- `$`: A method to extract slots in `OPPARList`

Slots

`profileMatrix` A matrix of 0, -1 and 1 representing outlier genes in samples

`upper.quantile` Numeric. The upper quantile cut-off for detection of outliers

`lower.quantile` Numeric. The lower quantile cut-off for detection of outliers

`group` A factor vector representing the group to which each sample belong

`.Data` matrix

Index