

Package ‘pathwayPCA’

April 20, 2018

Type Package

Title Test Pathways for Statistically Significant Relationships

Version 0.0.0.9000

Maintainer Gabriel Odom <gabriel.odom@med.miami.edu>

Description Apply the Supervised PCA and Adaptive, Elastic-Net, Sparse PCA methods to extract principal components from each pathway. Use these pathway-specific principal components as the design matrix relating the response to each pathway. Return the model fit statistic p-values, and adjust these values for False Discovery Rate. Return a data frame of the pathways sorted by their adjusted p-values. This package has corresponding vignettes hosted in the “User Guides” page of <<https://gabrielodom.github.io/pathwayPCA/index.html>>, and the website for the development information is hosted at <<https://github.com/gabrielodom/pathwayPCA>>.

License GPL-2

Depends R (>= 2.10)

Imports corpcor, lars, methods, parallel, survival

Suggests knitr, reshape2, rmarkdown, tidyverse

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Collate 'createClass_OmicsPath.R' 'createClass_validOmics.R'
'accessClass_OmicsPath.R' 'accessClass_OmicsRegCateg.R'
'createClass_OmicsSurv.R' 'accessClass_OmicsSurv.R'
'adjust_and_sort_pValues.R' 'aesPC_calculate_AESPCA.R'
'aesPC_calculate_LARS.R' 'subsetExpressed-omes.R'
'aesPC_extract_OmicsPath_PCs.R' 'aesPC_permtest_CoxPH.R'
'createClass_OmicsCateg.R' 'aesPC_permtest_GLM.R'
'createClass_OmicsReg.R' 'aesPC_permtest_LS.R'
'aesPC_unknown_matrixNorm.R' 'aesPC_wrapper.R'
'calculate_gene_rank.R' 'calculate_matrixRoot.R'
'calculate_multtest_pvalues.R' 'create_Omics_All.R'
'create_Omics_Wrapper.R' 'create_pathwaySet.R'
'data_colonSubset.R' 'data_genesetSubset.R'
'printClass_Omics_All.R' 'printClass_pathwaySet.R' 'read_gmt.R'
'superPC_model_CoxPH.R' 'superPC_model_GLM.R'
'superPC_model_LS.R' 'superPC_model_tStats.R'

```
'superPC_modifiedSVD.R' 'superPC_optimWeibullParams.R'
'superPC_optimWeibull_pValues.R' 'superPC_pathway_pValues.R'
'superPC_pathway_tControl.R' 'superPC_pathway_tScores.R'
'superPC_permuteSamples.R' 'superPC_train.R'
'superPC_wrapper.R' 'transpose_assay.R'
```

VignetteBuilder knitr

URL <https://github.com/gabrielodom/pathwayPCA>

BugReports <https://github.com/gabrielodom/pathwayPCA/issues>

Author Gabriel Odom [aut, cre],

James Ban [aut],

Lily Wang [aut],

Steven Chen [aut]

R topics documented:

adjustRaw_pVals	3
adjust_and_sort	5
aespca	6
AESPCA_pVals	7
colonSurv_df	9
colon_pathwaySet	10
coxTrain_fun	10
create_Omics	11
create_OmicsPath	13
create_pathwaySet	15
expressedOmes	16
extract_aesPCs	17
getAssay	18
getEventTime	20
getResponse	21
glmTrain_fun	22
lars.lsa	23
matrixRoot	24
mysvd	24
normalize	25
olsTrain_fun	26
OmicsCateg-class	27
OmicsPathway-class	28
OmicsReg-class	28
OmicsSurv-class	29
pathway_pValues	29
pathway_tControl	30
pathway_tScores	32
permTest_OmicsCateg	33
permTest_OmicsReg	34
permTest_OmicsSurv	35
print.pathwaySet	36
randomControlSample	37
read_gmt	38
show,OmicsPathway-method	39

adjustRaw_pVals 3

superpc.st	40
superpc.train	41
superPCA_pVals	42
topGenes	44
transpose_assay	45
valid_OmicsSurv	46
weibullMix_optimParams	47
weibullMix_pValues	48

Index 50

<i>adjustRaw_pVals</i>	<i>Adjust p-values for simple multiple-testing procedures</i>
------------------------	---

Description

This is a modification of the `mt.rawp2adjp` function from the Bioconductor package `multtest`. We did not write the original function. For more information, see <https://www.bioconductor.org/packages/3.7/bioc/manuals/multtest/man/multtest.pdf>.

Usage

```
adjustRaw_pVals(rawp, proc = c("Bonferroni", "Holm", "Hochberg", "SidakSS",  
  "SidakSD", "BH", "BY", "ABH", "TSBH"), alpha = 0.05, na.rm = FALSE,  
  as.multtest.out = FALSE)
```

Arguments

<code>rawp</code>	A vector of raw (unadjusted) p -values for each hypothesis under consideration. These could be nominal p -values, for example, from t -tables, or permutation p -values.
<code>proc</code>	A vector of character strings containing the names of the multiple testing procedures for which adjusted p -values are to be computed. This vector should include any of the options listed in the "Details" Section. Adjusted p -values are computed for simple FWER- and FDR- controlling procedures based on a vector of raw (unadjusted) p -values.
<code>alpha</code>	A nominal Type-I error rate, or a vector of error rates, used for estimating the number of true null hypotheses in the two-stage Benjamini & Hochberg procedure ("TSBH"). Default is 0.05.
<code>na.rm</code>	An option for handling NA values in a list of raw p - values. If FALSE, the number of hypotheses considered is the length of the vector of raw p -values. Otherwise, if TRUE, the number of hypotheses is the number of raw p -values which were not NAs.
<code>as.multtest.out</code>	Should the output match the output from the <code>mt.rawp2adjp</code> function? If not, the output will match the input (a vector). Defaults to FALSE.

Details

This function computes adjusted p -values for simple multiple testing procedures from a vector of raw (unadjusted) p -values. The procedures include the Bonferroni, Holm (1979), Hochberg (1988), and Sidak procedures for strong control of the family-wise Type-I error rate (FWER), and the Benjamini & Hochberg (1995) and Benjamini & Yekutieli (2001) procedures for (strong) control of the false discovery rate (FDR). The less conservative adaptive Benjamini & Hochberg (2000) and two-stage Benjamini & Hochberg (2006) FDR-controlling procedures are also included.

The proc options are

- "Bonferroni" : Bonferroni single-step adjusted p - values for strong control of the FWER.
- "Holm" : Holm (1979) step-down adjusted p -values for strong control of the FWER.
- "Hochberg" : Hochberg (1988) step-up adjusted p - values for strong control of the FWER (for raw (unadjusted) p - values satisfying the Simes inequality).
- "SidakSS" : Sidak single-step adjusted p -values for strong control of the FWER (for positive orthant dependent test statistics).
- "SidakSD" : Sidak step-down adjusted p -values for strong control of the FWER (for positive orthant dependent test statistics).
- "BH" : Adjusted p -values for the Benjamini & Hochberg (1995) step-up FDR-controlling procedure (independent and positive regression dependent test statistics).
- "BY" : Adjusted p -values for the Benjamini & Yekutieli (2001) step-up FDR-controlling procedure (general dependency structures).
- "ABH" : Adjusted p -values for the adaptive Benjamini & Hochberg (2000) step-up FDR-controlling procedure. This method amends the original step-up procedure using an estimate of the number of true null hypotheses obtained from p -values.
- "TSBH" : Adjusted p -values for the two-stage Benjamini & Hochberg (2006) step-up FDR-controlling procedure. This method amends the original step-up procedure using an estimate of the number of true null hypotheses obtained from a first-pass application of "BH". The adjusted p -values are α - dependent, therefore α must be set in the function arguments when using this procedure.

Value

A vector of the same length and order as `rawp`, unless the user specifies that the output should match the output from the `multtest` package. In that case, the use should specify `as.multtest.out = TRUE` and this function will return output identical to that of the `mt.rawp2adjp` function from package `multtest`. That output is as follows:

- `adjp` : A matrix of adjusted p -values, with rows corresponding to hypotheses and columns to multiple testing procedures. Hypotheses are sorted in increasing order of their raw (unadjusted) p -values.
- `index` : A vector of row indices, between 1 and `length(rawp)`, where rows are sorted according to their raw (unadjusted) p -values. To obtain the adjusted p -values in the original data order, use `adjp[order(index),\]`.
- `h0.ABH` : The estimate of the number of true null hypotheses (as proposed by Benjamini & Hochberg (2000)) used when computing adjusted p -values for the "ABH" procedure (see Dudoit et al., 2007).
- `h0.TSBH` : The estimate (or vector of estimates) of the number of true null hypotheses (as proposed by Benjamini et al. (2006)) when computing adjusted p -values for the "TSBH" procedure (see Dudoit et al., 2007).

Author(s)

Sandrine Dudoit, <http://www.stat.berkeley.edu/~sandrine>

Yongchao Ge, yongchao.ge@mssm.edu

Houston Gilbert, <http://www.stat.berkeley.edu/~houston>

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Call this function through AESPCA_pVals() or superPCA_pVals() instead.
```

adjust_and_sort

Adjust and sort pathway p-values

Description

Adjust the pathway p -values, then return a data frame of the relevant pathway information, sorted by adjusted significance.

Usage

```
adjust_and_sort(pVals_vec, genesets_ls, adjust = TRUE,
  proc_vec = c("Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD", "BH",
    "BY", "ABH", "TSBH"), ...)
```

Arguments

- | | |
|-------------|---|
| pVals_vec | A named vector of permutation p -values returned by the permTest_OmicsSurv , permTest_OmicsReg , or permTest_OmicsCateg functions when the analysis performed was AES-PCA. Otherwise, when the analysis was performed with Supervised PCA, a named vector of p -values from the weibullMix_pValues function. |
| genesets_ls | <p>A list of known gene pathways. This pathway list must contain:</p> <ul style="list-style-type: none"> • pathways : A named list of character vectors where each vector contains the names of the genes in that specific pathway. • TERMS : A character vector the same length as pathways containing the full pathway descriptions. • setsize : An integer vector the same length as pathways containing the number of genes contained in the original pathway set. • trim_setsize : An integer vector the same length as pathways containing the number of genes present in the pathway after trimming. Pathway set trimming is done in the expressedOmes function. |
| adjust | Should you adjust the p -values for multiple comparisons? Defaults to TRUE. |
| proc_vec | Character vector of procedures. The returned data frame will be sorted in ascending order by the first procedure in this vector, with ties broken by the unadjusted p -value. If only one procedure is selected, then it is necessarily the first procedure. |
| ... | Additional arguments to pass to the adjustRaw_pVals function. |

Details

This is a wrapper function for the `adjustRaw_pVals` function. The number of p -values passed to the `pVals_vec` argument *must* equal the number of pathways and set size values in the `genesets_ls` argument. If you trimmed a pathway from p -value calculation, then pad this missing value with an NA.

Value

A data frame with columns

- `pathways`: The names of the pathways in the `OmicS*` object (stored in `object@pathwaySet$pathways`).
- `setsize`: The number of genes in each of the original pathways (as stored in the `object@pathwaySet$setsize` object).
- `terms`: The pathway description, as stored in the `object@pathwaySet$TERMS` object.
- `rawp`: The unadjusted p -values of each pathway.
- `...`: Additional columns as specified through the adjustment argument.

The data frame will be sorted in ascending order by the method specified first in the adjustment argument. If `adjustpValues = FALSE`, then the data frame will be sorted by the raw p -values. If you have the suggested tidyverse package suite loaded, then this data frame will print as a `tibble`. Otherwise, it will stay a simple data frame.

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Call this function through AESPCA_pVals() or superPCA_pVals() instead.
```

aespca

Adaptive, elastic-net, sparse principal component analysis

Description

A function to perform adaptive, elastic-net, sparse principal component analysis (AES-PCA).

