Package 'pathwayPCA'

January 20, 2026

Type Package

Title Integrative Pathway Analysis with Modern PCA Methodology and Gene Selection

Version 1.26.0

Description pathwayPCA is an integrative analysis tool that implements the principal component analysis (PCA) based pathway analysis approaches described in Chen et al. (2008), Chen et al. (2010), and Chen (2011). pathwayPCA allows users to: (1) Test pathway association with binary, continuous, or survival phenotypes. (2) Extract relevant genes in the pathways using the SuperPCA and AES-PCA approaches. (3) Compute principal components (PCs) based on the selected genes. These estimated latent variables represent pathway activities for individual subjects, which can then be used to perform integrative pathway analysis, such as multi-omics analysis. (4) Extract relevant genes that drive pathway significance as well as data corresponding to these relevant genes for additional in-depth analysis. (5) Perform analyses with enhanced computational efficiency with parallel computing and enhanced data safety with S4-class data objects. (6) Analyze studies with complex experimental designs, with multiple covariates, and with interaction effects, e.g., testing whether pathway association with clinical phenotype is different between male and female subjects.
Citations: Chen et al. (2008) <https://doi.org/10.1093/bioinformatics/btn458>; Chen et al. (2010) <https://doi.org/10.1002/gepi.20532>; and Chen (2011) <https://doi.org/10.2202/1544-6115.1697>.

License GPL-3

Depends R (>= 3.1)

Imports lars, methods, parallel, stats, survival, utils

Suggests airway, circlize, grDevices, knitr, RCurl, reshape2, rmarkdown, SummarizedExperiment, survminer, testthat, tidyverse

biocViews CopyNumberVariation, DNAMethylation, GeneExpression, SNP, Transcription, GenePrediction, GeneSetEnrichment, GeneSignaling, GeneTarget, GenomeWideAssociation, GenomicVariation, CellBiology, Epigenetics, FunctionalGenomics, Genetics, Lipidomics, Metabolomics, Proteomics, SystemsBiology, Transcriptomics, Classification, DimensionReduction, FeatureExtraction, PrincipalComponent, Regression, Survival, MultipleComparison, Pathways

Encoding UTF-8

LazyData false

RoxygenNote 7.2.3

Collate 'CreatePathwayCollection.R' 'createClass_OmicsPath.R'
'createClass_validOmics.R' 'accessClass_OmicsPath.R'
'createClass_OmicsSurv.R' 'accessClass_OmicsSurv.R'
'accessClass_OmicsRegCateg.R' 'createClass_OmicsCateg.R'
'createClass_OmicsReg.R' 'accessClass_OmicsPathData.R'
'accessClass_pathwayCollection.R'
'accessClass_pathwayCollection_which.R' 'accessClass_pcOut.R'
'accessClass_pcOutpVals.R' 'aesPC_calculate_AESPCA.R'
'aesPC_calculate_LARS.R' 'aesPC_extract_OmicsPath_PCs.R'
'aesPC_permtest_CoxPH.R' 'aesPC_permtest_GLM.R'
'aesPC_permtest_LM.R' 'aesPC_unknown_matrixNorm.R'
'aesPC_wrapper.R' 'createOmics_All.R'
'createOmics_CheckAssay.R'
'createOmics_CheckPathwayCollection.R'
'createOmics_CheckSampleIDs.R' 'createOmics_JoinPhenoAssay.R'
'createOmics_TrimPathwayCollection.R' 'createOmics_Wrapper.R'
'data_colonSubset.R' 'data_genesetSubset.R'
'data_wikipathways.R' 'data_wikipathways_symbols.R'
'pathwayPCA.R' 'printClass_Omics_All.R'
'printClass_pathwayCollection.R' 'superPC_model_CoxPH.R'
'superPC_model_GLM.R' 'superPC_model_LS.R'
'superPC_model_tStats.R' 'superPC_model_train.R'
'superPC_modifiedSVD.R' 'superPC_optimWeibullParams.R'
'superPC_optimWeibull_pValues.R' 'superPC_pathway_tControl.R'
'superPC_pathway_tScores.R' 'superPC_pathway_tValues.R'
'superPC_permuteSamples.R' 'superPC_wrapper.R'
'utils_Contains.R' 'utils_adjust_and_sort_pValues.R'
'utils_load_test_data_onto_PCs.R' 'utils_multtest_pvalues.R'
'utils_read_gmt.R' 'utils_stdExpr_2_tidyAssay.R'
'utils_transpose_assay.R' 'utils_write_gmt.R'

VignetteBuilder knitr

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

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

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

git_branch RELEASE_3_22

git_last_commit e20c694

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-01-19

Author Gabriel Odom [aut, cre],
James Ban [aut],
Lizhong Liu [aut],
Lily Wang [aut],
Steven Chen [aut]

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

Contents

`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

`X`	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.
`d`	The number of principal components (PCs) to extract from the pathway. Defaults to 1.
`max.iter`	The maximum number of times an internal `while()` loop can make calls to the `lars.lsa()` function. Defaults to 10.
`eps.conv`	A numerical convergence threshold for the same `while()` loop. Defaults to 0.001.
`adaptive`	Internal argument of the `lars.lsa()` function. Defaults to TRUE.
`para`	Internal argument of the `lars.lsa()` 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. Note that, if the number of features in the pathway exceeds the number of samples, this decompostion will be an approximation; also, the internal `lars.lsa` function may require more computing time than usual to converge (which is one of the reasons why, in practice, we usually remove pathways that have more than 200-300 features).

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:

- `aesLoad` : A $d \times p$ projection matrix of the d AES-PCs.
- `oldLoad` : A $d \times p$ projection matrix of the d PCs from the singular value decomposition (SVD).
- `aesScore` : 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`; `ExtractAESPCs`; `AESPCA_pVals`

Examples

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

## Not run:
  data("colonSurv_df")
  aespca(as.matrix(colonSurv_df[, 5:50]))

## End(Not run)
```

`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 pathway-subset of the features 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,
  numReps = 0L,
  parallel = FALSE,
  numCores = NULL,
  asPCA = FALSE,
  adjustpValues = TRUE,
 adjustment = c("Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD", "BH", "BY",
    "ABH", "TSBH"),
  ...
)

## S4 method for signature 'OmicsPathway'
```

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

Arguments

object	An object of class OmicsPathway with a response matrix or vector.
numPCs	The number of PCs to extract from each pathway. Defaults to 1.
numReps	How many permutations to estimate the p-value? Defaults to 0 (that is, to estimate the p-value parametrically). If numReps > 0, then the non-parametric, permutation p-value will be returned based on the number of random samples specified.
parallel	Should the computation be completed in parallel? Defaults to FALSE.
numCores	If parallel = TRUE, how many cores should be used for computation? Internally defaults to the number of available cores minus 1.
asPCA	Should the computation return the eigenvectors and eigenvalues instead of the adaptive, elastic-net, sparse principal components and their corresponding loadings. Defaults to FALSE; this should be used for diagnostic or comparative purposes only.
adjustpValues	Should you adjust the p-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 p-value. If only one procedure is selected, then it is necessarily the first procedure. See the documentation for the ControlFDR function for the adjustment procedure definitions and citations.
...	Dots for additional internal arguments.

Details

This is a wrapper function for the ExtractAESPCs, PermTestSurv, PermTestReg, and PermTestCateg functions.

Please see our Quickstart Guide for this package: https://gabrielodom.github.io/pathwayPCA/articles/Supplement1-Quickstart_Guide.html

Value

A results list with class aespcOut. This list has three components: a data frame of pathway details, pathway p-values, and potential adjustments to those values (pVals_df); a list of the first numPCs *score* vectors for each pathway (PCs_ls); and a list of the first numPCs feature loading vectors for each pathway (loadings_ls). The p-value data frame has columns:

- pathways : The names of the pathways in the Omics* object (given in object@trimPathwayCollection$pathways

- `setsize` : The number of genes in each of the original pathways (given in the `object@trimPathwayCollection$se` object).
- `n_tested` : The number of genes in each of the trimmed pathways (given in the `object@trimPathwayCollection$` object).
- `terms` : The pathway description, as given in the `object@trimPathwayCollection$TERMS` object.
- `rawp` : The unadjusted p-values of each pathway.
- `...` : Additional columns of adjusted p-values 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 print as a data frame.

See Also

`CreateOmics`; `ExtractAESPCs`; `PermTestSurv`; `PermTestReg`; `PermTestCateg`; `TabulatepValues`; `clusterApply`

Examples

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

###  Create an OmicsSurv Object  ###
colon_Omics <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "surv"
)

###  Calculate Pathway p-Values  ###
colonSurv_aespc <- AESPCA_pVals(
  object = colon_Omics,
  numReps = 0,
  parallel = TRUE,
  numCores = 2,
  adjustpValues = TRUE,
  adjustment = c("Hoch", "SidakSD")
)
```

`CheckAssay` *Check an Input Assay*

Description

Check the classes, dimensions, missingness, feature variance, feature type, and feature names of a data frame.

Usage

```
CheckAssay(df, removeNear0 = TRUE, epsilon = 10^-6)
```

Arguments

df	An assay data frame supplied to the CreateOmics function. The first column is assumed to be the sample IDs, and will be ignored. See CheckSampleIDs for checking sample IDs.
removeNear0	Should columns of df with variance near 0 be removed? Defaults to TRUE.
epsilon	Threshold to consider the variance of a column equal to 0. Defaults to 0.000001.

Details

This function checks that the data frame is not a matrix, that the data frame has more columns than rows (tidy genomic data), that the data frame contains no missing or character values, that no features of the data frame have variance less than epsilon (and removes such features if removeNear0 = TRUE), and checks the data frame for valid column names.

Value

The same data frame, without features with 0 variance, **if** that data frame passes all checks.

Examples

```
 # DO NOT CALL THIS FUNCTION DIRECTLY. CALL FROM WITHIN CreateOmics().

