Package ‘GSSTDA’

June 7, 2023

Title Gene Structure Survival using Topological Data Analysis

Version 0.1.3

Description Mapper-based survival analysis with transcriptomics data is designed to carry out.
Mapper-based survival analysis is a modification of Progression Analysis of Disease (PAD)
where survival data is taken into account in the filtering function. More
details in: J. Fores-Martos, B. Suay-Garcia, R. Bosch-Romeu, M.C. Sanfeliu-Alonso,
A. Falco, J. Climent, ``Progression Analysis of Disease with Survival (PAD-S) by SurvMap
identifies different prognostic subgroups of breast cancer in a large combined set of transcrip-
tomics
and methylation studies'' <doi:10.1101/2022.09.08.507080>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

LazyData true

LazyDataCompression xz

DependsNote BioC (>= 3.2)

Imports base, cluster, stats, survival, utils, visNetwork,
ComplexHeatmap, circlize, devtools

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

SuggestsNote BioC (>= 3.0), Recommended: affy, oligoClasses, oligo,
GEOquery, DupChecker, arrayQualityMetrics, vctrs, frma,
a4Preproc, genefu, plyr, preprocessCore, hgu133plus2.db

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 3.5.0)

NeedsCompilation no

Author Miriam Esteve [aut, cre] (<https://orcid.org/0000-0002-5908-0581>),
Raquel Bosch [aut],
Jaume Forés [aut] (<https://orcid.org/0000-0002-9025-4877>),
Joan Climent [aut],
Antonio Falco [aut]

Maintainer Miriam Esteve <`miriam.estevecampello@uchceu.es`>

Repository CRAN

Date/Publication 2023-06-07 13:20:11 UTC

R topics documented:

`case_tag` *Case-control vector*

Description

Character vector of length 121 containing the group to which each sample belongs.

Usage

```
data(case_tag, package = "GSSTDA")
```

Format

Character vector length 121.

"NT": control, sample from healthy tissue; "T": case, sample from neoplastic tissue.

Source

The data are from the study GSE42568. Information extracted from the file GSE42568_family.soft.gz available at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE42568.

`check_arg_mapper` *check_arg_mapper*

Description

Checking the arguments introduces in the `mapper` object.

Usage

```
check_arg_mapper(
  full_data,
  filter_values,
  distance_type,
  clustering_type,
  linkage_type,
  na.rm = TRUE
)
```

Arguments

`full_data`	Matrix with the columns of the input matrix corresponding to the individuals belonging to the level.
`filter_values`	Vector obtained after applying the filtering function to the input matrix, i.e, a vector with the filtering function values for each included sample.
`distance_type`	Type of distance to be used for clustering. Choose between correlation ("cor") and euclidean ("euclidean"). "cor" default option.
`clustering_type`	Type of clustering method. Choose between "hierarchical" and "PAM" ("partition around medoids") options. "hierarchical" default option.
`linkage_type`	Linkage criteria used in hierarchical clustering. Choose between "single" for single-linkage clustering, "complete" for complete-linkage clustering or "average" for average linkage clustering (or UPGMA). Only necessary for hierarchical clustering. "single" default option.
`na.rm`	`logical`. If `TRUE`, `NA` rows are omitted. If `FALSE`, an error occurs in case of `NA` rows.

Value

`optimal_clustering_mode`

`check_filter_values` *check_filter_values*

Description

Checking the filter_values introduces in the `mapper` object.

Usage

```
check_filter_values(full_data, filter_values, na.rm = TRUE)
```

Arguments

`full_data`	Matrix with the columns of the input matrix corresponding to the individuals belonging to the level. This matrix could be the genes_disease_component.
`filter_values`	Vector obtained after applying the filtering function to the input matrix, i.e, a vector with the filtering function values for each included sample.
`na.rm`	`logical`. If `TRUE`, `NA` rows are omitted. If `FALSE`, an error occurs in case of `NA` rows.

Value

`filter_value` and `full_data` without NAN's

`check_full_data` *check_full_data*

Description

Checking the full_data introduces in the package

Usage

```
check_full_data(full_data, na.rm = TRUE)
```

Arguments

`full_data`	Matrix with the columns of the input matrix corresponding to the individuals belonging to the level.
`na.rm`	`logical`. If `TRUE`, `NA` rows are omitted. If `FALSE`, an error occurs in case of `NA` rows.

Value

Return `full_data` without NAN's and as a matrix

`check_gene_selection` *check_gene_selection*

Description

Checking the arguments introduces in the gene selection process.

Usage

```
check_gene_selection(num_genes, gen_select_type, percent_gen_select)
```

Arguments

`num_genes`	Number of genes in the full_data
`gen_select_type`	Type of gene selection to be used. Choose between "top_bot" (top-botton) and "abs" (absolute)
`percent_gen_select`	Percentage of genes to be selected

Value

num_gen_select Number of genes to be selected according to the percent_gen_select value

`check_vectors` *check_vectors*

Description

Checking the `survival_time`, `survival_event` and `case_tag` introduces in the `GSSTDA` object.

Usage

```
check_vectors(full_data, survival_time, survival_event, case_tag, na.rm = TRUE)
```

Arguments

`full_data`	The genes of the full_data (maybe remove by na.rm = TRUE)
`survival_time`	Time between disease diagnosis and death (if not dead until the end of follow-up).
`survival_event`	`logical`. Whether the patient has died or not.
`case_tag`	The tag of the healthy patient (healthy or not).
`na.rm`	`logical`. If `TRUE`, `NA` rows are omitted. If `FALSE`, an error occurs in case of `NA` rows.

