Package 'psichomics'

April 12, 2018

Title Graphical Interface for Alternative Splicing Quantification, Analysis and Visualisation

Version 1.4.5

Encoding UTF-8

Description Interactive R package with an intuitive Shiny-based graphical interface for alternative splicing quantification and integrative analyses of alternative splicing and gene expression from large transcriptomic datasets, including those from The Cancer Genome Atlas (TCGA) and the Genotype-Tissue Expression project (GTEx), as well as user-owned data. The tool interactively performs survival, dimensionality reduction and median- and variance-based differential splicing and gene expression analyses that benefit from the incorporation of clinical and molecular sample-associated features (such as tumour stage or survival). Interactive visual access to genomic mapping and functional annotation of selected alternative splicing events is also included.

Depends R (>= 3.4), shiny (>= 1.0.3), shinyBS

License MIT + file LICENSE

LazyData true

RoxygenNote 6.0.1

Imports AnnotationHub, cluster, colourpicker, data.table, digest, dplyr, DT (>= 0.2), edgeR, fastICA, fastmatch, ggplot2, ggrepel, grDevices, highcharter (>= 0.5.0), htmltools, httr, jsonlite, limma, miscTools, pairsD3, plyr, Rcpp (>= 0.12.14), R.utils, shinyjs, stringr, stats, survival, tools, utils, XML, xtable, methods

Suggests testthat, knitr, parallel, devtools, rmarkdown, gplots, covr, car

LinkingTo Rcpp

VignetteBuilder knitr

Collate 'RcppExports.R' 'utils.R' 'globalAccess.R' 'app.R' 'analysis.R' 'analysis_correlation.R' 'analysis_diffExpression.R' 'analysis_diffExpression_event.R' 'analysis_diffExpression_table.R' 'analysis_diffSplicing.R' 'analysis_diffSplicing_event.R' 'analysis_diffSplicing_table.R' 'analysis_dimReduction.R' 'analysis_dimReduction_ica.R' 'analysis_dimReduction_pca.R' 'analysis_information.R'

'analysis_survival.R' 'analysis_template.R' 'data.R'
'formats.R' 'data_firebrowse.R'
'data_geNormalisationFiltering.R' 'data_gtex.R'
'data_inclusionLevels.R' 'data_local.R' 'events_suppa.R'
'events_vastTools.R' 'events_miso.R' 'events_mats.R' 'events.R'
'formats_firebrowseGeneExpression.R'
'formats_firebrowseJunctionReads.R'
'formats_firebrowseMergeClinical.R'
'formats_firebrowseNormalizedGeneExpression.R'
'formats_genericClinical.R' 'formats_genericGeneExpression.R'
'formats_genericInclusionLevels.R'
'formats_genericJunctionReads.R' 'formats_genericSampleInfo.R'
'formats_gtexClinical.R' 'formats_gtexGeneReadsFormat.R'
'formats_gtexJunctionReads.R' 'formats_gtexSampleInfo.R'
'formats_gtexV7Clinical.R' 'formats_gtexV7JunctionReads.R'
'formats_psichomicsGeneExpression.R'
'formats_psichomicsInclusionLevels.R' 'groups.R' 'help.R'

biocViews Sequencing, RNASeq, AlternativeSplicing, DifferentialSplicing, Transcription, GUI, PrincipalComponent, Survival, BiomedicalInformatics, Transcriptomics, Visualization, MultipleComparison, GeneExpression, DifferentialExpression

URL https://github.com/nuno-agostinho/psichomics

BugReports https://github.com/nuno-agostinho/psichomics/issues

NeedsCompilation yes

Author Nuno Saraiva-Agostinho [aut, cre], Nuno Luís Barbosa-Morais [aut, led, ths], André Falcão [ths], Lina Gallego Paez [ctb], Marie Bordone [ctb], Teresa Maia [ctb], Mariana Ferreira [ctb], Ana Carolina Leote [ctb], Bernardo de Almeida [ctb]

Maintainer Nuno Saraiva-Agostinho <nunodanielagostinho@gmail.com>

R topics documented:

`addTCGAdata` *Creates a UI set with options to add data from TCGA/Firebrowse*

Description

Creates a UI set with options to add data from TCGA/Firebrowse

Usage

```
addTCGAdata(ns)
```

Arguments

`ns`	Namespace function

Value

A UI set that can be added to a UI definition

appendNewGroups *Append new groups to already existing groups*

Description

Retrieve previous groups, rename duplicated group names in the new groups and append new groups to the previous ones

Usage

```
appendNewGroups(type, new, clearOld = FALSE)
```

Arguments

type	Character: type of groups (either "Patients", "Samples", "ASevents" or "Genes")
new	Rows of groups to be added
clearOld	Boolean: clear old groups?

Value

NULL (this function is used to modify the Shiny session's state)

appServer *Server logic*

Description

Instructions to build the Shiny app

Usage

```
appServer(input, output, session)

analysesServer(input, output, session)

correlationServer(input, output, session)

diffExpressionServer(input, output, session)

diffExpressionEventServer(input, output, session)

diffExpressionTableServer(input, output, session)

diffSplicingServer(input, output, session)

diffSplicingEventServer(input, output, session)

diffSplicingTableServer(input, output, session)
```

```
dimReductionServer(input, output, session)

icaServer(input, output, session)

pcaServer(input, output, session)

infoServer(input, output, session)

survivalServer(input, output, session)

templateServer(input, output, session)

dataServer(input, output, session)

firebrowseServer(input, output, session)

geNormalisationFilteringServer(input, output, session)

gtexDataServer(input, output, session)

inclusionLevelsServer(input, output, session)

localDataServer(input, output, session)

groupsServer(input, output, session)

helpServer(input, output, session)
```

Arguments

input	Input object
output	Output object
session	Session object

Value

NULL (this function is used to modify the Shiny session's state)

---

appUI *User interface*

Description

The user interface (UI) controls the layout and appearance of the app. All CSS modifications are in the file `shiny/www/styles.css`

Usage

```
appUI()

analysesUI(id, tab)
```

```
correlationUI(id)

diffExpressionUI(id, tab)

diffExpressionEventUI(id)

diffExpressionTableUI(id)

diffSplicingUI(id, tab)

diffSplicingEventUI(id)

diffSplicingTableUI(id)

dimReductionUI(id, tab)

icaUI(id)

pcaUI(id)

infoUI(id)

survivalUI(id)

templateUI(id)

dataUI(id, tab)

firebrowseUI(id, panel)

geNormalisationFilteringUI(id, panel)

gtexDataUI(id, panel)

inclusionLevelsUI(id, panel)

localDataUI(id, panel)

groupsUI(id, tab)

helpUI(id, tab)
```

Arguments

id	Character: identifier
tab	Function to process HTML elements
panel	Function to enclose interface

Value

HTML elements

areSplicingEvents *Check if string identifies splicing events*

Description

Check if string identifies splicing events

Usage

```
areSplicingEvents(char, num = 6)
```

Arguments

char	Character vector
num	Integer: number of elements to check

Value

TRUE if first elements of the vector identify splicing events; FALSE, otherwise

articleUI *Return the interface to display an article*

Description

Return the interface to display an article

Usage

```
articleUI(article)
```

Arguments

article	PubMed article

Value

HTML to render an article's interface

ASquantFileInput *File input for alternative splicing quantification*

Description

File input for alternative splicing quantification

Usage

```
ASquantFileInput(ASquantFileId, speciesId, assemblyId)
```

Arguments

ASquantFileId	Character: identifier for alternative splicing quantification input
speciesId	Character: identifier for species selection input
assemblyId	Character: identifier for genome assembly selection input

Value

HTML elements

assignColours *Assign colours to groups*

Description

Assign colours to groups

Usage

```
assignColours(new, groups = NULL)
```

Arguments

new	Matrix: groups to which colours will be assigned
groups	Matrix: groups to check which colours are already assigned

Value

Groups with an added column to state the colour

basicStats *Basic statistics performed on data*

Description

Variance and median of each group. If data has 2 groups, also calculates the delta variance and delta median.

Usage

```
basicStats(data, groups)
```

Arguments

data	Numeric, data frame or matrix: data for one gene or alternative splicing event
groups	List of characters (list of groups containing data identifiers) or character vector (group of each value in data)

Value

HTML elements

blendColours *Blend two HEX colours*

Description

Blend two HEX colours

Usage

```
blendColours(colour1, colour2, colour1Percentage = 0.5)
```

Arguments

colour1	Character: HEX colour
colour2	Character: HEX colour
colour1Percentage	Character: percentage of colour 1 mixed in blended colour (default is 0.5)

Value

Character representing an HEX colour

Source

Code modified from https://stackoverflow.com/questions/5560248

Examples

```
psichomics:::blendColours("#3f83a3", "#f48000")
```

browserHistory *Enable history navigation*

Description

Navigate app according to the location given by the navigation bar. Code and logic adapted from https://github.com/daattali/advanced-shiny/blob/master/navigate-history

Usage

```
browserHistory(navId, input, session)
```

Arguments

navId	Character: identifier of the navigation bar
input	Input object
session	Session object

Value

NULL (this function is used to modify the Shiny session's state)

bsModal2 *Modified version of* shinyBS::bsModal

Description

bsModal is used within the UI to create a modal window. This allows to modify the modal footer.

Usage

```
bsModal2(id, title, trigger, ..., size = NULL, footer = NULL,
  style = NULL)
```

Arguments

id	A unique identifier for the modal window
title	The title to appear at the top of the modal
trigger	The id of a button or link that will open the modal.
...	UI elements to include within the modal
size	Character: Modal size (small, default or large)
footer	UI set: List of elements to include in the footer
style	Character: message style can be warning, error, info or NULL

Value

HTML elements

`calculateInclusionLevels` *Calculate inclusion levels using alternative splicing event annotation and junction quantification for many samples*

Description

Calculate inclusion levels using alternative splicing event annotation and junction quantification for many samples

Usage

```
calculateInclusionLevels(eventType, junctionQuant, annotation, minReads = 10)
```

Arguments

`eventType`	Character: type of the alternative event to calculate
`junctionQuant`	Matrix: junction quantification with samples as columns and junctions as rows
`annotation`	Data.frame: alternative splicing annotation related to event type
`minReads`	Integer: minimum of total reads required to consider the quantification as valid (10 by default)

Value

Matrix with inclusion levels

`calculateLoadingsContribution` *Calculate the contribution of PCA loadings to the selected principal components*

Description

Total contribution of a variable is calculated as per: ((Cx * Ex) + (Cy * Ey))/(Ex + Ey), where Cx and Cy are the contributions of a variable to principal components (x and y) and Ex and Ey are the eigenvalues of principal components (x and y)

Usage

```
calculateLoadingsContribution(pca, pcX = 1, pcY = 2)
```

Arguments

`pca`	`prcomp` object
`pcX`	Character: name of the X axis of interest from the PCA
`pcY`	Character: name of the Y axis of interest from the PCA

Value

Data frame containing the correlation between variables and selected principal components and the contribution of variables to the selected principal components (both individual and total contribution)

Source

http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/

---

checkFileFormat *Checks the format of a file*

Description

Checks the format of a file

Usage

```
checkFileFormat(format, head, filename = "")
```

Arguments

format	Environment: format of the file
head	Data.frame: head of the file to check
filename	Character: name of the file

Details

The name of the file may also be required to be considered of a certain format.

Value

TRUE if the file is of the given format; otherwise, returns FALSE

---

checkFirebrowse *Return an user interface depending on the status of the Firebrowse API*

Description

If the API is working, it'll be loaded. Else, a message will appear warning the user that the API is down and that will let check again if the API is back online.

Usage

```
checkFirebrowse(ns)
```

Arguments

ns	Namespace function

Value

HTML elements

---

checkIntegrity *Compute the 32-byte* MD5 *hashes of one or more files and check with given* md5 *file*

Description

Compute the 32-byte MD5 hashes of one or more files and check with given md5 file

Usage

```
checkIntegrity(filesToCheck, md5file)
```

Arguments

filesToCheck	Character: files to calculate and match MD5 hashes
md5file	Character: file containing correct MD5 hashes

Value

Logical vector showing TRUE for files with matching md5sums and FALSE for files with non-matching md5sums

---

checkSurvivalInput *Prepare survival terms in case of valid input*

Description

Prepare survival terms in case of valid input

Usage

```
checkSurvivalInput(session, input, coxph = FALSE)
```

Arguments

session	Shiny session
input	Shiny input
coxph	Boolean: prepare data for Cox models? FALSE by default

Value

NULL (this function is used to modify the Shiny session's state)

closeProgress *Close the progress even if there's an error*

Description

Close the progress even if there's an error

Usage

```
closeProgress(message = NULL, global = if (isRunning()) sharedData else
  getHidden())
```

Arguments

message	Character: message to show in progress bar
global	Global Shiny variable where all data is stored

Value

NULL (this function is used to modify the Shiny session's state)

clusterICAset *Server logic for clustering ICA data*

Description

Server logic for clustering ICA data

Usage

```
clusterICAset(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

clusterSet *Server logic for clustering PCA data*

Description

Server logic for clustering PCA data

Usage

```
clusterSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

correlateGEandAS *Correlate gene expression data against alternative splicing quantification*

Description

Test for association between paired samples' gene expression (for any genes of interest) and alternative splicing quantification.

Usage

```
correlateGEandAS(geneExpr, psi, gene, ASevents = NULL, ...)
```

Arguments

geneExpr	Matrix or data frame: gene expression data
psi	Matrix or data frame: alternative splicing quantification data
gene	Character: gene symbol for genes of interest
ASevents	Character: alternative splicing events to correlate with gene expression of a gene (if NULL, the events will be automatically retrieved from the given gene)
...	Arguments passed on to stats:::cor.test.default

alternative indicates the alternative hypothesis and must be one of "two.sided", "greater" or "less". You can specify just the initial letter. "greater" corresponds to positive association, "less" to negative association.

method a character string indicating which correlation coefficient is to be used for the test. One of "pearson", "kendall", or "spearman", can be abbreviated.

exact a logical indicating whether an exact p-value should be computed. Used for Kendall's τ and Spearman's ρ. See 'Details' for the meaning of NULL (the default).

conf.level confidence level for the returned confidence interval. Currently only used for the Pearson product moment correlation coefficient if there are at least 4 complete pairs of observations.

continuity logical: if true, a continuity correction is used for Kendall's τ and Spearman's ρ when not computed exactly.

Value

List of correlations where each element is organised as such:

eventID	Alternative splicing event identifier
cor	Correlation between gene expression and alternative splicing quantification of one alternative splicing event
geneExpr	Gene expression for the selected gene
psi	Alternative splicing quantification for the alternative splicing event

Examples

```
annot <- readFile("ex_splicing_annotation.RDS")
junctionQuant <- readFile("ex_junctionQuant.RDS")
psi <- quantifySplicing(annot, junctionQuant, eventType=c("SE", "MXE"))

geneExpr <- readFile("ex_gene_expression.RDS")
correlateGEandAS(geneExpr, psi, "ALDOA")
```

---

createDataTab *Render a specific data tab (including data table and related interface)*

Description

Render a specific data tab (including data table and related interface)

Usage

```
createDataTab(index, data, name, session, input, output)
```

Arguments

index	Integer: index of the data to load
data	Data frame: data with everything to load
name	Character: name of the dataset
session	Shiny session
input	Shiny session input
output	Shiny session output

Value

NULL (this function is used to modify the Shiny session's state)

createDensitySparklines *Create density sparklines for inclusion levels*

Description

Create density sparklines for inclusion levels

Usage

```
createDensitySparklines(data, events, areSplicingEvents = TRUE,
  groups = NULL, geneExpr = NULL)
```

Arguments

data	Character: HTML-formatted data series of interest
events	Character: event identifiers
areSplicingEvents	Boolean: are these splicing events (TRUE) or gene expression (FALSE)?
groups	Character: name of the groups used for differential analyses
geneExpr	Character: name of the gene expression dataset

Value

HTML element with sparkline data

createEventPlotting *Create plot for events*

Description

Create plot for events

Usage

```
createEventPlotting(df, x, y, params, highlightX, highlightY, highlightParams,
  selected, selectedParams, labelled, labelledParams, xlim, ylim)
```

Arguments

df	Data frame
x	Character: name of the variable used for the X axis
y	Character: name of the variable used for the Y axis
params	List of parameters to pass to geom_point related to most points
highlightX	Integer: region of points in X axis to highlight
highlightY	Integer: region of points in Y axis to highlight

highlightParams	List of parameters to pass to geom_point related to highlighted points
selected	Integer: index of rows/points to be coloured
selectedParams	List of parameters to pass to geom_point related to selected points
labelled	Integer: index of rows/points to be labelled
labelledParams	List of parameters to pass to geom_label_repel related to labelled points
xlim	Numeric: limits of X axis
ylim	Numeric: limits of Y axis

Value

List containing HTML elements and highlighted points

`createGroup` *Prepare to create group according to specific details*

Description

Prepare to create group according to specific details

Usage

```
createGroup(session, input, output, id, type)
```

Arguments

session	Shiny session
input	Shiny input
output	Shiny output
id	Character: identifier of the group selection
type	Character: type of group to create

Value

NULL (this function is used to modify the Shiny session's state)

createGroupByColumn *Split elements into groups based on a given column of a dataset*

Description

Elements are identified by their respective row name.