## Not run:
 data("colonSurv_df")
 CheckAssay(colonSurv_df[, -(1:3)])

## End(Not run)
```

CheckPwyColl *Check an Input Pathway Collection*

Description

Check the class and names of a pathwayCollection object. Add or fix names as appropriate. Add the setsize vector to the object.

Usage

```
CheckPwyColl(pwyColl_ls)
```

Arguments

pwyColl_ls	A pathway collection supplied to the CreateOmics function

Details

If there are no names, create them. If there are missing names, label them. If there are duplicated names (because R is stupid and allows duplicate element names in a list—but not a data frame!), then use the `data.frame` name rule to append a period followed by integers to the end of the name string.

Notes: if the supplied `pathways` object within your `pwyColl_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).

Value

The same pathway collection, but with names modified as described in "Details" and the number of genes per pathway as the `setsize` element of the collection object.

Examples

```
 # DO NOT CALL THIS FUNCTION DIRECTLY. CALL FROM WITHIN CreateOmics().

## Not run:
  data("colon_pathwayCollection")
  CheckPwyColl(colon_pathwayCollection)

## End(Not run)
```

CheckSampleIDs *Check Input Sample IDs*

Description

Check the class of the sample IDs and if they are unique. This assumes that the sample IDs are in the first column.

Usage

```
CheckSampleIDs(df)
```

Arguments

`df`	An assay or phenotype data frame supplied to the `CreateOmics` function

Details

This function checks that the sample IDs are unique, then coerces them from factor to character (if necessary), stores these IDs as the first column, then returns the same data frame.

Value

The same data frame, **if** the sample IDs pass sanity checks, with the sample IDs as a character vector.

Examples

```
 # DO NOT CALL THIS FUNCTION DIRECTLY. CALL FROM WITHIN CreateOmics().

## Not run:
 data("colonSurv_df")
 CheckSampleIDs(colonSurv_df[, -(2:3)])

## 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

```
data(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

GEO GSE17538 https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE17538

---

colon_pathwayCollection *Gene Pathway Subset*

Description

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

Usage

```
data(colon_pathwayCollection)
```

Format

A pathwayCollection 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 pathways 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

---

Contains *Check if a long atomic vector contains a short atomic vector*

Description

Check if any or all of the elements of a short atomic vector are contained within a supplied long atomic vector.

Usage

```
Contains(long, short, matches = c("any", "all"), partial = FALSE)
```

Arguments

long	A vector to possibly containing any or all elements of short
short	A short vector or scalar, some elements of which may be contained in long
matches	Should partial set matching of short be allowed? Defaults to "any", signifying that the function should return TRUE if any of the elements of short are contained in long. The other option is "all".
partial	Should partial string matching be allowed? Defaults to FALSE. Partial string matching means that the character string **starts with** the supplied value.

Details

This is a helper function to find out if a gene symbol or some similar character string (or character vector) is contained in a pathway. Currently, this function uses base R, but we can write it in a compiled language (such as C++) to increase speed later.

For partial matching (partial = TRUE), long must be an atomic vector of type character, short must be an atomic scalar (a vector with length of 1) of type character, and matches should be set to "any". Because this function is designed to match gene symbols or CpG locations, we care if the symbol or location starts with the string supplied. For example, if we set short = "PIK", then we want to find if any of the gene symbols in the supplied long vector belong to the PIK gene family. We don't care if this string appears elsewhere in a gene symbol.

Value

A logical scalar. If `matches = "any"`, this indicates if any of the elements of `short` are contained in `long`. If `matches = "all"`, this indicates if all of the elements of `short` are contained in `long`. If `partial = TRUE`, the returned logical indicates whether or not any of the character strings in `long` **start with** the character scalar supplied to `short`.

Examples

```
Contains(1:10, 8)
Contains(LETTERS, c("A", "!"), matches = "any")
Contains(LETTERS, c("A", "!"), matches = "all")

genesPI <- c(
  "PI4K2A", "PI4K2B", "PI4KA", "PI4KB", "PIK3C2A", "PIK3C2B", "PIK3C2G",
  "PIK3C3", "PIK3CA", "PIK3CB", "PIK3CD", "PIK3CG", "PIK3R1", "PIK3R2",
  "PIK3R3", "PIK3R4", "PIK3R5", "PIK3R6", "PIKFYVE", "PIP4K2A",
  "PIP4K2B", "PIP5K1B", "PIP5K1C", "PITPNB"
)
Contains(genesPI, "PIK3", partial = TRUE)
```

`ControlFDR` *Adjust p-values for simple multiple-testing procedures*

Description

This is a modification of the `mt.rawp2adjp` function from the Bioconductor package `multtest`. We fixed an error wherein selecting the `"TSBH"` option overwrote the results of any previous adjustment methods, and another error created when the `"BY"` and `"TSBH"` methods were called simultaneously. 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

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

Arguments

`rawp`	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.
`proc`	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. Defaults to `"BH"`.

`alpha` 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.

`na.rm` 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 `NA`s.

`as.multtest.out` Should the output match the output from the `mt.rawp2adjp` 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

- `"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. This method is not guaranteed to return finite 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.
- `"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).

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

See Also

AESPCA_pVals SuperPCA_pVals

Examples

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

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

  ###  Create an OmicsSurv Object  ###
  colon_Omics <- CreateOmics(
    assayData_df = colonSurv_df[, -(2:3)],
    pathwayCollection_ls = colon_pathwayCollection,
    response = colonSurv_df[, 1:3],
    respType = "surv"
  )

  ###  Extract Pathway PCs and Loadings  ###
  colonPCs_ls <- ExtractAESPCs(
    object = colon_Omics,
    parallel = TRUE,
    numCores = 2
  )

  ###  Pathway p-Values  ###
  pVals <- PermTestSurv(
    OmicsSurv = colon_Omics,
    pathwayPCs_ls = colonPCs_ls$PCs,
    parallel = TRUE,
    numCores = 2
```

```
  )

  ###  Adjust p-Values  ###
  ControlFDR(rawp = pVals)

## End(Not run)
```

`coxTrain_fun` *Train Cox Proportional Hazards model for supervised PCA*

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

## Not run:
  p <- 500
  n <- 50

  x_mat <- matrix(rnorm(n * p), nrow = p, ncol = n)
  x_df <- data.frame(x_mat)
  time_int <- rpois(n, lambda = 365 * 2)
  obs_logi <- sample(
    c(FALSE, TRUE),
    size = n,
    replace = TRUE,
    prob = c(0.2, 0.8)
  )

  coxTrain_fun(
    x = x_df,
    y = time_int,
    censoring.status = !obs_logi
  )

## End(Not run)
```

`CreateOmics` *Generation Wrapper function for* `-Omics*-class objects`

Description

This function calls the `CreateOmicsPath`, `CreateOmicsSurv`, `CreateOmicsReg`, and `CreateOmicsCateg` functions to create valid objects of the classes `OmicsPathway`, `OmicsSurv`, `OmicsReg`, or `OmicsCateg`, respectively.

Usage

```
CreateOmics(
  assayData_df,
  pathwayCollection_ls,
  response = NULL,
  respType = c("none", "survival", "regression", "categorical"),
  centerScale = c(TRUE, TRUE),
  minPathSize = 3,
  ...
)
```

Arguments

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

`pathwayCollection_ls`
A `pathwayCollection` list of known gene pathways with two or 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. 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.
- description : An optional character vector the same length as the pathways list with additional information about the pathways.

If your gene pathways list is stored in a .gmt file, use the read_gmt function to import your pathways list as a pathwayCollection list object.

response — An optional response object. See "Details" for more information. Defaults to NULL.

respType — What type of response has been supplied. Options are "none", "survival", "regression", and "categorical". Defaults to "none" to match the default response = NULL value.

centerScale — Should the values in assayData_df be centered and scaled? Defaults to TRUE for centering and scaling, respectively. See scale for more information.

minPathSize — What is the smallest number of genes allowed in each pathway? Defaults to 3.

... — Dots for additional arguments passed to the internal CheckAssay function.

Details

This function is a wrapper around the four CreateOmics* functions. The values supplied to the response function argument can be in a list, data frame, matrix, vector, Surv object, or any class which extends these. Because this function makes "best guess" type conversions based on the respType argument, this argument is mandatory if response is non-NULL. Further, it is the responsibility of the user to ensure that the coerced response contained in the resulting Omics object accurately reflects the supplied response.

For respType = "survival", response is assumed to be ordered by event time, then event indicator. For example, if the response is a data frame or matrix, this function assumes that the first column is the time and the second column the death indicator. If the response is a list, then this function assumes that the first entry in the list is the event time and the second entry the death indicator. The death indicator must be a logical or binary (0-1) vector, where 1 or TRUE represents a death and 0 or FALSE represents right-censoring.

Some of the pathways in the supplied pathways list will be removed, or "trimmed", during object creation. For the pathway-testing methods, these trimmed pathways will have p-values given as NA. For an explanation of pathway trimming, see the documentation for the IntersectOmicsPwyCollct function.

Value

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

See Also

OmicsPathway, CreateOmicsPath, OmicsSurv, CreateOmicsSurv, OmicsCateg, CreateOmicsCateg OmicsReg, CreateOmicsReg, CheckAssay, CheckPwyColl, and IntersectOmicsPwyCollct

Examples

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

###  Create an OmicsPathway Object  ###
colon_OmicsPath <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection
)

###  Create an OmicsSurv Object  ###
colon_OmicsSurv <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "surv"
)

###  Create an OmicsReg Object  ###
colon_OmicsReg <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:2],
  respType = "reg"
)

###  Create an OmicsCateg Object  ###
colon_OmicsCateg <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, c(1,3)],
  respType = "cat"
)
```