Value

control_tag Return the tag of the healthy patient

`clust_all_levels` *Get clusters for all data level*

Description

It performs the clustering of the samples in each of the levels. That is to say, in each interval of values of the filtering function, the samples with a value within that interval are clustered using the proposed clustering algorithm and the proposed method to determine the optimal number of clusters.

Usage

```
clust_all_levels(
  full_data,
  samp_in_lev,
  distance_type,
  clustering_type,
  linkage_type,
  optimal_clustering_mode,
  num_bins_when_clustering
)
```

Arguments

`full_data`	Input data matrix whose columns are the individuals and rows are the features.BR cambiar nombre.
`samp_in_lev`	A list of character vectors with the individuals included in each of the levels (i.e. each of the intervals of the values of the filter functions). It is the output of the `samples_in_levels` function.
`distance_type`	Type of distance to be used for clustering. Choose between correlation ("cor") and euclidean ("euclidean").
`clustering_type`	Type of clustering method. Choose between "hierarchical" and "PAM" ("partition around medoids") options.
`linkage_type`	Linkage criteria used in hierarchical clustering. Choose between "single" for single-linkage clustering, "complete" for complete-linkage clustering or "average" for average linkage clustering (or UPGMA). Only necessary for hierarchical clustering.
`optimal_clustering_mode`	Method for selection optimal number of clusters. It is only necessary if the chosen type of algorithm is hierarchical. In this case, choose between "standard" (the method used in the original mapper article) or "silhouette". In the case of the PAM algorithm, the method will always be "silhouette". "silhouette"
`num_bins_when_clustering`	Number of bins to generate the histogram employed by the standard optimal number of cluster finder method. Parameter not necessary if the "optimal_clust_mode" option is "silhouette" or the "clust_type" is "PAM".

Value

List of interger vectors. Each of the vectors contains information about the nodes at each level and the individuals contained in them. The names of the vector values are the names of the samples and the vector values are the node number of that level to which the individual belongs.

`clust_lev` *Get clusters for a particular data level*

Description

It performs clustering of the samples belonging to a particular level (to a particular interval of the filter function) with the proposed clustering algorithm and the proposed method to determine the optimal number of clusters.

Usage

```
clust_lev(
  full_data_i,
  distance_type,
  clustering_type,
  linkage_type,
  optimal_clustering_mode,
  num_bins_when_clustering,
  level_name
)
```

Arguments

`full_data_i`	Matrix with the columns of the input matrix corresponding to the individuals belonging to the level.
`distance_type`	Type of distance to be used for clustering. Choose between correlation ("cor") and euclidean ("euclidean").
`clustering_type`	Type of clustering method. Choose between "hierarchical" and "PAM" (“partition around medoids”) options.
`linkage_type`	Linkage criteria used in hierarchical clustering. Choose between "single" for single-linkage clustering, "complete" for complete-linkage clustering or "average" for average linkage clustering (or UPGMA). Only necessary for hierarchical clustering. The value provided if the type of clustering chosen is hierarchical will be ignored
`optimal_clustering_mode`	Method for selection optimal number of clusters. It is only necessary if the chosen type of algorithm is hierarchical. In this case, choose between "standard" (the method used in the original mapper article) or "silhouette". In the case of the PAM algorithm, the method will always be "silhouette".
`num_bins_when_clustering`	Number of bins to generate the histogram employed by the standard optimal number of cluster finder method. Parameter not necessary if the "optimal_clust_mode" option is "silhouette" or the "clust_type" is "PAM".
`level_name`	Name of the studied level. # ERROR No usado

Value

Returns a interger vector with the samples included in each cluster for the specific level analyzed. The names of the vector values are the names of the samples and the vector values are the node number to which the individual belongs.

`compute_node_adjacency` *Computes the adjacency matrix.*

Description

It computes the adjacency matrix between nodes. Two nodes are considered connected if they share at least one individual.

Usage

```
compute_node_adjacency(nodes_list)
```

Arguments

`nodes_list`	Output of the `levels_to_nodes` function. List of character vectors. Each of the vectors contains the names of the individuals at each node.

Value

It returns a matrix of magnitude n nodes x n nodes that stores a 1 if there are shared samples in two given nodes and a 0 otherwise.

`cox_all_genes` *Survival analysis based on gene expression levels.*

Description

It carries out univariate cox proportional hazard models for the expression levels of each gene included in the provided dataset (case_disease_component) and their link with relapse-free or overall survival.

Usage

```
cox_all_genes(case_disease_component, survival_time, survival_event)
```

Arguments

`case_disease_component`	Disease component matrix (output of the function `generate_disease_component`) having selected only the columns belonging to disease samples. The names of the rows must be the names of the genes.
`survival_time`	Numeric vector that includes time to the event information
`survival_event`	Numeric vector that indicates if relapse or death have been produced (0 and 1s).

Value

A matrix with the results of the application of proportional hazard models using the expression levels of each gene as covariate. The `coef` column corresponds to the regression coefficient; the `exp_coef` column corresponds to the value of e^coef (which is interpreted as the odds ratio); the `se_coef` column corresponds to the standard error of each coefficient; the `Z` column corresponds to the value of coef/se_coef (the higher the Z value, the higher the significance of the variable) and the `Pr_z` column corresponds to the p-value for each Z value.

`denoise_rectangular_matrix`
Rectangular Matrix Denoiser.