Usage

```
aespca(X, d = 1, max.iter = 10, eps.conv = 0.001, adaptive = TRUE,
       para = NULL)
```

Arguments

<code>X</code>	A pathway design matrix: the data matrix should be $n \times p$, where n is the sample size and p is the number of variables included in the pathway.
<code>d</code>	The number of principal components (PCs) to extract from the pathway. Defaults to 1.
<code>max.iter</code>	The maximum number of times an internal <code>while()</code> loop can make calls to the <code>lars.lsa()</code> function. Defaults to 10.
<code>eps.conv</code>	A numerical convergence threshold for the same <code>while()</code> loop. Defaults to 0.001.
<code>adaptive</code>	Internal argument of the <code>lars.lsa()</code> function. Defaults to TRUE.
<code>para</code>	Internal argument of the <code>lars.lsa()</code> function. Defaults to NULL.

Details

This function calculates the loadings and reduced-dimension predictor matrix using both the Singular Value Decomposition and AES-PCA Decomposition (as described in Efron et al (2003)) of the data matrix.

See https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf.

For potential enhancement details, see the comment in the "Details" section of [normalize](#).

Value

A list of four elements containing the loadings and projected predictors:

- `loadings` : A $p \times d$ projection matrix of the d AES-PCs.
- `B0` : A $p \times d$ projection matrix of the d PCs from the singular value decomposition (SVD).
- `score` : An $n \times d$ predictor matrix: the original n observations loaded onto the d AES-PCs.
- `oldscore` : An $n \times d$ predictor matrix: the original n observations loaded onto the d SVD-PCs.

See Also

[normalize](#); [lars.lsa](#); [extract_aesPCs](#); [AESPCA_pVals](#)

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Call this function through AESPCA_pVals() instead.
```

AESPCA_pVals

Test pathway association with AES-PCA

Description

Given a supervised OmicsPath object (one of OmicsSurv, OmicsReg, or OmicsCateg), extract the first k adaptive, elastic-net, sparse principal components (PCs) from each expressed pathway in the -Omics assay design matrix, test their association with the response matrix, and return a data frame of the adjusted p -values for each pathway.

Usage

```
AESPCA_pVals(object, numPCs = 1, min.features = 3, numReps = 1000,
  parallel = FALSE, numCores = NULL, adjustpValues = TRUE,
  adjustment = c("Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD", "BH",
    "BY", "ABH", "TSBH"), ...)

## S4 method for signature 'OmicsPathway'
AESPCA_pVals(object, numPCs = 1, min.features = 3,
  numReps = 1000, parallel = FALSE, numCores = NULL,
  adjustpValues = TRUE, adjustment = c("Bonferroni", "Holm", "Hochberg",
    "SidakSS", "SidakSD", "BH", "BY", "ABH", "TSBH"), ...)
```

Arguments

<code>object</code>	An object of class <code>OmicsPathway</code> with a response matrix or vector.
<code>numPCs</code>	The number of PCs to extract from each pathway. Defaults to 1.
<code>min.features</code>	What is the smallest number of genes allowed in each pathway? This argument must be kept constant across all calls to this function which use the same pathway list. Defaults to 3.
<code>numReps</code>	The number of permutations to take of the data to calculate a p -value for each pathway. Defaults to 1000.
<code>parallel</code>	Should the computation be completed in parallel? Defaults to FALSE.
<code>numCores</code>	If <code>parallel = TRUE</code> , how many cores should be used for computation? Defaults to NULL.
<code>adjustpValues</code>	Should you adjust the p -values for multiple comparisons? Defaults to TRUE.
<code>adjustment</code>	Character vector of procedures. The returned data frame will be sorted in ascending order by the first procedure in this vector, with ties broken by the unadjusted p -value. If only one procedure is selected, then it is necessarily the first procedure. See the documentation for the adjustRaw_pVals function for the adjustment procedure definitions and citations.
<code>...</code>	Dots for additional internal arguments.

Details

This is a wrapper function for the [expressedOmes](#), [extract_aesPCs](#), [permTest_OmicsSurv](#), [permTest_OmicsReg](#), and [permTest_OmicsCateg](#) functions.

Value

A data frame with columns:

- `pathways` : The names of the pathways in the `Omics*` object (given in `object@pathwaySet$pathways`.)
- `setsize` : The number of genes in each of the original pathways (given in the `object@pathwaySet$setsize` object).
- `terms` : The pathway description, as given in the `object@pathwaySet$TERMS` object.
- `rawp` : The unadjusted p -values of each pathway.
- `...` : Additional columns as specified through the `adjustment` argument.

Some of the pathways in the supplied pathway set list will be removed, or "trimmed", during function execution. These trimmed pathways will have p -values given as NA. For an explanation of pathway trimming, see the documentation for the [expressedOmes](#) function.

The data frame will be sorted in ascending order by the method specified first in the `adjustment` argument. If `adjustpValues = FALSE`, then the data frame will be sorted by the raw p -values. If you have the suggested tidyverse package suite loaded, then this data frame will print as a [tibble](#). Otherwise, it will print as a data frame.

See Also

[expressedOmes](#); [create_OmicsPath](#); [create_OmicsSurv](#); [create_OmicsReg](#); [create_OmicsCateg](#); [extract_aesPCs](#); [permTest_OmicsSurv](#); [permTest_OmicsReg](#); [permTest_OmicsCateg](#); [adjust_and_sort](#)

Examples

```
## Not run:
### Load the Example Data ###
data("colonSurv_df")
data("colon_pathwaySet")

### Create an OmicsSurv Object ###
colon_OmicsSurv <- create_OmicsSurv(assayData_df = colonSurv_df[, -(1:2)],
                                   pathwaySet_ls = colon_pathwaySet,
                                   eventTime_num = colonSurv_df$OS_time,
                                   eventObserved_lgl = as.logical(colonSurv_df$OS_event))

### Calculate Pathway p-Values ###
colonSurv_pVals_df <- AESPCA_pVals(object = colon_OmicsSurv,
                                   numReps = 500,
                                   parallel = TRUE,
                                   numCores = 2,
                                   adjustpValues = TRUE,
                                   adjustment = c("Hoch", "SidakSD"))

## End(Not run)
```

colonSurv_df

Colon Cancer -Omics Data

Description

Subset of a colon cancer survival data set, with subject response and assay values.

Usage

```
colonSurv_df
```

Format

A subset of a data frame containing 656 of 2022 genes measured on 250 subjects. The first two columns are the Overall Survival time (OS_time) and death indicator (OS_event).

Source

Xi Steven Chen

colon_pathwaySet	<i>Gene Pathway Subset</i>
------------------	----------------------------

Description

An example Canonical Pathways Gene Subset from the Broad Institute: File: c2.cp.v6.0.symbols.gmt.

Usage

```
colon_pathwaySet
```

Format

A pathwaySet list of two elements:

- pathways : A list of 15 character vectors. Each vector contains the names of the individual genes within that pathway as a vector of character strings.
- TERMS : A character vector of length 15 containing the names of the gene pathways.

Details

This is a subset of 15 pathways from the Broad Institute gene set list. This subset contains seven pathways which are related to the response information in the [colonSurv_df](#) data file.

Source

<http://software.broadinstitute.org/gsea/msigdb/collections.jsp>

coxTrain_fun	<i>Train Cox Proportional Hazards model for supervised PCA</i>
--------------	--

Description

Main and utility functions for training the Cox PH model.

Usage

```
coxTrain_fun(x, y, censoring.status, s0.perc = NULL)
```

Arguments

x	A "tall" pathway data frame ($p \times n$).
y	A response vector of follow-up / event times.
censoring.status	A censoring vector.
s0.perc	A stabilization parameter. This is an optional argument to each of the functions called internally. Defaults to NULL.

Details

See https://web.stanford.edu/~hastie/Papers/spca_JASA.pdf, Section 5, for a description of Supervised PCA applied to survival data. The internal utility functions defined in this file (`.coxscor`, `.coxvar`, and `.coxstuff`) are not called anywhere else, other than in the `coxTrain_fun` function itself. Therefore, we do not document these functions.

NOTE: No missing values allowed.

Value

A list containing:

- `tt` : The scaled p-dimensional score vector: each value has been divided by the respective standard deviation plus the fudge value.
- `numer` : The original p-dimensional score vector. From the internal `.coxscor` function.
- `sd` : The standard deviations of the scores. From the internal `.coxvar` function.
- `fudge` : A regularization scalar added to the standard deviation. If `s0.perc` is supplied, `fudge = quantile(sd, s0.perc)`.

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use superPCA_pVals() instead
```

create_Omics	<i>Generation Wrapper function for -Omics*-class objects</i>
--------------	--

Description

This function calls the `create_OmicsPath`, `create_OmicsSurv`, `create_OmicsReg`, and `create_OmicsCateg` functions to create valid objects of the classes `OmicsPathway`, `OmicsSurv`, `OmicsReg`, or `OmicsCateg`, respectively.

Usage

```
create_Omics(assayData_df, pathwaySet_ls, response = NULL,
  respType = c("none", "survival", "regression", "categorical"))
```

Arguments

- | | |
|---------------|---|
| assayData_df | An $N \times p$ data frame with named columns. |
| pathwaySet_ls | A pathwaySet list of known gene pathways with two elements: <ul style="list-style-type: none"> • <code>pathways</code> : A named list of character vectors. Each vector contains the names of the individual genes within that pathway as a vector of character strings. The names contained in these vectors must have non-empty overlap with the <i>column names</i> of the <code>assayData_df</code> data frame. The names of the pathways (the list elements themselves) should be the a shorthand representation of the full pathway name. • <code>TERMS</code>: A character vector the same length as the <code>pathways</code> list with the proper names of the pathways. |

create_OmicsPath	<i>Generation functions for -Omics*-class objects</i>
------------------	---

Description

These functions create valid objects of class OmicsPathway, OmicsSurv, OmicsReg, or OmicsCateg.