`CreateOmicsPath` *Generation functions for* `-Omics*-class objects`

Description

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

Usage

```
CreateOmicsPath(assayData_df, sampleIDs_char, pathwayCollection_ls)

CreateOmicsSurv(
  assayData_df,
  sampleIDs_char,
  pathwayCollection_ls,
  eventTime_num,
  eventObserved_lgl
```

```
)

CreateOmicsReg(
  assayData_df,
  sampleIDs_char,
  pathwayCollection_ls,
  response_num
)

CreateOmicsCateg(
  assayData_df,
  sampleIDs_char,
  pathwayCollection_ls,
  response_fact
)
```

Arguments

`assayData_df`	An $N \times p$ data frame with named columns.
`sampleIDs_char`	A character vector with the N sample names.
`pathwayCollection_ls`	A `pathwayCollection` list of known gene pathways with two or 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. 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. • `description` : An optional character vector the same length as the `pathways` list with additional information about 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 assay data *must* be or extend the class `data.frame`, and the response values (for a survival-, regression-, or categorical-response 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.

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 bioassay ("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 categorial (factor) observations, respectively.

See Also

OmicsPathway, OmicsSurv, OmicsReg, and OmicsCateg

Examples

```
# DO NOT CALL THESE FUNCTIONS DIRECTLY. USE CreateOmics() INSTEAD.

  data("colon_pathwayCollection")
  data("colonSurv_df")

## Not run:
  CreateOmicsPath(
    assayData_df = colonSurv_df[, -(1:3)],
    sampleIDs_char = colonSurv_df$sampleID,
    pathwayCollection_ls = colon_pathwayCollection
  )

  CreateOmicsSurv(
    assayData_df = colonSurv_df[, -(1:3)],
    sampleIDs_char = colonSurv_df$sampleID,
    pathwayCollection_ls = colon_pathwayCollection,
    eventTime_num = colonSurv_df$OS_time,
    eventObserved_lgl = as.logical(colonSurv_df$OS_event)
  )

  CreateOmicsReg(
    assayData_df = colonSurv_df[, -(1:3)],
    sampleIDs_char = colonSurv_df$sampleID,
    pathwayCollection_ls = colon_pathwayCollection,
    response_num = colonSurv_df$OS_time
  )

  CreateOmicsCateg(
```

```
    assayData_df = colonSurv_df[, -(1:3)],
    sampleIDs_char = colonSurv_df$sampleID,
    pathwayCollection_ls = colon_pathwayCollection,
    response_fact = as.factor(colonSurv_df$OS_event)
  )

## End(Not run)
```

`CreatePathwayCollection` *Manually Create a* `pathwayCollection`*-class Object.*

Description

Manually create a `pathwayCollection` list similar to the output of the `read_gmt` function.

Usage

```
CreatePathwayCollection(
  sets_ls,
  TERMS,
  setType = c("pathways", "genes", "regions"),
  ...
)
```

Arguments

`sets_ls`	A named list of character vectors. Each vector should contain the names of the individual genes, proteins, sits, or CpGs within that set as a vector of character strings. If you create this pathway collection to integrate with data of class `Omics*`, 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.
`TERMS`	A character vector the same length as the `sets_ls` list with the proper names of the sets.
`setType`	What is the type of the set: pathway set of gene, gene sites in RNA or DNA, or regions of CpGs. Defaults to `''pathway''`.
`...`	Additional vectors or data components related to the `sets_ls` list. These values should be passed as a name-value pair. See "Details" for more information.

Details

This function checks the set list and set term inputs and then creates a `pathwayCollection` object from them. Pass additional list elements (such as the `description` of each set) 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 `pathwayCollection` objects, please do not create `pathwayCollection` list items named `setsize` or `n_tested`.

Value

A list object with class `pathwayCollection`.

See Also

read_gmt

Examples

```
data("colon_pathwayCollection")

CreatePathwayCollection(
  sets_ls = colon_pathwayCollection$pathways,
  TERMS = colon_pathwayCollection$TERMS
)
```

ExtractAESPCs *Extract AES-PCs from recorded pathway-subsets of a mass spectrometry or bio-assay data frame*

Description

Given a clean OmicsPath object (cleaned by the IntersectOmicsPwyCollct function), extract the first principal components (PCs) from each pathway with features recorded in the assay design matrix.

Usage

```
ExtractAESPCs(
  object,
  numPCs = 1,
  parallel = FALSE,
  numCores = NULL,
  standardPCA = FALSE,
  ...
)

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

Arguments

object	An object of class OmicsPathway.
numPCs	The number of PCs to extract from each pathway. Defaults to 1.
parallel	Should the computation be completed in parallel? Defaults to FALSE.
numCores	If parallel = TRUE, how many cores should be used for computation? Internally defaults to the number of available cores minus 2.

standardPCA	Should the function return the AES-PCA PCs and loadings (FALSE) or the standard PCA PCs and loadings (TRUE)? Defaults to FALSE.
...	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 unrecorded -Omes removed from the corresponding pathway collection by the IntersectOmicsPwyCollct 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 pathways, so these will not be extracted to the list. It is then vitally important to use either a very broad and generic list of pathways or a pathways list that is compatible to the assay data supplied.

Value

Two lists of matrices: PCs and loadings. Each element of both lists will be named by its pathway. The elements of the PCs list will be $N\times$ numPCs matrices containing the first numPCs principal components from each pathway. The elements of the loadings list will be numPCs $\times p$ projection matrices containing the loadings corresponding to the first numPCs principal components from each pathway. See "Details" for more information.

See Also

CreateOmicsPath; aespca IntersectOmicsPwyCollct

Examples

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


###  Load the Example Data  ###
data("colonSurv_df")
data("colon_pathwayCollection")

###  Create an OmicsSurv Object  ###
colon_Omics <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "surv"
)

###  Extract Pathway PCs and Loadings  ###
ExtractAESPCs(
  object = colon_Omics,
  parallel = TRUE,
  numCores = 2
)
```

getPathPCLs *Extract PCs and Loadings from a* superpcOut- *or* aespcOut-*class Object.*

Description

Given an object of class aespcOut or superpcOut, as returned by the functions AESPCA_pVals or SuperPCA_pVals, respectively, and the name or unique ID of a pathway, return a data frame of the principal components and a data frame of the loading vectors corresponding to that pathway.

Usage

```
getPathPCLs(pcOut, pathway_char, ...)

## S3 method for class 'superpcOut'
getPathPCLs(pcOut, pathway_char, ...)

## S3 method for class 'aespcOut'
getPathPCLs(pcOut, pathway_char, ...)
```

Arguments

pcOut	An object of classes superpcOut or aespcOut as returned by the SuperPCA_pVals or AESPCA_pVals functions, respectively.
pathway_char	A character string of the name or unique identifier of a pathway
...	Dots for additional arguments (currently unused).

Details

Match the supplied pathway character string to either the pathways or terms columns of the pVals_df data frame within the pcOut object. Then, subset the loadings_ls and PCs_ls lists for their entries which match the supplied pathway. Finally, return a list of the PCs, loadings, and the pathway ID and name.

Value

A list of four elements:

- PCs : A data frame of the principal components
- Loadings : A matrix of the loading vectors with features in the row names
- pathway : The unique pathway identifier for the pcOut object
- term : The name of the pathway

NULL

NULL

Examples

```
###  Load Data  ###
data("colonSurv_df")
data("colon_pathwayCollection")

###  Create -Omics Container  ###
colon_Omics <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "survival"
)

###  Calculate Supervised PCA Pathway p-Values  ###
colon_superpc <- SuperPCA_pVals(
  colon_Omics,
  numPCs = 2,
  parallel = TRUE,
  numCores = 2,
  adjustment = "BH"
)

###  Extract PCs and Loadings  ###
getPathPCLs(
  colon_superpc,
  "KEGG_PENTOSE_PHOSPHATE_PATHWAY"
)
```

`getPathpVals` *Extract Table of p-values from a* `superpcOut`- *or* `aespcOut`- *class Object.*

Description

Given an object of class `aespcOut` or `superpcOut`, as returned by the functions `AESPCA_pVals` or `SuperPCA_pVals`, respectively, return a data frame of the p-values for the top pathways.

Usage

```
getPathpVals(pcOut, score = FALSE, numPaths = 20L, alpha = NULL, ...)

## S3 method for class 'superpcOut'
getPathpVals(pcOut, score = FALSE, numPaths = 20L, alpha = NULL, ...)

## S3 method for class 'aespcOut'
getPathpVals(pcOut, score = FALSE, numPaths = 20L, alpha = NULL, ...)
```

Arguments

`pcOut`	An object of classes `superpcOut` or `aespcOut` as returned by the `SuperPCA_pVals` or `AESPCA_pVals` functions, respectively.

score	Should the unadjusted p-values be returned transformed to negative natural logarithm scores or left as is? Defaults to FALSE; that is, the raw p-values are returned instead of the transformed p-values.
numPaths	The number of top pathways by raw p-value. Defaults to the top 20 pathways. We do not permit users to specify numPaths and alpha concurrently.
alpha	The significance threshold for raw p-values. Defaults to NULL. If alpha is given, then numPaths will be ignored.
...	Dots for additional arguments (currently unused).