Description

It takes a rectangular matrix composed by the addition of a signal matrix and a Gaussian noise matrix and returns a matrix of the same dimension that is denoised through a Singular Value Decomposition truncation process. The selection of the number of singular values is chosen following the proposal by "The optimal hard threshold for singular values is $\sqrt{(4/3)}$". It should be used after the function `flatten_normal_tiss`.

Usage

```
denoise_rectangular_matrix(matrix_flatten_normal_tiss)
```

Arguments

`matrix_flatten_normal_tiss`	A rectangular noisy matrix to denoise. It is return by `flatten_normal_tiss` function.

Value

A the normal space which has the same dimension denoised version of the matrix returned by `flatten_normal_tiss`.

Examples

```
denoise_rectangular_matrix(matrix(c(1,2,3,4,5,2,3,1,2,3),ncol = 2))
```

DGSA *Disease-Specific Genomic Analysis*

Description

Disease-Specific Genomic Analysis (DGSA). This analysis, developed by Nicolau *et al.*, allows the calculation of the "disease component" of a expression matrix which consists of, through linear models, eliminating the part of the data that is considered normal or healthy and keeping only the component that is due to the disease. It is intended to precede other techniques like classification or clustering. For more information see *Disease-specific genomic analysis: identifying the signature of pathologic biology* (doi: 10.1093/bioinformatics/btm033).

Usage

```
DGSA(full_data, survival_time, survival_event, case_tag, na.rm = TRUE)
```

Arguments

`full_data`	Input matrix whose columns correspond to the patients and rows to the genes.
`survival_time`	Numerical vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. For the patients with tumour sample should be indicated the time between disease diagnosis and death (if not dead until the end of follow-up) and healthy patients must have an NA value.
`survival_event`	Numerical vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. For the patients with tumour sample should be indicated whether the patient has died (1) or not (0). Only these values are valid and healthy patients must have an NA value.
`case_tag`	Character vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. It must be indicated for each patient whether he/she is healthy or not. One value should be used to indicate whether the patient is healthy and another value should be used to indicate whether the patient's sample is tumourous. The user will then be asked which one indicates whether the patient is healthy. Only two values are valid in the vector in total.
`na.rm`	`logical`. If `TRUE`, `NA` rows are omitted. If `FALSE`, an error occurs in case of `NA` rows. TRUE default option.

Value

A `DGSA` object. It contains: the `full_data` without NAN's values, the `case_control` vector without NAN's values, the label designated for healthy samples (`control_tag`), the matrix with the normal space (linear space generated from normal tissue samples) and the matrix of the disease components (the transformed full_data matrix from which the normal component has been removed).

Examples

```
DGSA_obj <- DGSA(full_data,  survival_time, survival_event, case_tag)
```

`flatten_normal_tiss` *Flatten normal tissues*

Description

Given a matrix containing the expression values of `n` healthy tissue samples, it produces the flattened vector matrix as reported in "Disease-specific genomic analysis: identifying the signature of pathology biology".

Usage

```
flatten_normal_tiss(normal_tiss)
```

Arguments

`normal_tiss`	A normal tissue data gene expression matrix. The columns should be the samples and the rows should be the genes.

Value

A gene expression matrix containing the flattened version of the vectors.

Examples

```
normal_tissue_matrix <- matrix(stats::rnorm(36), nrow=6)
flatten_normal_tiss(normal_tissue_matrix)
```

`full_data` *Gene expression matrix*

Description

Matrix containing gene expression profiling of 104 breast cancer and 17 normal breast biopsies. Expression profiling data by array (platform HG-U133_Plus_2).

Usage

```
data(full_data, package = "GSSTDA")
```

Format

Gene expression matrix with 20825 rows and 121 columns.

The columns correspond to the patients and the rows to the genes. The column names correspond to the patient identifier in GEO. The row names correspond to the gene names.

Details

Normalized gene expression data GSE42568. Background correction, summarization, and quantile normalization were carried out using the fRMA method implemented in the `fRMA` package. Filtered probes that did not target genes with valid gene id were filtered and those probes targeting the same gene were collapse by thanking those presenting the highest row variance using the `WGCNA::collapseRows` function.

Source

The data are from the study GSE42568 available in https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE42568. The data were processed as explained in the `details` section.

`fun_to_int` *Marcenko-Pastur distribution to integrate.*

Description

This function is an auxiliary function that includes the Marcenco-Pastur function that has to be integrated to find the upper bound of integration that produces and area of 1/2.

Usage

```
fun_to_int(t, bet)
```

Arguments

`t`	Parameter t.
`bet`	Beta value. Aspect ratio of the input matrix, $\frac{m}{n}$, were m is the number of rows of the input matrix and n the number of columns.

Value

It returns the function value for a specific t and a particular aspect ration m/n.

Examples

```
fun_to_int(1,0.3)
```

`generate_disease_component` *Generate disease component matrix.*

Description

This function produces a disease component matrix from an expression matrix and the denoised flattened matrix constructed from "healthy tissue data".

Usage

```
generate_disease_component(full_data, normal_space)
```

Arguments

`full_data`	Input matrix whose columns correspond to the patients and rows to the gens. Both tumour and healthy samples should be included.
`normal_space`	Denoised flattened matrix constructed from "healthy tissue data". Output of the function `denoise_rectangular_matrix`.