Usage

```
create_OmicsPath(assayData_df, pathwaySet_ls)
```

```
create_OmicsSurv(assayData_df, pathwaySet_ls, eventTime_num, eventObserved_lgl)
```

```
create_OmicsReg(assayData_df, pathwaySet_ls, response_num)
```

```
create_OmicsCateg(assayData_df, pathwaySet_ls, response_fact)
```

Arguments

assayData_df An $N \times p$ data frame with named columns.

pathwaySet_ls A pathwaySet list of known gene pathways with two elements:

- pathways : A named list of character vectors. Each vector contains the names of the individual genes within that pathway as a vector of character strings. The names contained in these vectors must have non-empty overlap with the *column names* of the assayData_df data frame. The names of the pathways (the list elements themselves) should be the a shorthand representation of the full pathway name.
- TERMS: A character vector the same length as the pathways list with the proper names of the pathways.

eventTime_num A numeric vector with N observations corresponding to the last observed time of follow up.

eventObserved_lgl
A logical vector with N observations indicating right-censoring. The values will be FALSE if the observation was censored (i.e., we did not observe an event).

response_num A numeric vector of length N : the dependent variable in an ordinary regression exercise.

response_fact A factor vector of length N : the dependent variable of a generalized linear regression exercise.

Details

Please note that the classes of the parameters are *not* flexible. The -Omics measurement data *must* be or extend the class data.frame, and the response values (for a survival, regression, or classification object) *must* match their expected classes *exactly*. The reason for this is to encourage the end user to pay attention to the quality and format of their input data. Because the functions internal to this package have only been tested on the classes described in the Arguments section, these class checks prevent unexpected errors (or worse, incorrect computational results without an error). These draconian input class restrictions protect the accuracy of your data analysis.

Also note the following: if the supplied pathways object within your pathwaySet_ls list has no names, then this pathway list will be named path1, path2, path3, ...; if any of the pathways are missing names, then the missing pathways will be named noName followed by the index of the pathway. For example, if the 112th pathway in the pathways list has no name (but other pathways do), then this pathway will be named noName112. Furthermore, if any of the pathway names are duplicated, then the duplicates will have .1, .2, .3, ... appended to the duplicate names until all pathway names are unique. Once all pathways have been verified to have unique names, then the pathway names are attached as attributes to the TERMS and setsize vectors (the setsize vector is calculated at object creation).

If your gene pathway set information is stored in a .gmt file, use the [read_gmt](#) function to import your gene set list as a pathwaySet list object.

Value

A valid object of class OmicsPathway, OmicsSurv, OmicsReg, or OmicsCateg.

OmicsPathway

Valid OmicsPathway objects will have no response information, just the mass spectrometry or bio-assay ("design") matrix and the pathway list. OmicsPathway objects should be created only when unsupervised pathway extraction is needed (not possible with Supervised PCA). Because of the missing response, no pathway testing can be performed on an OmicsPathway object.

OmicsSurv

Valid OmicsSurv objects will have two response vectors: a vector of the most recently recorded follow-up times and a logical vector if that time marks an event (TRUE: observed event; FALSE: right- censored observation).

OmicsReg and OmicsCateg

Valid OmicsReg and OmicsCateg objects with have one response vector of continuous (numeric) or categorical (factor) observations, respectively.

See Also

[OmicsPathway](#), [OmicsSurv](#), [OmicsReg](#), and [OmicsCateg](#)

Examples

```
## Not run:
### Load the Example Data ###
data("colonSurv_df")
data("colon_pathwaySet")

### Create an OmicsPathway Object ###
colon_OmicsPath <- create_OmicsPath(assayData_df = colonSurv_df[, -(1:2)],
                                   pathwaySet_ls = colon_pathwaySet)

### Create an OmicsSurv Object ###
colon_OmicsSurv <- create_OmicsSurv(assayData_df = colonSurv_df[, -(1:2)],
                                   pathwaySet_ls = colon_pathwaySet,
                                   eventTime_num = colonSurv_df$OS_time,
                                   eventObserved_lgl = as.logical(colonSurv_df$OS_event))
```

```

### Create an OmicsReg Object ###
colon_OmicsReg <- create_OmicsReg(assayData_df = colonSurv_df[, -(1:2)],
                                pathwaySet_ls = colon_pathwaySet,
                                response_num = colonSurv_df$OS_time)

### Create an OmicsCateg Object ###
colon_OmicsCateg <- create_OmicsCateg(assayData_df = colonSurv_df[, -(1:2)],
                                     pathwaySet_ls = colon_pathwaySet,
                                     response_fact = as.factor(colonSurv_df$OS_event))

## End(Not run)

```

create_pathwaySet	<i>Manually Create a pathwaySet-class Object.</i>
-------------------	---

Description

Manually create a pathwaySet list similar to the output of the [read_gmt](#) function.

Usage

```
create_pathwaySet(pathways, TERMS, ...)
```

Arguments

pathways	A named list of character vectors. Each vector should contain the names of the individual genes or proteins within that pathway as a vector of character strings. The names contained in these vectors should have non-empty overlap with the feature names of the assay data frame that will be paired with this list in the subsequent analysis. The names of the pathways (the list elements themselves) should be the a shorthand representation of the full pathway name, but this is not required.
TERMS	A character vector the same length as the pathways list with the proper names of the pathways.
...	Additional vectors or data components related to the pathways list. These values should be passed as a name-value pair. See "Details" for more information.

Details

This function checks the pathway list and pathway term inputs and then creates a pathwaySet object from them. Pass additional list elements using the form tag = value through the ... argument (as in the [list](#) function). Because some functions in the pathwayPCA package add and edit elements of pathwaySet objects, please do not create pathwaySet list items named setsize or trim_setsize.

Value

A list object with class pathwaySet.

See Also

[read_gmt](#)

Examples

```
data("colon_pathwaySet")

create_pathwaySet(pathways = colon_pathwaySet$pathways,
                  TERMS = colon_pathwaySet$TERMS)
```

expressedOmes	<i>Extract expressed -Omes matching a gene set from a mass spectrometry or assay data frame</i>
---------------	---

Description

Given a bio-assay design matrix and a pathwaySet gene pathways list (each within an Omics*-class object), extract the genes / proteins / lipids / metabolomes / transcriptomes contained in each gene pathway set which are expressed in the assay data frame.

Usage

```
expressedOmes(object, trim = 3, message = TRUE, ...)

## S4 method for signature 'OmicsPathway'
expressedOmes(object, trim = 3, message = TRUE,
              ...)
```

Arguments

object	An object of class OmicsPathway, OmicsSurv, OmicsReg, or OmicsCateg.
trim	The minimum cutoff of expressed -Ome measures before a pathway is excluded. Defaults to 3.
message	Should this function return diagnostic messages? Messages concern the percentage of genes included in the pathway set but not measured in the data, genes measured in the data but not called for in the pathways, and the number of pathways ignored due to too few number of genes present after trimming. Defaults to TRUE.
...	Dots for additional internal arguments (as necessary).

Details

This function takes in a data frame with named columns and a pathwaySet list, all through one of the Omics* classes. This function will then iterate over the list of pathways, extract columns from the bio-assay design matrix which match the genes listed in that pathway, and remove any pathways with fewer than trim expressed genes. The genes not expressed in the bio-assay design matrix are removed from the pathwaySet list.

NOTE: some genes will be included in more than one pathway, so these pathways are not mutually exclusive. Further note that there may be many genes in the assay design matrix that are not included in the pathway sets, so these will not be extracted to the list. It is then vitally important to use either a very broad and generic pathwaySet list or a pathwaySet list that is appropriate for the assay data supplied. While you can create your own pathway lists, create proper pathwaySet list objects by importing .gmt files with the [read_gmt](#) function.

Value

A valid `Omics*`-class object. This output object will be identical to the input object, except that any genes present in the pathways list, but not present in the MS design matrix, will have been removed. Additionally, the pathway list will have the number of genes in each trimmed pathway stored as the `trim_setsize` object.

Examples

```
## Not run:
### Load the Example Data ###
data("colonSurv_df")
data("colon_pathwaySet")

### Create an OmicsSurv Object ###
colon_OmicsSurv <- create_OmicsSurv(assayData_df = colonSurv_df[, -(1:2)],
                                   pathwaySet_ls = colon_pathwaySet,
                                   eventTime_num = colonSurv_df$OS_time,
                                   eventObserved_lgl = as.logical(colonSurv_df$OS_event))

### Extract Expressed Genes ###
expressed0mes(colon_OmicsSurv)

## End(Not run)
```

extract_aesPCs	<i>Extract AES-PCs from expressed pathway-subsets of a mass spectrometry or bio-assay data frame</i>
----------------	--

Description

Given a clean `OmicsPath` object (cleaned by the [expressed0mes](#) function), extract the first principal components from each expressed pathway in the assay design matrix.

Usage

```
extract_aesPCs(object, trim = 3, numPCs = 1, parallel = FALSE,
               numCores = NULL, ...)

## S4 method for signature 'OmicsPathway'
extract_aesPCs(object, trim = 3, numPCs = 1,
               parallel = FALSE, numCores = NULL, ...)
```

Arguments

<code>object</code>	An object of class <code>OmicsPathway</code> .
<code>trim</code>	The minimum cutoff of expressed -Ome measures before a pathway is excluded. Defaults to 3.
<code>numPCs</code>	The number of PCs to extract from each pathway. Defaults to 1.
<code>parallel</code>	Should the computation be completed in parallel? Defaults to FALSE.
<code>numCores</code>	If <code>parallel = TRUE</code> , how many cores should be used for computation? Defaults to NULL.
<code>...</code>	Dots for additional internal arguments (currently unused).

Details

This function takes in a data frame with named columns and a pathway list as an `OmicsPathway` object which has had unexpressed -Omes removed by the `expressedOmes` function. This function will then iterate over the list of pathways, extracting columns from the assay design matrix which match the genes listed in that pathway as a sub-matrix (as a `data.frame` object). This function will then call the `aespca` on each data frame in the list of pathway-specific design matrices, extracting the first `numPCs` AES principal components from each pathway data frame. These PC matrices are returned as a named list.

NOTE: some genes will be included in more than one pathway, so these pathways are not mutually exclusive. Further note that there may be many genes in the assay design matrix that are not included in the pathway sets, so these will not be extracted to the list. It is then vitally important to use either a very broad and generic pathway set list or a pathway set list that is appropriate for the assay data supplied.

Value

A list of matrices. Each element of the list will be named by its pathway, and the elements will be $N \times \text{numPCs}$ matrices containing the first `numPCs` principal components from each pathway. See "Details" for more information.