Details

Row-subset the pVals_df entry of an object of class aespcOut or superpcOut by the number of pathways requested (via the nPaths argument) or by the unadjusted significance level for each pathway (via the alpha argument). Return a data frame of the pathway names, FDR-adjusted significance levels (if available), and the raw score (negative natural logarithm of the p-values) of each pathway.

Value

A data frame with the following columns:

- terms : The pathway name, as given in the object@trimPathwayCollection$TERMS object.
- description : (OPTIONAL) The pathway description, as given in the object@trimPathwayCollection$descri object, if supplied.
- rawp : The unadjusted p-values of each pathway. Included if score = FALSE.
- ... : Additional columns of FDR-adjusted p-values as specified through the adjustment argument of the SuperPCA_pVals or AESPCA_pVals functions.
- score : The negative natural logarithm of the unadjusted p-values of each pathway. Included if score = TRUE.

NULL

NULL

Examples

```
###  Load Data  ###
data("colonSurv_df")
data("colon_pathwayCollection")

###  Create -Omics Container  ###
colon_Omics <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "survival"
)

###  Calculate Supervised PCA Pathway p-Values  ###
colon_superpc <- SuperPCA_pVals(
  colon_Omics,
  numPCs = 2,
  parallel = TRUE,
  numCores = 2,
```

```
  adjustment = "BH"
)

###  Extract Table of p-Values  ###
# Top 5 Pathways
getPathpVals(
  colon_superpc,
  numPaths = 5
)

# Pathways with Unadjusted p-Values < 0.01
getPathpVals(
  colon_superpc,
  alpha = 0.01
)
```

`glmTrain_fun` *Gene-specific Generalized Linear Model fit statistics for supervised PCA*

Description

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

Usage

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

Arguments

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

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

## Not run:
  p <- 500
  n <- 50

  x_mat <- matrix(rnorm(n * p), nrow = p, ncol = n)
  obs_logi <- sample(
    c(FALSE, TRUE),
    size = n,
    replace = TRUE,
    prob = c(0.2, 0.8)
  )

  glmTrain_fun(
    x = x_mat,
    y = obs_logi
  )

## End(Not run)
```

`GumbelMixpValues` *Calculate the p-values from an optimal mixture of Weibull Extreme Value distributions for supervised PCA*

Description

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

Usage

```
GumbelMixpValues(tScore_vec, pathwaySize_vec, optimParams_vec)
```

Arguments

`tScore_vec`	A vector of the maximum absolute t-scores for each pathway (returned by the `pathway_tScores` function) when under the alternative model.
`pathwaySize_vec`	A vector of the number of genes in each pathway.
`optimParams_vec`	The *NAMED* vector of optimal Weibull Extreme Value mixture distribution parameters returned by the `OptimGumbelMixParams` 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 `mu1`, `mu2` and `s1`, `s2` 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

OptimGumbelMixParams; pathway_tScores; SuperPCA_pVals

Examples

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

## Not run:
  ###  Load the Example Data  ###
  data("colon_pathwayCollection")
  n_int <- lengths(colon_pathwayCollection$pathways)


  ###  Simulate Maximum Absolute Control t-Values  ###
  # The SuperPCA algorithm defaults to 20 threshold values; the example
  #   pathway collection has 15 pathways.
  t_mat <- matrix(rt(15 * 20, df = 5), nrow = 15)

  absMax <- function(vec){
    vec[which.max(abs(vec))]
  }
  tAbsMax_num <- apply(t_mat, 1, absMax)


  ###  Calculate Optimal Parameters for the Gumbel Distribution  ###
  optParams_num <- OptimGumbelMixParams(
    max_tControl_vec = tAbsMax_num,
    pathwaySize_vec = n_int
  )


  ###  Simulate Maximum Absolute t-Values  ###
  tObs_mat <- matrix(rt(15 * 20, df = 3), nrow = 15)
  tObsAbsMax_num <- apply(tObs_mat, 1, absMax)


  ###  Calculate Observed-t-score p-Values  ###
  GumbelMixpValues(
    tScore_vec = tObsAbsMax_num,
    pathwaySize_vec = n_int,
    optimParams_vec = optParams_num
  )

## End(Not run)
```

IntersectOmicsPwyCollct *Delete -Ome symbols or IDs without matching features recorded in a given assay data frame from a pathway collection*

Description

Given a bio-assay design matrix and a pathwayCollection gene pathways list (each within an Omics*-class object), delete the genes / proteins / lipids / metabolomes / transcriptomes symbols or IDs recorded in each pathway which are not recorded in the assay data frame.

Usage

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

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

Arguments

object	An object of class OmicsPathway, OmicsSurv, OmicsReg, or OmicsCateg.
trim	The minimum cutoff of matching -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 pathways list 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 pathwayCollection list, all through one of the Omics* classes. This function will then copy the pathway collection, iterate over the list of copied pathways, delete symbols or IDs from that pathway without matches from the bio-assay design matrix column names, and remove any pathways that have fewer than trim genes with corresponding columns in the assay. The genes not recorded in the bio-assay design matrix are removed from the copy of the pathway collection (the trimPathwayCollection object), but remain in the original pathway collection.

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 pathwayCollection list or a pathwayCollection list that is appropriate for the assay data supplied. While you can create your own pathway lists, create proper pathwayCollection 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 n_tested object.

Examples

```
# DO NOT CALL THIS FUNCTION DIRECTLY. USE CreateOmics() INSTEAD.

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

  ###  Create an OmicsSurv Object  ###
  colon_Omics <- CreateOmics(
    assayData_df = colonSurv_df[, -(2:3)],
    pathwayCollection_ls = colon_pathwayCollection
  )

## End(Not run)
```

JoinPhenoAssay *Merge Phenotype and Assay Data by First Column (Sample ID)*

Description

Match the records from the phenotype data to the values in the assay data by sample ID. Return rows from each data frame with matches in both data frames. The sample ID must be the first column in both data frames.

Usage

```
JoinPhenoAssay(pheno_df, assay_df)
```

Arguments

pheno_df	Phenotype data frame with the sample IDs in the first column
assay_df	Assay data frame with the sample IDs in the first column

Details

Don't use this function. This is simply a wrapper around the merge function with extra checks for the class of the ID column. If you want to merge your two data frames by sample ID, you should use the inner_join function from the dplyr package instead. It's easier. See https://dplyr.tidyverse.org/reference/join.html.

Value

A list of three elements:

- assay : A data frame with the rows from assay_df which are contained in pheno_df, ordered by their position in pheno_df.
- response : A data frame with the rows from pheno_df which are contained in assay_df.
- sampleID : A vector of the sample IDs shared by both data frames, ordered by their position in pheno_df.

Examples

```
  # DO NOT CALL THIS FUNCTIONS DIRECTLY. USE CreateOmics() INSTEAD.

## Not run:
 data("colonSurv_df")
 JoinPhenoAssay(
   pheno_df = colonSurv_df[, 1:3],
   assay_df = colonSurv_df[, -(2:3)]
 )

## End(Not run)
```

---

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 Grammian / 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 Grammian 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

## Not run:
  X_mat <- as.matrix(colonSurv_df[, 5:50])
  X_mat <- scale(X_mat)

  XtX <- t(X_mat) %*% X_mat
  A_mat <- svd(XtX)$v

  lars.lsa(
    Sigma0 = XtX,
    b0 = A_mat[, 1] * sign(A_mat[1, 1]),
    n = ncol(X_mat)
  )

## End(Not run)
```

LoadOntoPCs *Calculate Test Data PCs from Training-Data Estimated Loadings*

Description

Given a list of loading vectors from a training data set, calculate the PCs of the test data set.

Usage

```
LoadOntoPCs(design_df, loadings_ls, sampleID = c("firstCol", "rowNames"))
```

Arguments

design_df	A test data frame with rows as samples and named features as columns
loadings_ls	A list of $p \times d$ loading vectors or matrices as returned by either the SuperPCA_pVals, AESPCA_pVals, or ExtractAESPCs functions. These lists of loadings will have feature names as their row names. Such feature names must match a subset of the column names of design_df exactly, as pathway-specific test-data subsetting is performed by column name.
sampleID	Are the sample IDs in the first column of design_df or in accessible by rownames(design_df)? Defaults to the first column. If your data does not have sample IDs for some reason, set this to rowNames.

Details

This function takes in a list of loadings and a training-centered test data set, applies over the list of loadings, subsets the columns of the test data by the row names of the loading vectors, right-multiplies the test-data subset matrix by the loading vector / matrix, and returns a data frame of the test-data PCs for each loading vector.

Value

A data frame with the PCs from each pathway concatenated by column. If you have the tidyverse loaded, this object will display as a tibble.

Examples

```
###  Load the Data  ###
data("colonSurv_df")
data("colon_pathwayCollection")

###  Create -Omics Container  ###
colon_Omics <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "survival"
)

###  Extract AESPCs  ###
colonSurv_aespc <- AESPCA_pVals(
  object = colon_Omics,
  numReps = 0,
  parallel = TRUE,
  numCores = 2,
  adjustpValues = TRUE,
  adjustment = c("Hoch", "SidakSD")
)