Value

Disease component matrix that contains the disease component of the provided normal space

Examples

```
full_data <- matrix(stats::rnorm(120),ncol=20)
normal_tissue <- full_data[,11:20]
normal_tissue_f <- flatten_normal_tiss(normal_tissue)
normal_tissue_f_d <- denoise_rectangular_matrix(normal_tissue_f)
disease_component <- generate_disease_component(full_data,normal_tissue_f_d)
```

`geneSelection` *Gene selection and filter function*

Description

Gene selection and calculation of filter function values. After fitting a Cox proportional hazard model to each gene, this function makes a selection of genes according to both their variability within the database and their relationship with survival. Subsequently, with the genes selected, the values of the filtering functions are calculated for each patient. The filter function allows to summarise each vector of each individual in a single data. This function takes into account the survival associated with each gene. In particular, the implemented filter function performs the vector magnitude in the [L_p] norm (as well as k powers of this magnitude) of the vector resulting of weighting each element of the column vector by the Z score obtained in the cox proportional hazard model.

Usage

```
geneSelection(data_object, gen_select_type, percent_gen_select, na.rm = TRUE)
```

Arguments

`data_object`	Object with: • full_data Input matrix whose columns correspond to the patients and rows to the genes. • survival_time Numerical vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. For the patients with tumour sample should be indicated the time between disease diagnosis and death (if not dead until the end of follow-up) and healthy patients must have an NA value. • survival_event Numerical vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. For the patients with tumour sample should be indicated whether the patient has died (1) or not (0). Only these values are valid and healthy patients must have an NA value. • case_tag Character vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. It must be indicated for each patient whether he/she is healthy or not. One value should be used to indicate whether the patient is healthy and another value should be used to indicate whether the patient's sample is tumourous. The user will then be asked which one indicates whether the patient is healthy. Only two values are valid in the vector in total.
`gen_select_type`	Option. Options on how to select the genes to be used in the mapper. Select the "Abs" option, which means that the genes with the highest absolute value are chosen, or the "Top_Bot" option, which means that half of the selected genes are those with the highest value (positive value, i.e. worst survival prognosis) and the other half are those with the lowest value (negative value, i.e. best prognosis). "Top_Bot" default option.
`percent_gen_select`	Percentage (from zero to one hundred) of genes to be selected to be used in mapper. 10 default option.
`na.rm`	`logical`. If `TRUE`, `NA` rows are omitted. If `FALSE`, an error occurs in case of `NA` rows. TRUE default option.

Value

A `geneSelection_object`. It contains:

- the `full_data` without NAN's values (`data`)
- the `cox_all_matrix` (a matrix with the results of the application of proportional hazard models: with the regression coefficients, the odds ratios, the standard errors of each coefficient, the Z values (coef/se_coef) and the p-values for each Z value)
- a vector with the name of the selected genes

- the matrix of disease components with only the rows of the selected genes (`genes_disease_component`)
- and the vector of the values of the filter function.

Examples

```
data_object <- list("full_data" = full_data, "survival_time" = survival_time,
"survival_event" = survival_event, "case_tag" = case_tag)
class(data_object) <- "data_object"
geneSelection_obj <- geneSelection(data_object,
gen_select_type ="top_bot", percent_gen_select=10)
```

`geneSelection.default` *gene_selection_classes.default*

Description

Private function to select Gene without DGSA process

Usage

```
## Default S3 method:
geneSelection(data_object, gen_select_type, percent_gen_select, na.rm = TRUE)
```

Arguments

`data_object` Object with:

- full_data Input matrix whose columns correspond to the patients and rows to the genes.
- survival_time Numerical vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. For the patients with tumour sample should be indicated the time between disease diagnosis and death (if not dead until the end of follow-up) and healthy patients must have an NA value.
- survival_event Numerical vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. For the patients with tumour sample should be indicated whether the patient has died (1) or not (0). Only these values are valid and healthy patients must have an NA value.
- case_tag Character vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. It must be indicated for each patient whether he/she is healthy or not. One value should be used to indicate whether the patient is healthy and another value should be used to indicate whether the patient's sample is tumourous. The user will then be asked which one indicates whether the patient is healthy. Only two values are valid in the vector in total.

`gen_select_type`	Option. Options on how to select the genes to be used in the mapper. Select the "Abs" option, which means that the genes with the highest absolute value are chosen, or the "Top_Bot" option, which means that half of the selected genes are those with the highest value (positive value, i.e. worst survival prognosis) and the other half are those with the lowest value (negative value, i.e. best prognosis). "Top_Bot" default option.
`percent_gen_select`	Percentage (from zero to one hundred) of genes to be selected to be used in mapper. 10 default option.
`na.rm`	`logical`. If `TRUE`, `NA` rows are omitted. If `FALSE`, an error occurs in case of `NA` rows. TRUE default option.

Value

A `geneSelection` object. It contains: the full_data without NAN's values, the control tag of the healthy patient, the matrix with the normal space and the matrix of the disease components.

Examples

```
data_object <- list("full_data" = full_data, "survival_time" = survival_time,
"survival_event" = survival_event, "case_tag" = case_tag)
class(data_object) <- "data_object"
geneSelection_object <- geneSelection(data_object, gen_select_type ="top_bot",
                                     percent_gen_select = 10)
```

`geneSelection.DGSA_object` *gene_selection_classes.DGSA_object*

Description

Private function to select Gene with DGSA object

Usage

```
## S3 method for class 'DGSA_object'
geneSelection(data_object, gen_select_type, percent_gen_select, na.rm = TRUE)
```

Arguments

`data_object`	DGSA object information
`gen_select_type`	Option. Options on how to select the genes to be used in the mapper. Select the "Abs" option, which means that the genes with the highest absolute value are chosen, or the "Top_Bot" option, which means that half of the selected genes

	are those with the highest value (positive value, i.e. worst survival prognosis) and the other half are those with the lowest value (negative value, i.e. best prognosis). "Top_Bot" default option.
`percent_gen_select`	Percentage (from zero to one hundred) of genes to be selected to be used in mapper. 10 default option.
`na.rm`	`logical`. If `TRUE`, `NA` rows are omitted. If `FALSE`, an error occurs in case of `NA` rows. TRUE default option.