See Also

`create_OmicsPath`; `expressedOmes`; `aespca`

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use AESPCA_pVals() instead
```

getAssay

Access and Edit Assay or pathwaySet Values in Omics Objects*

Description

"Get" or "Set" the values of the `assayData_df` or `pathwaySet` slots of an object of class `OmicsPathway` or a class that extends this class (`OmicsSurv`, `OmicsReg`, or `OmicsCateg`).

Usage

```
getAssay(object, ...)

getAssay(object) <- value

getPathwaySet(object, ...)

getPathwaySet(object) <- value

## S4 method for signature 'OmicsPathway'
getAssay(object, ...)
```

```
## S4 replacement method for signature 'OmicsPathway'
getAssay(object) <- value

## S4 method for signature 'OmicsPathway'
getPathwaySet(object, ...)

## S4 replacement method for signature 'OmicsPathway'
getPathwaySet(object) <- value
```

Arguments

object	An object of or extending OmicsPathway-class : that class, OmicsSurv-class , OmicsReg-class , or OmicsCateg-class .
...	Dots for additional internal arguments (currently unused).
value	The replacement object to be assigned to the specified slot.

Details

These functions can be useful to set or extract the assay data or gene set list from an `Omics*`-class object. However, we recommend that users simply create a new, valid `Omics*` object instead of modifying an existing one. The validity of edited objects is checked with the [valid_OmicsSurv](#), [valid_OmicsCateg](#), or [valid_OmicsReg](#) functions.

Value

The "get" functions return the objects in the slots specified: `getAssay` returns the `assayData_df` data frame object and `getPathwaySet` returns the `pathwaySet` list object. These functions can extract these values from any valid `OmicsPathway`, `OmicsSurv`, `OmicsReg`, or `OmicsCateg` object.

The "set" functions enable the user to edit or replace objects in the `assayData_df` or `pathwaySet` slots for any `OmicsPathway`, `OmicsSurv`, `OmicsReg`, or `OmicsCateg` objects, provided that the new values do not violate the validity checks of their respective objects. See "Details" for more information.

See Also

[create_OmicsPath](#), [create_OmicsSurv](#), [create_OmicsReg](#), [create_OmicsCateg](#)

Examples

```
## Not run:
getAssay(colon_OmicsSurv)
getPathwaySet(colon_OmicsSurv)

getAssay(colon_OmicsSurv) <- newAssay_df
getPathwaySet(colon_OmicsSurv) <- new_pathwaySet

## End(Not run)
```

getEventTime	<i>Access and Edit Event Time or Indicator in an OmicsSurv Object</i>
--------------	---

Description

"Get" or "Set" the values of the eventTime_num or eventObserved_lgl slots of an object of class OmicsSurv.

Usage

```
getEventTime(object, ...)

getEventTime(object) <- value

getEvent(object, ...)

getEvent(object) <- value

## S4 method for signature 'OmicsSurv'
getEventTime(object, ...)

## S4 replacement method for signature 'OmicsSurv'
getEventTime(object) <- value

## S4 method for signature 'OmicsSurv'
getEvent(object, ...)

## S4 replacement method for signature 'OmicsSurv'
getEvent(object) <- value
```

Arguments

object	An object of class OmicsSurv-class .
...	Dots for additional internal arguments (currently unused).
value	The replacement object to be assigned to the specified slot.

Details

These functions can be useful to set or extract the event time or death indicator from an OmicsSurv object. However, we recommend that users simply create a new, valid OmicsSurv object instead of modifying an existing one. The validity of edited objects is checked with the [valid_OmicsSurv](#) function.

Value

The "get" functions return the objects in the slots specified: getEventTime returns the eventTime_num vector object and getEvent returns the eventObserved_lgl vector object. These functions can extract these values from any valid OmicsSurv object.

The "set" functions enable the user to edit or replace objects in the eventTime_num or eventObserved_lgl slots for any OmicsSurv object, provided that the new values do not violate the validity check of an OmicsSurv object. See "Details" for more information.

See Also[create_OmicsSurv](#)**Examples**

```
## Not run:
  getEventTime(colon_OmicsSurv)
  getEvent(colon_OmicsSurv)

  getEventTime(colon_OmicsSurv) <- newTime_num
  getEvent(colon_OmicsSurv) <- newEvent_lgl

## End(Not run)
```

getResponse

*Access and Edit Response of an OmicsReg or OmicsReg Object***Description**

"Get" or "Set" the values of the response_num or response_fact slots of an object of class OmicsReg or OmicsReg, respectively.

Usage

```
getResponse(object, ...)

getResponse(object) <- value

## S4 method for signature 'OmicsPathway'
getResponse(object, ...)

## S4 replacement method for signature 'OmicsPathway'
getResponse(object) <- value
```

Arguments

object	An object of class OmicsReg-class or OmicsCateg-class .
...	Dots for additional internal arguments (currently unused).
value	The replacement object to be assigned to the response slot.

Details

These functions can be useful to set or extract the response vector from an object of class OmicsReg or OmicsReg. However, we recommend that users simply create a new, valid object instead of modifying an existing one. The validity of edited objects is checked with their respective [valid_OmicsCateg](#) or [valid_OmicsReg](#) function. Because both classes have a response slot, we set this method for the parent class, [OmicsPathway-class](#).

Value

The "get" functions return the objects in the slots specified: `getResponse` returns the `response_num` vector from objects of class `OmicsReg` and the `response_fact` vector from objects of class `OmicsCateg`. These functions can extract these values from any valid object of those classes.

The "set" functions enable the user to edit or replace the object in the `response_num` slot for any `OmicsReg` object or `response_fact` slot for any `OmicsCateg` object, provided that the new values do not violate the validity check of such an object. See "Details" for more information.

See Also

[create_OmicsReg](#), [create_OmicsCateg](#)

Examples

```
## Not run:
getResponse(colon_OmicsReg)
getResponse(colon_OmicsCateg)

getResponse(colon_OmicsReg) <- newResponse_num
getResponse(colon_OmicsCateg) <- newResponse_fct

## End(Not run)
```

glmTrain_fun	<i>Gene-specific Generalized Linear Model fit statistics for supervised PCA</i>
--------------	---

Description

Model statistics for Generalized Linear Model (GLM) regression by gene

Usage

```
glmTrain_fun(x, y, family = binomial)
```

Arguments

<code>x</code>	An $p \times n$ predictor matrix.
<code>y</code>	A response vector.
<code>family</code>	A description of the error distribution and link function to be used in the model. The default is <code>binomial(link = "logit")</code> .

Details

While this function currently supports any GLM family from the `family` function, this function is only called in the model fitting step (via the internal `superpc.train` function) and not in the test statistic calculation step (in the `superpc.st` function). We would like to support Poisson regression through the `glm` function, as well as n-ary classification through `multinom` and ordinal logistic regression through `polr`.

Value

The slope coefficient from the GLM for each gene.

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use superPCA_pVals() instead
```

lars.lsa

Least Angle Regression and LASSO Regression

Description

These are all variants of LASSO, and provide the entire sequence of coefficients and fits, starting from zero to the least squares fit.

Usage

```
lars.lsa(Sigma0, b0, n, type = c("lar", "lasso"), max.steps = NULL,
  eps = .Machine$double.eps, adaptive = TRUE, para = NULL)
```

Arguments

Sigma0	A Gramian / covariance matrix of pathway predictors.
b0	An eigenvector of Sigma0.
n	The sample size.
type	Option between "lar" and "lasso". Defaults to "lasso".
max.steps	How many steps should the LAR or LASSO algorithms take? Defaults to 8 times the pathway dimension.
eps	What should we consider to be numerically 0? Defaults to the machine's default error limit for doubles (.Machine\$double.eps).
adaptive	Ignore.
para	Ignore.

Details

LARS is described in detail in Efron, Hastie, Johnstone and Tibshirani (2002). With the "lasso" option, it computes the complete LASSO solution simultaneously for *all* values of the shrinkage parameter in the same computational cost as a least squares fit. This function is adapted from the [lars](#) function in the lars package to apply to covariance or Gramian pathway design matrices.

Value

An object of class "lars".

See Also

https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use AESPCA_pVals() instead
```

matrixRoot	<i>Positive root of a symmetric matrix</i>
------------	--

Description

Calculate the matrix root of a symmetric matrix via the Spectral Decomposition.

Usage

```
matrixRoot(x, root = 2)
```

Arguments

x	A symmetric (necessarily square) matrix.
root	A positive real number.

Details

This function decomposes x into $V \times D \times V^T$ via the [eigen](#) function, sets any numerically negative eigenvalues to 0, calculates the root of these eigenvalues as D^r , then returns the matrix $V \times D^r \times V^T$.

See https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix.

Value

A matrix, that when multiplied by itself root times, yields x.

Examples

```
X <- matrix(rnorm(25), ncol = 5);  xTx <- t(X) %*% X
matrixRoot(xTx)
matrixRoot(xTx, root = 3)
```

mysvd	<i>Singular Value Decomposition wrapper for supervised PCA</i>
-------	--

Description

Center and compute the fast SVD of a matrix

Usage

```
mysvd(mat, n.components = NULL)
```


Arguments

mat	A matrix of data frame in "tall" format ($p \times n$).
n.components	How many singular values / vectors to return? Must be an integer less than $\min(p, n)$. Best performance increase is for values much less than $\min(p, n)$. Defaults to NULL.

Details

The mysvd function takes in a tall -Omics data matrix, extracts the feature means, centers the matrix on this mean vector, and calculates the Singular Value Decomposition (SVD) of the centered data matrix. Currently, the SVD is calculated via the [fast.svd](#) function from corpcor package. However, this function calculates all the singular vectors, even when n.components is non-NULL. We should experiment with other SVD functions, such as the [rsvd](#) function from the rsvd package.

ENHANCEMENT.

Value

A list containing:

- u : The first n.components left singular vectors of mat.
- d : The largest n.component singular values of mat.
- v : The first n.components right singular vectors of mat.
- feature.means : A named vector of the feature means of mat.

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use superPCA_pVals() instead
```

normalize	<i>Normalize and reconstruct the eigenvalues of a data matrix for supervised PCA</i>
-----------	--

Description

Normalize the columns of a project matrix. For each eigenvector, swap the signs of the vector elements if the first entry is negative. See "Details" for more information.

Usage

```
normalize(B, d)
```

Arguments

B	A projection matrix: often the matrix of the left singular vectors given by the Singular Value Decomposition of a data matrix or Grammian.
d	The number of columns of B to normalize.

Details

This function is designed to reconstruct the original first d left singular vectors of a data matrix from the first d eigenvectors of the Grammian of that data matrix. Basically, after the data matrix has been centred, the left singular vectors of that data matrix and the left singular vectors of the Grammian of that data matrix are equal up to a sign. This function reverses that sign so that the two sets of singular vectors are equal.

Consider the internal workings of the `aespca` function. This "sign flipping" changes the eigenvectors of xtx into the left singular vectors of `scale(X, , center = TRUE, scale = TRUE)`. Instead of calculating the Grammian, regularising it (by adding some small λ value to the diagonal), taking the SVD of the regularized Grammian, and extracting the first d eigenvectors, why don't we just extract the first d singular vectors directly from the scaled data matrix itself? The regularisation effect only inflates the singular- or eigen-values anyway, so it has no effect on the singular vectors in any way. Moreover, the `aespca` function does not even call for the eigen-values at all, so this whole process is superfluous. The only wrinkle is adapting the `lars.lsa` and `aespca` functions to only operate on the data matrix.

Furthermore, the `lars` function *can* take in the full data, instead of just a Grammian. As an enhancement, we should either update our copy of the `lars` function in `lars.lsa`, or make a call to the exported `lars` function. ENHANCEMENT.

Value

A matrix of the eigenvectors or left singular vectors in B transformed to be the left singular values of the original data matrix.