###  Project Data onto Pathway First PCs  ###
LoadOntoPCs(
  design_df = colonSurv_df,
  loadings_ls = colonSurv_aespc$loadings_ls
)
```

mysvd *Singular Value Decomposition wrapper for supervised PCA*

Description

Center and compute the SVD of a matrix

Usage

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

Arguments

mat	A matrix of data frame in "tall" format ($p \times n$).
method	What function should be used to extract the left- and right- singular vectors and singular values? Any function that returns the values as a list with components u, v, and d is appropriate. Defaults to svd.
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

## Not run:
  data("colon_pathwayCollection")
  data("colonSurv_df")

  colon_OmicsSurv <- CreateOmics(
    assayData_df = colonSurv_df[,-(2:3)],
    pathwayCollection_ls = colon_pathwayCollection,
```

```
  response = colonSurv_df[, 1:3],
  respType = "surv"
)

asthmaGenes_char <-
  getTrimPathwayCollection(colon_OmicsSurv)[["KEGG_ASTHMA"]]$IDs

mysvd(t(getAssay(colon_OmicsSurv))[asthmaGenes_char, ])

## End(Not run)
```

`normalize` *Normalize and reconstruct the eigenvalues of a data matrix for supervised PCA*

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 supurfluous. 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 *Gene-specific Regularized Ordinary Least Squares fit statistics for supervised PCA*

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

## Not run:
  p <- 500
  n <- 50

  x_mat <- matrix(rnorm(n * p), nrow = p, ncol = n)
  time_int <- rpois(n, lambda = 365 * 2)

  olsTrain_fun(
    x = x_mat,
    y = time_int
  )

## End(Not run)
```

`OmicsCateg-class` *An S4 class for categorical responses within an* `OmicsPathway` *object*

Description

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

Slots

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

`pathwayCollection` A list of known gene pathways with three or four 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.
- `description` : An optional character vector the same length as the `pathways` list with additional information about 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 `CreateOmics` 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`, `CreateOmics`

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

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.

`sampleIDs_char` A character vector with the N sample names.

`pathwayCollection` A list of known gene pathways with three or four 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.
- `description` : An optional character vector the same length as the `pathways` list with additional information about 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 `CreateOmics` function call.

`trimPathwayCollection` A subset of the list stored in the `pathwayCollection` slot. This list will have pathways that only contain genes that are present in the assay data frame.

See Also

`CreateOmics`

OmicsReg-class *An S4 class for continuous responses within an* `OmicsPathway` *object*

Description

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

Slots

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

`pathwayCollection` A list of known gene pathways with three or four 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.
- description : An optional character vector the same length as the pathways list with additional information about 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 CreateOmics function call.

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

See Also

OmicsPathway, CreateOmics

OmicsSurv-class *An S4 class for survival responses within an* OmicsPathway *object*

Description

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

Slots

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

pathwayCollection A list of known gene pathways with three or four 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.
- description : An optional character vector the same length as the pathways list with additional information about 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 CreateOmics 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, CreateOmics

`OptimGumbelMixParams` *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

```
OptimGumbelMixParams(
  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 `optim` 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 `initialVals`. Defaults to `c(0, -Inf, 0, -Inf, 0)`.

`upperBD`
: A vector of the upper bounds on the `initialVals`. Defaults to `c(1, Inf, Inf, Inf, Inf)`.

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; GumbelMixpValues; pathway_tControl; SuperPCA_pVals

Examples

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

## Not run:
  ###  Load the Example Data  ###
  data("colon_pathwayCollection")


  ###  Simulate Maximum Absolute Control t-Values  ###
  # The SuperPCA algorithm defaults to 20 threshold values; the example
  #   pathway collection has 15 pathways.
  t_mat <- matrix(rt(15 * 20, df = 5), nrow = 15)

  absMax <- function(vec){
    vec[which.max(abs(vec))]
  }
  tAbsMax_num <- apply(t_mat, 1, absMax)


  ###  Calculate Optimal Parameters for the Gumbel Distribution  ###
  OptimGumbelMixParams(
    max_tControl_vec = tAbsMax_num,
    pathwaySize_vec = lengths(colon_pathwayCollection$pathways)
  )

## End(Not run)
```

pathwayPCA — *Extract and Test the Significance of Pathway-Specific Principal Components*

Description

To introduce this package, please see our "Integrative Pathway Analysis" vignette: https://gabrielodom.github.io/pathwayPCA/articles//Introduction_to_pathwayPCA.html.

The pathwayPCA package has three main components:

- Import and Tidy Data: https://gabrielodom.github.io/pathwayPCA/articles/Supplement2-Importing_Data.html
- Create Omics Data Objects https://gabrielodom.github.io/pathwayPCA/articles/Supplement3-Create_Omics_Objects.html
- Test Pathway Significance https://gabrielodom.github.io/pathwayPCA/articles/Supplement4-Methods_Walkthrough.html
- Analyze and Visualize Results https://gabrielodom.github.io/pathwayPCA/articles/Supplement5-Analyse_Results.html

For an overview of these four topics in context, please see our Quickstart Guide: https://gabrielodom.github.io/pathwayPCA/articles/Supplement1-Quickstart_Guide.html

PathwaytValues — *Calculate pathway-specific Student's t-scores from a null distribution or the true distribution for supervised PCA*

Description

If we sample from the null, distribution, first parametrically resample the response vector before model analysis (f we calculate Student t statistics from the true distribution instead, the response matrix is untouched). 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 values of the parametrically resampled response (for the null distribution) or the response itself (for the true distribution).

Usage

```
PathwaytValues(
  pathway_vec,
  geneArray_df,
  response_mat,
  responseType = c("survival", "regression", "categorical"),
  control = 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 "categorical", this will be an $N \times 1$ factor 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. You can create a factor matrix of a factor a with the command dim(a) <- c(k, 1), where k = length(a).
responseType	A character string. Options are "survival", "regression", and "categorical".
control	Should the responses be parametrically resampled to generate a control distribution? 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. 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.

If control = TRUE, the RandomControlSample suite of functions first parametrically bootstrapps the response. This control response will be used to contrstruct a null distribution against which to compare the results calculated with the original response values.

Value

If control = TRUE, a matrix with numPCs rows and n.threshold columns. The matrix values are model t-statisics for each PC included (rows) at each threshold level (columns).

If control = TRUE, the same matrix as above is contained as the tscor element of a list (the first element). The other list elements are PCs_mat (the matrix of PCs) and loadings (the matrix of -Ome loadings corresponding to the PCs).

See Also

pathway_tScores; pathway_tControl; RandomControlSample; superpc.train; superpc.st

Examples

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

## Not run:
  data("colon_pathwayCollection")
  data("colonSurv_df")
```

```
colon_OmicsSurv <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "surv"
)

asthmaGenes_char <-
  getTrimPathwayCollection(colon_OmicsSurv)[["KEGG_ASTHMA"]]$IDs
resp_mat <- matrix(
  c(getEventTime(colon_OmicsSurv), getEvent(colon_OmicsSurv)),
  ncol = 2
)

PathwaytValues(
  pathway_vec = asthmaGenes_char,
  geneArray_df = t(getAssay(colon_OmicsSurv)),
  response_mat = resp_mat,
  responseType = "survival"
)

PathwaytValues(
  pathway_vec = asthmaGenes_char,
  geneArray_df = t(getAssay(colon_OmicsSurv)),
  response_mat = resp_mat,
  responseType = "survival",
  control = TRUE
)

## End(Not run)
```

`pathway_tControl` *Calculate pathway-specific Student's t-scores from a null distribution for supervised PCA*

Description

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 resampled values of the response.

Usage

```
pathway_tControl(
  pathway_vec,
  geneArray_df,
  response_mat,
  responseType = c("survival", "regression", "categorical"),
  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 "categorical", 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 "categorical".
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 parametric bootstrapping with the RandomControlSample suite of functions. This response sampling 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-statisics 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

## Not run:
  data("colon_pathwayCollection")
  data("colonSurv_df")

  colon_OmicsSurv <- CreateOmics(
    assayData_df = colonSurv_df[, -(2:3)],
    pathwayCollection_ls = colon_pathwayCollection,
    response = colonSurv_df[, 1:3],
    respType = "surv"
  )
```

```
  asthmaGenes_char <-
    getTrimPathwayCollection(colon_OmicsSurv)[["KEGG_ASTHMA"]]$IDs
  resp_mat <- matrix(
    c(getEventTime(colon_OmicsSurv), getEvent(colon_OmicsSurv)),
    ncol = 2
  )

  pathway_tControl(
    pathway_vec = asthmaGenes_char,
    geneArray_df = t(getAssay(colon_OmicsSurv)),
    response_mat = resp_mat,
    responseType = "survival"
  )

## End(Not run)
```

`pathway_tScores` *Calculate pathway-specific Student's t-scores for supervised PCA*

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", "categorical"),
  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 `"categorical"`, 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 `"categorical"`.
`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-statisics 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

## Not run:
  data("colon_pathwayCollection")
  data("colonSurv_df")

  colon_OmicsSurv <- CreateOmics(
    assayData_df = colonSurv_df[, -(2:3)],
    pathwayCollection_ls = colon_pathwayCollection,
    response = colonSurv_df[, 1:3],
    respType = "surv"
  )

  asthmaGenes_char <-
    getTrimPathwayCollection(colon_OmicsSurv)[["KEGG_ASTHMA"]]$IDs
  resp_mat <- matrix(
    c(getEventTime(colon_OmicsSurv), getEvent(colon_OmicsSurv)),
    ncol = 2
  )

  pathway_tScores(
    pathway_vec = asthmaGenes_char,
    geneArray_df = t(getAssay(colon_OmicsSurv)),
    response_mat = resp_mat,
    responseType = "survival"
  )

## End(Not run)
```

PermTestCateg *AES-PCA permutation test of categorical response for pathway PCs*

Description

Given an OmicsCateg object and a list of pathway PCs from the ExtractAESPCs function, test if each pathway with features recorded in the bio-assay design matrix is significantly related to the categorical response.