Value

A `geneSelection` object. It contains: the full_data without NAN's values, the control tag of the healthy patient, the matrix with the normal space and the matrix of the disease components.

Examples

```
DGSA_obj <- DGSA(full_data, survival_time, survival_event, case_tag, na.rm = "checked")

geneSelection_object <- geneSelection(DGSA_obj, gen_select_type ="top_bot",
                                      percent_gen_select = 10)
```

`gene_selection` *gene_selection*

Description

Private function to gene selection

Usage

```
gene_selection(
  full_data,
  survival_time,
  survival_event,
  control_tag_cases,
  gen_select_type,
  num_gen_select
)
```

Arguments

`full_data`	Input matrix whose columns correspond to the patients and rows to the genes.
`survival_time`	Numerical vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. For the patients with tumour sample should be indicated the time between disease diagnosis and death (if not dead until the end of follow-up) and healthy patients must have an NA value.

`survival_event` Numerical vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. For the patients with tumour sample should be indicated whether the patient has died (1) or not (0). Only these values are valid and healthy patients must have an NA value.

`control_tag_cases`
Character vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. It must be indicated for each patient whether he/she is healthy or not. One value should be used to indicate whether the patient is healthy and another value should be used to indicate whether the patient's sample is tumourous. The user will then be asked which one indicates whether the patient is healthy. Only two values are valid in the vector in total.

`gen_select_type`
Option. Options on how to select the genes to be used in the mapper. Select the "Abs" option, which means that the genes with the highest absolute value are chosen, or the "Top_Bot" option, which means that half of the selected genes are those with the highest value (positive value, i.e. worst survival prognosis) and the other half are those with the lowest value (negative value, i.e. best prognosis). "Top_Bot" default option.

`num_gen_select` Number of genes to be selected to be used in mapper.

Value

A `geneSelection_object`. It contains:

- the `full_data` without NAN's values (`data`)
- the `cox_all_matrix` (a matrix with the results of the application of proportional hazard models: with the regression coefficients, the odds ratios, the standard errors of each coefficient, the Z values (coef/se_coef) and the p-values for each Z value)
- a vector with the name of the selected genes
- the matrix of disease components with only the rows of the selected genes (`genes_disease_component`)
- and the vector of the values of the filter function.

Examples

```
gen_select_type <- "Top_Bot"
percent_gen_select <- 10
control_tag_cases <- which(case_tag == "NT")
geneSelection_obj <- gene_selection(full_data, survival_time, survival_event, control_tag_cases,
gen_select_type ="top_bot", num_gen_select = 10)
```

`gene_selection_surv` *Gene selection based on variability and the relationship to survival.*

Description

It selects genes for mapper based on the product of standard deviation of the rows (genes) in the disease component matrix plus one times the Z score obtained by fitting a cox proportional hazard model to the level of each gene. For further information see "Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival"

Usage

```
gene_selection_surv(
  case_disease_component,
  cox_all_matrix,
  gen_select_type,
  num_gen_select
)
```

Arguments

`case_disease_component`	Disease component matrix (output of the function `generate_disease_component`) having selected only the columns belonging to disease samples. The names of the rows must be the names of the genes.
`cox_all_matrix`	Output from the `cox_all_genes` function. Data.frame with information on the relationship between genes and survival.
`gen_select_type`	Option. Select the "Abs" option, which means that the genes with the highest absolute value are chosen, or the "Top_Bot" option, which means that half of the selected genes are those with the highest value (positive value, i.e. worst survival prognosis) and the other half are those with the lowest value (negative value, i.e. best prognosis).
`num_gen_select`	Number of genes to be selected (those with the highest product value).

Value

Character vector with the names of the selected genes.

`get_intervals_One_D` *Extract intervals from filter function output values.*

Description

It calculates the intervals of the given output values of a filter function according to the given number and percentage of overlap.

Usage

```
get_intervals_One_D(filter_values, num_intervals, percent_overlap)
```

Arguments

`filter_values`	Vector obtained after applying the filtering function to the input matrix, i.e, a vector with the filtering function values for each included sample.
`num_intervals`	Number of intervals to divide the filtering function values in.
`percent_overlap`	Percentage of overlap between intervals.

Value

Returns a list with the set of intervals for the filtering function values.

`get_lambda` *Computes lambda*

Description

Computes the value of lambda as defined in: "The Optimal Hard Threshold for Singular Values is $\sqrt{(4/3)}$".

Usage

```
get_lambda(bet)
```

Arguments

`bet`	Beta value. Aspect ratio of the input matrix. $$\frac{m}{n}$$, were m is the number of rows of the input matrix and n the number of columns.

Value

Numeric. Lambda value.

Examples

```
get_lambda(0.3)
```

`get_mu_beta` *Get mu sub beta*

Description

This function identifies the upper bound of integration of the Marcenko-Pastur distribution, as described in "The Optimal Hard Threshold for Singular Values is $\sqrt{(4/3)}$". It explores 100 values in a given interval. Then it selects the values closest to 1/2 on the left and on the right. As the upper bound of integration, if the distance between the left and right approximations is lower than a given threshold (1e-10), it converges and the upper bound that produces an area of 1/2 is defined as the mean of the left and right approximations.