Usage

```
createGroupByColumn(col, dataset)

createGroupByAttribute(col, dataset)
```

Arguments

col	Character: column name
dataset	Matrix or data frame: dataset

Value

Named list with each unique value from a given column and respective elements

Examples

```
df <- data.frame(gender=c("male", "female"),
                 stage=paste("stage", c(1, 3, 1, 4, 2, 3, 2, 2)))
rownames(df) <- paste0("patient-", LETTERS[1:8])
createGroupByAttribute(col="stage", dataset=df)
```

createGroupById *Create groups based on given row indexes or identifiers*

Description

Create groups based on given row indexes or identifiers

Usage

```
createGroupById(session, rows, identifiers)
```

Arguments

session	Shiny session
rows	Character: comma-separated row indexes or identifiers
identifiers	Character: available identifiers

Value

Character: values based on given row indexes or identifiers

createGroupFromInput *Set new groups according to the user input*

Description

Set new groups according to the user input

Usage

```
createGroupFromInput(session, input, output, dataset, id, type)
```

Arguments

session	Shiny session
input	Shiny input
output	Shiny output
dataset	Data frame or matrix: dataset of interest
id	Character: identifier of the group selection
type	Character: type of group to create

Value

Matrix with the group names and respective elements

createJunctionsTemplate *Creates a template of alternative splicing junctions*

Description

Creates a template of alternative splicing junctions

Usage

```
createJunctionsTemplate(nrow, program = character(0),
  event.type = character(0), chromosome = character(0),
  strand = character(0), id = character(0))
```

Arguments

nrow	Integer: Number of rows
program	Character: Program used to get the junctions
event.type	Character: Event type of the respective events
chromosome	Character: Chromosome of the junctions
strand	Character: positive ("+") or negative ("-") strand of the event
id	Character: events' ID

Value

A data frame with the junctions coordinate names pre-filled with NAs

Examples

```
psichomics:::createJunctionsTemplate(nrow = 8)
```

`createOptimalSurvData` *Create survival data based on a PSI cutoff*

Description

Data is presented in the table for statistical analyses

Usage

```
createOptimalSurvData(eventPSI, clinical, censoring, event, timeStart, timeStop,
  match, patients, samples)
```

Arguments

`eventPSI`	Numeric: alternative splicing quantification for multiple samples relative to a single splicing event
`clinical`	Data frame: clinical data
`censoring`	Character: censor using "left", "right", "interval" or "interval2"
`event`	Character: name of column containing time of the event of interest
`timeStart`	Character: name of column containing starting time of the interval or follow up time
`timeStop`	Character: name of column containing ending time of the interval (only relevant for interval censoring)
`match`	Matrix: match between samples and patients
`patients`	Character: patient identifiers (only required if the `clinical` argument is not handed)
`samples`	Character: samples to use when assigning values per patient (if NULL, all samples will be used)

Value

Survival data including optimal PSI cutoff, minimal survival p-value and HTML element required to plot survival curves

createSparklines *Create sparkline charts to be used in a data table*

Description

Create sparkline charts to be used in a data table

Usage

```
createSparklines(hc, data, events, FUN, groups = NULL, geneExpr = NULL)
```

Arguments

hc	Highcharts object
data	Character: HTML-formatted data series of interest
events	Character: event identifiers
FUN	Character: JavaScript function to execute when clicking on a chart
groups	Character: name of the groups used for differential analyses
geneExpr	Character: name of the gene expression dataset

Value

HTML element with sparkline data

diffAnalyses *Perform statistical analyses*

Description

Perform statistical analyses

Usage

```
diffAnalyses(data, groups = NULL, analyses = c("wilcoxRankSum", "ttest",
  "kruskal", "levene", "fligner"), pvalueAdjust = "BH", geneExpr = NULL,
  psi = NULL)

diffAnalysis(data, groups = NULL, analyses = c("wilcoxRankSum", "ttest",
  "kruskal", "levene", "fligner"), pvalueAdjust = "BH", geneExpr = NULL,
  psi = NULL)
```

Arguments

data	Data frame or matrix: gene expression or alternative splicing quantification
groups	Named list of characters (containing elements belonging to each group) or character vector (containing the group of each individual sample); if NULL, sample types are used instead when available, e.g. normal, tumour and metastasis
analyses	Character: statistical tests to perform (see Details)
pvalueAdjust	Character: method used to adjust p-values (see Details)
geneExpr	Character: name of the gene expression dataset (only required for density sparklines available in the interactive mode)
psi	Data frame or matrix: alternative splicing quantification (defunct argument, use data instead)

Details

The following statistical analyses may be performed by including the respective string in the analysis argument:

- ttest - Unpaired t-test (2 groups)
- wilcoxRankSum - Wilcoxon Rank Sum test (2 groups)
- kruskal - Kruskal test (2 or more groups)
- levene - Levene's test (2 or more groups)
- fligner - Fligner-Killeen test (2 or more groups)
- density - Sample distribution per group (only usable through the visual interface)

The following methods for p-value adjustment are supported by using the respective string in the pvalueAdjust argument:

- none: do not adjust p-values
- BH: Benjamini-Hochberg's method (false discovery rate)
- BY: Benjamini-Yekutieli's method (false discovery rate)
- bonferroni: Bonferroni correction (family-wise error rate)
- holm: Holm's method (family-wise error rate)
- hochberg: Hochberg's method (family-wise error rate)
- hommel: Hommel's method (family-wise error rate)

Value

Table of statistical analyses

Examples

```
# Calculate PSI for skipped exon (SE) and mutually exclusive (MXE) events
eventType <- c("SE", "MXE")
annot <- readFile("ex_splicing_annotation.RDS")
junctionQuant <- readFile("ex_junctionQuant.RDS")

psi <- quantifySplicing(annot, junctionQuant, eventType=c("SE", "MXE"))
group <- c(rep("Normal", 3), rep("Tumour", 3))
diffAnalyses(psi, group)
```

diffAnalysesPlotSet *Set of functions to plot differential analyses*

Description

Set of functions to plot differential analyses

Usage

```
diffAnalysesPlotSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

diffAnalysesSet *Set of functions to perform differential analyses*

Description

Set of functions to perform differential analyses

Usage

```
diffAnalysesSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

diffAnalysesTableSet *Set of functions to render data table for differential analyses*

Description

Set of functions to render data table for differential analyses

Usage

```
diffAnalysesTableSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

diffExpressionPlotSet *Set of functions to plot differential analyses*

Description

Set of functions to plot differential analyses

Usage

```
diffExpressionPlotSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

diffExpressionSet *Set of functions to perform differential analyses*

Description

Set of functions to perform differential analyses

Usage

```
diffExpressionSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

diffExpressionTableSet *Set of functions to render data table for differential analyses*

Description

Set of functions to render data table for differential analyses

Usage

```
diffExpressionTableSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

disableTab *Enable or disable a tab from the* navbar

Description

Enable or disable a tab from the navbar

Usage

```
disableTab(tab)

enableTab(tab)
```

Arguments

tab	Character: tab

Value

NULL (this function is used to modify the Shiny session's state)

display *Display characters in the command-line*

Description

Display characters in the command-line

Usage

```
display(char, timeStr = "Time difference of")
```

Arguments

char	Character: message
timeStr	Character: message when a difftime object is passed to the char argument

Value

NULL (display message in command-line)

downloadFiles *Download files to a given directory*

Description

Download files to a given directory

Usage

```
downloadFiles(url, folder, download = download.file, ...)
```

Arguments

url	Character: download links
folder	Character: directory to store the downloaded archives
download	Function to use to download files
...	Extra parameters passed to the download function

Value

Invisible TRUE if every file was successfully downloaded

Examples

```
## Not run:
url <- paste0("https://unsplash.it/400/300/?image=", 570:572)
downloadFiles(url, "~/Pictures")

# Download without printing to console
downloadFiles(url, "~/Pictures", quiet = TRUE)

## End(Not run)
```

endProcess *Signal the program that a process has ended*

Description

Style button to show processing is not occurring. Also, close the progress bar (if TRUE) and print the difference between the current time and a given time (if given time is not NULL)

Usage

```
endProcess(id, time = NULL, closeProgressBar = TRUE)
```

Arguments

id	Character: button identifier
time	POSIXct object: start time needed to show the interval time (if NULL, the time interval is not displayed)
closeProgressBar	Boolean: close progress bar? TRUE by default

Value

NULL (this function is used to modify the Shiny session's state)

---

ensemblToUniprot *Convert an Ensembl identifier to the respective UniProt identifier*

Description

Convert an Ensembl identifier to the respective UniProt identifier

Usage

```
ensemblToUniprot(protein)
```

Arguments

protein	Character: Ensembl identifier

Value

UniProt protein identifier

Examples

```
gene <- "ENSG00000173262"
ensemblToUniprot(gene)

protein <- "ENSP00000445929"
ensemblToUniprot(protein)
```

escape *Escape symbols for use in regular expressions*

Description

Escape symbols for use in regular expressions

Usage

```
escape(...)
```

Arguments

...	Characters to be pasted with no space

Value

Escaped string

eventPlotOptions *Options for event plotting*

Description

Options for event plotting

Usage

```
eventPlotOptions(session, df, xAxis, yAxis, labelSortBy)
```

Arguments

session	Shiny session
df	Data frame
xAxis	Character: currently selected variable for the X axis
yAxis	Character: currently selected variable for the Y axis
labelSortBy	Character: currently selected variable for the selectize element to sort differentially analysis

Value

HTML elements

export_highcharts *Add an exporting feature to a* highcharts *object*

Description

Add an exporting feature to a highcharts object

Usage

```
export_highcharts(hc, fill = "transparent", text = "Export")
```

Arguments

hc	A highcharts object
fill	Character: colour fill
text	Character: button text

Value

A highcharts object with an export button

fileBrowser *Interactive folder selection using a native dialogue*

Description

Interactive folder selection using a native dialogue

Usage

```
fileBrowser(default = NULL, caption = NULL, multiple = FALSE,
  directory = FALSE, system = Sys.info()["sysname"])
```

Arguments

default	Character: path to initial folder
caption	Character: caption on the selection dialogue
multiple	Boolean: allow to select multiple files?
directory	Boolean: allow to select directories instead of files?
system	Character: system name

Details

For macOS, it uses an Apple Script to display a folder selection dialogue. With default = NA, the initial folder selection is determined by default behaviour of the "choose folder" Apple Script command. Otherwise, paths are expanded with path.expand.

In Windows, it uses either ‘utils::choose.files‘ or ‘utils::choose.dir‘.

Value

A length one character vector, character NA if 'Cancel' was selected.

Source

Original code by wleepang: https://github.com/wleepang/shiny-directory-input

---

`fileBrowserInput` *File browser input*

Description

Input to interactively select a file or directory on the server

Usage

```
fileBrowserInput(id, label, value = NULL, placeholder = NULL,
  info = FALSE, infoFUN = NULL, infoPlacement = "right", infoTitle = "",
  infoContent = "")
```

Arguments

`id`	Character: input identifier
`label`	Character: input label (NULL to show no label)
`value`	Character: initial value (paths are expanded via `path.expand`)
`placeholder`	Character: placeholder when no file or folder is selected
`info`	Boolean: add information icon for tooltips and popovers
`infoFUN`	Function to use to provide information (e.g. `shinyBS::bsTooltip` and `shinyBS::bsPopover`)
`infoPlacement`	Character: placement of the information (top, bottom, right or left)
`infoTitle`	Character: text to show as title of information
`infoContent`	Character: text to show as content of information

Details

To show the dialog for file input, the `prepareFileBrowser` function needs to be included in the server logic.

This widget relies on `fileBrowser` to present an interactive dialogue to users for selecting a directory on the local filesystem. Therefore, this widget is intended for shiny apps that are run locally - i.e. on the same system that files/directories are to be accessed - and not from hosted applications (e.g. from https://www.shinyapps.io).

Value

A file browser input control that can be added to a UI definition.

Source

Original code by wleepang: https://github.com/wleepang/shiny-directory-input

See Also

updateFileBrowserInput and prepareFileBrowser

`filterGroups` *Filter groups with less data points than the threshold*

Description

Groups containing a number of non-missing values less than the threshold are discarded.

Usage

```
filterGroups(vector, group, threshold = 1)
```

Arguments

`vector`	Unnamed elements
`group`	Character: group of the elements
`threshold`	Integer: number of valid non-missing values by group

Value

Named vector with filtered elements from valid groups. The group of the respective element is given in the name.

Examples

```
# Removes groups with less than two elements
filterGroups(1:4, c("A", "B", "B", "D"), threshold=2)
```

`fisher` *Perform Fisher's exact test and return interface to show the results*

Description

Perform Fisher's exact test and return interface to show the results

Usage

```
fisher(data, groups)
```

Arguments

`data`	Numeric, data frame or matrix: data for one gene or alternative splicing event
`groups`	List of characters (list of groups containing data identifiers) or character vector (group of each value in `data`)

Value

HTML elements

fligner *Perform Fligner-Killeen test and return interface to show the results*

Description

Perform Fligner-Killeen test and return interface to show the results

Usage

```
fligner(data, groups, stat = NULL)
```

Arguments

data	Numeric, data frame or matrix: data for one gene or alternative splicing event
groups	List of characters (list of groups containing data identifiers) or character vector (group of each value in data)
stat	Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)

Value

HTML elements

geneExprFileInput *File input for gene expression*

Description

File input for gene expression

Usage

```
geneExprFileInput(geneExprFileId)
```

Arguments

geneExprFileId	Character: identifier for gene expression input

Value

HTML elements

geneExprSurvSet *Logic set to perform survival analysis based on gene expression cut-offs*

Description

Logic set to perform survival analysis based on gene expression cut-offs

Usage

```
geneExprSurvSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

geNormalisationFilteringInterface *Interface to normalise and filter gene expression*

Description

Interface to normalise and filter gene expression

Usage

```
geNormalisationFilteringInterface(ns)
```

Arguments

ns	Namespace function

Value

HTML elements

getASevent *Get or set globally accessible elements*

Description

Get or set globally accessible elements

Usage

```
getASevent()
```

Value

Getters return globally accessible data, whereas setters return NULL as they are only used to modify the Shiny session's state

See Also

getEvent, getClinicalMatchFrom, getGroups and getDifferentialAnalyses

getASevents *Get or set globally accessible elements*

Description

Get or set globally accessible elements

Usage

```
getASevents()
```

Value

Getters return globally accessible data, whereas setters return NULL as they are only used to modify the Shiny session's state

See Also

getEvent, getClinicalMatchFrom, getGroups and getDifferentialAnalyses

getAttributesTime *Retrieve the time for given columns in a clinical dataset*

Description

Retrieve the time for given columns in a clinical dataset

Usage

```
getAttributesTime(clinical, event, timeStart, timeStop = NULL,
  followup = "days_to_last_followup")

getColumnsTime(clinical, event, timeStart, timeStop = NULL,
  followup = "days_to_last_followup")
```

Arguments

clinical	Data frame: clinical data
event	Character: name of column containing time of the event of interest
timeStart	Character: name of column containing starting time of the interval or follow up time
timeStop	Character: name of column containing ending time of the interval (only relevant for interval censoring)
followup	Character: name of column containing follow up time

Value

Data frame containing the time for the given columns

Examples

```
df <- data.frame(followup=c(200, 300, 400), death=c(NA, 300, NA))
rownames(df) <- paste("patient", 1:3)
getAttributesTime(df, event="death", timeStart="death", followup="followup")
```

getClinicalDataForSurvival *Retrieve clinical data based on attributes required for survival analysis*

Description

Retrieve clinical data based on attributes required for survival analysis

Usage

```
getClinicalDataForSurvival(..., formulaStr = NULL)
```

Arguments

...	Character: names of columns to retrieve
formulaStr	Character: right-side of the formula for survival analysis

Value

Filtered clinical data

---

getClinicalMatchFrom *Get or set clinical matches from a given data type*

Description

Get or set clinical matches from a given data type

Usage