Usage

```
PermTestCateg(
  OmicsCateg,
  pathwayPCs_ls,
  numReps = 0L,
  parallel = FALSE,
  numCores = NULL,
  ...
)

## S4 method for signature 'OmicsCateg'
PermTestCateg(
  OmicsCateg,
  pathwayPCs_ls,
  numReps = 0L,
  parallel = FALSE,
  numCores = NULL,
  ...
)
```

Arguments

OmicsCateg	A data object of class OmicsCateg, created by the CreateOmics function.
pathwayPCs_ls	A list of pathway PC matrices returned by the ExtractAESPCs function.
numReps	How many permutations to estimate the p-value? Defaults to 0 (that is, to estimate the p-value parametrically). If numReps > 0, then the non-parametric, permutation p-value will be returned based on the number of random samples specified.
parallel	Should the computation be completed in parallel? Defaults to FALSE.
numCores	If parallel = TRUE, how many cores should be used for computation? Internally defaults to the number of available cores minus 2.
...	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 categorical 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. Note that the AIC and log-likelihood are proportional because the number of parameters in each pathway is constant.

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

CreateOmics; ExtractAESPCs; glm; binomial; SampleCateg

Examples

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

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

  ###  Create an OmicsSurv Object  ###
  colon_Omics <- CreateOmics(
    assayData_df = colonSurv_df[, -(2:3)],
    pathwayCollection_ls = colon_pathwayCollection,
    response = colonSurv_df[, c(1,3)],
    respType = "categ"
  )

  ###  Extract Pathway PCs and Loadings  ###
  colonPCs_ls <- ExtractAESPCs(
    object = colon_Omics,
    parallel = TRUE,
    numCores = 2
  )

  ###  Pathway p-Values  ###
  PermTestCateg(
    OmicsCateg = colon_Omics,
    pathwayPCs_ls = colonPCs_ls$PCs,
    parallel = TRUE,
    numCores = 2
  )

## End(Not run)
```

PermTestReg *AES-PCA permutation test of continuous response for pathway PCs*

Description

Given an OmicsReg object and a list of pathway PCs from the ExtractAESPCs function, test if each pathway with features recorded in the bio-assay design matrix is significantly related to the continuous response.

Usage

```
PermTestReg(
  OmicsReg,
  pathwayPCs_ls,
  numReps = 0L,
  parallel = FALSE,
  numCores = NULL,
  ...
)

## S4 method for signature 'OmicsReg'
PermTestReg(
  OmicsReg,
  pathwayPCs_ls,
  numReps = 0L,
  parallel = FALSE,
  numCores = NULL,
  ...
)
```

Arguments

OmicsReg	A data object of class OmicsReg, created by the CreateOmics function.
pathwayPCs_ls	A list of pathway PC matrices returned by the ExtractAESPCs function.
numReps	How many permutations to estimate the p-value? Defaults to 0 (that is, to estimate the p-value parametrically). If numReps > 0, then the non-parametric, permutation p-value will be returned based on the number of random samples specified.
parallel	Should the computation be completed in parallel? Defaults to FALSE.
numCores	If parallel = TRUE, how many cores should be used for computation? Internally defaults to the number of available cores minus 2.
...	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. Note that the AIC and log-likelihood are proportional because the number of parameters in each pathway is constant.

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

CreateOmics; ExtractAESPCs; lm; SampleReg

Examples

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

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

  ###  Create an OmicsSurv Object  ###
  colon_Omics <- CreateOmics(
    assayData_df = colonSurv_df[, -(2:3)],
    pathwayCollection_ls = colon_pathwayCollection,
    response = colonSurv_df[, 1:2],
    respType = "reg"
  )

  ###  Extract Pathway PCs and Loadings  ###
  colonPCs_ls <- ExtractAESPCs(
    object = colon_Omics,
    parallel = TRUE,
    numCores = 2
  )

  ###  Pathway p-Values  ###
  PermTestReg(
    OmicsReg = colon_Omics,
    pathwayPCs_ls = colonPCs_ls$PCs,
    parallel = TRUE,
    numCores = 2
  )

## End(Not run)
```

PermTestSurv *AES-PCA permutation test of survival response for pathway PCs*

Description

Given an OmicsSurv object and a list of pathway principal components (PCs) from the ExtractAESPCs function, test if each pathway with features recorded in the bio-assay design matrix is significantly related to the survival output.

Usage

```
PermTestSurv(
  OmicsSurv,
  pathwayPCs_ls,
  numReps = 0L,
  parallel = FALSE,
  numCores = NULL,
  ...
)

## S4 method for signature 'OmicsSurv'
PermTestSurv(
  OmicsSurv,
  pathwayPCs_ls,
  numReps = 0L,
  parallel = FALSE,
  numCores = NULL,
  ...
)
```

Arguments

OmicsSurv	A data object of class OmicsSurv, created by the CreateOmics function.
pathwayPCs_ls	A list of pathway PC matrices returned by the ExtractAESPCs function.
numReps	How many permutations to estimate the p-value? Defaults to 0 (that is, to estimate the p-value parametrically). If numReps > 0, then the non-parametric, permutation p-value will be returned based on the number of random samples specified.
parallel	Should the computation be completed in parallel? Defaults to FALSE.
numCores	If parallel = TRUE, how many cores should be used for computation? Internally defaults to the number of available cores minus 2.
...	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. Note that the AIC and log-likelihood are proportional because the number of parameters in each pathway is constant.

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

CreateOmics; ExtractAESPCs; coxph; SampleSurv

Examples

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

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

  ###  Create an OmicsSurv Object  ###
  colon_Omics <- CreateOmics(
    assayData_df = colonSurv_df[, -(2:3)],
    pathwayCollection_ls = colon_pathwayCollection,
    response = colonSurv_df[, 1:3],
    respType = "surv"
  )

  ###  Extract Pathway PCs and Loadings  ###
  colonPCs_ls <- ExtractAESPCs(
    object = colon_Omics,
    parallel = TRUE,
    numCores = 2
  )

  ###  Pathway p-Values  ###
  PermTestSurv(
    OmicsSurv = colon_Omics,
    pathwayPCs_ls = colonPCs_ls$PCs,
    parallel = TRUE,
    numCores = 2
  )

## End(Not run)
```

---

print.pathwayCollection

Display the Summary of a pathwayCollection*-class Object.*

Description

The display method for pathways lists as returned by the read_gmt function.

Usage

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

Arguments

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

Details

This function sets a print method for pathwayCollection objects.

Value

x, returned invisibly (with the invisible function).

See Also

read_gmt; write_gmt

Examples

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

###  Print / Show  ###
colon_pathwayCollection
```

RandomControlSample *Parametric bootstrap and non-parametric permutations of a response vector or matrix*

Description

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

Usage

```
SampleResponses(
  response_vec,
  event_vec = NULL,
  respType = c("survival", "regression", "categorical"),
  parametric = TRUE
)

SampleSurv(response_vec, event_vec, parametric = TRUE)

SampleReg(response_vec, parametric = TRUE)

SampleCateg(response_vec, parametric = TRUE)
```

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 occurence and 1 for a recorded event.
respType	What type of response has been supplied. Options are "none", "survival", "regression", and "categorical". Defaults to "none" to match the default response = NULL value.
parametric	Should the random sample be taken using a parametric bootstrap sample? Defaults to TRUE.

Details

The distributions (for parametric = TRUE) are Weibull for survival times, Normal for regression response, and n-ary Multinomial for categorical response. Distributional parameters are estimated with their maximum likelihood estimates. When parametric = FALSE, the response vector or survival matrix is randomly ordered by row. This option should only be used when called from the AES-PCA method.

Value

If parametric = FALSE, a permutation of the supplied response is returned (for AES-PCA). If parametric = TRUE, we return a parametric bootstrap sample of the response.

Examples

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

## Not run:
  data("colon_pathwayCollection")
  data("colonSurv_df")

  SampleResponses(
    response_vec = colonSurv_df$OS_time,
    event_vec = colonSurv_df$OS_event,
    respType = "survival"
  )

## End(Not run)
```

---

read_gmt *Read a* .gmt *file in as a* pathwayCollection *object*

Description

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

Usage

```
read_gmt(
  file,
  setType = c("pathways", "genes", "regions"),
  description = FALSE,
  nChars = 1e+07,
  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.
setType	What is the type of the set: pathway set of gene, gene sites in RNA or DNA, or regions of CpGs. Defaults to ''pathway''.
description	Should the "description" field (the second field in the .gmt file on each line) be included in the output? Defaults to FALSE.
nChars	The number of characters to read from a connection. The largest .gmt file we have encountered is the full C5 pathway collection from MSigDB (5917 pathways), which has roughly 5 million characters in UTF-8 encoding. Therefore, we default this argument to be twice the size of the largest pathway collection we have seen so far, 10,000,000.
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 pathwayCollection list of sets. This list has three elements:

- 'setType' : A named list of character vectors. Each vector contains the names of the individual genes, sites, or CpGs within that set as a vector of character strings. The name of this list entry is equal to the value specified in setType.
- TERMS : A character vector the same length as the 'setType' list with the proper names of the sets.
- description : (OPTIONAL) A character vector the same length as the 'setType' list with a note on that set (for the .gmt file included with this package, this field contains hyperlinks to the MSigDB description card for that pathway). This field is included when description = TRUE.