Usage

```
get_mu_beta(bet)
```

Arguments

`bet`	Beta value. Aspect ratio of the input matrix, $$\frac{m}{n}$$, were m is the number of rows of the input matrix and n the number of columns.

Value

It returns the mu beta value. This is the upper limit of integration where the Marcenko-Pastur distribution is equal to 1/2.

Examples

```
get_mu_beta(0.3)
```

`get_omega` *Compute the omega value*

Description

It computes the omega value as described in "The Optimal Hard Threshold for Singular Values is $\sqrt(4/3)$".

Usage

```
get_omega(bet)
```

Arguments

`bet`	Beta value. Aspect ratio of the input matrix, $$\frac{m}{n}$$, were m is the number of rows of the input matrix and n the number of columns.

Value

Numeric. Omega value.

Examples

```
get_omega(0.3)
```

`GSSTDA` *Gene Structure Survival using Topological Data Analysis (GSSTDA).*

Description

Gene Structure Survival using Topological Data Analysis. This function implements an analysis for expression array data based on the *Progression Analysis of Disease* developed by Nicolau *et al.* (doi: 10.1073/pnas.1102826108) that allows the information contained in an expression matrix to be condensed into a combinatory graph. The novelty is that information on survival is integrated into the analysis.

The analysis consists of 3 parts: a preprocessing of the data, the gene selection and the filter function, and the mapper algorithm. The preprocessing is specifically the Disease Specific Genomic Analysis (proposed by Nicolau *et al.*) that consists of, through linear models, eliminating the part of the data that is considered "healthy" and keeping only the component that is due to the disease. The genes are then selected according to their variability and whether they are related to survival and the values of the filtering function for each patient are calculated taking into account the survival associated with each gene. Finally, the mapper algorithm is applied from the disease component matrix and the values of the filter function obtaining a combinatory graph.

Usage

```
GSSTDA(
  full_data,
  survival_time,
  survival_event,
  case_tag,
  gen_select_type = "Top_Bot",
  percent_gen_select = 10,
  num_intervals = 5,
  percent_overlap = 40,
  distance_type = "cor",
  clustering_type = "hierarchical",
  num_bins_when_clustering = 10,
  linkage_type = "single",
  na.rm = TRUE
)
```

Arguments

`full_data`	Input matrix whose columns correspond to the patients and rows to the genes.
`survival_time`	Numerical vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. For the patients with tumour sample should be indicated the time between disease diagnosis and death (if not dead until the end of follow-up) and healthy patients must have an NA value.
`survival_event`	Numerical vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. For the patients with tumour sample should be indicated whether the patient has died (1) or not (0). Only these values are valid and healthy patients must have an NA value.
`case_tag`	Character vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. It must be indicated for each patient whether he/she is healthy or not. One value should be used to indicate whether the patient is healthy and another value should be used to indicate whether the patient's sample is tumourous. The user will then be asked which one indicates whether the patient is healthy. Only two values are valid in the vector in total.
`gen_select_type`	Option. Options on how to select the genes to be used in the mapper. Select the "Abs" option, which means that the genes with the highest absolute value are chosen, or the "Top_Bot" option, which means that half of the selected genes are those with the highest value (positive value, i.e. worst survival prognosis) and the other half are those with the lowest value (negative value, i.e. best prognosis). "Top_Bot" default option.
`percent_gen_select`	Percentage (from zero to one hundred) of genes to be selected to be used in mapper. 10 default option.
`num_intervals`	Parameter for the mapper algorithm. Number of intervals used to create the first sample partition based on filtering values. 5 default option.

`percent_overlap`
: Parameter for the mapper algorithm. Percentage of overlap between intervals. Expressed as a percentage. 40 default option.

`distance_type`
: Parameter for the mapper algorithm. Type of distance to be used for clustering. Choose between correlation ("cor") and euclidean ("euclidean"). "cor" default option.

`clustering_type`
: Parameter for the mapper algorithm. Type of clustering method. Choose between "hierarchical" and "PAM" ("partition around medoids") options. "hierarchical" default option.

`num_bins_when_clustering`
: Parameter for the mapper algorithm. Number of bins to generate the histogram employed by the standard optimal number of cluster finder method. Parameter not necessary if the "optimal_clust_mode" option is "silhouette" or the "clust_type" is "PAM". 10 default option.

`linkage_type`
: Parameter for the mapper algorithm. Linkage criteria used in hierarchical clustering. Choose between "single" for single-linkage clustering, "complete" for complete-linkage clustering or "average" for average linkage clustering (or UPGMA). Only necessary for hierarchical clustering. "single" default option.

`na.rm`
: `logical`. If `TRUE`, `NA` rows are omitted. If `FALSE`, an error occurs in case of `NA` rows. TRUE default option.

Value

A `GSSTDA` object. It contains:

- the matrix with the normal space `normal_space`,
- the matrix of the disease components normal_space `matrix_disease_component`,
- a matrix with the results of the application of proportional hazard models for each gene (`cox_all_matrix`),
- the genes selected for mapper `genes_disease_componen`,
- the matrix of the disease components with information from these genes only `genes_disease_component`
- and a `mapper_obj` object. This `mapper_obj` object contains the values of the intervals (interval_data), the samples included in each interval (sample_in_level), information about the cluster to which the individuals in each interval belong (clustering_all_levels), a list including the individuals contained in each detected node (node_samples), their size (node_sizes), the average of the filter function values of the individuals of each node (node_average_filt) and the adjacency matrix linking the nodes (adj_matrix). Moreover, information is provided on the number of nodes, the average node size, the standard deviation of the node size, the number of connections between nodes, the proportion of connections to all possible connections and the number of ramifications.