```
getClinicalMatchFrom(dataset, category = getCategory())

setClinicalMatchFrom(dataset, matches, category = getCategory())
```

Arguments

dataset	Character: data set name (e.g. "Junction quantification")
category	Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category
matches	Vector of integers: clinical matches of dataset

Value

Getters return globally accessible data, whereas setters return NULL as they are only used to modify the Shiny session's state

See Also

getEvent, getClinicalMatchFrom, getGroups and getDifferentialAnalyses

---

getData *Get global data*

Description

Get global data

Usage

```
getData()
```

Value

Variable containing all data of interest

getDataRows *Get rows of a data frame between two row indexes*

Description

Get rows of a data frame between two row indexes

Usage

```
getDataRows(i, data, firstRow, lastRow)
```

Arguments

i	Integer: current iteration
data	Data.frame: contains the data of interest
firstRow	Vector of integers: First row index of interest; value must be less than the respective last row index and less than the number of rows in the data frame
lastRow	Vector of integers: Last row index of interest; value must be higher than the respective first row index and less than the number of rows in the data frame

Details

For a given iteration i, returns data from `firstRow[i]` to `lastRow[i]`

Value

Data frame subset from two row indexes (returns NA if the first row index is NA)

getDifferentialAnalyses *Get or set differential splicing' elements for a data category*

Description

Get or set differential splicing' elements for a data category

Usage

```
getDifferentialAnalyses(category = getCategory())

setDifferentialAnalyses(differential, category = getCategory())

getDifferentialAnalysesFiltered(category = getCategory())

setDifferentialAnalysesFiltered(differential, category = getCategory())

getDifferentialAnalysesSurvival(category = getCategory())

setDifferentialAnalysesSurvival(survival, category = getCategory())
```

```
getDifferentialAnalysesResetPaging(category = getCategory())

setDifferentialAnalysesResetPaging(reset, category = getCategory())

getDifferentialAnalysesColumns(category = getCategory())

setDifferentialAnalysesColumns(columns, category = getCategory())
```

Arguments

category	Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category
differential	Data frame or matrix: differential analyses table
survival	Data frame or matrix: differential analyses' survival data
reset	Character: reset paging of differential analyses table?
columns	Character: differential analyses' column names

Value

Getters return globally accessible data, whereas setters return NULL as they are only used to modify the Shiny session's state

See Also

getEvent, getClinicalMatchFrom, getGroups and getDifferentialAnalyses

---

getDifferentialExpression

Get or set differential expression' elements for a data category

---

Description

Get or set differential expression' elements for a data category

Usage

```
getDifferentialExpression(category = getCategory())

setDifferentialExpression(differential, category = getCategory())

getDifferentialExpressionFiltered(category = getCategory())

setDifferentialExpressionFiltered(differential, category = getCategory())

getDifferentialExpressionResetPaging(category = getCategory())

setDifferentialExpressionResetPaging(reset, category = getCategory())

getDifferentialExpressionColumns(category = getCategory())

setDifferentialExpressionColumns(columns, category = getCategory())
```

Arguments

category	Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category
differential	Data frame or matrix: differential analyses table
reset	Character: reset paging of differential analyses table?
columns	Character: differential analyses' column names

Value

Getters return globally accessible data, whereas setters return NULL as they are only used to modify the Shiny session's state

See Also

getEvent, getClinicalMatchFrom, getGroups and getDifferentialAnalyses

---

getDownloadsFolder *Get the Downloads folder of the user*

Description

Get the Downloads folder of the user

Usage

```
getDownloadsFolder()
```

Value

Path to Downloads folder

Examples

```
getDownloadsFolder()
```

---

getFirebrowseCohorts *Query the Firebrowse web API for available cohorts*

Description

Query the Firebrowse web API for available cohorts

Usage

```
getFirebrowseCohorts(cohort = NULL)

getFirehoseCohorts(cohort = NULL)

getTCGAcohorts(cohort = NULL)
```

Arguments

cohort	Character: filter by given cohorts (optional)

Value

Character with cohort abbreviations (as values) and description (as names)

Examples

```
if (isFirebrowseUp()) getFirebrowseCohorts()
```

---

getFirebrowseDataTypes

Get data types available from Firebrowse

Description

Get data types available from Firebrowse

Usage

```
getFirebrowseDataTypes()

getFirehoseDataTypes()
```

Value

Named character vector

Examples

```
getFirebrowseDataTypes()
```

---

getFirebrowseDateFormat

Returns the date format used by the Firebrowse web API

Description

Returns the date format used by the Firebrowse web API

Usage

```
getFirebrowseDateFormat()
```

Value

Named list with Firebrowse web API's date formats

Examples

```
format <- psichomics:::getFirebrowseDateFormat()

# date format to use in a query to Firebrowse web API
format$query

# date format to parse a date in a response from Firebrowse web API
format$response
```

---

getFirebrowseDates *Query the Firebrowse web API for the available data datestamps*

Description

Query the Firebrowse web API for the available data datestamps

Usage

```
getFirebrowseDates()

getFirehoseDates()

getTCGAdates()
```

Value

Parsed date with datestamps of the data available

Examples

```
if (isFirebrowseUp()) getFirebrowseDates()
```

---

getGenes *Get or set globally accessible elements*

Description

Get or set globally accessible elements

Usage

```
getGenes()
```

Value

Getters return globally accessible data, whereas setters return NULL as they are only used to modify the Shiny session's state

See Also

getEvent, getClinicalMatchFrom, getGroups and getDifferentialAnalyses

`getGenesFromSplicingEvents` *Retrieve genes based on given alternative splicing events*

Description

Retrieve genes based on given alternative splicing events

Usage

```
getGenesFromSplicingEvents(ASevents)
```

Arguments

`ASevents`	Character: alternative splicing events

Value

Named character containing alternative splicing events and their respective genes as names

`getGlobal` *Get or set globally accessible elements*

Description

Get or set globally accessible elements

Usage

```
getGlobal(category = getCategory(), ..., sep = "_")

setGlobal(category = getCategory(), ..., value, sep = "_")

setData(data)

setDataTable(name, value, category = getCategory())

getAutoNavigation()

setAutoNavigation(auto)

getCores()

setCores(integer)

getSignificant()

setSignificant(integer)

getPrecision()

setPrecision(integer)
```

Arguments

`category`	Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category
`...`	Arguments to identify a variable
`sep`	Character to separate identifiers
`value`	Value to attribute to an element
`data`	List of data frame or matrix to set as data
`name`	Character: data table name
`auto`	Boolean: enable automatic navigation of browser history?
`integer`	Integer: value of the setting

Value

Getters return globally accessible data, whereas setters return NULL as they are only used to modify the Shiny session's state

Note

Needs to be called inside a reactive function

See Also

`getEvent`, `getClinicalMatchFrom`, `getGroups` and `getDifferentialAnalyses`

`getGroups` *Get or set groups*

Description

Get or set groups

Usage

```
getGroups(type = c("Patients", "Samples", "ASevents", "Genes"),
  complete = FALSE, category = getCategory())

setGroups(type = c("Patients", "Samples", "ASevents", "Genes"), groups,
  category = getCategory())
```

Arguments

`type`	Character: type of groups (either "Patients", "Samples", "ASevents" or "Genes")
`complete`	Boolean: return all the information on groups (TRUE) or just the group names and respective indexes (FALSE)? FALSE by default
`category`	Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category
`groups`	Matrix: groups of dataset

Value

Getters return globally accessible data, whereas setters return NULL as they are only used to modify the Shiny session's state

See Also

`getEvent`, `getClinicalMatchFrom`, `getGroups` and `getDifferentialAnalyses`

---

getGtexTissues *Get GTEx tissues from given GTEx sample attributes*

Description

Get GTEx tissues from given GTEx sample attributes

Usage

```
getGtexTissues(sampleMetadata)

getGTExTissues(sampleMetadata)
```

Arguments

`sampleMetadata`	Character: path to sample attributes

Value

Character: available tissues

---

getHidden *Get or set hidden globally accessible elements*

Description

Get or set hidden globally accessible elements

Usage

```
getHidden()

setHidden(val)
```

Arguments

`val`	Value to attribute

Value

Getters return hidden globally accessible data, whereas setters return NULL as they are only used to modify the state of hidden elements

`getHighlightedPoints` *Get or set points or regions for plots*

Description

Get or set points or regions for plots

Usage

```
getHighlightedPoints(id, category = getCategory())

setHighlightedPoints(id, events, category = getCategory())

getZoom(id, category = getCategory())

setZoom(id, zoom, category = getCategory())

getSelectedPoints(id, category = getCategory())

setSelectedPoints(id, events, category = getCategory())

getLabelledPoints(id, category = getCategory())

setLabelledPoints(id, events, category = getCategory())
```

Arguments

`id`	Character: identifier
`category`	Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category
`events`	Integer: index of events
`zoom`	Integer: range of X and Y coordinates for zooming

Value

Getters return globally accessible data, whereas setters return NULL as they are only used to modify the Shiny session's state

See Also

`getEvent`, `getClinicalMatchFrom`, `getGroups` and `getDifferentialAnalyses`

`getMatchingSamples` *Get samples matching the given patients*

Description

Get samples matching the given patients

Usage

```
getMatchingSamples(patients, samples, clinical = NULL, rm.NA = TRUE,
  match = NULL, showMatch = FALSE)

getSampleFromPatient(patients, samples, clinical = NULL, rm.NA = TRUE,
  match = NULL, showMatch = FALSE)

getSampleFromSubject(patients, samples, clinical = NULL, rm.NA = TRUE,
  match = NULL, showMatch = FALSE)
```

Arguments

`patients`	Character or list of characters: patient identifiers
`samples`	Character: sample identifiers
`clinical`	Data frame or matrix: clinical dataset
`rm.NA`	Boolean: remove NAs? TRUE by default
`match`	Integer: vector of patient index with the sample identifiers as name to save time (optional)
`showMatch`	Boolean: show matching patient index? FALSE by default

Value

Names of the matching samples (if `showMatch` is `TRUE`, a character with the patients as values and their respective samples as names is returned)

Examples

```
patients <- c("GTEX-ABC", "GTEX-DEF", "GTEX-GHI", "GTEX-JKL", "GTEX-MNO")
samples <- paste0(patients, "-sample")
clinical <- data.frame(samples=samples)
rownames(clinical) <- patients
getMatchingSamples(patients[c(1, 4)], samples, clinical)
```

getNumerics *Convert a column to numeric if possible and ignore given columns composed of lists*

Description

Convert a column to numeric if possible and ignore given columns composed of lists

Usage

```
getNumerics(table, by = NULL, toNumeric = FALSE)
```

Arguments

table	Data matrix: table
by	Character: column names of interest
toNumeric	Boolean: which columns to convert to numeric (FALSE by default)

Value

Processed data matrix

Examples

```
event <- read.table(text = "ABC123 + 250 300 350
                            DEF456 - 900 800 700")
names(event) <- c("Event ID", "Strand", "C1.end", "A1.end", "A1.start")

# Let's change one column to character
event[ , "C1.end"] <- as.character(event[ , "C1.end"])
is.character(event[ , "C1.end"])

event <- psichomics:::getNumerics(event, by = c("Strand", "C1.end", "A1.end",
                                  "A1.start"),
                                  toNumeric = c(FALSE, TRUE, TRUE, TRUE))
# Let's check if the same column is now integer
is.numeric(event[ , "C1.end"])
```

getPatientFromSample *Get patients from given samples*

Description

Get patients from given samples

Usage

```
getPatientFromSample(sampleId, patientId = NULL, na = FALSE,
  sampleInfo = NULL)

getSubjectFromSample(sampleId, patientId = NULL, na = FALSE,
  sampleInfo = NULL)
```

Arguments

sampleId	Character: sample identifiers
patientId	Character: patient identifiers to filter by (optional; if a matrix or data frame is given, its rownames will be used to infer the patient identifiers)
na	Boolean: return NA for samples with no matching patients
sampleInfo	Data frame or matrix: sample information containing the sample identifiers as rownames and a column named "Subject ID" with the respective subject identifiers

Value

Character: patient identifiers corresponding to the given samples

Examples

```
samples <- paste0("GTEX-", c("ABC", "DEF", "GHI", "JKL", "MNO"), "-sample")
getPatientFromSample(samples)

# Filter returned samples based on available patients
patients <- paste0("GTEX-", c("DEF", "MNO"))
getPatientFromSample(samples, patients)
```

---

getServerFunctions *Matches server functions from a given loader*

Description

Matches server functions from a given loader

Usage

```
getServerFunctions(loader, ..., priority = NULL)
```

Arguments

loader	Character: loader to run the functions
...	Extra arguments to pass to server functions
priority	Character: name of functions to prioritise by the given order; for instance, c("data", "analyses") would load "data", then "analyses" then remaining functions

Value

Invisible TRUE

getSplicingEventCoordinates

Returns the coordinates of interest for a given event type

Description

Returns the coordinates of interest for a given event type

Usage

```
getSplicingEventCoordinates(type, sorting = FALSE)
```

Arguments

type	Character: alternative splicing event type
sorting	Boolean: get coordinates used for sorting and comparison between different programs? FALSE by default

Value

Coordinates of interest according to the alternative splicing event type

getSplicingEventFromGenes

Retrieve alternative splicing events based on given genes

Description

Retrieve alternative splicing events based on given genes

Usage

```
getSplicingEventFromGenes(genes, ASevents)
```

Arguments

genes	Character: gene symbols (or TCGA-styled gene symbols)
ASevents	Character: alternative splicing events to be matched

Value

Character containing respective alternative splicing events

Examples

```
ASevents <- c("SE_1_+_201763003_201763300_201763374_201763594_NAV1",
              "SE_1_+_183515472_183516238_183516387_183518343_SMG7",
              "SE_1_+_183441784_183471388_183471526_183481972_SMG7",
              "SE_1_+_181019422_181022709_181022813_181024361_MR1",
              "SE_1_+_181695298_181700311_181700367_181701520_CACNA1E")
genes <- c("NAV1", "SMG7", "MR1", "HELLO")
getSplicingEventFromGenes(genes, ASevents)
```

getSplicingEventTypes *Splicing event types available*

Description

Splicing event types available

Usage

```
getSplicingEventTypes(acronymsAsNames = FALSE)
```

Arguments

acronymsAsNames	Boolean: return acronyms as names? FALSE by default

Value

Named character vector with splicing event types

Examples

```
getSplicingEventTypes()
```

getUiFunctions *Matches user interface (UI) functions from a given loader*

Description

Matches user interface (UI) functions from a given loader

Usage

```
getUiFunctions(ns, loader, ..., priority = NULL)
```

Arguments

ns	Shiny function to create IDs within a namespace
loader	Character: loader to run the functions
...	Extra arguments to pass to the user interface (UI) functions
priority	Character: name of functions to prioritise by the given order; for instance, c("data", "analyses") would load "data", then "analyses" then remaining functions

Value

List of functions related to the given loader

getValidEvents *Filters the events with valid elements according to the given validator*

Description

Filters the events with valid elements according to the given validator

Usage

```
getValidEvents(event, validator, areMultipleExonsValid = FALSE)
```

Arguments

event	Data.frame containing only one event with at least 7 columns as retrieved from the alternative splicing annotation files from MISO (GFF3 files)
validator	Character: valid elements for each event
areMultipleExonsValid	Boolean: consider runs of exons as valid when comparing with the validator? Default is FALSE (see details)

Details

areMultipleExonsValid allows to consider runs of exons (i.e. sequences where "exon" occurs consecutively) as valid when comparing with given validator. For example, if the validator is c("gene", "mRNA","exon") and areMultipleExonsValid = FALSE, this function will only considerate events as valid if they have the exact same elements. If areMultipleExonsValid = TRUE, a valid events could include the elements c("gene", "mRNA", "exon", "exon", "exon").

Value

Data.frame with valid events

Examples

```
event <- read.table(text = "
 chr1 SE gene 17233 18061  .  -  .
 chr1 SE dkfd 00000 30000  .  -  .
 chr1 SE mRNA 17233 18061  .  -  .
 chr1 SE exon 17233 17368  .  -  .
 chr1 SE exon 17526 17742  .  -  .
 chr1 SE exon 17915 18061  .  -  .
 chr1 SE mRNA 17233 18061  .  -  .
 chr1 SE exon 17233 17368  .  -  .
 chr1 SE exon 17915 18061  .  -  .
 chr1 SE gene 17233 18061  .  -  .
 chr1 SE mRNA 17233 18061  .  -  .
 chr1 SE exon 17233 17368  .  -  .
 chr1 SE exon 17606 17742  .  -  .
 chr1 SE exon 17915 18061  .  -  .
 chr1 SE mRNA 17233 18061  .  -  .
 chr1 SE exon 17233 17368  .  -  .
 chr1 SE exon 17915 18061  .  -  .
")
```