See Also

print.pathwayCollection; write_gmt

Examples

```
# 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, description = TRUE)

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

SE2Tidy *Tidy a SummarizedExperiment Assay*

Description

Extract the assay information from a SummarizedExperiment-class-object, transpose it, and and return it as a tidy data frame that contains assay measurements, feature names, and sample IDs

Usage

```
SE2Tidy(summExperiment, whichAssay = 1)
```

Arguments

summExperiment	A SummarizedExperiment-class object
whichAssay	Because SummarizedExperiment objects can store multiple related assays, which assay will be paired with a given pathway collection to create an Omics*-class data container? Defaults to 1, for the first assay in the object.

Details

This function is designed to extract and transpose a "tall" assay data frames (where genes or proteins are the rows and patient or tumour samples are the columns) from a SummarizedExperiment object. This function also transposes the row (feature) names to column names and the column (sample) names to row names via the TransposeAssay function.

NOTE: if this function stops working (again), please add a comment here: https://github.com/gabrielodom/pathwayPCA/issues/83

Value

The transposition of the assay in summExperiment to tidy form, with the column data (from the colData slot of the object) appended as the first columns of the data frame.

Examples

```
# THIS REQUIRES THE SummarizedExperiment PACKAGE.
library(SummarizedExperiment)
data(airway, package = "airway")

airway_df <- SE2Tidy(airway)
```

---

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_pathwayCollection")

###  Create an OmicsSurv Object  ###
colon_OmicsSurv <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "surv"
)

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

SubsetOmicsPath *Access and Edit Assay or* pathwayCollection *Values in* Omics* *Objects*

Description

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

Usage

```
getAssay(object, ...)

getAssay(object) <- value

getSampleIDs(object, ...)

getSampleIDs(object) <- value

getPathwayCollection(object, ...)

getPathwayCollection(object) <- value

getTrimPathwayCollection(object, ...)

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

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

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

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

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

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

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

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 pathways 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 ValidOmicsSurv, ValidOmicsCateg, or ValidOmicsReg functions.

Further, because the pathwayPCA methods require a cleaned (trimmed) pathway collection, the trimPathwayCollection slot is read-only. Users may only edit this slot by updating the pathway collection provided to the pathwayCollection slot. Despite this functionality, we **strongly** recommend that users create a new object with the updated pathway collection, rather than attempting to overwrite the slots within an existing object. See IntersectOmicsPwyCollct for details on trimmed pathway collection.

Value

The "get" functions return the objects in the slots specified: getAssay returns the assayData_df data frame object, getSampleIDs returns the sampleIDs_char character vector, getPathwayCollection returns the pathwayCollection list object, and getTrimPathwayCollection returns the trimPathwayCollection. 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, sampleIDs_char, or pathwayCollection slots for any OmicsPathway, OmicsSurv, OmicsReg, or OmicsCateg objects, provided that the new values do not violate the validity checks of their respective objects. Because the slot for trimPathwayCollection is filled upon object creation, and to ensure that this pathway collection is "clean", there is no "set" function for the trimmed pathway collection slot. Instead, users can update the pathway collection, and the trimmed pathway collection will be updated automatically. See "Details" for more information on the "set" functions.

See Also

CreateOmics

Examples

```
data("colonSurv_df")
data("colon_pathwayCollection")

colon_Omics <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection
)

getAssay(colon_Omics)
getPathwayCollection(colon_Omics)
```

SubsetOmicsResponse *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 ValidOmicsCateg or ValidOmicsReg 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

CreateOmics

Examples

```
data("colonSurv_df")
data("colon_pathwayCollection")

colon_Omics <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, c(1, 2)],
  respType = "reg"
)

getResponse(colon_Omics)
```

SubsetOmicsSurv *Access and Edit Event Time or Indicator in an* OmicsSurv *Object*

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 ValidOmicsSurv 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

CreateOmics

Examples

```
data("colonSurv_df")
data("colon_pathwayCollection")

colon_Omics <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "survival"
)

getEventTime(colon_Omics)
getEvent(colon_Omics)
```

SubsetPathwayCollection

Subset a pathwayCollection*-class Object by Pathway.*

Description

The subset method for pathways lists as returned by the read_gmt function.

Usage

```
## S3 method for class 'pathwayCollection'
x[[name_char]]
```

Arguments

x	An object of class pathwayCollection.
name_char	The name of a pathway in the collection or its unique ID.

Details

This function finds the index matching the name_char argument to the TERMS field of the pathwayCollection-class Object, then subsets the pathways list, TERMS vector, description vector, and setsize vector by this index. If you subset a trimmed pathwayCollection object, and the function errors with "Pathway not found.", then the pathway specified has been trimmed from the pathway collection.

Also, this function does not allow for users to overwrite any portion of a pathway collection. These objects should rarely, if ever, be changed. If you absolutely must change the components of a pathwayCollection object, then create a new one with the codeCreatePathwayCollection function.

Value

A list of the pathway name (Term), unique ID (pathID), contents (IDs), description (description), and number of features (Size).

Examples

```
data("colon_pathwayCollection")
colon_pathwayCollection[["KEGG_RETINOL_METABOLISM"]]
```

SubsetPathwayData *Subset Pathway-Specific Data*

Description

Given an Omics object and the name of a pathway, return the -omes in the assay and the response as a (tibble) data frame.

Usage

```
SubsetPathwayData(object, pathName, ...)

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

Arguments

object	An object of class OmicsPathway, or an object extending this class.
pathName	The name of a pathway contained in the pathway collection in the object.
...	Dots for additional internal arguments (currently unused).

Details

This function subsets the assay by the matching gene symbols or IDs in the specified pathway.

Value

A data frame of the columns of the assay in the Omics object which are listed in the specified pathway, with a leading column for sample IDs. If the Omics object has response information, these are also included as the first column(s) of the data frame, after the sample IDs. 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.

Examples

```
data("colonSurv_df")
data("colon_pathwayCollection")

colon_Omics <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "survival"
)

SubsetPathwayData(
  colon_Omics,
  "KEGG_RETINOL_METABOLISM"
)
```

`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 `superpc.train`.
`data`	A list of test data: • `x` : A "tall" pathway data frame ($p_{path} \times N$). • `y` : A response vector corresponding to `type`. • `censoring.status` : If `type = "survival"`, the censoring indicator ($1-$ the observed event indicator). Otherwise, `NULL`. • `featurenames` : 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 `fit` 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-statisics 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

## Not run:
  data("colon_pathwayCollection")
  data("colonSurv_df")

  colon_OmicsSurv <- CreateOmics(
    assayData_df = colonSurv_df[,-(2:3)],
    pathwayCollection_ls = colon_pathwayCollection,
    response = colonSurv_df[, 1:3],
    respType = "surv"
  )

  asthmaGenes_char <-
```

```
    getTrimPathwayCollection(colon_OmicsSurv)[["KEGG_ASTHMA"]]$IDs

  data_ls <- list(
    x = t(getAssay(colon_OmicsSurv))[asthmaGenes_char, ],
    y = getEventTime(colon_OmicsSurv),
    censoring.status = getEvent(colon_OmicsSurv),
    featurenames = asthmaGenes_char
  )

  superpcFit <- superpc.train(
    data = data_ls,
    type = "surv"
  )

  superpc.st(
    fit = superpcFit,
    data = data_ls
  )

## End(Not run)
```

`superpc.train` *Train a supervised PCA model*

Description

Computes feature scores for p_{path} features of a pathway via a linear model fit.

Usage

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

Arguments

`data`	A list of test data: • `x` : A "tall" pathway data frame ($p_{path} \times N$). • `y` : A response vector corresponding to `type`. • `censoring.status` : If `type` = `"survival"`, the censoring indicator ($1-$ the observed event indicator. Otherwise, `NULL`. • `featurenames` : A character vector of the measured -Omes in `x`.
`type`	What model relates `y` and `x`? Options are `"survival"`, `"regression"`, or `"categorical"`.
`s0.perc`	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 = "categorical").

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

## Not run:
  data("colon_pathwayCollection")
  data("colonSurv_df")

  colon_OmicsSurv <- CreateOmics(
    assayData_df = colonSurv_df[,-(2:3)],
    pathwayCollection_ls = colon_pathwayCollection,
    response = colonSurv_df[, 1:3],
    respType = "surv"
  )

  asthmaGenes_char <-
    getTrimPathwayCollection(colon_OmicsSurv)[["KEGG_ASTHMA"]]$IDs

  data_ls <- list(
    x = t(getAssay(colon_OmicsSurv))[asthmaGenes_char, ],
    y = getEventTime(colon_OmicsSurv),
    censoring.status = getEvent(colon_OmicsSurv),
    featurenames = asthmaGenes_char
  )

  superpc.train(
    data = data_ls,
    type = "surv"
  )

## End(Not run)
```

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 pathway-subset of 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,
  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,
  parallel = FALSE,
  numCores = NULL,
  adjustpValues = TRUE,
 adjustment = c("Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD", "BH", "BY",
    "ABH", "TSBH"),
  ...
)
```

Arguments

object	An object of superclass OmicsPathway with a response matrix or vector.
n.threshold	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.
parallel	Should the computation be completed in parallel? Defaults to FALSE.
numCores	If parallel = TRUE, how many cores should be used for computation? Internally defaults to the number of available cores minus 1.
adjustpValues	Should you adjust the p-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 p-value. If only one procedure is selected, then it is necessarily the first procedure. See the documentation for the ControlFDR 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, OptimGumbelMixParams, GumbelMixpValues, and TabulatepValues functions.