Examples

```
GSSTDA_object <- GSSTDA(full_data,  survival_time, survival_event, case_tag,
                gen_select_type="Top_Bot", percent_gen_select=10,
                num_intervals = 4, percent_overlap = 50,
                distance_type = "euclidean", num_bins_when_clustering = 8,
                clustering_type = "hierarchical", linkage_type = "single")
```

`levels_to_nodes` *Extract Information about Nodes*

Description

Extract the nodes information based on information about clustering. The individuals who are part of each node are identified

Usage

```
levels_to_nodes(clust_all_levels_list)
```

Arguments

`clust_all_levels_list`	A list with information on the levels obtained from the `clust_all_levels` function.

Value

A list including the individuals content of each detected node. List of character vectors. Each of the vectors contains the names of the individuals at each node.

`lp_norm_k_powers_surv` *Filtering function*

Description

A filtering function for mapper that projects $$R^n into R. It calculates for each column of the matrix (each patient), its value of the filtering function. Specifically, it computes the vector magnitude in the [L_p] norm (as well as k powers of this magnitude) of the vector resulting of weighting each element of the column vector by the Z score obtained by fitting a cox proportional hazard model to the level of each gene. For further information see "Progression Analysis of Disease with Survival (PAD-S) by SurvMap identifies different prognostic subgroups of breast cancer in a large combined set of transcriptomics and methylation studies"

Usage

```
lp_norm_k_powers_surv(genes_disease_component, p, k, cox_all_matrix)
```

Arguments

`genes_disease_component`	Disease component matrix (output of the function of `generate_disease_component`), after having selected the rows corresponding to the selected genes.
`p`	integer. It indicates the p norm to be calculated. If k = 1 and p = 2, the function computes the standard (Euclidean) vector magnitude of each column. For larger values of p the weight of genes with larger levels is greater.
`k`	integer. Powers of the vector magnitude. If k = 1 and p = 2, the function computes the standard (Euclidean) vector magnitude of each column.
`cox_all_matrix`	A matrix with the output of the `cox_all_genes` function that stores the information of all cox proportional hazard model tests for each gene in the dataset.

Value

A numeric vector including the values produced by the function for each sample in the dataset.

`mapper` *Mapper object*

Description

TDA are persistent homology and mapper. Persistent homology borrows ideas from abstract algebra to identify particular aspects related to the shape of the data such as the number of connected components and the presence of higher-dimensional holes, whereas mapper condenses the information of high-dimensional datasets into a combinatory graph or simplicial complex that is referred to as the skeleton of the dataset. This implementation is the mapper of one dimension, i.e. using only one filter function value.

Usage

```
mapper(
  full_data,
  filter_values,
  num_intervals = 5,
  percent_overlap = 40,
  distance_type = "cor",
  clustering_type = "hierarchical",
  num_bins_when_clustering = 10,
  linkage_type = "single",
  optimal_clustering_mode = "",
  na.rm = TRUE
)
```

Arguments

`full_data`	Input matrix whose columns correspond to the individuals and rows to the features.
`filter_values`	Vector obtained after applying the filtering function to the input matrix, i.e, a vector with the filtering function values for each included sample.
`num_intervals`	Number of intervals used to create the first sample partition based on filtering values. 5 default option.
`percent_overlap`	Percentage of overlap between intervals. Expressed as a percentage. 40 default option.
`distance_type`	Type of distance to be used for clustering. Choose between correlation ("cor") and euclidean ("euclidean"). "cor" default option.
`clustering_type`	Type of clustering method. Choose between "hierarchical" and "PAM" ("partition around medoids") options. "hierarchical" default option.
`num_bins_when_clustering`	Number of bins to generate the histogram employed by the standard optimal number of cluster finder method. Parameter not necessary if the "optimal_clust_mode" option is "silhouette" or the "clust_type" is "PAM". 10 default option.
`linkage_type`	Linkage criteria used in hierarchical clustering. Choose between "single" for single-linkage clustering, "complete" for complete-linkage clustering or "average" for average linkage clustering (or UPGMA). Only necessary for hierarchical clustering. "single" default option.
`optimal_clustering_mode`	Method for selection optimal number of clusters. It is only necessary if the chosen type of algorithm is hierarchical. In this case, choose between "standard" (the method used in the original mapper article) or "silhouette". In the case of the PAM algorithm, the method will always be "silhouette".
`na.rm`	`logical`. If `TRUE`, `NA` rows are omitted. If `FALSE`, an error occurs in case of `NA` rows. TRUE default option.

Value

A `mapper_obj` object. It contains the values of the intervals (interval_data), the samples included in each interval (sample_in_level), information about the cluster to which the individuals in each interval belong (clustering_all_levels), a list including the individuals contained in each detected node (node_samples), their size (node_sizes), the average of the filter function values of the individuals of each node (node_average_filt) and the adjacency matrix linking the nodes (adj_matrix).

Examples

```
control_tag_cases <- which(case_tag == "NT")
geneSelection_object <- gene_selection(full_data, survival_time, survival_event, control_tag_cases,
gen_select_type ="top_bot", num_gen_select = 10)

mapper_object <- mapper(full_data = geneSelection_object[["genes_disease_component"]],
filter_values = geneSelection_object[["filter_values"]],
num_intervals = 5,
percent_overlap = 40, distance_type = "cor",
clustering_type = "hierarchical",
linkage_type = "single")
```

---

map_to_color *Map to color*

Description

Auxiliary function that maps a numeric vector, the average node filtering function values, to a color vector.