```
validator <- c("gene", "mRNA", rep("exon", 3), "mRNA", rep("exon", 2))
psichomics:::getValidEvents(event, validator)
```

`getValuePerPatient` *Assign average sample values to their corresponding patients*

Description

Assign average sample values to their corresponding patients

Usage

```
getValuePerPatient(data, match, clinical = NULL, patients = NULL,
  samples = NULL)

getValuePerSubject(data, match, clinical = NULL, patients = NULL,
  samples = NULL)

assignValuePerPatient(data, match, clinical = NULL, patients = NULL,
  samples = NULL)

assignValuePerSubject(data, match, clinical = NULL, patients = NULL,
  samples = NULL)

getPSIperPatient(psi, match, clinical = NULL, patients = NULL, ...)
```

Arguments

`data`	One-row data frame/matrix or vector: values per sample for a single gene
`match`	Matrix: match between samples and patients
`clinical`	Data frame or matrix: clinical dataset (only required if the `patients` argument is not handed)
`patients`	Character: patient identifiers (only required if the `clinical` argument is not handed)
`samples`	Character: samples to use when assigning values per patient (if NULL, all samples will be used)
`psi`	Data frame or matrix: values per sample
`...`	Deprecated arguments

Value

Values per patient

ggplotServer *Logic set to create an interactive ggplot*

Description

Logic set to create an interactive ggplot

Usage

```
ggplotServer(input, output, id, plot = NULL, df = NULL, x = NULL,
  y = NULL)

ggplotAuxServer(input, output, id)
```

Arguments

input	Shiny input
output	Shiny output
id	Character: identifier
plot	Character: plot expression (NULL renders no plot)
df	Data frame
x	Character: name of the variable used for the X axis
y	Character: name of the variable used for the Y axis

Value

NULL (this function is used to modify the Shiny session's state)

Note

Insert `ggplotAuxSet` outside any observer (so it is only run once)

ggplotTooltip *Create the interface for the tooltip of a plot*

Description

Create the interface for the tooltip of a plot

Usage

```
ggplotTooltip(df, hover, x, y)
```

Arguments

df	Data frame
hover	Mouse hover information for a given plot as retrieved from `hoverOpts`
x	Character: name of the variable used for the X axis
y	Character: name of the variable used for the Y axis

Value

HTML elements

---

ggplotUI *Interface for interactive ggplot*

Description

Interface for interactive ggplot

Usage

```
ggplotUI(id)
```

Arguments

id	Character: identifier

Value

HTML elements

---

globalSelectize *Create a selectize input available from any page*

Description

Create a selectize input available from any page

Usage

```
globalSelectize(id, placeholder)
```

Arguments

id	Character: input identifier
placeholder	Character: input placeholder

Value

HTML element for a global selectize input

groupByAttribute *User interface to group by attribute*

Description

User interface to group by attribute

Usage

```
groupByAttribute(ns, cols, id, example)
```

Arguments

ns	Namespace function
cols	Character or list: name of columns to show
id	Character: identifier
example	Character: text to show as an example

Value

HTML elements

groupByExpression *User interface to group by subset expression*

Description

User interface to group by subset expression

Usage

```
groupByExpression(ns, id)
```

Arguments

ns	Namespace function
id	Character: identifier

Value

HTML elements

groupByGrep *User interface to group by grep expression*

Description

User interface to group by grep expression

Usage

```
groupByGrep(ns, cols, id)
```

Arguments

ns	Namespace function
cols	Character or list: name of columns to show
id	Character: identifier

Value

HTML elements

groupById *User interface to group by row*

Description

User interface to group by row

Usage

```
groupById(ns, id)
```

Arguments

ns	Namespace function
id	Character: identifier

Value

HTML elements

groupManipulation *Logic server to manipulate data grouping*

Description

Logic server to manipulate data grouping

Usage

```
groupManipulation(input, output, session, type)
```

Arguments

input	Input object
output	Output object
session	Session object
type	Character: type of data for each the interface is intended

Value

HTML elements

groupManipulationInput *Interface to manipulate data grouping*

Description

Interface to manipulate data grouping

Usage

```
groupManipulationInput(id, type)
```

Arguments

id	Character: identifier
type	Character: type of data for each the interface is intended

Value

HTML elements

groupPerElem *Assign one group to each element*

Description

Assign one group to each element

Usage

```
groupPerElem(groups, elem = NULL, outerGroupName = NA)

groupPerPatient(groups, patients = NULL, includeOuterGroup = FALSE,
  outerGroupName = "(Outer data)")

groupPerSample(groups, samples, includeOuterGroup = FALSE,
  outerGroupName = "(Outer data)")
```

Arguments

groups	List of integers: groups of elements
elem	Character: all elements available (NULL by default)
outerGroupName	Character: name to give to outer group (NA by default; set to NULL to only show elements matched to their respective groups)
patients	Integer: total number of patients
includeOuterGroup	Boolean: join the patients that have no groups?
samples	Character: all available samples

Value

Character vector where each element corresponds to the group of the respective element

Examples

```
groups <- list(1:3, 4:7, 8:10)
names(groups) <- paste("Stage", 1:3)
groupPerElem(groups)
```

groupsServerOnce *Server function for data grouping (one call)*

Description

These functions only run once instead of running for every instance of groups

Usage

```
groupsServerOnce(input, output, session)
```

Arguments

input	Input object
output	Output object
session	Session object

Value

NULL (this function is used to modify the Shiny session's state)

---

hchart.survfit *Plot survival curves using Highcharts*

Description

Plot survival curves using Highcharts

Usage

```
## S3 method for class 'survfit'
hchart(object, ..., fun = NULL, markTimes = TRUE,
  symbol = "plus", markerColor = "black", ranges = FALSE,
  rangesOpacity = 0.3)
```

Arguments

object	A survfit object as returned from the survfit function
...	Arguments passed on to highcharter::hc_add_series
fun	Name of function or function used to transform the survival curve: log will put y axis on log scale, event plots cumulative events (f(y) = 1-y), cumhaz plots the cumulative hazard function (f(y) = -log(y)), and cloglog creates a complimentary log-log survival plot (f(y) = log(-log(y)) along with log scale for the x-axis.
markTimes	Label curves marked at each censoring time? TRUE by default
symbol	Symbol to use as marker (plus sign by default)
markerColor	Colour of the marker ("black" by default); use NULL to use the respective colour of each series
ranges	Plot interval ranges? FALSE by default
rangesOpacity	Opacity of the interval ranges (0.3 by default)

Value

highcharter object to plot survival curves

Examples

```
# Plot Kaplan-Meier curves
require("survival")
require("highcharter")
leukemia.surv <- survfit(Surv(time, status) ~ x, data = aml)
hchart(leukemia.surv)

# Plot the cumulative hazard function
lsurv2 <- survfit(Surv(time, status) ~ x, aml, type='fleming')
hchart(lsurv2, fun="cumhaz")

# Plot the fit of a Cox proportional hazards regression model
fit <- coxph(Surv(futime, fustat) ~ age, data = ovarian)
ovarian.surv <- survfit(fit, newdata=data.frame(age=60))
hchart(ovarian.surv, ranges = TRUE)
```

---

hc_scatter *Create scatter plot*

Description

Create a scatter plot using highcharter

Usage

```
hc_scatter(hc, x, y, z = NULL, label = NULL, showInLegend = FALSE, ...)
```

Arguments

hc	Highchart object
x	Numeric: X axis
y	Numeric: Y axis
z	Numeric: Z axis to set the bubble size (optional)
label	Character: data label for each point (optional)
showInLegend	Boolean: show the data in the legend box? FALSE by default
...	Arguments passed on to highcharter::hc_add_series

Value

highcharter object containing information for a scatter plot

inclusionLevelsInterface *Interface to quantify alternative splicing*

Description

Interface to quantify alternative splicing

Usage

```
inclusionLevelsInterface(ns)
```

Arguments

ns	Namespace function

Value

HTML elements

inlineDialog *Alert in the style of a dialogue box with a button*

Description

Alert in the style of a dialogue box with a button

Usage

```
inlineDialog(description, ..., buttonLabel = NULL, buttonIcon = NULL,
  buttonId = NULL, id = NULL, type = c("error", "warning"),
  bigger = FALSE)

errorDialog(description, ...)

warningDialog(description, ...)
```

Arguments

description	Character: description
...	Extra parameters when creating the alert
buttonLabel	Character: button label (NULL by default)
buttonIcon	Character: button icon (NULL by default)
buttonId	Character: button identifier (NULL by default)
id	Character: identifier (NULL by default)
type	Character: type of alert (error or warning)
bigger	Boolean: wrap the description in a h4 tag?

Value

HTML elements

insideFile *Get psichomics file inside a given directory*

Description

Get psichomics file inside a given directory

Usage

```
insideFile(...)
```

Arguments

...	character vectors, specifying subdirectory and file(s) within some package. The default, none, returns the root of the package. Wildcards are not supported.

Value

Loaded file

is.whole *Check if a number is whole*

Description

Check if a number is whole

Usage

```
is.whole(x, tol = .Machine$double.eps^0.5)
```

Arguments

x	Object to be tested
tol	Numeric: tolerance used for comparison

Value

TRUE if number is whole; otherwise, FALSE

isFirebrowseUp *Check whether the Firebrowse web API is running*

Description

The Firebrowse web API is running if it returns the status condition 200; if this is not the status code obtained from the API, the function will raise a warning with the status code and a brief explanation.

Usage

```
isFirebrowseUp()

isFirehoseUp()
```

Value

Invisible TRUE if the Firebrowse web API is working; otherwise, raises a warning

Examples

```
isFirebrowseUp()
```

joinEventsPerType *Full outer join all given events based on select columns*

Description

Full outer join all given events based on select columns

Usage

```
joinEventsPerType(events, types)
```

Arguments

events	Data frame or matrix: alternative splicing events
types	Character: alternative splicing types

Value

List of events joined by alternative splicing event type

junctionString *String used to search for matches in a junction quantification file*

Description

String used to search for matches in a junction quantification file

Usage

```
junctionString(chr, strand, junc5, junc3, showStrand)
```

Arguments

chr	Character: chromosome
strand	Character: strand
junc5	Integer: 5' end junction
junc3	Integer: 3' end junction
showStrand	Boolean: include strand?

Value

Formatted character string

kruskal *Perform Kruskal's test and return interface to show the results*

Description

Perform Kruskal's test and return interface to show the results

Usage

```
kruskal(data, groups, stat = NULL)
```

Arguments

data	Numeric, data frame or matrix: data for one gene or alternative splicing event
groups	List of characters (list of groups containing data identifiers) or character vector (group of each value in data)
stat	Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)

Value

HTML elements

labelBasedOnCutoff *Label groups based on a given cutoff*

Description

Label groups based on a given cutoff

Usage

```
labelBasedOnCutoff(data, cutoff, label = NULL, gte = TRUE)
```

Arguments

data	Numeric: test data
cutoff	Numeric: test cutoff
label	Character: label to prefix group names (NULL by default)
gte	Boolean: test with greater than or equal to cutoff (TRUE) or use less than or equal to cutoff (FALSE)? TRUE by default

Value

Labelled groups

Examples

```
labelBasedOnCutoff(data=c(1, 0, 0, 1, 0, 1), cutoff=0.5)

labelBasedOnCutoff(data=c(1, 0, 0, 1, 0, 1), cutoff=0.5, "Ratio")

# Use "greater than" instead of "greater than or equal to"
labelBasedOnCutoff(data=c(1, 0, 0, 0.5, 0, 1), cutoff=0.5, gte=FALSE)
```

levene *Perform Levene's test and return interface to show the results*

Description

Perform Levene's test and return interface to show the results

Usage

```
levene(data, groups, stat = NULL)
```

Arguments

data	Numeric, data frame or matrix: data for one gene or alternative splicing event
groups	List of characters (list of groups containing data identifiers) or character vector (group of each value in data)
stat	Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)

Value

HTML elements

`leveneTest` *Levene's test*

Description

Performs a Levene's test to assess the equality of variances

Usage

```
leveneTest(x, g, centers = median)
```

Arguments

`x`	a numeric vector of data values, or a list of numeric data vectors. Non-numeric elements of a list will be coerced, with a warning.
`g`	a vector or factor object giving the group for the corresponding elements of `x`. Ignored with a warning if `x` is a list.
`centers`	Function used to calculate how much values spread (`median` by default; another common function used is `mean`)

Details

The implementation of this function is based on `car:::leveneTest.default` with a more standard result.

Value

A list with class `"htest"` containing the following components:

`statistic`	the value of the test statistic with a name describing it.
`p.value`	the p-value for the test.
`method`	the type of test applied.
`data.name`	a character string giving the names of the data.

Examples

```
vals <- sample(30, replace=TRUE)
group <- lapply(list("A", "B", "C"), rep, 10)
group <- unlist(group)
psichomics:::leveneTest(vals, group)

## Using Levene's test based on the mean
psichomics:::leveneTest(vals, group, mean)
```

linkToArticle *Interface that directs users to original article*

Description

Interface that directs users to original article

Usage

```
linkToArticle()
```

Value

HTML elements

linkToRunJS *Link to run arbitrary JavaScript code*

Description

Link to run arbitrary JavaScript code

Usage

```
linkToRunJS(text, code)
```

Arguments

text	Character: text label
code	Character: JavaScript code

Value

HTML elements

listAllAnnotations *List alternative splicing annotation files available, as well as custom annotation*

Description

List alternative splicing annotation files available, as well as custom annotation

Usage

```
listAllAnnotations(...)
```

Arguments

...	Custom annotation loaded

Value

Named character vector with splicing annotation files available#'

Examples

```
psichomics:::listAllAnnotations()
```

listSplicingAnnotations *List the alternative splicing annotation files available*

Description

List the alternative splicing annotation files available

Usage

```
listSplicingAnnotations()
```

Value

Named character vector with splicing annotation files available

Examples

```
listSplicingAnnotations()
```

loadAnnotation *Load alternative splicing annotation from* AnnotationHub

Description

Load alternative splicing annotation from AnnotationHub

Usage

```
loadAnnotation(annotation)
```

Arguments

annotation	Character: annotation to load

Value

List of data frames containing the alternative splicing annotation per event type

Examples

```
human <- listSplicingAnnotations()[[1]]
## Not run:
annot <- loadAnnotation(human)

## End(Not run)
```

loadBy *Check if a given function should be loaded by the calling module*

Description

Check if a given function should be loaded by the calling module

Usage

```
loadBy(loader, FUN)
```

Arguments

loader	Character: name of the file responsible to load such function
FUN	Function

Value

Boolean vector

loadCustomSplicingAnnotationSet *Set of functions to load a custom alternative splicing annotation*

Description

Set of functions to load a custom alternative splicing annotation

Usage

```
loadCustomSplicingAnnotationSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

loadedDataModal *Create a modal warning the user of already loaded data*

Description

Create a modal warning the user of already loaded data

Usage

```
loadedDataModal(session, modalId, replaceButtonId, keepButtonId)
```

Arguments

session	Shiny session
modalId	Character: identifier of the modal
replaceButtonId	Character: identifier of the button to replace data
keepButtonId	Character: identifier of the button to append data

Value

HTML elements for a warning modal reminding data is loaded

loadFile *Loads a file according to its format*

Description

Loads a file according to its format

Usage

```
loadFile(format, file, ...)
```

Arguments

format	Environment: format of the file
file	Character: file to load
...	Extra parameters passed to fread

Details

The resulting data frame includes the attribute tablename with the name of the data frame

Value

Data frame with the loaded file

loadFileFormats *Loads file formats*

Description

Loads file formats

Usage

```
loadFileFormats()
```

Value

Loaded file formats available

`loadFirebrowseData` *Downloads and processes data from the Firebrowse web API and loads it into R*

Description

Downloads and processes data from the Firebrowse web API and loads it into R

Usage

```
loadFirebrowseData(folder = NULL, data = NULL, exclude = c(".aux.",
  ".mage-tab.", "MANIFEST.txt"), ..., download = TRUE)

loadFirehoseData(folder = NULL, data = NULL, exclude = c(".aux.",
  ".mage-tab.", "MANIFEST.txt"), ..., download = TRUE)

loadTCGAdata(folder = NULL, data = NULL, exclude = c(".aux.",
  ".mage-tab.", "MANIFEST.txt"), ..., download = TRUE)
```

Arguments

`folder`	Character: directory to store the downloaded archives (by default, it saves in the user's "Downloads" folder)
`data`	Character: data to load
`exclude`	Character: files and folders to exclude from downloading and from loading into R (by default, it excludes `.aux.`, `.mage-tab.` and `MANIFEST.TXT` files)
`...`	Arguments passed on to `queryFirebrowseData` **format** Character: response format as `JSON` (default), `CSV` or `TSV` **date** Character: dates of the data retrieval by Firebrowse (by default, it uses the most recent data available) **cohort** Character: abbreviation of the cohorts (by default, returns data for all cohorts) **data_type** Character: data types (optional) **tool** Character: data produced by the selected Firebrowse tools (optional) **platform** Character: data generation platforms (optional) **center** Character: data generation centres (optional) **level** Integer: data levels (optional) **protocol** Character: sample characterization protocols (optional) **page** Integer: page of the results to return (optional) **page_size** Integer: number of records per page of results; max is 2000 (optional) **sort_by** String: column used to sort the data (by default, it sorts by cohort)
`download`	Boolean: download missing files through the function `download.file` (TRUE by default)

Value

URL of missing files ("missing" class) if files need to be downloaded and if the argument `download` is `FALSE`; else, a list with loaded data

Examples

```
## Not run:
loadFirebrowseData(cohort = "ACC", data_type = "Clinical")

## End(Not run)
```

---

loadFirebrowseFolders *Load Firebrowse folders*

Description

Loads the files present in each folder as a data.frame.