Please see our Quickstart Guide for this package: https://gabrielodom.github.io/pathwayPCA/articles/Supplement1-Quickstart_Guide.html

Value

A data frame with columns:

- pathways : The names of the pathways in the Omics* object (given in object@trimPathwayCollection$pathways
- setsize : The number of genes in each of the original pathways (given in the object@trimPathwayCollection$se object).
- terms : The pathway description, as given in the object@trimPathwayCollection$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 print as a data frame.

See Also

CreateOmics; TabulatepValues; pathway_tScores; pathway_tControl; OptimGumbelMixParams; GumbelMixpValues; clusterApply

Examples

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

###  Create an OmicsSurv Object  ###
colon_OmicsSurv <- CreateOmics(
  assayData_df = colonSurv_df[, -(2:3)],
  pathwayCollection_ls = colon_pathwayCollection,
  response = colonSurv_df[, 1:3],
  respType = "surv"
)

###  Calculate Pathway p-Values  ###
colonSurv_superpc <- SuperPCA_pVals(
  object = colon_OmicsSurv,
```

```
  parallel = TRUE,
  numCores = 2,
  adjustpValues = TRUE,
  adjustment = c("Hoch", "SidakSD")
)
```

TabulatepValues *Tabulate, 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

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

Arguments

pVals_vec	A named vector of permutation p-values returned by the PermTestSurv, PermTestReg, or PermTestCateg 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 GumbelMixpValues function.
genesets_ls	A list of known gene pathways, trimmed to match the given assay data by the IntersectOmicsPwyCollct function. This pathway list must contain: • 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. • n_tested : An integer vector the same length as pathways containing the number of genes present in the pathway after trimming. Pathways list trimming is done in the IntersectOmicsPwyCollct 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. Defaults to "BH" (Benjamini and Hochberg, 1995).
...	Additional arguments to pass to the ControlFDR function.

Details

This is a wrapper function for the ControlFDR 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@trimPathwayCollection$pathway
- n_tested : The number of genes in each pathway after being trimmed to match the assay. Given in the n_tested element of the trimmed pathway collection.
- terms : The pathway title, as stored in the object@trimPathwayCollection$TERMS object.
- description : The pathway description, if it is stored in the object@trimPathwayCollection$description 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.

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

  ###  Create an OmicsSurv Object  ###
  colon_Omics <- CreateOmics(
    assayData_df = colonSurv_df[, -(2:3)],
    pathwayCollection_ls = colon_pathwayCollection,
    response = colonSurv_df[, 1:3],
    respType = "surv"
  )

  ###  Extract Pathway PCs and Loadings  ###
  colonPCs_ls <- ExtractAESPCs(
    object = colon_Omics,
    parallel = TRUE,
    numCores = 2
  )

  ###  Pathway p-Values  ###
  pVals <- PermTestSurv(
    OmicsSurv = colon_Omics,
    pathwayPCs_ls = colonPCs_ls$PCs,
    parallel = TRUE,
```

```
  numCores = 2
)

###  Create Table of p-Values  ###
trimmed_PC <- getTrimPathwayCollection(colon_Omics)
TabulatepValues(
  pVals_vec = pVals,
  genesets_ls = trimmed_PC
)

## End(Not run)
```

`TransposeAssay` *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

```
TransposeAssay(
  assay_df,
  omeNames = c("firstCol", "rowNames"),
  stringsAsFactors = FALSE
)
```

Arguments

`assay_df`	A data frame with numeric values to transpose
`omeNames`	Are the data feature names in the first column or in the row names of `df`? Defaults to the first column. If the feature names are in the row names, this function assumes that these names are accesible by the `rownames` function called on `df`.
`stringsAsFactors`	Should columns containing string information be coerced to factors? Defaults to `FALSE`.

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.

Recall that data frames require that all elements of a single column to have the same `class`. Therefore, sample IDs of a "tall" data frame **must** be stored as the column names rather than in the first row.

Value

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

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))

TransposeAssay(x_df, omeNames = "rowNames")
```

`ValidOmicsSurv` *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

```
ValidOmicsSurv(object)

ValidOmicsReg(object)

ValidOmicsCateg(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 with have one response vector of continuous (`numeric`) or categorial (`factor`) observations, respectively.

WhichPathways *Filter and Subset a* pathwayCollection*-class Object by Symbol.*

Description

The filter-subset method for pathways lists as returned by the read_gmt function. This function returns the subset of pathways which contain the set of symbols requested

Usage

```
WhichPathways(x, symbols_char, ...)
```

Arguments

x	An object of class pathwayCollection.
symbols_char	A character vector or scalar of gene symbols or regions
...	Additional arguments passed to the Contains function

Details

This function finds the index of each set that contains the symbols supplied, then returns those sets as a new pathwayCollection object. Find pathways that contain geneA OR geneB by passing the argument matches = "any" through ... to Contains (this is the default value). Find pathways that contain geneA AND geneB by changing this argument to matches = "all". Find all genes in a specified family by passing in one value to short and setting partial = TRUE.

Value

An object of class pathwayCollection, but containing only the sets which contain the symbols supplied to symbols_char. If no sets are found to contain the symbols supplied, this function returns NULL and prints a warning.

Examples

```
data("colon_pathwayCollection")

WhichPathways(colon_pathwayCollection, "MAP", partial = TRUE)

WhichPathways(
  colon_pathwayCollection,
  c("MAP4K5", "RELA"),
  matches = "all"
)
```

wikipwsHS_Entrez_pathwayCollection

Wikipathways Homosapiens EntrezIDs

Description

A pathwayCollection object containing the homosapiens pathways list from Wikipathways (https://www.wikipathways.org/).

Usage

```
data(wikipwsHS_Entrez_pathwayCollection)
```

Format

A pathwayCollection list of three elements:

- pathways : A named list of 443 character vectors. Each vector contains the Entrez Gene IDs of the individual genes within that pathway as a vector of character strings. The names are the shorthand pathway names.
- TERMS : A character vector of length 443 containing the shorthand names of the gene pathways.
- description : A character vector of length 443 containing the full names of the gene pathways.

Details

This pathwayCollection was sent to us from Dr. Alexander Pico at the Gladstone Institute (https://gladstone.org/our-science/people/alexander-pico).

Source

Dr. Alexander Pico, Wikipathways

wikipwsHS_Symbol_pathwayCollection

Wikipathways Homosapiens Gene Symbols

Description

A pathwayCollection object containing the homosapiens pathways list from Wikipathways (https://www.wikipathways.org/).

Usage

```
data(wikipwsHS_Symbol_pathwayCollection)
```

Format

A pathwayCollection list of three elements:

- pathways : A named list of 457 character vectors. Each vector contains the Gene Symbols of the individual genes within that pathway as a vector of character strings. The names are the shorthand pathway names.
- TERMS : A character vector of length 457 containing the shorthand names of the gene pathways.
- description : A character vector of length 457 containing the full names of the gene pathways.

Details

This pathwayCollection was sent to us from Dr. Alexander Pico at the Gladstone Institute (https://gladstone.org/our-science/people/alexander-pico).

This pathway collection was translated from EntrezIDs to HGNC Symbols with the script convert_EntrezID_to_HGNC_ in scripts.

Source

Dr. Alexander Pico, Wikipathways

---

write_gmt *Write a* pathwayCollection *Object to a* .gmt *File*

Description

Write a pathwayCollection object as a pathways list file in Gene Matrix Transposed (.gmt) format.

Usage

```
write_gmt(pathwayCollection, file, setType = c("pathways", "genes", "regions"))
```

Arguments

pathwayCollection

A pathwayCollection list of sets. This list contains the following two or three elements:

- 'setType' : A named list of character vectors. Each vector contains the names of the individual genes, sites, or CpGs within that set as a vector of character strings. If you are using genes, these genes can be represented by HGNC gene symbols, Entrez IDs, Ensembl IDs, GO terms, etc.
- TERMS : A character vector the same length as the 'setType' list with the proper names of the sets.
- description : An optional character vector the same length as the 'setType' list with a note on that set (such as a url to the description if the set is a pathway). If this element of the pathwayCollection is NULL, then the file will be written with "" (the empty character string) as its second field in each line.

file	Either a character string naming a file or a connection open for writing. File names should end in .gmt for clarity.
setType	What is the type of the set: pathway set of gene, gene sites in RNA or DNA, or regions of CpGs. Defaults to ''pathway''.

Details

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

NULL. Output written to the file path specified.

See Also

print.pathwayCollection; read_gmt

Examples

```
# Toy pathway set
toy_pathwayCollection <- list(
  pathways = list(
    c("C1orf27", "NR5A1", "BLOC1S4", "C4orf50"),
    c("TARS2", "DUSP5", "GPR88"),
    c("TRX-CAT3-1", "LINC01333", "LINC01499", "LINC01046", "LINC01149")
  ),
  TERMS = c("C-or-f_paths", "randomPath2", "randomLINCs"),
  description = c("these are", "totally made up", "pathways")
)
class(toy_pathwayCollection) <- c("pathwayCollection", "list")
toy_pathwayCollection

# write_gmt(toy_pathwayCollection, file = "example_pathway.gmt")
```

Index