See Also

`aespca`; `lars.lsa`; `AESPCA_pVals`

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.  
# Use AESPCA_pVals() instead
```

olsTrain_fun	<i>Gene-specific Regularized Ordinary Least Squares fit statistics for supervised PCA</i>
--------------	---

Description

Model statistics for Ordinary Least Squares (OLS) regression by gene.

Usage

```
olsTrain_fun(x, y, s0.perc = NULL)
```

Arguments

- `x` An $p \times n$ predictor matrix.
- `y` A response vector.
- `s0.perc` Percentile of the standard error of the slope estimate to be used for regularization. The Default value of `NULL` will use the median of this distribution.

Details

This function calculates the Sxx, Syy, and Sxy sums from the gene-specific OLS models, then calculates estimates of the regression slopes for each gene and their corresponding regularized test statistics,

$$t = \hat{\beta} / (sd + e),$$

where e is a regularization parameter.

If `s0.perc` is `NULL`, then e is median of the `sd` values. Otherwise, e is set equal to `quantile(sd, s0.perc)`.

Value

A list of OLS model statistics:

- `tt` : The Student's t test statistic the slopes (β).
- `numer` : The estimate of β .
- `sd` : The standard error of the estimates for β (the standard error divided by the square root of `Sxx`).
- `fudge` : A regularization parameter. See Details for description.

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use superPCA_pVals() instead
```

OmicsCateg-class	<i>An S4 class for categorical responses within an OmicsPathway object</i>
------------------	--

Description

This creates the `OmicsCateg` class which extends the `OmicsPathway` master class.

Slots

`assayData_df` An $N \times p$ data frame with named columns.

`pathwaySet` A list of known gene pathways with two elements:

- `pathways` : A named list of character vectors. Each vector contains the names of the individual genes within that pathway as a vector of character strings. The names contained in these vectors must have non-empty overlap with the *column names* of the `assayData_df` data frame. The names of the pathways (the list elements themselves) should be the a shorthand representation of the full pathway name.
- `TERMS` : A character vector the same length as the `pathways` list with the proper names of the pathways.
- `setsize` : A named integer vector the same length as the `pathways` list with the number of genes in each pathway. This list item is calculated during the creation step of a `create_OmicsCateg` function call.

`response` A factor vector of length N : the dependent variable of a generalized linear regression exercise. Currently, we support binary factors only. We expect to extend support to n -ary responses in the next package version.

See Also

[OmicsPathway](#), [create_OmicsCateg](#)

OmicsPathway-class	<i>An S4 class for mass spectrometry or bio-assay data and gene pathway lists</i>
--------------------	---

Description

An S4 class for mass spectrometry or bio-assay data and gene pathway lists

Slots

assayData_df An $N \times p$ data frame with named columns.

pathwaySet A list of known gene pathways with two elements:

- pathways : A named list of character vectors. Each vector contains the names of the individual genes within that pathway as a vector of character strings. The names contained in these vectors must have non-empty overlap with the *column names* of the assayData_df data frame. The names of the pathways (the list elements themselves) should be the a shorthand representation of the full pathway name.
- TERMS : A character vector the same length as the pathways list with the proper names of the pathways.
- setsize : A named integer vector the same length as the pathways list with the number of genes in each pathway. This list item is calculated during the creation step of a create_OmicsPath function call.

See Also

[create_OmicsPath](#)

OmicsReg-class	<i>An S4 class for continuous responses within an OmicsPathway object</i>
----------------	---

Description

This creates the OmicsReg class which extends the OmicsPathway master class.

Slots

assayData_df An $N \times p$ data frame with named columns.

pathwaySet A list of known gene pathways with two elements:

- pathways : A named list of character vectors. Each vector contains the names of the individual genes within that pathway as a vector of character strings. The names contained in these vectors must have non-empty overlap with the *column names* of the assayData_df data frame. The names of the pathways (the list elements themselves) should be the a shorthand representation of the full pathway name.
- TERMS : A character vector the same length as the pathways list with the proper names of the pathways.
- setsize : A named integer vector the same length as the pathways list with the number of genes in each pathway. This list item is calculated during the creation step of a create_OmicsReg function call.

response A numeric vector of length N : the dependent variable in a regression exercise.

See Also

[OmicsPathway](#), [create_OmicsReg](#)

OmicsSurv-class	<i>An S4 class for survival responses within an OmicsPathway object</i>
-----------------	---

Description

This creates the OmicsSurv class which extends the OmicsPathway master class.

Slots

assayData_df An $N \times p$ data frame with named columns.

pathwaySet A list of known gene pathways with two elements:

- pathways : A named list of character vectors. Each vector contains the names of the individual genes within that pathway as a vector of character strings. The names contained in these vectors must have non-empty overlap with the *column names* of the assayData_df data frame. The names of the pathways (the list elements themselves) should be the a shorthand representation of the full pathway name.
- TERMS : A character vector the same length as the pathways list with the proper names of the pathways.
- setsize : A named integer vector the same length as the pathways list with the number of genes in each pathway. This list item is calculated during the creation step of a create_OmicsSurv function call.

eventTime A numeric vector with N observations corresponding to the last observed time of follow up.

eventObserved A logical vector with N observations indicating right-censoring. The values will be FALSE if the observation was censored (i.e., we did not observe an event).

See Also

[OmicsPathway](#), [create_OmicsSurv](#)

pathway_pValues	<i>Calculate the p-values from a mixture of Weibull Extreme Value Distributions for supervised PCA</i>
-----------------	--

Description

Calculate pathway-specific p -values for supervised PCA and their associated False Discovery Rates (FDR).

Usage

```
pathway_pValues(optimParams_vec, max_tScores_vec, genelist_ls,
  FDRadjust = TRUE, multTestProc = "BH")
```

Arguments

optimParams_vec	A named vector of the estimated values for the parameters which minimize the likelihood as returned by the function weibullMix_optimParams .
max_tScores_vec	A vector of the maximum absolute t -scores for each pathway when under the alternative model (the response vector as is).
genelist_ls	A list of three elements: <ul style="list-style-type: none"> • pathways : A list of character vectors such that each vector contains the ID numbers (as a character) of the individual genes within that pathway as a vector of character strings. • TERMS : A character vector containing the names of the gene pathways. • setsize : A named integer vector containing the number of genes in each gene pathway.
FDRadjust	Should the p -values be adjusted for multiple comparisons? Defaults to TRUE.
multTestProc	If the p -values should be adjusted, which procedure should be used? Options are passed to the adjustRaw_pVals function. Specify multiple procedures via <code>c(...)</code> . Defaults to "BH".

Details

This function takes in the optimal parameters returned by the [weibullMix_optimParams](#) function, the maximum t -scores for each gene pathway, and the list of gene pathway information. This function will calculate the p -value for each t -score given the Gumbel Extreme Value mixture distribution parametrized by the values returned by the [weibullMix_optimParams](#) function. If requested, this function will also calculate the FDR associated with all pathway p -values via requested FDR-adjustment procedure. The default procedure is the Benjamini & Hochberg (1995) step-up FDR-controlling procedure, but any procedure implemented in the [adjustRaw_pVals](#) function is available.

Value

A data frame with columns for the pathway names, pathway set sizes, raw pathway p -values, and a column of FDR-adjusted p -values for each adjustment method specified.

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use superPCA_pVals() instead
```

pathway_tControl	<i>Calculate pathway-specific Student's t-scores from a null distribution for supervised PCA</i>
------------------	---

Description

Randomly permute or parametrically resample the response vector before model analysis. Then extract principal components (PCs) from the gene pathway, and return the test statistics associated with the first numPCs principal components at a set of threshold values based on the permuted values of the response.

Usage

```
pathway_tControl(pathway_vec, geneArray_df, response_mat,
  responseType = c("survival", "regression", "classification"),
  parametric = FALSE, n.threshold = 20, numPCs = 1, min.features = 3)
```

Arguments

pathway_vec	A character vector of the measured -Omes in the chosen gene pathway. These should match a subset of the rownames of the gene array.
geneArray_df	A "tall" pathway data frame ($p \times N$). Each subject or tissue sample is a column, and the rows are the -Ome measurements for that sample.
response_mat	A response matrix corresponding to responseType. For "regression" and "classification", this will be an $N \times 1$ matrix of response values. For "survival", this will be an $N \times 2$ matrix with event times in the first column and observed event indicator in the second.
responseType	A character string. Options are "survival", "regression", and "classification".
parametric	Should the random sample be taken using a parametric bootstrap sample? Defaults to FALSE.
n.threshold	The number of bins into which to split the feature scores in the fit object returned internally by the superpc.train function.
numPCs	The number of PCs to extract from the pathway.
min.features	What is the smallest number of genes allowed in each pathway? This argument must be kept constant across all calls to this function which use the same pathway list. Defaults to 3.

Details

This is a wrapper function to call [superpc.train](#) and [superpc.st](#) after response sampling or permutation with the [randomControlSample](#) suite of functions. This response randomization will act as a null distribution against which to compare the results from the [pathway_tScores](#) function.

This wrapper is designed to facilitate apply calls (in parallel or serially) of these two functions over a list of gene pathways. When numPCs is equal to 1, we recommend using a simplify-style apply variant, such as [sapply](#) (shown in [lapply](#)) or [parSapply](#) (shown in [clusterApply](#)), then transposing the resulting matrix.

Value

A matrix with numPCs rows and n.threshold columns. The matrix values are model t -statistics for each PC included (rows) at each threshold level (columns).

See Also

[pathway_tScores](#); [randomControlSample](#); [superpc.train](#); [superpc.st](#)

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use superPCA_pVals() instead
```

pathway_tScores	<i>Calculate pathway-specific Student's t-scores for supervised PCA</i>
-----------------	---

Description

Extract principal components (PCs) from the gene pathway, and return the test statistics associated with the first numPCs principal components at a set of threshold values.

Usage

```
pathway_tScores(pathway_vec, geneArray_df, response_mat,
  responseType = c("survival", "regression", "classification"),
  n.threshold = 20, numPCs = 1, min.features = 3)
```

Arguments

pathway_vec	A character vector of the measured -Omes in the chosen gene pathway. These should match a subset of the rownames of the gene array.
geneArray_df	A "tall" pathway data frame ($p \times N$). Each subject or tissue sample is a column, and the rows are the -Ome measurements for that sample.
response_mat	A response matrix corresponding to responseType. For "regression" and "classification", this will be an $N \times 1$ matrix of response values. For "survival", this will be an $N \times 2$ matrix with event times in the first column and observed event indicator in the second.
responseType	A character string. Options are "survival", "regression", and "classification".
n.threshold	The number of bins into which to split the feature scores in the fit object returned internally by the superpc.train function.
numPCs	The number of PCs to extract from the pathway.
min.features	What is the smallest number of genes allowed in each pathway? This argument must be kept constant across all calls to this function which use the same pathway list. Defaults to 3.

Details

This is a wrapper function to call [superpc.train](#) and [superpc.st](#). This wrapper is designed to facilitate apply calls (in parallel or serially) of these two functions over a list of gene pathways. When numPCs is equal to 1, we recommend using a simplify-style apply variant, such as [sapply](#) (shown in [lapply](#)) or [parSapply](#) (shown in [clusterApply](#)), then transposing the resulting matrix.

Value

A matrix with numPCs rows and n.threshold columns. The matrix values are model t -statistics for each PC included (rows) at each threshold level (columns).