Usage

```
loadFirebrowseFolders(folder, exclude = "")

loadFirehoseFolders(folder, exclude = "")

loadTCGAfolders(folder, exclude = "")
```

Arguments

folder	Character: folder(s) in which to look for Firebrowse files
exclude	Character: files to exclude from the loading

Value

List with loaded data.frames

Note

For faster execution, this function uses the `readr` library. This function ignores subfolders of the given folder (which means that files inside subfolders are NOT loaded).

---

loadGeneExpressionSet *Set of functions to load splicing quantification*

Description

Set of functions to load splicing quantification

Usage

```
loadGeneExpressionSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

`loadGtexData` *Load GTEx data*

Description

Load GTEx data

Usage

```
loadGtexData(clinical = NULL, sampleMetadata = NULL, junctionQuant = NULL,
  geneExpr = NULL, tissue = NULL)
```

Arguments

`clinical`	Character: path to subject information (the TXT file)
`sampleMetadata`	Character: path to sample metadata (the TXT file)
`junctionQuant`	Character: path to junction quantification
`geneExpr`	Character: path to gene read counts, RPKMs or TPMs
`tissue`	Character: tissue(s) of interest when loading data (all tissues are loaded by default); if only some tissue(s) are of interest, this may speed up loading the data

Value

List with loaded data

`loadGtexDataShiny` *Shiny wrapper to load GTEx data*

Description

Shiny wrapper to load GTEx data

Usage

```
loadGtexDataShiny(session, input, replace = TRUE)
```

Arguments

`session`	Shiny session
`input`	Shiny input
`replace`	Boolean: replace loaded data? TRUE by default

Value

NULL (this function is used to modify the Shiny session's state)

loadGtexFile *Load GTEx file*

Description

Load GTEx file

Usage

```
loadGtexFile(path, pattern, samples = NULL)

loadGTExFile(path, pattern, samples = NULL)
```

Arguments

path	Character: path to file
pattern	Character: pattern of the format type to load file
samples	Character: samples to filter datasets

Value

Loaded file as a data frame

loadLocalFiles *Load local files*

Description

Load local files

Usage

```
loadLocalFiles(folder, ignore = c(".aux.", ".mage-tab."), name = "Data")
```

Arguments

folder	Character: path to folder containing files of interest
ignore	Character: skip folders and filenames that match the expression
name	Character: name of the category containing all loaded datasets

Value

List of data frames from valid files

Examples

```
## Not run:
folder <- "~/Downloads/ACC 2016"
data <- loadLocalFiles(folder)

ignore <- c(".aux.", ".mage-tab.", "junction quantification")
loadLocalFiles(folder, ignore)

## End(Not run)
```

`loadRequiredData` *Missing information modal template*

Description

Missing information modal template

Usage

```
loadRequiredData(modal = NULL)

missingDataModal(session, dataType, buttonId)

missingDataGuide(dataType)
```

Arguments

`modal`	Character: modal identifier
`session`	Shiny session
`dataType`	Character: type of data missing
`buttonId`	Character: identifier of button to take user to load missing data

Value

NULL (this function is used to modify the Shiny session's state)

Examples

```
## Not run:
 session <- session$ns
 buttonInput <- "takeMeThere"
 buttonId <- ns(buttonInput)
 dataType <- "Inclusion levels"
 missingDataModal(session, buttonId, dataType)
 observeEvent(input[[buttonInput]], missingDataGuide(dataType))

## End(Not run)
```

loadSplicingQuantificationSet

Set of functions to load splicing quantification

Description

Set of functions to load splicing quantification

Usage

```
loadSplicingQuantificationSet(session, input, output)
```

Arguments

session	Session object
input	Input object
output	Output object

Value

NULL (this function is used to modify the Shiny session's state)

loadTCGAsampleMetadata

Prepare TCGA sample metadata from loaded datasets

Description

If no TCGA datasets apply, the input is returned

Usage

```
loadTCGAsampleMetadata(data)
```

Arguments

data	List of list of data frames

Value

List of list of data frames

matchSplicingEventsWithGenes *Match splicing events with respective genes*

Description

Match splicing events with respective genes

Usage

```
matchSplicingEventsWithGenes(ASevents)
```

Arguments

ASevents	Character: alternative splicing events to be matched

Value

Named character vector containing the splicing events and their respective gene as their name

modTabPanel *Modified* tabPanel *function to show icon and title*

Description

Modified tabPanel function to show icon and title

Usage

```
modTabPanel(title, ..., icon = NULL, menu = FALSE)
```

Arguments

title	Character: title of the tab
...	HTML elements to render
icon	Character: name of the icon
menu	Boolean: create a dropdown menu-like tab? FALSE by default

Value

HTML interface

Note

Icon is hidden at small viewports

navSelectize *Create a special selectize input in the navigation bar*

Description

Create a special selectize input in the navigation bar

Usage

```
navSelectize(id, label, placeholder = label)
```

Arguments

id	Character: input identifier
label	Character: input label
placeholder	Character: input placeholder

Value

HTML element to be included in a navigation bar

noinfo *Interface when no information could be retrieved*

Description

Interface when no information could be retrieved

Usage

```
noinfo(output, description = paste("No information available for this gene."),
  ...)
```

Arguments

output	Shiny output
description	Character: description of the message to show to the user
...	Arguments passed on to inlineDialog **id** Character: identifier (NULL by default) **buttonId** Character: button identifier (NULL by default) **buttonLabel** Character: button label (NULL by default) **buttonIcon** Character: button icon (NULL by default) **type** Character: type of alert (error or warning) **bigger** Boolean: wrap the description in a h4 tag?

Value

NULL (this function is used to modify the Shiny session's state)

normaliseGeneExpression *Filter and normalise gene expression*

Description

Filter and normalise gene expression

Usage

```
normaliseGeneExpression(geneExpr, geneFilter = NULL, method = "TMM",
  p = 0.75, log2transform = TRUE, priorCount = 0.25)
```

Arguments

geneExpr	Matrix or data frame: gene expression
geneFilter	Boolean: filtered genes
method	normalization method to be used
p	percentile (between 0 and 1) of the counts that is aligned when method="upperquartile"
log2transform	Boolean: perform log2-transformation?
priorCount	Average count to add to each observation to avoid zeroes after log-transformation

Value

Gene expression filtered and normalised

Examples

```
geneExpr <- readFile("ex_gene_expression.RDS")
normaliseGeneExpression(geneExpr)
```

operateOnGroups *Set operations on groups*

Description

This function can be used on groups to merge, intersect, subtract, etc.

Usage

```
operateOnGroups(input, session, operation, buttonId, symbol = " ", type,
  sharedData = sharedData)
```

Arguments

input	Shiny input
session	Shiny session
operation	Character: set operation
buttonId	Character: ID of the button to trigger operation
symbol	Character: Unicode symbol to visually indicate the operation performed (" " by default)
type	Character: type of group where set operations are to be performed
sharedData	Shiny app's global variable

Value

NULL (this function is used to modify the Shiny session's state)

---

optimalSurvivalCutoff *Calculate optimal data cutoff that best separates survival curves*

Description

Uses stats::optim with the Brent method to test multiple cutoffs and to find the minimum log-rank p-value.

Usage

```
optimalSurvivalCutoff(clinical, data, censoring, event, timeStart,
  timeStop = NULL, followup = "days_to_last_followup", session = NULL,
  filter = TRUE, survTime = NULL, lower = NULL, upper = NULL)

optimalPSIcutoff(clinical, psi, censoring, event, timeStart, timeStop = NULL,
  followup = "days_to_last_followup", session = NULL, filter = TRUE,
  survTime = NULL)
```

Arguments

clinical	Data frame: clinical data
data	Numeric: data values
censoring	Character: censor using "left", "right", "interval" or "interval2"
event	Character: name of column containing time of the event of interest
timeStart	Character: name of column containing starting time of the interval or follow up time
timeStop	Character: name of column containing ending time of the interval (only relevant for interval censoring)
followup	Character: name of column containing follow up time
session	Shiny session (only used for the visual interface)
filter	Boolean or numeric: elements to use (all by default)
survTime	survTime object: times to follow up, time start, time stop and event (optional)

lower, upper	Bounds in which to search (if NULL, they will be automatically set to 0 and 1 if all data values are within that interval; otherwise, they will be set to the minimum and maximum values of data)
psi	Numeric: PSI values to test against the cutoff

Value

List containg the optimal cutoff (par) and the corresponding p-value (value)

Examples

```
clinical <- read.table(text = "2549   NA ii  female
                                840   NA i   female
                                 NA 1204 iv    male
                                 NA  383 iv  female
                               1293   NA iii   male
                                 NA 1355 ii    male")
names(clinical) <- c("patient.days_to_last_followup",
                     "patient.days_to_death",
                     "patient.stage_event.pathologic_stage",
                     "patient.gender")
timeStart  <- "days_to_death"
event      <- "days_to_death"

psi <- c(0.1, 0.2, 0.9, 1, 0.2, 0.6)
opt <- optimalSurvivalCutoff(clinical, psi, "right", event, timeStart)
```

---

optimSurvDiffSet *Optimal survival difference given an inclusion level cutoff for a specific alternative splicing event*

Description

Optimal survival difference given an inclusion level cutoff for a specific alternative splicing event

Usage

```
optimSurvDiffSet(session, input, output)
```

Arguments

session	Shiny session
input	Shiny input
output	Shiny output

Value

NULL (this function is used to modify the Shiny session's state) Calculate optimal survival cutoff for the inclusion levels of a given alternative splicing event

parseCategoricalGroups *Parse categorical columns in a data frame*

Description

Retrieve elements grouped by their unique group based on each categorical column

Usage

```
parseCategoricalGroups(df)
```

Arguments

df	Data frame

Value

List of lists containing values based on rownames of df

See Also

testGroupIndependence and plotGroupIndependence

Examples

```
df <- data.frame("race"=c("caucasian", "caucasian", "asian"),
                 "gender"=c("male", "female", "male"))
rownames(df) <- paste("patient", 1:3)
parseCategoricalGroups(df)
```

parseDateResponse *Parse the date from a response*

Description

Parse the date from a response

Usage

```
parseDateResponse(string)
```

Arguments

string	Character: dates

Value

Parsed date

parseFirebrowseMetadata *Query the Firebrowse web API for metadata*

Description

Query the Firebrowse web API for metadata

Usage

```
parseFirebrowseMetadata(type, ...)

parseFirehoseMetadata(type, ...)

parseTCGAmetadata(type, ...)
```

Arguments

type	Character: metadata to retrieve
...	Character: parameters to pass to query (optional)

Value

List with parsed response

Examples

```
psichomics:::parseFirebrowseMetadata("Dates")
psichomics:::parseFirebrowseMetadata("Centers")
psichomics:::parseFirebrowseMetadata("HeartBeat")

# Get the abbreviation and description of all cohorts available
psichomics:::parseFirebrowseMetadata("Cohorts")
# Get the abbreviation and description of the selected cohorts
psichomics:::parseFirebrowseMetadata("Cohorts", cohort = c("ACC", "BRCA"))
```

parseMatsEvent *Parse alternative splicing events from MATS*

Description

Parse alternative splicing events from MATS

Usage

```
parseMatsEvent(event, event_type)
```

Arguments

event	Data frame row: MATS splicing event
event_type	Character: Type of event to parse (see details)

Details

The following event types can be parsed:

- **SE**: Skipped exon
- **MXE**: Mutually exclusive exons
- **RI**:Retained intron
- **A3SS**: Alternative 3' splice site
- **A5SS**: Alternative 5' splice site

Value

List containing the event attributes and junctions

Examples

```
# MATS event (alternative 3' splice site)
event <- read.table(text = "
    2 ENSG00000166012 TAF1D chr11 - 93466515 93466671 93466515 93466563 93467790 93467826
    5 ENSG00000166012 TAF1D chr11 - 93466515 93466671 93466515 93466585 93467790 93467826
    6 ENSG00000166012 TAF1D chr11 - 93466515 93466585 93466515 93466563 93467790 93467826
")
psichomics:::parseMatsEvent(event, "A3SS")
```

---

parseMatsGeneric *Parse junctions of an alternative splicing event from MATS according to event type*

Description

Parse junctions of an alternative splicing event from MATS according to event type

Usage

```
parseMatsGeneric(junctions, strand, coords, plus_pos, minus_pos)

parseMatsSE(junctions, strand)

parseMatsMXE(junctions, strand)

parseMatsRI(junctions, strand)

parseMatsA3SS(junctions, strand)

parseMatsA5SS(junctions, strand)

parseMatsAFE(junctions, strand)

parseMatsALE(junctions, strand)
```

Arguments

junctions	Integer: event's junctions
strand	Character: strand of the event
coords	Character: names of the alternative splicing coordinates
plus_pos	Integer: match of each junction in the respective coordinate for the plus strand
minus_pos	Integer: match of each junction in the respective coordinate for the minus strand

Details

The following event types are ready to be parsed:

- **SE** (skipped exon)
- **MXE** (mutually exclusive exon)
- **RI** (intron retention)
- **A5SS** (alternative 5' splice site)
- **A3SS** (alternative 3' splice site)
- **AFE** (alternative first exon)
- **ALE** (alternative last exon)

You can use `parseMatsGeneric` to parse other event types.

Value

Data frame with parsed junctions

See Also

`parseMatsEvent`

Examples

```
# Parse generic event (in this case, an exon skipping event)
junctions <- read.table(text=
    "79685787 79685910 79685796 79685910 79679566 79679751")
coords <- c("A1.start", "A1.end",
            "C1.start", "C1.end",
            "C2.start", "C2.end")
plus  <- c(1:6)
minus <- c(2:1, 6:3)
psichomics:::parseMatsGeneric(junctions, strand = "+", coords, plus, minus)

# Parse exon skipping event
junctions <- read.table(text=
    "79685787 79685910 79685796 79685910 79679566 79679751")
psichomics:::parseMatsSE(junctions, strand = "+")

# Parse mutually exclusive exon event
junctions <- read.table(text=
"158282161 158282276 158282689 158282804 158281047 158281295 158283950 158284199")
psichomics:::parseMatsMXE(junctions, strand = "+")

# Parse intron retention event
```

```
junctions <- read.table(text=
    "15929853 15932100 15929853 15930016 15930687 15932100")
psichomics:::parseMatsRI(junctions, strand = "+")

# Parse alternative 3' splicing site event
junctions <- read.table(text=
    "79685787 79685910 79685796 79685910 79679566 79679751")
psichomics:::parseMatsA3SS(junctions, strand = "+")

# Parse alternative 5' splicing site event
junctions <- read.table(text=
    "102884421 102884501 102884421 102884489 102884812 102885881")
psichomics:::parseMatsA5SS(junctions, strand = "+")

# Parse alternative first exon event
junctions <- read.table(text=
    "16308723 16308879 16308967 16309119 16314269 16314426")
psichomics:::parseMatsAFE(junctions, strand = "+")

# Parse alternative last exon event
junctions <- read.table(text=
    "111858645 111858828 111851063 111851921 111850441 111850543")
psichomics:::parseMatsAFE(junctions, strand = "+")
```

`parseMisoEvent` *Parse an alternative splicing event from MISO*

Description

Parse an alternative splicing event from MISO

Usage