Usage

```
map_to_color(x, limits = NULL)
```

Arguments

`x`	A vector of numeric values storing the average filtering function values found in the samples placed into a specific node.
`limits`	A two element numeric vector including the range of values. This is optional.

Value

A vector of the same length of x with colors ranging from blue to red.

---

one_D_Mapper *one_D_Mapper*

Description

Wrapping function to carry out the complete process.

Usage

```
one_D_Mapper(mapper_object_ini)
```

Arguments

`mapper_object_ini`	Mapper TDA initializated object generated by `mapper` function.

Value

A `mapper_obj` object. It contains the values of the intervals (interval_data), the samples included in each interval (sample_in_level), information about the cluster to which the individuals in each interval belong (clustering_all_levels), a list including the individuals contained in each detected node (node_samples), their size (node_sizes), the average of the filter function values of the individuals of each node (node_average_filt) and the adjacency matrix linking the nodes (adj_matrix). Moreover, information is provided on the number of nodes, the average node size, the standard deviation of the node size, the number of connections between nodes, the proportion of connections to all possible connections and the number of ramifications.

`plot_DGSA` *plot DGSA*

Description

It draws the heatmap of the DGSA result by selecting the 100 genes with the highest variability between samples.

Usage

```
plot_DGSA(selected_matrix_disease_component, case_tag)
```

Arguments

`selected_matrix_disease_component`	Disease component matrix of the selected genes that contains the disease component of all patients. Output of the function `generate_disease_component`.
`case_tag`	Character vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. It must be indicated for each patient whether he/she is healthy or not. One value should be used to indicate whether the patient is healthy and another value should be used to indicate whether the patient's sample is tumourous. The user will then be asked which one indicates whether the patient is healthy. Only two values are valid in the vector in total.

Value

The heatmap of the DGSA result.

plot_mapper *Plot mapper*

Description

This function produces an interactive network plot using the `visNetork` function from the mapper results.

Usage

```
plot_mapper(mapper_object, trans_node_size = TRUE, exp_to_res = 1/2)
```

Arguments

`mapper_object`	A list produced as an output of the `one_D_Mapper` function.
`trans_node_size`	Logical, it indicates whether you want to resize the size of the nodes. `TRUE` default option.
`exp_to_res`	Only necessary if trans_node_size is `TRUE`. An exponent of the form 1/n to which the node sizes must be raised in order to resize them.

Value

Plots an interactive network using the `visNetwork` function.

Examples

```
# Create data object
data_object <- list("full_data" = full_data, "survival_time" = survival_time,
                   "survival_event" = survival_event, "case_tag" = case_tag)
class(data_object) <- "data_object"

#Select gene from data object
geneSelection_object <- geneSelection(data_object, gen_select_type="top_bot",
 percent_gen_select=10)

mapper_object <- mapper(full_data = geneSelection_object[["genes_disease_component"]],
filter_values = geneSelection_object[["filter_values"]],
num_intervals = 5,
percent_overlap = 40, distance_type = "cor",
clustering_type = "hierarchical",
linkage_type = "single")
plot_mapper(mapper_object)
```

`results_DGSA` *results DGSA*

Description

It calculates the 100 genes with the highest variability in the matrix disease component between samples and use them to draw the heat map.

Usage

```
results_DGSA(matrix_disease_component, case_tag)
```

Arguments

`matrix_disease_component`	Disease component matrix that contains the disease component of all patients. Output of the function `generate_disease_component`.
`case_tag`	Character vector of the same length as the number of columns of full_data. Patients must be in the same order as in full_data. It must be indicated for each patient whether he/she is healthy or not. One value should be used to indicate whether the patient is healthy and another value should be used to indicate whether the patient's sample is tumourous. The user will then be asked which one indicates whether the patient is healthy. Only two values are valid in the vector in total.

Value

A heatmap of the 100 genes with the highest variability in the matrix disease component.

`samples_in_levels` *Samples in levels*

Description

This function returns a list of vectors containing the individuals included at each level, i.e. the vectors of individuals with a value of the filter function within each of the intervals.

Usage

```
samples_in_levels(interval_data, filter_values)
```

Arguments

`interval_data`	Filter function intervals. List with the set of intervals for the filtering function values produced by the `get_intervals_One_D` function.
`filter_values`	Vector obtained after applying the filtering function to the input matrix, i.e, a vector with the filtering function values for each included sample.

Value

A list of character vectors with the samples included in each of the levels (i.e. each of the intervals of the values of the filter functions).

`survival_event` *Survival event vector*

Description

Character vector of length 121 containing whether or not the patient is deceased.

Usage

```
data(survival_event, package = "GSSTDA")
```

Format

Character vector of length 121.

A value of "0" indicates that the patient did not pass away during follow-up, a value of "1" indicates that the patient did. Samples from healthy tissue contain a value of `NA`.

Source

The data are from the study GSE42568. Information extracted from the file GSE42568_family.soft.gz available at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE42568.

`survival_time` *Survival time vector*

Description

Numeric vector of length 121 containing the time in months until the death of the patient or until the end of the follow-up in case the patient has not passed away.

Usage

```
data(survival_time, package = "GSSTDA")
```

Format

Numeric vector of length 121.

Time in months. Samples from healthy tissue contain a value of `NA`.

Source

The data are from the study GSE42568. Information extracted from the file GSE42568_family.soft.gz available at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE42568

Index