See Also

[superpc.train](#); [superpc.st](#)

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use superPCA_pVals() instead
```

permTest_OmicsCateg	<i>AES-PCA permutation test of categorical response for pathway PCs</i>
---------------------	---

Description

Given an OmicsCateg object and a list of pathway PCs from the [extract_aesPCs](#) function, test if each expressed pathway in the bio-assay design matrix is significantly related to the categorical response.

Usage

```
permTest_OmicsCateg(OmicsCateg, pathwayPCs_ls, numReps = 1000,
  parallel = FALSE, numCores = NULL, ...)
```

```
## S4 method for signature 'OmicsCateg'
permTest_OmicsCateg(OmicsCateg, pathwayPCs_ls,
  numReps = 1000, parallel = FALSE, numCores = NULL, ...)
```

Arguments

OmicsCateg	A data object of class OmicsCateg, created by the create_OmicsCateg function.
pathwayPCs_ls	A list of pathway PC matrices returned by the extract_aesPCs function.
numReps	How many permuted models to fit? Defaults to 1000.
parallel	Should the computation be completed in parallel? Defaults to FALSE.
numCores	If parallel = TRUE, how many cores should be used for computation? Defaults to NULL.
...	Dots for additional internal arguments (currently unused).

Details

This function takes in a list of the first principal components from each pathway and an object of class OmicsCateg. This function will then calculate the AIC of a multivariate generalized linear model (via the [glm](#) function with a [binomial](#) error family) with the original observations as response and the pathway principal components as the predictor matrix.

Then, this function will create numReps permutations of the classification response, fit models to each of these permuted responses (holding the path predictor matrix fixed), and calculate the AIC of each model. This function will return a named vector of permutation p -values, where the value for each pathway is the proportion of models for which the AIC of the permuted response model is less than the AIC of the original model.

In future versions, this function will also be able to calculate permuted p -values for multinomial logistic regression and proportional odds logistic regression models, for n-ary and ordered categorical responses, respectively.

Value

A named vector of pathway permutation p -values.

See Also

[create_OmicsCateg](#); [extract_aesPCs](#); [glm](#); [binomial](#); [sample_Classifresp](#)

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use AESPCA_pVals() instead
```

permTest_OmicsReg	<i>AES-PCA permutation test of continuous response for pathway PCs</i>
-------------------	--

Description

Given an OmicsReg object and a list of pathway PCs from the [extract_aesPCs](#) function, test if each expressed pathway in the bio-assay design matrix is significantly related to the continuous response.

Usage

```
permTest_OmicsReg(OmicsReg, pathwayPCs_ls, numReps = 1000, parallel = FALSE,
  numCores = NULL, ...)

## S4 method for signature 'OmicsReg'
permTest_OmicsReg(OmicsReg, pathwayPCs_ls,
  numReps = 1000, parallel = FALSE, numCores = NULL, ...)
```

Arguments

OmicsReg	A data object of class OmicsReg, created by the create_OmicsReg function.
pathwayPCs_ls	A list of pathway PC matrices returned by the extract_aesPCs function.
numReps	How many permuted models to fit? Defaults to 1000.
parallel	Should the computation be completed in parallel? Defaults to FALSE.
numCores	If parallel = TRUE, how many cores should be used for computation? Defaults to NULL.
...	Dots for additional internal arguments (currently unused).

Details

This function takes in a list of the first principal components from each pathway and an object of class OmicsReg. This function will then calculate the AIC of a multivariate linear model (via the [lm](#) function) with the original observations as response and the pathway principal components as the predictor matrix.

Then, this function will create numReps permutations of the regression response, fit models to each of these permuted responses (holding the path predictor matrix fixed), and calculate the AIC of each model. This function will return a named vector of permutation p -values, where the value for each pathway is the proportion of models for which the AIC of the permuted response model is less than the AIC of the original model.

Value

A named vector of pathway permutation p -values.

See Also

[create_OmicsReg](#); [extract_aesPCs](#); [lm](#); [sample_Regresp](#)

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use AESPCA_pVals() instead
```

permTest_OmicsSurv	<i>AES-PCA permutation test of survival response for pathway PCs</i>
--------------------	--

Description

Given an OmicsSurv object and a list of pathway principal components (PCs) from the [extract_aesPCs](#) function, test if each expressed pathway in the bio-assay design matrix is significantly related to the survival output.

Usage

```
permTest_OmicsSurv(OmicsSurv, pathwayPCs_ls, numReps = 1000,
  parallel = FALSE, numCores = NULL, ...)

## S4 method for signature 'OmicsSurv'
permTest_OmicsSurv(OmicsSurv, pathwayPCs_ls,
  numReps = 1000, parallel = FALSE, numCores = NULL, ...)
```

Arguments

OmicsSurv	A data object of class OmicsSurv, created by the create_OmicsSurv function.
pathwayPCs_ls	A list of pathway PC matrices returned by the extract_aesPCs function.
numReps	How many permuted models to fit? Defaults to 1000.
parallel	Should the computation be completed in parallel? Defaults to FALSE.
numCores	If parallel = TRUE, how many cores should be used for computation? Defaults to NULL.
...	Dots for additional internal arguments (currently unused).

Details

This function takes in a list of the first principal components from each pathway and an object of class OmicsSurv. This function will then calculate the AIC of a Cox Proportional Hazards model (via the [coxph](#) function) with the original observations as response and the pathway principal components as the predictor matrix.

Then, this function will create numReps permutations of the survival response, fit models to each of these permuted responses (holding the path predictor matrix fixed), and calculate the AIC of each model. This function will return a named vector of permutation p -values, where the value for each pathway is the proportion of models for which the AIC of the permuted response model is less than the AIC of the original model.

Value

A named vector of pathway permutation p -values.

See Also

[create_OmicsSurv](#); [extract_aesPCs](#); [coxph](#); [sample_Survivalresp](#)

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use AESPCA_pVals() instead
```

print.pathwaySet	<i>Display the Summary of a pathwaySet-class Object.</i>
------------------	--

Description

The display method for gene set / pathway set lists as returned by the [read_gmt](#) function.

Usage

```
## S3 method for class 'pathwaySet'
print(x, ...)
```

Arguments

x	An object of class pathwaySet.
...	Lazy dots for additional internal arguments (currently unused).

Details

This function sets a print method for pathwaySet objects.

Value

x, returned invisibly (with the [invisible](#) function).

See Also

[read_gmt](#)

Examples

```
### Load the Example Data ###
data("colon_pathwaySet")

### Print / Show ###
colon_pathwaySet
```

randomControlSample	<i>Parametric bootstrap and non-parametric permutations of a response vector or matrix</i>
---------------------	--

Description

Create a random parametric bootstrap sample or a permutation of the input response vector or matrix (for survival outcomes).

Usage

```
sample_Survivalresp(response_vec, event_vec, parametric = FALSE)
```

```
sample_Regresp(response_vec, parametric = FALSE)
```

```
sample_Classifresp(response_vec, parametric = FALSE)
```

Arguments

response_vec	The dependent vector to sample from. For survival response, this is the vector of event times. For regression or n-ary classification, this is the vector of responses.
event_vec	The death / event observation indicator vector for survival response. This is coded as 0 for a right-censoring occurrence and 1 for a recorded event.
parametric	Should the random sample be taken using a parametric bootstrap sample? Defaults to FALSE.

Details

The distributions (for `parametric = TRUE`) are Weibull for survival times, Normal for regression, and n-ary Multinomial for classification. Distributional parameters are estimated with their maximum likelihood estimates. When `parametric = FALSE`, the response vector or survival matrix is simply permuted by row.

Value

A permutation of the supplied response (if `parametric = TRUE`). Otherwise, a parametric bootstrap sample of the response.

Examples

```
# DO NOT CALL THESE FUNCTIONS DIRECTLY.  
# Use AESPCA_pVals() or superPCA_pVals() instead
```

read_gmt

*Read a .gmt file in as a pathwaySet object***Description**

Read a gene set file in Gene Matrix Transposed (.gmt) format, with special performance consideration for large files. Present this object as a pathwaySet object.

Usage

```
read_gmt(file, delim = "\t")
```

Arguments

file	A path to a file or a connection. This file must be a .gmt file, otherwise input will likely be nonsense. See the "Details" section for more information.
delim	The .gmt delimiter. As proper .gmt files are tab delimited, this defaults to "\t".

Details

This function uses R's [readChar](#) function to improve character input performance over [readLines](#) (and far improve input performance over [scan](#)).

See the Broad Institute's "Data Formats" page for a description of the Gene Matrix Transposed file format: https://software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats#GMT:_Gene_Matrix_Transposed_file_format_.28.2A.gmt.29

Value

A pathwaySet list of gene sets. This list has three elements:

- pathways : A named list of character vectors. Each vector contains the names of the individual genes within that pathway as a vector of character strings.
- TERMS : A character vector the same length as the pathways list with the proper names of the pathways.
- GSEA_link : A character vector the same length as the pathways list with hyperlinks to the MSigDB description card for that pathway.

See Also

[print.pathwaySet](#)

Examples

```
## Not run:
# If you have installed the package:
data_path <- system.file("extdata", "c2.cp.v6.0.symbols.gmt",
                        package = "pathwayPCA", mustWork = TRUE)
geneset_ls <- read_gmt(data_path)

# If you are using the development version from GitHub:
geneset_ls <- read_gmt("inst/extdata/c2.cp.v6.0.symbols.gmt")
```

```
## End(Not run)
```

```
show,OmicsPathway-method
```

Display the Summary of an Omics-class Object.*

Description

The display method for objects of class OmicsPathway, OmicsSurv, OmicsReg, or OmicsCateg.

Usage

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

Arguments

object	An object inheriting the super-class OmicsPathway. This class includes objects of class OmicsSurv, OmicsReg, or OmicsCateg.
--------	---

Details

S4 objects print to the screen via the [show](#) function. This function sets a show method for OmicsPathway objects.

Value

A copy of object, returned invisibly (with the [invisible](#) function).

Examples

```
### Load the Example Data ###
data("colonSurv_df")
data("colon_pathwaySet")

### Create an OmicsSurv Object ###
colon_OmicsSurv <- create_OmicsSurv(assayData_df = colonSurv_df[, -(1:2)],
                                   pathwaySet_ls = colon_pathwaySet,
                                   eventTime_num = colonSurv_df$OS_time,
                                   eventObserved_lgl = as.logical(colonSurv_df$OS_event))

### Print / Show ###
colon_OmicsSurv
```

superpc.st

*Extract and test principal components from supervised PCA***Description**

Identify p_{path} significant features, extract principal components (PCs) from those specific features to construct a data matrix, predict the response with this data matrix, and record the model fit statistic of this prediction.

Usage

```
superpc.st(fit, data, n.threshold = 20, threshold.ignore = 0, n.PCs = 1,
  min.features = 3, epsilon = 1e-06)
```

Arguments

fit	An object of class superpc returned by the function <code>superpc.train</code> .
data	A list of test data: <ul style="list-style-type: none"> • x : A "tall" pathway data frame ($p_{path} \times N$). • y : A response vector corresponding to type. • <code>censoring.status</code> : If type = "survival", the censoring indicator (1 – the observed event indicator). Otherwise, NULL. • <code>featurenames</code> : A character vector of the measured -Omes in x.
n.threshold	The number of bins into which to split the feature scores returned in the <code>fit</code> object.
threshold.ignore	Calculate the model for feature scores above this percentile of the threshold. We have observed that the smallest threshold values (0% - 40%) largely have no effect on model t -scores. Defaults to 0.00 (0%).
n.PCs	The number of PCs to extract from the pathway.
min.features	What is the smallest number of genes allowed in each pathway? This argument must be kept constant across all calls to this function which use the same pathway list. Defaults to 3.
epsilon	I'm not sure why this is important. It's called when comparing the absolute score values to each value of the threshold vector. Defaults to 10^{-6} .