```
parseMisoEvent(event)
```

Arguments

`event`	Data.frame containing only one event with at least 7 columns as retrieved from the alternative splicing annotation files from MISO (GFF3 files)

Details

More information about MISO available at http://miso.readthedocs.org

Value

List with event attributes and junction positions for the exons (depends on the events)

Examples

```
# example of alternative splicing event: skipped exon (SE)
event <- read.table(text = "
  chr1 SE gene 16854 18061 . - .
  chr1 SE mRNA 16854 18061 . - .
  chr1 SE exon 16854 17055 . - .
  chr1 SE exon 17233 17742 . - .
  chr1 SE exon 17915 18061 . - .
  chr1 SE mRNA 16854 18061 . - .
  chr1 SE exon 16854 17955 . - .
  chr1 SE exon 17915 18061 . - .")
psichomics:::parseMisoEvent(event)
```

`parseMisoEventID` *Match MISO's splicing event IDs with the IDs present in the alternative splicing annotation file and get events in a data frame*

Description

Match MISO's splicing event IDs with the IDs present in the alternative splicing annotation file and get events in a data frame

Usage

```
parseMisoEventID(eventID, annotation, IDcolumn)
```

Arguments

`eventID`	Character: alternative event IDs
`annotation`	Data.frame: alternative event annotation file
`IDcolumn`	Integer: index of the column with the event ID's in the alternative event annotation file

Details

For faster execution times, provide a vector of event IDs.

For more information about MISO, see http://miso.readthedocs.org.

Value

Data frame of the matching events (or NA when nothing is matched)

Note

If possible, it's recommend to use smaller subsets of the alternative events' annotation instead of all data for faster runs. For example, when trying to match only skipped exons event IDs, only use the annotation of skipped exons instead of using a mega annotation with all event types.

Examples

```
eventID <- c("114785@uc001sok.1@uc001soj.1", "114784@uc001bxm.1@uc001bxn.1")
# the annotation is one of the GFF3 files needed to run MISO
gff3 <- system.file("extdata", "miso_AS_annot_example.gff3",
                    package="psichomics")
annotation <- read.delim(gff3, header=FALSE, comment.char="#")
IDcolumn <- 9
psichomics:::parseMisoEventID(eventID, annotation, IDcolumn)
```

`parseMisoGeneric` *Parse junctions of an event from MISO according to event type*

Description

Parse junctions of an event from MISO according to event type

Usage

```
parseMisoGeneric(event, validator, eventType, coord, plusIndex, minusIndex)

parseMisoSE(event)

parseMisoMXE(event)

parseMisoRI(event, strand)

parseMisoA5SS(event)

parseMisoA3SS(event, plusIndex, minusIndex)

parseMisoTandemUTR(event, minusIndex)

parseMisoAFE(event)

parseMisoALE(event)
```

Arguments

`event`	Data.frame containing only one event with at least 7 columns as retrieved from the alternative splicing annotation files from MISO (GFF3 files)
`validator`	Character: valid elements for each event
`eventType`	Character: event type (see details for available events)
`coord`	Character: coordinate positions to fill
`plusIndex`	Integer: index of the coordinates for a plus strand event
`minusIndex`	Integer: index of the coordinates for a minus strand event
`strand`	Character: "+" or "-" strand

Details

The following event types are available to be parsed:

- **SE** (exon skipping)
- **MXE** (mutually exclusive exon)
- **RI** (intron retention)
- **A5SS** (alternative 5' splice site)
- **A3SS** (alternative 3' splice site)
- **AFE** (alternative first exon)
- **ALE** (alternative last exon)
- **Tandem UTR**

Value

List of parsed junctions

See Also

parseMisoEvent

Examples

```
# skipped exon event (SE)
event <- read.table(text = "
  chr1 SE gene 16854 18061 . - .
  chr1 SE mRNA 16854 18061 . - .
  chr1 SE exon 16854 17055 . - .
  chr1 SE exon 17233 17742 . - .
  chr1 SE exon 17915 18061 . - .
  chr1 SE mRNA 16854 18061 . - .
  chr1 SE exon 16854 17955 . - .
  chr1 SE exon 17915 18061 . - .")
psichomics:::parseMisoSE(event)

# mutually exclusive exon (MXE) event
event <- read.table(text = "
 chr1 MXE gene 764383 788090 . + .
 chr1 MXE mRNA 764383 788090 . + .
 chr1 MXE exon 764383 764484 . + .
 chr1 MXE exon 776580 776753 . + .
 chr1 MXE exon 787307 788090 . + .
 chr1 MXE mRNA 764383 788090 . + .
 chr1 MXE exon 764383 764484 . + .
 chr1 MXE exon 783034 783186 . + .
 chr1 MXE exon 787307 788090 . + .")
psichomics:::parseMisoMXE(event)

# intron retention (RI) event
event <- read.table(text = "
 chr1 RI gene 17233 17742 . - .
 chr1 RI mRNA 17233 17742 . - .
 chr1 RI exon 17233 17742 . - .
 chr1 RI mRNA 17233 17742 . - .
```

```
 chr1 RI exon 17233 17364 . - .
 chr1 RI exon 17601 17742 . - .")
psichomics:::parseMisoRI(event)

# alternative 5' splice site (A5SS) event
event <- read.table(text = "
 chr1 A5SS gene 17233 17742 . - .
 chr1 A5SS mRNA 17233 17742 . - .
 chr1 A5SS exon 17233 17368 . - .
 chr1 A5SS exon 17526 17742 . - .
 chr1 A5SS mRNA 17233 17742 . - .
 chr1 A5SS exon 17233 17368 . - .
 chr1 A5SS exon 17606 17742 . - .")
psichomics:::parseMisoA5SS(event)

# alternative 3' splice site (A3SS) event
event <- read.table(text = "
 chr1 A3SS gene 15796 16765 . - .
 chr1 A3SS mRNA 15796 16765 . - .
 chr1 A3SS exon 15796 15947 . - .
 chr1 A3SS exon 16607 16765 . - .
 chr1 A3SS mRNA 15796 16765 . - .
 chr1 A3SS exon 15796 15942 . - .
 chr1 A3SS exon 16607 16765 . - .")
psichomics:::parseMisoA3SS(event)

# Tandem UTR event
event <- read.table(text = "
 chr19 TandemUTR gene  10663759  10664625  .  -  .
 chr19 TandemUTR mRNA  10663759  10664625  .  -  .
 chr19 TandemUTR exon  10663759  10664625  .  -  .
 chr19 TandemUTR mRNA  10664223  10664625  .  -  .
 chr19 TandemUTR exon  10664223  10664625  .  -  .")
psichomics:::parseMisoTandemUTR(event)

# alternative first exon (AFE) event
event <- read.table(text = "
 chr12 AFE gene 57916659 57920171  .  +  .
 chr12 AFE mRNA 57919131 57920171  .  +  .
 chr12 AFE exon 57919131 57920171  .  +  .
 chr12 AFE mRNA 57916659 57918199  .  +  .
 chr12 AFE exon 57916659 57916794  .  +  .
 chr12 AFE exon 57917812 57917875  .  +  .
 chr12 AFE exon 57918063 57918199  .  +  .")
psichomics:::parseMisoAFE(event)

# alternative last exon (ALE) event
event <- read.table(text = "
 chr6 ALE gene 30620579 30822593  .  +  .
 chr6 ALE mRNA 30822190 30822593  .  +  .
 chr6 ALE exon 30822190 30822593  .  +  .
 chr6 ALE mRNA 30620579 30620982  .  +  .
 chr6 ALE exon 30620579 30620982  .  +  .")
psichomics:::parseMisoALE(event)
```

parseMisoId *Parse MISO's alternative splicing event identifier*

Description

Parse MISO's alternative splicing event identifier

Usage

```
parseMisoId(id)
```

Arguments

id	Character: MISO alternative splicing event identifier

Value

Character with the parsed ID

Examples

```
id <- paste0(
    "ID=ENSMUSG00000026150.chr1:82723803:82723911:+@chr1:82724642:82724813:",
    "+@chr1:82725791:82726011:+.B;Parent=ENSMUSG00000026150.chr1:82723803:",
    "82723911:+@chr1:82724642:82724813:+@chr1:82725791:82726011:+")
psichomics:::parseMisoId(id)
```

parseSampleGroups *Return the type of a given sample*

Description

Return the type of a given sample

Usage

```
parseSampleGroups(sample, filename = system.file("extdata",
  "TCGAsampleType.RDS", package = "psichomics"))
```

Arguments

sample	Character: ID of the sample
filename	Character: path to RDS file containing corresponding type

Value

Types of the TCGA samples

Examples

```
parseSampleGroups(c("TCGA-01A-Tumour", "TCGA-10B-Normal"))
```

parseSplicingEvent *Parse an alternative splicing event based on a given identifier*

Description

Parse an alternative splicing event based on a given identifier

Usage

```
parseSplicingEvent(event, char = FALSE, pretty = FALSE, extra = NULL)
```

Arguments

event	Character: event identifier
char	Boolean: return a single character instead of list with parsed values? FALSE by default
pretty	Boolean: return a prettier name of the event identifier? FALSE by default
extra	Character: extra information to add (such as species and assembly version)

Value

Parsed event

Examples

```
events <- c("SE_1_-_123_456_789_1024_TST",
            "MXE_3_+_473_578_686_736_834_937_HEY/YOU")
parseSplicingEvent(events)
```

parseSuppaAnnotation *Get events from alternative splicing annotation*

Description

Get events from alternative splicing annotation

Usage

```
parseSuppaAnnotation(folder, types = c("SE", "AF", "AL", "MX", "A5", "A3",
  "RI"), genome = "hg19")

parseVastToolsAnnotation(folder, types = c("ALT3", "ALT5", "COMBI", "IR",
  "MERGE3m", "MIC", "EXSK", "MULTI"), genome = "Hsa", complexEvents = FALSE)

parseMisoAnnotation(folder, types = c("SE", "AFE", "ALE", "MXE", "A5SS",
  "A3SS", "RI", "TandemUTR"), genome = "hg19")

parseMatsAnnotation(folder, types = c("SE", "AFE", "ALE", "MXE", "A5SS",
  "A3SS", "RI"), genome = "fromGTF", novelEvents = TRUE)
```

Arguments

folder	Character: path to folder
types	Character: type of events to retrieve (depends on the program of origin; see details)
genome	Character: genome of interest (for instance, hg19; depends on the program of origin)
complexEvents	Boolean: should complex events in A3SS and A5SS be parsed? FALSE by default
novelEvents	Boolean: parse events dedected due to novel splice sites (TRUE by default)

Details

Type of parsable events:

- Alternative 3' splice site
- Alternative 5' splice site
- Alternative first exon
- Alternative last exon
- Skipped exon (may include skipped micro-exons)
- Mutually exclusive exon
- Retained intron
- Tandem UTR

Value

Retrieve data frame with events based on a given alternative splicing annotation

Examples

```
# Load sample files
folder <- "extdata/eventsAnnotSample/suppa_output/suppaEvents"
suppaOutput <- system.file(folder, package="psichomics")

suppa <- parseSuppaAnnotation(suppaOutput)
# Load sample files
folder <- "extdata/eventsAnnotSample/VASTDB/Hsa/TEMPLATES"
vastToolsOutput <- system.file(folder, package="psichomics")

vast <- parseVastToolsAnnotation(vastToolsOutput)
# Load sample files
folder <- "extdata/eventsAnnotSample/miso_annotation"
misoOutput <- system.file(folder, package="psichomics")

miso <- parseMisoAnnotation(misoOutput)
# Load sample files
folder <- "extdata/eventsAnnotSample/mats_output/ASEvents"
matsOutput <- system.file(folder, package="psichomics")

mats <- parseMatsAnnotation(matsOutput)

# Do not parse novel events
mats <- parseMatsAnnotation(matsOutput, novelEvents=FALSE)
```

`parseSuppaEvent` *Parses splicing events of a specific event type from SUPPA*

Description

Parses splicing events of a specific event type from SUPPA

Usage

```
parseSuppaEvent(event)
```

Arguments

`event`	Character vector: Splicing event attributes and junction positions

Details

More information about SUPPA available at https://bitbucket.org/regulatorygenomicsupf/suppa

The following event types are available to be parsed:

- **SE** (skipped exon)
- **RI** (intron retention)
- **MX** (mutually exclusive exons)
- **A5** (alternative 5' splice site)
- **A3** (alternative 3' splice site)
- **AL** (alternative last exon)
- **AF** (alternative first exon)

Value

List with the event attributes (chromosome, strand, event type and the position of the exon boundaries)

Note

It only allows to parse one event type at once.

Examples

```
event <- "ENSG00000000419;A3:20:49557492-49557642:49557470-49557642:-"
psichomics:::parseSuppaEvent(event)
```

`parseSuppaGeneric` *Parse junctions of an event from SUPPA*

Description

Parse junctions of an event from SUPPA

Usage

```
parseSuppaGeneric(junctions, strand, coords, plus_pos, minus_pos)

parseSuppaSE(junctions, strand)

parseSuppaRI(junctions, strand)

parseSuppaALE(junctions, strand)

parseSuppaAFE(junctions, strand)

parseSuppaMXE(junctions, strand)

parseSuppaA3SS(junctions, strand)

parseSuppaA5SS(junctions, strand)
```

Arguments

`junctions`	List of integers: exon-exon junctions of an event
`strand`	Character: positive ("+") or negative ("-") strand
`coords`	Character: coordinate positions to fill
`plus_pos`	Integer: index of the coordinates for a plus strand event
`minus_pos`	Integer: index of the coordinates for a minus strand event

Details

The following event types are available to be parsed:

- **SE** (exon skipping)
- **RI** (intron retention)
- **MXE** (mutually exclusive exons)
- **A5SS** (alternative 5' splice site)
- **A3SS** (alternative 3' splice site)
- **ALE** (alternative last exon)
- **AFE** (alternative first exon)

Value

Data frame of parsed junctions

See Also

parseSuppaEvent

Examples

```
# Parse generic event (in this case, an exon skipping event)
junctions <- read.table(text = "169768099 169770024 169770112 169771762")
coords <- c("C1.end", "A1.start", "A1.end", "C2.start")
plus  <- 1:4
minus <- 1:4
psichomics:::parseSuppaGeneric(junctions, strand = "+", coords, plus, minus)

junctions <- read.table(text = "169768099 169770024 169770112 169771762")
psichomics:::parseSuppaSE(junctions, "+")

junctions <- read.table(text = "196709749 196709922 196711005 196711181")
psichomics:::parseSuppaRI(junctions, "+")

junctions <- read.table(
    text = "24790610 24792494 24792800 24790610 24795476 24795797")
psichomics:::parseSuppaALE(junctions, "+")

junctions <- read.table(
    text = "169763871 169764046 169767998 169764550 169765124 169767998")
psichomics:::parseSuppaAFE(junctions, "+")

junctions <- read.table(
   text = "202060671 202068453 202068489 202073793 202060671 202072798 202072906 202073793")
psichomics:::parseSuppaMXE(junctions, "+")

junctions <- read.table(text = "169772450 169773216 169772450 169773253")
psichomics:::parseSuppaA3SS(junctions, "+")

junctions <- read.table(text = "50193276 50197008 50192997 50197008")
psichomics:::parseSuppaA5SS(junctions, "+")
```

---

parseTcgaSampleInfo *Parse sample information from TCGA samples*

Description

Parse sample information from TCGA samples

Usage

```
parseTcgaSampleInfo(samples, match = NULL)

parseTCGAsampleInfo(samples, match = NULL)
```

Arguments

samples	Character: sample identifiers
match	Integer: match between samples and patients (NULL by default; performs the match)

Value

Data frame containing metadata associated with each TCGA sample

Examples

```
samples <- c("TCGA-3C-AAAU-01A-11R-A41B-07", "TCGA-3C-AALI-01A-11R-A41B-07",
             "TCGA-3C-AALJ-01A-31R-A41B-07", "TCGA-3C-AALK-01A-11R-A41B-07",
             "TCGA-4H-AAAK-01A-12R-A41B-07", "TCGA-5L-AAT0-01A-12R-A41B-07")

parseTcgaSampleInfo(samples)
```

---

parseUniprotXML *Parse XML from UniProt's RESTful service*

Description

Parse XML from UniProt's RESTful service

Usage

```
parseUniprotXML(xml)
```

Arguments

xml	response from UniProt

Value

List containing protein length and data frame of protein features

---

parseUrlsFromFirebrowseResponse *Retrieve URLs from a response to a Firebrowse data query*

Description

Retrieve URLs from a response to a Firebrowse data query

Usage

```
parseUrlsFromFirebrowseResponse(res)

parseUrlsFromFirehoseResponse(res)
```

Arguments

res	Response from `httr::GET` to a Firebrowse data query

Value

Named character with URLs

Examples

```
res <- psichomics:::queryFirebrowseData(cohort = "ACC")
url <- psichomics:::parseUrlsFromFirebrowseResponse(res)
```

---

parseValidFile *Parse file given a list of file formats*

Description

Tries to recognise the file format and parses the content of the given file accordingly.

Usage

```
parseValidFile(file, formats, ...)
```

Arguments

file	Character: file to parse
formats	List of file formats to check
...	Extra parameters passed to fread

Details

The resulting data frame includes the attribute tablename with the name of the data frame

Value

Data frame with the contents of the given file if the file format is recognised; otherwise, returns NULL

---

parseVastToolsEvent *Parses an alternative splicing event from VAST-TOOLS*

Description

Parses an alternative splicing event from VAST-TOOLS

Usage

```
parseVastToolsEvent(event)
```

Arguments

event	Data.frame: VAST-TOOLS event containing gene symbol, event ID, length, junctions coordinates, event type and inclusion levels for both samples

Details

Junctions are parsed from

Value

List with the event attributes (chromosome, strand, event type and the position of the exon boundaries)

Note

Only supports to parse one event at a time.

Examples

```
event <- read.table(text =
"NFYA HsaEX0042823 chr6:41046768-41046903 136 chr6:41040823,41046768-41046903,41051785 C2 0 N 0 N"
)
psichomics:::parseVastToolsEvent(event)
```

---

parseVastToolsSE *Parse junctions of an event from VAST-TOOLS according to event type*

Description

Parse junctions of an event from VAST-TOOLS according to event type

Usage

```
parseVastToolsSE(junctions)

parseVastToolsRI(junctions, strand)

parseVastToolsA3SS(junctions)

parseVastToolsA5SS(junctions)
```

Arguments

junctions	Data.frame or matrix: exon-exon junctions of alternative splicing events (it must have 4 columns)
strand	Character: positive (+) or negative (-) strand

Details

The following event types are available to be parsed:

- **SE** (skipped exon)
- **RI** (intron retention)
- **A5SS** (alternative 5' splice site)
- **A3SS** (alternative 3' splice site)

Value

List of parsed junctions

See Also

parseVastToolsEvent

Examples

```
junctions <- read.table(text = "41040823 41046768 41046903 41051785")
psichomics:::parseVastToolsSE(junctions)

# these functions are vectorised!
junctions <- read.table(text = "41040823 41046768 41046903 41051785
                                58864658 58864693 58864294 58864563")
psichomics:::parseVastToolsSE(junctions)

junctions <- read.table(text = "58864658 58864693 58864294 58864563")
psichomics:::parseVastToolsRI(junctions, strand = "+")

junctions <- rbind(
    c(36276385, list(c(36277798, 36277315)), 36277974),
    c(7133604, 7133377, list(c(7133474, 7133456)))
)
psichomics:::parseVastToolsA3SS(junctions)

junctions <- rbind(
    c(74650610, list(c(74650654, 74650658)), 74650982),
    c(list(c(49557666, 49557642), 49557746, 49557470))
)
psichomics:::parseVastToolsA5SS(junctions)
```

---

patientMultiMatchWarning

Helper text to explain what happens when a patient matches multiple samples when performing survival analysis

---

Description

Helper text to explain what happens when a patient matches multiple samples when performing survival analysis

Usage

```
patientMultiMatchWarning()
```

Value

Character

`performICA` *Perform independent component analysis after processing missing values*

Description

Perform independent component analysis after processing missing values

Usage

```
performICA(data, n.comp = min(5, ncol(data)), center = TRUE,
  scale. = FALSE, missingValues = round(0.05 * nrow(data)),
  alg.typ = c("parallel", "defaltion"), fun = c("logcosh", "exp"),
  alpha = 1, ...)
```

Arguments

`data`	an optional data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`n.comp`	number of components to be extracted
`center`	a logical value indicating whether the variables should be shifted to be zero centered. Alternately, a vector of length equal the number of columns of `x` can be supplied. The value is passed to `scale`.
`scale.`	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is `FALSE` for consistency with S, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of `x` can be supplied. The value is passed to `scale`.
`missingValues`	Integer: number of tolerated missing values per column to be replaced with the mean of the values of that same column (5 rows by default)
`alg.typ`	if `alg.typ == "parallel"` the components are extracted simultaneously (the default). if `alg.typ == "deflation"` the components are extracted one at a time.
`fun`	the functional form of the G function used in the approximation to neg-entropy (see 'details').
`alpha`	constant in range [1, 2] used in approximation to neg-entropy when `fun == "logcosh"`
`...`	Arguments passed on to `fastICA::fastICA`

Value

ICA result in a `prcomp` object

See Also

`plotICA`, `performPCA` and `plotPCA`

Examples

```
performICA(USArrests)
```

performPCA *Perform principal component analysis after processing missing values*

Description

Perform principal component analysis after processing missing values

Usage

```
performPCA(data, center = TRUE, scale. = FALSE, missingValues = round(0.05
  * nrow(data)), ...)
```

Arguments

data	an optional data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
center	a logical value indicating whether the variables should be shifted to be zero centered. Alternately, a vector of length equal the number of columns of x can be supplied. The value is passed to scale.
scale.	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is FALSE for consistency with S, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of x can be supplied. The value is passed to scale.
missingValues	Integer: number of tolerated missing values per column to be replaced with the mean of the values of that same column (5 rows by default)
...	Arguments passed on to stats::prcomp

Value

PCA result in a prcomp object

See Also

plotPCA, performICA and plotICA

Examples

```
performPCA(USArrests)
```

plotClusters *Add clusters to* highchart *object*

Description

Clusters are added as coloured polygons.

Usage

```
plotClusters(hc, data, clustering)
```

Arguments

hc	highchart object
data	Data frame
clustering	Character: group of each sample

Value

highcharter object

plotCorrelation *Plot correlations*

Description

Plot correlation results from correlateGEandAS

Usage

```
plotCorrelation(corr, autoZoom = FALSE, loessSmooth = TRUE,
  loessFamily = c("gaussian", "symmetric"), colour = "black", alpha = 0.2,
  size = 1.5, loessColour = "red", loessAlpha = 1, loessWidth = 0.5,
  ...)
```

Arguments

corr	List of correlations
autoZoom	Boolean: automatically set the range of PSI values based on available data? If FALSE, the axis relative to PSI values will range from 0 to 1
loessSmooth	Boolean: plot a smooth curve computed by stats::loess.smooth?
loessFamily	Character: if gaussian, loess fitting is by least-squares, and if symmetric, a re-descending M estimator is used
colour	Character: points' colour
alpha	Numeric: points' alpha
size	Numeric: points' size
loessColour	Character: loess line's colour

loessAlpha	Numeric: loess line's opacity
loessWidth	Numeric: loess line's width
...	Arguments passed on to stats::loess.smooth **span** smoothness parameter for loess. **degree** degree of local polynomial used. **evaluation** number of points at which to evaluate the smooth curve.

Value

Renders plots for each correlation in corr

Examples

```
annot <- readFile("ex_splicing_annotation.RDS")
junctionQuant <- readFile("ex_junctionQuant.RDS")
psi <- quantifySplicing(annot, junctionQuant, eventType=c("SE", "MXE"))

geneExpr <- readFile("ex_gene_expression.RDS")
corr <- correlateGEandAS(geneExpr, psi, "ALDOA")
plotCorrelation(corr)
```

---

plotDistribution *Plot distribution through a density plot*

Description

The tooltip shows the median, variance, max, min and number of non-NA samples of each data series.

Usage

```
plotDistribution(data, groups = "All samples", rug = TRUE, vLine = TRUE,
  ..., title = NULL, psi = TRUE)

plotDensity(data, groups = "All samples", rug = TRUE, vLine = TRUE, ...,
  title = NULL, psi = TRUE)
```

Arguments

data	Numeric, data frame or matrix: data for one gene or alternative splicing event
groups	List of characters (list of groups containing data identifiers) or character vector (group of each value in data)
rug	Boolean: include rug plot to better visualise data distribution
vLine	Boolean: include vertical plot lines to indicate the mean and median of each group even when those groups are omitted
...	Extra parameters passed to density to create the kernel density estimates
title	Character: plot title
psi	Boolean: are data composed of PSI values?

Value

Highcharter object with density plot

Examples

```
data <- sample(20, rep=TRUE)/20
groups <- c(rep("A", 10), rep("B", 10))
plotDistribution(data, groups)
```

`plotGroupIndependence` *Plot -log10(p-values) of the results obtained after multiple group independence testing*

Description

Plot -log10(p-values) of the results obtained after multiple group independence testing

Usage

```
plotGroupIndependence(groups, top = 50, textSize = 10,
  colourLow = "lightgrey", colourMid = "blue", colourHigh = "orange",
  colourMidpoint = 150)
```

Arguments

`groups`	`multiGroupIndependenceTest` object (obtained after running `testGroupIndependence`)
`top`	Integer: number of attributes to render
`textSize`	Integer: size of the text
`colourLow`	Character: name or HEX code of colour for lower values
`colourMid`	Character: name or HEX code of colour for middle values
`colourHigh`	Character: name or HEX code of colour for higher values
`colourMidpoint`	Numeric: midpoint to identify middle values

Value

`ggplot` object

See Also

`parseCategoricalGroups` and `testGroupIndependence`

Examples

```
elements <- paste("patients", 1:50)
ref      <- elements[10:50]
groups   <- list(race=list(asian=elements[1:3],
                           white=elements[4:7],
                           black=elements[8:10]),
                 region=list(european=elements[c(4, 5, 9)],
                             african=elements[c(6:8, 10:50)]))
groupTesting <- testGroupIndependence(ref, groups, elements)
plotGroupIndependence(groupTesting)
```

plotICA *Create multiple scatterplots from ICA*

Description

Create multiple scatterplots from ICA

Usage

```
plotICA(ica, components = seq(10), groups = NULL, ...)
```

Arguments

ica	Object resulting from performICA
components	Numeric: independent components to plot
groups	Matrix: groups to plot indicating the index of interest of the samples (use clinical or sample groups)
...	Arguments passed on to pairsD3::pairsD3

group a optional vector specifying the group each observation belongs to. Used for tooltips and colouring the observations.

subset an optional vector specifying a subset of observations to be used for plotting. Useful when you have a large number of observations, you can specify a random subset.

labels the names of the variables (column names of x used by default).

cex the magnification of the plotting symbol (default=3)

width the width (and height) of the plot when viewed externally.

col an optional (hex) colour for each of the levels in the group vector.

big a logical parameter. Prevents inadvertent plotting of huge data sets. Default limit is 10 variables, to plot more than 10 set big=TRUE.

theme a character parameter specifying whether the theme should be colour colour (default) or black and white bw.

opacity numeric between 0 and 1. The opacity of the plotting symbols (default 0.9).

tooltip an optional vector with the tool tip to be displayed when hovering over an observation. You can include basic html.

leftmar space on the left margin

topmar space on the bottom margin

Value

Multiple scatterplots as a pairsD3 object

Examples

```
data <- scale(USArrests)
ica  <- fastICA::fastICA(data, n.comp=4)
plotICA(ica)

# Colour by groups
```

```
groups <- NULL
groups$sunny <- c("California", "Hawaii", "Florida")
groups$ozEntrance <- c("Kansas")
groups$novel <- c("New Mexico", "New York", "New Hampshire", "New Jersey")
plotICA(ica, groups=groups)
```

---

plotPCA *Create a scatterplot from a PCA object*

Description

Create a scatterplot from a PCA object

Usage

```
plotPCA(pca, pcX = 1, pcY = 2, groups = NULL, individuals = TRUE,
  loadings = FALSE, nLoadings = NULL)
```

Arguments

pca	prcomp object
pcX	Character: name of the X axis of interest from the PCA
pcY	Character: name of the Y axis of interest from the PCA
groups	Matrix: groups to plot indicating the index of interest of the samples (use clinical or sample groups)
individuals	Boolean: plot PCA individuals (TRUE by default)
loadings	Boolean: plot PCA loadings/rotations (FALSE by default)
nLoadings	Integer: Number of variables to plot, ordered by those that most contribute to selected principal components (this allows for faster performance as only the variables that most contribute are rendered); if NULL, all variables are plotted

Value

Scatterplot as an highcharter object

Examples

```
pca <- prcomp(USArrests, scale=TRUE)
plotPCA(pca)
plotPCA(pca, pcX=2, pcY=3)

# Plot both individuals and loadings
plotPCA(pca, pcX=2, pcY=3, loadings=TRUE)
```

plotPointsStyle *Interface to modify the style of the plot points*

Description

Interface to modify the style of the plot points

Usage

```
plotPointsStyle(ns, id, description, help = NULL, size = 2,
  colour = "black", alpha = 1)
```

Arguments

ns	Namespace function
id	Character: identifier
description	Character: display text for user
help	Character: extra text to help the user
size	Integer: default size (2 by default)
colour	Character: default colour ("black" by default)
alpha	Numeric: default transparency value; (opaque by default)

Value

HTML elements

plotProtein *Plot protein features*

Description

Plot protein features

Usage

```
plotProtein(molecule)
```

Arguments

molecule	Character: UniProt protein or Ensembl transcript identifier

Value

highcharter object

Examples

```
## Not run:
protein <- "P38398"
plotProtein(protein)

transcript <- "ENST00000488540"
plotProtein(transcript)

## End(Not run)
```

`plotSingleICA` *Create a scatterplot for ICA*

Description

Create a scatterplot for ICA

Usage

```
plotSingleICA(ica, icX = 1, icY = 2, groups = NULL)
```

Arguments

`ica`	Object containing an ICA
`icX`	Character: name of the X axis
`icY`	Character: name of the Y axis
`groups`	Matrix: groups to plot indicating the index of interest of the samples (use clinical or sample groups)

Value

Scatterplot as an `highcharter` object

Examples

```
ica <- performICA(USArrests, scale=TRUE)
psichomics:::plotSingleICA(ica)
psichomics:::plotSingleICA(ica, icX=2, icY=3)

# Colour by groups
groups <- NULL
groups$sunny <- c("California", "Hawaii", "Florida")
groups$ozEntrance <- c("Kansas")
groups$novel <- c("New Mexico", "New York", "New Hampshire", "New Jersey")
psichomics:::plotSingleICA(ica, groups=groups)
```

plotSurvivalCurves *Plot survival curves*

Description

Plot survival curves

Usage

```
plotSurvivalCurves(surv, mark = TRUE, interval = FALSE, pvalue = NULL,
  title = "Survival analysis", scale = NULL, auto = TRUE)
```

Arguments

surv	Survival object
mark	Boolean: mark times? TRUE by default
interval	Boolean: show interval ranges? FALSE by default
pvalue	Numeric: p-value of the survival curves
title	Character: plot title
scale	Character: time scale; default is "days"
auto	Boolean: return the plot automatically prepared (TRUE) or only the bare minimum (FALSE)? TRUE by default

Value

Plot of survival curves

Examples

```
require("survival")
fit <- survfit(Surv(time, status) ~ x, data = aml)
plotSurvivalCurves(fit)
```

plottableXranges *HTML code to plot a X-ranges series*

Description

HTML code to plot a X-ranges series

Usage

```
plottableXranges(hc, shiny = FALSE)
```

Arguments

hc	highcharter object
shiny	Boolean: is the function running in a Shiny session? FALSE by default

Value

HTML elements

`plotTranscripts` *Plot transcripts*

Description

Plot transcripts

Usage

```
plotTranscripts(info, eventPosition = NULL, shiny = FALSE)
```

Arguments

`info`	Information retrieved from Ensembl
`eventPosition`	Numeric: coordinates of the alternative splicing event; NULL by default
`shiny`	Boolean: is the function running in a Shiny session? FALSE by default

Value

NULL (this function is used to modify the Shiny session's state)

Examples

```
event <- "SE_12_-_7985318_7984360_7984200_7982602_SLC2A14"
info  <- queryEnsemblByEvent(event, species="human", assembly="hg19")
pos   <- parseSplicingEvent(event)$pos[[1]]
## Not run:
plotTranscripts(info, pos)

## End(Not run)
```

`plotVariance` *Create the explained variance plot*

Description

Create the explained variance plot

Usage

```
plotVariance(pca)
```

Arguments

`pca`	PCA values

Value

Plot variance as an Highcharter object

Examples

```
pca <- prcomp(USArrests)
plotVariance(pca)
```

---

prepareAnnotationFromEvents

Prepare annotation from alternative splicing events

Description

In case more than one data frame with alternative splicing events is given, the events are cross-referenced according to the chromosome, strand and relevant coordinates per event type (see details).

Usage