Details

NOTE: the number of thresholds at which to test (`n.threshold`) can be larger than the number of features to bin. This will result in constant t -statistics for the first few bins because the model isn't changing.

See https://web.stanford.edu/~hastie/Papers/spca_JASA.pdf.

Value

A list containing:

- `thresholds` : A labelled vector of quantile values of the score vector in the `fit` object.

- `n.threshold` : The number of splits to make in the score vector.
- `scor` : A matrix of model fit statistics. Each column is the threshold level of predictors allowed into the model, and each row is a PC included. Which genes are included in the matrix before PC extraction is governed by comparing their model score to the quantile value of the scores at each threshold value.
- `tscor` : A matrix of model *t*-statistics for each PC included (rows) at each threshold level (columns).
- `type` : Which model was called? Options are survival, regression, or binary.

See Also

[superpc.train](#); [superPCA_pVals](#)

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use superPCA_pVals() instead
```

superpc.train	<i>Train a supervised PCA model</i>
---------------	-------------------------------------

Description

Computes feature scores for p_{path} features of a pathway via supervised principal component analysis.

Usage

```
superpc.train(data, type = c("survival", "regression", "classification"),
  s0.perc = NULL)
```

Arguments

<code>data</code>	A list of test data: <ul style="list-style-type: none"> • <code>x</code> : A "tall" pathway data frame ($p_{path} \times N$). • <code>y</code> : A response vector corresponding to type. • <code>censoring.status</code> : If type = "survival", the censoring indicator (1—the observed event indicator. Otherwise, NULL. • <code>featurenames</code> : A character vector of the measured -Omes in x.
<code>type</code>	What model relates y and x? Options are "survival", "regression", or "classification".
<code>s0.perc</code>	A stabilization parameter on the interval $[0, 1]$. This is an internal argument to each of the called functions. The default value is NULL to ensure an appropriate value is determined internally.

Details

This function is a [switch](#) call to [coxTrain_fun](#) (for type = "survival"), [olsTrain_fun](#) (for type = "regression"), or [glmTrain_fun](#) (for type = "classification").

Value

A list containing:

- `feature.scores` : The scaled p -dimensional score vector: each value has been divided by its respective standard deviation plus epsilon (governed by `s0.perc`). NA values returned by the logistic model are replaced with 0.
- `type` : The argument for `type`.
- `s0.perc` : The user-supplied value of `s0.perc`, or the internally-calculated default value from the chosen model.
- `call` : The output of `match.call` for the user-supplied function arguments.

See Also

[superpc.st](#); [superPCA_pVals](#)

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use superPCA_pVals() instead
```

superPCA_pVals

Test pathways with supervised PCA

Description

Given a supervised OmicsPath object (one of OmicsSurv, OmicsReg, or OmicsCateg), extract the first k principal components (PCs) from each expressed pathway in the -Omics assay design matrix, test their association with the response matrix, and return a data frame of the adjusted p -values for each pathway.

Usage

```
superPCA_pVals(object, n.threshold = 20, numPCs = 1, min.features = 3,
  parallel = FALSE, numCores = NULL, adjustpValues = TRUE,
  adjustment = c("Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD", "BH",
    "BY", "ABH", "TSBH"), ...)
```

```
## S4 method for signature 'OmicsPathway'
superPCA_pVals(object, n.threshold = 20,
  numPCs = 1, min.features = 3, parallel = FALSE, numCores = NULL,
  adjustpValues = TRUE, adjustment = c("Bonferroni", "Holm", "Hochberg",
    "SidakSS", "SidakSD", "BH", "BY", "ABH", "TSBH"), ...)
```

Arguments

<code>object</code>	An object of superclass OmicsPathway with a response matrix or vector.
<code>n.threshold</code>	The number of bins into which to split the feature scores in the fit object returned internally by the superpc.train function to the pathway_tScores and pathway_tControl functions. Defaults to 20. Smaller values may result in less accurate pathway p -values while larger values increase computation time.

numPCs	The number of PCs to extract from each pathway. Defaults to 1.
min.features	What is the smallest number of genes allowed in each pathway? This argument must be kept constant across all calls to this function which use the same pathway list. Defaults to 3.
parallel	Should the computation be completed in parallel? Defaults to FALSE.
numCores	If parallel = TRUE, how many cores should be used for computation? Defaults to NULL.
adjustpValues	Should you adjust the <i>p</i> -values for multiple comparisons? Defaults to TRUE.
adjustment	Character vector of procedures. The returned data frame will be sorted in ascending order by the first procedure in this vector, with ties broken by the unadjusted <i>p</i> -value. If only one procedure is selected, then it is necessarily the first procedure. See the documentation for the adjustRaw_pVals function for the adjustment procedure definitions and citations.
...	Dots for additional internal arguments.

Details

This is a wrapper function for the [pathway_tScores](#), [pathway_tControl](#), [weibullMix_optimParams](#), [weibullMix_pValues](#), and [adjust_and_sort](#) functions.

Value

A data frame with columns:

- pathways : The names of the pathways in the Omics* object (given in object@pathwaySet\$pathways.)
- setsize : The number of genes in each of the original pathways (given in the object@pathwaySet\$setsize object).
- terms : The pathway description, as given in the object@pathwaySet\$TERMS object.
- rawp : The unadjusted *p*-values of each pathway.
- ... : Additional columns as specified through the adjustment argument.

Some of the pathways in the supplied pathway set list will be removed, or "trimmed", during function execution. These trimmed pathways will have *p*-values given as NA. For an explanation of pathway trimming, see the documentation for the [expressedOmes](#) function.

The data frame will be sorted in ascending order by the method specified first in the adjustment argument. If adjustpValues = FALSE, then the data frame will be sorted by the raw *p*-values. If you have the suggested tidyverse package suite loaded, then this data frame will print as a [tibble](#). Otherwise, it will print as a data frame.

See Also

[expressedOmes](#); [create_OmicsPath](#); [create_OmicsSurv](#); [create_OmicsReg](#); [create_OmicsCateg](#); [pathway_tScores](#); [pathway_tControl](#); [weibullMix_optimParams](#); [weibullMix_pValues](#); [adjust_and_sort](#)

Examples

```
## Not run:
### Load the Example Data ###
data("colonSurv_df")
data("colon_pathwaySet")
```

```

#### Create an OmicsSurv Object ####
colon_OmicsSurv <- create_OmicsSurv(assayData_df = colonSurv_df[, -(1:2)],
                                   pathwaySet_ls = colon_pathwaySet,
                                   eventTime_num = colonSurv_df$OS_time,
                                   eventObserved_lgl = as.logical(colonSurv_df$OS_event))

#### Calculate Pathway p-Values ####
colonSurv_pVals_df <- superPCA_pVals(object = colon_OmicsSurv,
                                     parallel = TRUE,
                                     numCores = 2,
                                     adjustpValues = TRUE,
                                     adjustment = c("Hoch", "SidakSD"))

## End(Not run)

```

topGenes	<i>Rank -Omes by adjusted significance given in a ranked-pathways data frame</i>
----------	--

Description

Given a supervised Omics*-class object and a ranked pathways data frame returned by either the [AESPCA_pVals](#) or [superPCA_pVals](#) functions, calculate the weighted rank the genes / proteins / lipids / metabolomes / transcriptomes contained in each pathway by the significance of their container pathways.

Usage

```

topGenes(object, pVals_df, percentile = 0.01)

## S4 method for signature 'OmicsPathway'
topGenes(object, pVals_df, percentile = 0.01)

```

Arguments

object	An object of class OmicsSurv, OmicsReg, or OmicsCateg.
pVals_df	The ranked pathways data frame returned by either the AESPCA_pVals or superPCA_pVals functions. Missing <i>p</i> -values (from trimmed pathways) are omitted.
percentile	Return the most significant <i>q</i> percent of the features contained in all pathways. Defaults to 0.01.

Details

This function takes in the pathway set information in a valid Omics*-class object and a data frame of ranked pathways (as returned by either the [AESPCA_pVals](#) or [superPCA_pVals](#) functions). This function creates a matrix with pathways as the columns and all genes included in those pathways as the rows: the i, j entry of the matrix equals 1 if gene i is an element of pathway j . (This is created after trimming the pathways to the assay data frame supplied using the [expressedOmes](#) function.) The topGenes function then multiplies each pathway membership indicator column by the negative natural logarithm of the adjusted *p*-values for that pathway; if multiple FDR adjustment methods are used, then the score is the average of each negative logged *p*-values. This function then returns two named numeric vectors: the sum of these gene scores and the means of the non- zero gene scores, sorted in descending order.

Value

A list of two named numeric vectors. For both vectors, the names are the genes, and the values are the scores for those genes. The first vector is the sum of scores across all pathways; the second vector is this score sum divided by the number of pathways which contain that particular gene. The summed vector does not adjust for genes which appear more frequently in pathways, while the averaged vector does. See "Details" for more information.

Examples

```
## Not run:
### Load the Example Data ###
data("colonSurv_df")
data("colon_pathwaySet")

### Create an OmicsSurv Object ###
colon_OmicsSurv <- create_OmicsSurv(assayData_df = colonSurv_df[, -(1:2)],
                                   pathwaySet_ls = colon_pathwaySet,
                                   eventTime_num = colonSurv_df$OS_time,
                                   eventObserved_lgl = as.logical(colonSurv_df$OS_event))

### Calculate Pathway p-Values ###
colonSurv_pVals_df <- superPCA_pVals(object = colon_OmicsSurv,
                                     parallel = TRUE,
                                     numCores = 2,
                                     adjustpValues = TRUE,
                                     adjustment = c("Hoch", "SidakSD"))

### Find the Top Genes ###
topGenes(object = colon_OmicsSurv, pVals_df = colonSurv_pVals_df)

## End(Not run)
```

transpose_assay

Transpose an Assay (Data Frame)

Description

Transpose an object of class `data.frame` that contains assay measurements while preserving row (feature) and column (sample) names.

Usage

```
transpose_assay(assay_df, firstColIsFeatureNames = TRUE)
```

Arguments

`assay_df` A data frame with numeric values to transpose

`firstColIsFeatureNames`

Are the data row names in the first column of `df`? Defaults to `TRUE`. If not, this function assumes that the data row names are accessible by the [rownames](#) function.

Details

This function is designed to transpose "tall" assay data frames (where genes or proteins are the rows and patient or tumour samples are the columns). This function also transposes the row (feature) names to column names and the column (sample) names to row names. Notice that all rows and columns (other than the feature name column, as applicable) are numeric.

Value

The transposition of df, with row and column names preserved.

Examples

```
x_mat <- matrix(rnorm(5000), ncol = 20, nrow = 250)
rownames(x_mat) <- paste0("gene_", 1:250)
colnames(x_mat) <- paste0("sample_", 1:20)
x_df <- as.data.frame(x_mat, row.names = rownames(x_mat))

transpose_assay(x_df, firstColIsFeatureNames = FALSE)
```

valid_OmicsSurv

Check validity of new Omics-class objects*

Description

These functions check the validity of new objects created in the OmicsSurv, OmicsReg, and OmicsCateg classes.

Usage

```
valid_OmicsSurv(object)
```

```
valid_OmicsReg(object)
```

```
valid_OmicsCateg(object)
```

Arguments

object An object potentially of class OmicsSurv, OmicsReg, or OmicsCateg.

Details

We have currently written checks to make sure the dimensions of the mass spectrometry or bio-assay data frame and response matrices or vectors match. Other checks should be added in response to user feedback during or after beta testing. ENHANCEMENT.

Value

TRUE if the object is a valid object, else an error message with the rule broken.

OmicsSurv

Valid OmicsSurv objects will have two response vectors: a vector of the most recently recorded follow-up times and a logical vector if that time marks a death or event (TRUE: observed event; FALSE: right-censored observation).

OmicsReg and OmicsCateg

Valid OmicsReg and OmicsCateg objects will have one response vector of continuous (numeric) or categorical (factor) observations, respectively.

weibullMix_optimParams

Calculate the optimal parameters for a mixture of Weibull Extreme Value Distributions for supervised PCA

Description

Calculate the parameters which minimise the negative log-likelihood of a mixture of two Weibull Extreme Value distributions.

Usage

```
weibullMix_optimParams(max_tControl_vec, pathwaySize_vec, initialVals = c(p =
  0.5, mu1 = 1, s1 = 0.5, mu2 = 1, s2 = 0.5), optimMethod = "L-BFGS-B",
  lowerBD = c(0, -Inf, 0, -Inf, 0), upperBD = c(1, Inf, Inf, Inf, Inf))
```

Arguments

max_tControl_vec

A vector of the maximum absolute t -scores for each pathway (returned by the [pathway_tControl](#) function) when under the null model. Under the null model, the response vector will have been randomly generated or parametrically bootstrapped.

pathwaySize_vec

A vector of the number of genes in each pathway.

initialVals

A named vector of initial values for the Weibull parameters. The values are

- p : The mixing proportion between the Gumbel minimum and Gumbel maximum distributions. This parameter is bounded by $[0, 1]$ and defaults to 0.5.
- μ_1 : The mean of the first distribution. This parameter is unbounded and defaults to 1.
- s_1 : The precision of the first distribution. This parameter is bounded below by 0 and defaults to 0.5.
- μ_2 : The mean of the second distribution. This parameter is unbounded and defaults to 1.
- s_2 : The precision of the second distribution. This parameter is bounded below by 0 and defaults to 0.5.

optimMethod	Which numerical optimization routine to pass to the <code>optim</code> function. Defaults to "L-BFGS-B", which allows for lower and upper bound constraints. When this option is specified, lower and upper bounds for ALL parameters must be supplied.
lowerBD	A vector of the lower bounds on the <code>initialVals</code> . Defaults to <code>c(0, -Inf, 0, -Inf, 0)</code> .
upperBD	A vector of the upper bounds on the <code>initialVals</code> . Defaults to <code>c(1, Inf, Inf, Inf, Inf)</code> .

Details

The likelihood function is equation (4) in Chen et al (2008): a mixture of two Gumbel Extreme Value probability density functions, with mixing proportion p . Within the code of this function, the values `mu1`, `mu2` and `s1`, `s2` are placeholders for the mean and precision, respectively.

A computational note: the "L-BFGS-B" option within the `optim` function requires a bounded function or likelihood. We therefore replaced `Inf` with 10^{200} in the check for boundedness. As we are attempting to minimise the negative log-likelihood, this maximum machine value is effectively `+Inf`.

See <https://doi.org/10.1093/bioinformatics/btn458> for more information.

Value

A named vector of the estimated values for the parameters which minimize the negative log-likelihood of the mixture Weibull Extreme Value distributions.

See Also

`optim`; `weibullMix_pValues`; `pathway_tControl`; `superPCA_pVals`

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.
# Use superPCA_pVals() instead.
```

<code>weibullMix_pValues</code>	<i>Calculate the p-values from an optimal mixture of Weibull Extreme Value distributions for supervised PCA</i>
---------------------------------	---

Description

Calculate the p -values of test statistics from a mixture of two Weibull Extreme Value distributions.

Usage

```
weibullMix_pValues(tScore_vec, pathwaySize_vec, optimParams_vec)
```

Arguments

<code>tScore_vec</code>	A vector of the maximum absolute t -scores for each pathway (returned by the <code>pathway_tScores</code> function) when under the alternative model.
<code>pathwaySize_vec</code>	A vector of the number of genes in each pathway.
<code>optimParams_vec</code>	The <i>NAMED</i> vector of optimal Weibull Extreme Value mixture distribution parameters returned by the <code>weibullMix_optimParams</code> function.

Details

The likelihood function is equation (4) in Chen et al (2008): a mixture of two Gumbel Extreme Value probability density functions, with mixing proportion p . Within the code of this function, the values μ_1 , μ_2 and s_1 , s_2 are placeholders for the mean and precision, respectively.

See <https://doi.org/10.1093/bioinformatics/btn458> for more information.

Value

A named vector of the estimated raw p -values for each gene pathway.

See Also

[weibullMix_optimParams](#); [pathway_tScores](#); [superPCA_pVals](#)

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY.  
# Use superPCA_pVals() instead.
```

Index

*Topic **datasets**

- colon_pathwaySet, 10
- colonSurv_df, 9
- adjust_and_sort, 5, 8, 43
- adjustRaw_pVals, 3, 5, 6, 8, 30, 43
- aespca, 6, 18, 26
- AESPCA_pVals, 7, 7, 26, 44
- AESPCA_pVals, OmicsPathway-method (AESPCA_pVals), 7
- binomial, 33, 34
- clusterApply, 31, 32
- colon_pathwaySet, 10
- colonSurv_df, 9, 10
- coxph, 35, 36
- coxTrain_fun, 10, 41
- create_Omics, 11
- create_OmicsCateg, 8, 11, 12, 19, 22, 27, 33, 34, 43
- create_OmicsCateg (create_OmicsPath), 13
- create_OmicsPath, 8, 11, 12, 13, 18, 19, 28, 43
- create_OmicsReg, 8, 11, 12, 19, 22, 29, 34, 35, 43
- create_OmicsReg (create_OmicsPath), 13
- create_OmicsSurv, 8, 11, 12, 19, 21, 29, 35, 36, 43
- create_OmicsSurv (create_OmicsPath), 13
- create_pathwaySet, 15
- eigen, 24
- expressedOmes, 5, 8, 16, 17, 18, 43, 44
- expressedOmes, OmicsPathway-method (expressedOmes), 16
- extract_aesPCs, 7, 8, 17, 33–36
- extract_aesPCs, OmicsPathway-method (extract_aesPCs), 17
- family, 22
- fast.svd, 25
- getAssay, 18
- getAssay, OmicsPathway-method (getAssay), 18
- getAssay<- (getAssay), 18
- getAssay<- , OmicsPathway-method (getAssay), 18
- getEvent (getEventTime), 20
- getEvent, OmicsSurv-method (getEventTime), 20
- getEvent<- (getEventTime), 20
- getEvent<- , OmicsSurv-method (getEventTime), 20
- getEventTime, 20
- getEventTime, OmicsSurv-method (getEventTime), 20
- getEventTime<- (getEventTime), 20
- getEventTime<- , OmicsSurv-method (getEventTime), 20
- getPathwaySet (getAssay), 18
- getPathwaySet, OmicsPathway-method (getAssay), 18
- getPathwaySet<- (getAssay), 18
- getPathwaySet<- , OmicsPathway-method (getAssay), 18
- getResponse, 21
- getResponse, OmicsPathway-method (getResponse), 21
- getResponse<- (getResponse), 21
- getResponse<- , OmicsPathway-method (getResponse), 21
- glm, 22, 33, 34
- glmTrain_fun, 22, 41
- invisible, 36, 39
- lapply, 31, 32
- lars, 23, 26
- lars.lsa, 7, 23, 26
- list, 15
- lm, 34, 35
- match.call, 42
- matrixRoot, 24
- multinom, 22
- mysvd, 24

- normalize, [7, 25](#)
- olsTrain_fun, [26, 41](#)
- OmicsCateg, [12, 14](#)
- OmicsCateg-class, [27](#)
- OmicsPathway, [12, 14, 27, 29](#)
- OmicsPathway-class, [28](#)
- OmicsReg, [12, 14](#)
- OmicsReg-class, [28](#)
- OmicsSurv, [12, 14](#)
- OmicsSurv-class, [29](#)
- optim, [48](#)
- pathway_pValues, [29](#)
- pathway_tControl, [30, 42, 43, 47, 48](#)
- pathway_tScores, [31, 32, 42, 43, 48, 49](#)
- permTest_OmicsCateg, [5, 8, 33](#)
- permTest_OmicsCateg,OmicsCateg-method
(permTest_OmicsCateg), [33](#)
- permTest_OmicsReg, [5, 8, 34](#)
- permTest_OmicsReg,OmicsReg-method
(permTest_OmicsReg), [34](#)
- permTest_OmicsSurv, [5, 8, 35](#)
- permTest_OmicsSurv,OmicsSurv-method
(permTest_OmicsSurv), [35](#)
- polr, [22](#)
- print.pathwaySet, [36, 38](#)
- randomControlSample, [31, 37](#)
- read_gmt, [14–16, 36, 38](#)
- readChar, [38](#)
- readLines, [38](#)
- rownames, [45](#)
- rsvd, [25](#)
- sample_Classifresp, [34](#)
- sample_Classifresp
(randomControlSample), [37](#)
- sample_Regresp, [35](#)
- sample_Regresp(randomControlSample), [37](#)
- sample_Survivalresp, [36](#)
- sample_Survivalresp
(randomControlSample), [37](#)
- scan, [38](#)
- show, [39](#)
- show,OmicsPathway-method, [39](#)
- superpc.st, [22, 31, 32, 40, 42](#)
- superpc.train, [22, 31, 32, 40, 41, 41, 42](#)
- superPCA_pVals, [41, 42, 42, 44, 48, 49](#)
- superPCA_pVals,OmicsPathway-method
(superPCA_pVals), [42](#)
- Surv, [12](#)
- switch, [41](#)
- tibble, [6, 8, 43](#)
- topGenes, [44](#)
- topGenes,OmicsPathway-method
(topGenes), [44](#)
- transpose_assay, [45](#)
- valid_OmicsCateg, [19, 21](#)
- valid_OmicsCateg(valid_OmicsSurv), [46](#)
- valid_OmicsReg, [19, 21](#)
- valid_OmicsReg(valid_OmicsSurv), [46](#)
- valid_OmicsSurv, [19, 20, 46](#)
- weibullMix_optimParams, [30, 43, 47, 48, 49](#)
- weibullMix_pValues, [5, 43, 48, 48](#)