```
prepareAnnotationFromEvents(...)
```

Arguments

...	Data frame(s) of alternative splicing events to include in the annotation

Details

Events from two or more data frames are cross-referenced based on each event's chromosome, strand and specific coordinates relevant for each event type:

- Skipped exon: constitutive exon 1 end, alternative exon (start and end) and constitutive exon 2 start
- Mutually exclusive exon: constitutive exon 1 end, alternative exon 1 and 2 (start and end) and constitutive exon 2 start
- Alternative 5' splice site: constitutive exon 1 end, alternative exon 1 end and constitutive exon 2 start
- Alternative first exon: same as alternative 5' splice site
- Alternative 3' splice site: constitutive exon 1 end, alternative exon 1 start and constitutive exon 2 start
- Alternative last exon: same as alternative 3' splice site

Value

List of data frames with the annotation from different data frames joined by event type

Note

When cross-referencing events, gene information is discarded.

Examples

```
# Load sample files (SUPPA annotation)
folder <- "extdata/eventsAnnotSample/suppa_output/suppaEvents"
suppaOutput <- system.file(folder, package="psichomics")

# Parse and prepare SUPPA annotation
suppa <- parseSuppaAnnotation(suppaOutput)
annot <- prepareAnnotationFromEvents(suppa)

# Load sample files (rMATS annotation)
folder <- "extdata/eventsAnnotSample/mats_output/ASEvents/"
matsOutput <- system.file(folder, package="psichomics")

# Parse rMATS annotation and prepare combined annotation from rMATS and SUPPA
mats <- parseMatsAnnotation(matsOutput)
annot <- prepareAnnotationFromEvents(suppa, mats)
```

---

prepareEventPlotOptions

Prepare event plot options

Description

Prepare event plot options

Usage

```
prepareEventPlotOptions(id, ns, labelsPanel = NULL)
```

Arguments

id	Character: identifier
ns	Namespace identifier
labelsPanel	Tab panel containing options to label points

Value

HTML elements

---

prepareFileBrowser

Prepare file browser dialogue and update the input's value accordingly to selected file or directory

---

Description

Prepare file browser dialogue and update the input's value accordingly to selected file or directory

Usage

```
prepareFileBrowser(session, input, id, ...)
```

Arguments

session	Shiny session
input	Shiny input
id	Character: input identifier
...	Arguments passed on to fileBrowser
	default Character: path to initial folder
	caption Character: caption on the selection dialogue
	multiple Boolean: allow to select multiple files?
	directory Boolean: allow to select directories instead of files?
	system Character: system name

Value

NULL (this function is used to modify the Shiny session's state)

---

prepareFirebrowseArchives

Prepares Firebrowse archives in a given directory

Description

Checks Firebrowse archives' integrity using the MD5 files, extracts the content of the archives, moves the content to newly-created folders and removes the original downloaded archives.

Usage

```
prepareFirebrowseArchives(archive, md5, folder, outdir)

prepareFirehoseArchives(archive, md5, folder, outdir)

prepareTCGAarchives(archive, md5, folder, outdir)
```

Arguments

archive	Character: path to downloaded archives
md5	Character: path to MD5 files of each archive
folder	Character: master directory where every archive will be extracted
outdir	Character: subdirectories where to move the extracted content

Value

Invisible TRUE if successful

Examples

```
file <- paste0(
    "~/Downloads",
    "ACC/20151101/gdac.broadinstitute.org_ACC.",
    "Merge_Clinical.Level_1.2015110100.0.0.tar.gz")
md5 <- paste0(file, ".md5")
## Not run:
prepareFirebrowseArchives(archive = file, md5 = paste0(file, ".md5"))

## End(Not run)
```

`processButton` *Style button used to initiate a process*

Description

Style button used to initiate a process

Usage

```
processButton(id, label, ..., class = "btn-primary")
```

Arguments

`id`	Character: button identifier
`label`	Character: label
`...`	Arguments passed on to `shiny::actionButton` **icon** An optional `icon` to appear on the button. **width** The width of the input, e.g. `'400px'`, or `'100%'`; see `validateCssUnit`.
`class`	Character: class

Value

HTML for a button

`processDatasetNames` *Process dataset names*

Description

Process dataset names

Usage

```
processDatasetNames(data)
```

Arguments

`data`	List of lists of data frames

Details

Avoid duplicated names and append the technology used for junction quantification

Value

Processed list of lists of data frames

`processSurvData` *Process survival data to calculate survival curves*

Description

Process survival data to calculate survival curves

Usage

```
processSurvData(event, timeStart, timeStop, followup, group, clinical,
  survTime = NULL)
```

Arguments

`event`	Character: name of column containing time of the event of interest
`timeStart`	Character: name of column containing starting time of the interval or follow up time
`timeStop`	Character: name of column containing ending time of the interval (only relevant for interval censoring)
`followup`	Character: name of column containing follow up time
`group`	Character: group relative to each patient
`clinical`	Data frame: clinical data
`survTime`	`survTime` object: Times to follow up, time start, time stop and event (optional)

Details

The event time will only be used to determine whether the event has occurred (1) or not (0) in case of missing values.

If `survTime` is NULL, the survival times will be fetch from the clinical dataset according to the names given in `timeStart`, `timeStop`, `event` and `followup`. This can became quite slow when using the function in a for loop. If these variables are constant, consider running the function `getAttributesTime` to retrieve the time of such columns once and hand the result to the `survTime` argument of this function.

Value

Data frame with terms needed to calculate survival curves

processSurvival *Check if survival analyses successfully completed or returned errors*

Description

Check if survival analyses successfully completed or returned errors

Usage

```
processSurvival(session, ...)
```

Arguments

session	Shiny session
...	Arguments passed on to processSurvTerms
	censoring Character: censor using "left", "right", "interval" or "interval2"
	scale Character: rescale the survival time to "days", "weeks", "months" or "years"
	formulaStr Character: formula to use
	coxph Boolean: fit a Cox proportional hazards regression model? FALSE by default
	survTime survTime object: times to follow up, time start, time stop and event (optional)
	clinical Data frame: clinical data
	event Character: name of column containing time of the event of interest
	timeStart Character: name of column containing starting time of the interval or follow up time
	timeStop Character: name of column containing ending time of the interval (only relevant for interval censoring)
	group Character: group relative to each patient
	followup Character: name of column containing follow up time

Value

List with survival analysis results

processSurvTerms *Process survival curves terms to calculate survival curves*

Description

Process survival curves terms to calculate survival curves

Usage

```
processSurvTerms(clinical, censoring, event, timeStart, timeStop = NULL,
  group = NULL, formulaStr = NULL, coxph = FALSE, scale = "days",
  followup = "days_to_last_followup", survTime = NULL)
```

Arguments

clinical	Data frame: clinical data
censoring	Character: censor using "left", "right", "interval" or "interval2"
event	Character: name of column containing time of the event of interest
timeStart	Character: name of column containing starting time of the interval or follow up time
timeStop	Character: name of column containing ending time of the interval (only relevant for interval censoring)
group	Character: group relative to each patient
formulaStr	Character: formula to use
coxph	Boolean: fit a Cox proportional hazards regression model? FALSE by default
scale	Character: rescale the survival time to "days", "weeks", "months" or "years"
followup	Character: name of column containing follow up time
survTime	survTime object: times to follow up, time start, time stop and event (optional)

Details

If survTime is NULL, the survival times will be fetch from the clinical dataset according to the names given in timeStart, timeStop, event and followup. This can became quite slow when using the function in a for loop. If these variables are constant, consider running the function getAttributesTime to retrieve the time of such columns once and hand the result to the survTime argument of this function.

Value

A list with a formula object and a data frame with terms needed to calculate survival curves

Examples

```
clinical <- read.table(text = "2549   NA ii  female
                                840   NA i   female
                                 NA 1204 iv    male
                                 NA  383 iv  female
                               1293   NA iii   male
                                 NA 1355 ii    male")
names(clinical) <- c("patient.days_to_last_followup",
                     "patient.days_to_death",
                     "patient.stage_event.pathologic_stage",
                     "patient.gender")
timeStart  <- "days_to_death"
event      <- "days_to_death"
formulaStr <- "patient.stage_event.pathologic_stage + patient.gender"
survTerms  <- processSurvTerms(clinical, censoring="right", event, timeStart,
                               formulaStr=formulaStr)
```

`psichomics` *Start graphical interface of psichomics*

Description

Start graphical interface of psichomics

Usage

```
psichomics(..., reset = FALSE)
```

Arguments

`...`	Arguments passed on to `shiny::runApp` **port** The TCP port that the application should listen on. If the `port` is not specified, and the `shiny.port` option is set (with `options(shiny.port = XX)`), then that port will be used. Otherwise, use a random port. **host** The IPv4 address that the application should listen on. Defaults to the `shiny.host` option, if set, or `"127.0.0.1"` if not. See Details. **workerId** Can generally be ignored. Exists to help some editions of Shiny Server Pro route requests to the correct process. **quiet** Should Shiny status messages be shown? Defaults to FALSE. **display.mode** The mode in which to display the application. If set to the value `"showcase"`, shows application code and metadata from a `DESCRIPTION` file in the application directory alongside the application. If set to `"normal"`, displays the application normally. Defaults to `"auto"`, which displays the application in the mode given in its `DESCRIPTION` file, if any. **test.mode** Should the application be launched in test mode? This is only used for recording or running automated tests. Defaults to the `shiny.testmode` option, or FALSE if the option is not set.
`reset`	Boolean: reset Shiny session? FALSE by default; requires the package `devtools` to reset data

Value

NULL (this function is used to modify the Shiny session's state)

Examples

```
## Not run:
psichomics()

## End(Not run)
```

pubmedUI *Return the interface of relevant PubMed articles for a given gene*

Description

Return the interface of relevant PubMed articles for a given gene

Usage

```
pubmedUI(gene, ...)
```

Arguments

gene	Character: gene
...	Arguments passed on to queryPubMed
	top Numeric: number of articles to retrieve (3 by default)
	field Character: field of interest where to look for terms ("abstract" by default)
	sort Character: sort by a given parameter ("relevance" by default)

Value

HTML interface of relevant PubMed articles

quantifySplicing *Quantify alternative splicing events*

Description

Quantify alternative splicing events

Usage

```
quantifySplicing(annotation, junctionQuant, eventType = c("SE", "MXE", "ALE",
  "AFE", "A3SS", "A5SS"), minReads = 10, genes = NULL)
```

Arguments

annotation	List of data frames: annotation for each alternative splicing event type
junctionQuant	Data frame: junction quantification
eventType	Character: splicing event types to quantify
minReads	Integer: discard alternative splicing quantified using a number of reads below this threshold
genes	Character: gene symbols for which the splicing quantification of associated splicing events is performed (by default, all splicing events undergo splicing quantification)

Value

Data frame with the quantification of the alternative splicing events

Examples

```
# Calculate PSI for skipped exon (SE) and mutually exclusive (MXE) events
annot <- readFile("ex_splicing_annotation.RDS")
junctionQuant <- readFile("ex_junctionQuant.RDS")

psi <- quantifySplicing(annot, junctionQuant, eventType=c("SE", "MXE"))
```

---

quantifySplicingSet *Set of functions to quantify alternative splicing*

Description

Set of functions to quantify alternative splicing

Usage

```
quantifySplicingSet(session, input)
```

Arguments

session	Session object
input	Input object

Value

NULL (this function is used to modify the Shiny session's state)

---

queryEnsembl *Query the Ensembl REST API*

Description

Query the Ensembl REST API

Usage

```
queryEnsembl(path, query, grch37 = TRUE)
```

Arguments

path	Character: API path
query	Character: API query
grch37	Boolean: query the Ensembl GRCh37 API? TRUE by default; otherwise, query the most recent API

Value

Parsed response or NULL if there's no response

Examples

```
path  <- "overlap/region/human/7:140424943-140624564"
query <- list(feature = "gene")
psichomics:::queryEnsembl(path, query, grch37 = TRUE)

path  <- "lookup/symbol/human/BRCA2"
query <- list(expand=1)
psichomics:::queryEnsembl(path, query, grch37 = TRUE)
```

---

queryEnsemblByEvent *Query information from Ensembl by a given alternative splicing event*

Description

Query information from Ensembl by a given alternative splicing event

Usage

```
queryEnsemblByEvent(event, ...)
```

Arguments

event	Character: alternative splicing event identifier
...	Arguments passed on to queryEnsemblByGene
	species Character: species (can be NULL when handling an Ensembl identifier)
	assembly Character: assembly version (can be NULL when handling an Ensembl identifier)

Value

Information from Ensembl

Examples

```
event <- c("SE_17_-_41251792_41249306_41249261_41246877_BRCA1")
queryEnsemblByEvent(event, species="human", assembly="hg19")
```

queryEnsemblByGene *Query information from Ensembl by a given gene*

Description

Query information from Ensembl by a given gene

Usage

```
queryEnsemblByGene(gene, species = NULL, assembly = NULL)
```

Arguments

gene	Character: gene identifier
species	Character: species (can be NULL when handling an Ensembl identifier)
assembly	Character: assembly version (can be NULL when handling an Ensembl identifier)

Value

Information from Ensembl

Examples

```
queryEnsemblByGene("BRCA1", "human", "hg19")
queryEnsemblByGene("ENSG00000139618")
```

queryFirebrowseData *Query the Firebrowse web API for TCGA data*

Description

Query the Firebrowse web API for TCGA data

Usage

```
queryFirebrowseData(format = "json", date = NULL, cohort = NULL,
  data_type = NULL, tool = NULL, platform = NULL, center = NULL,
  level = NULL, protocol = NULL, page = NULL, page_size = NULL,
  sort_by = NULL)

queryFirehoseData(format = "json", date = NULL, cohort = NULL,
  data_type = NULL, tool = NULL, platform = NULL, center = NULL,
  level = NULL, protocol = NULL, page = NULL, page_size = NULL,
  sort_by = NULL)
```

Arguments

format	Character: response format as JSON (default), CSV or TSV
date	Character: dates of the data retrieval by Firebrowse (by default, it uses the most recent data available)
cohort	Character: abbreviation of the cohorts (by default, returns data for all cohorts)
data_type	Character: data types (optional)
tool	Character: data produced by the selected Firebrowse tools (optional)
platform	Character: data generation platforms (optional)
center	Character: data generation centres (optional)
level	Integer: data levels (optional)
protocol	Character: sample characterization protocols (optional)
page	Integer: page of the results to return (optional)
page_size	Integer: number of records per page of results; max is 2000 (optional)
sort_by	String: column used to sort the data (by default, it sorts by cohort)

Value

Response from the Firebrowse web API (it needs to be parsed)

Examples

```
cohort <- psichomics:::getFirebrowseCohorts()[1]
psichomics:::queryFirebrowseData(cohort = cohort, data_type = "mRNASeq")

# Querying for data from a specific date
dates <- psichomics:::getFirebrowseDates()
dates <- format(dates, psichomics:::getFirebrowseDateFormat()$query)

psichomics:::queryFirebrowseData(date = dates[2], cohort = cohort)
```

---

queryPubMed *Query the PubMed REST API*

Description

Query the PubMed REST API

Usage

```
queryPubMed(primary, ..., top = 3, field = "abstract", sort = "relevance")
```

Arguments

primary	Character: primary search term
...	Character: other relevant search terms
top	Numeric: number of articles to retrieve (3 by default)
field	Character: field of interest where to look for terms ("abstract" by default)
sort	Character: sort by a given parameter ("relevance" by default)

Value

Parsed response

Examples

```
psichomics:::queryPubMed("BRCA1", "cancer", "adrenocortical carcinoma")
```

---

queryUniprot *Query the UniProt REST API*

Description

Query the UniProt REST API

Usage

```
queryUniprot(molecule, format = "xml")
```

Arguments

molecule	Character: protein or transcript to query
format	Character: format of the response

Value

Parsed response

Examples

```
protein <- "P51587"
format <- "xml"
psichomics:::queryUniprot(protein, format)

transcript <- "ENST00000488540"
format <- "xml"
psichomics:::queryUniprot(transcript, format)
```

---

readAnnot *Read custom or remote annotation*

Description

Read custom or remote annotation

Usage

```
readAnnot(session, annotation, showProgress = FALSE)
```

Arguments

session	Session object
annotation	Character: chosen annotation
showProgress	Boolean: show progress? FALSE by default

Value

NULL (this function is used to modify the Shiny session's state)

---

readFile *Load local file*

Description

Load local file

Usage

```
readFile(file)
```

Arguments

file	Character: path to the file

Value

Loaded file

Examples

```
junctionQuant <- readFile("ex_junctionQuant.RDS")
```

---

reduceDimensionality *Reduce dimensionality after processing missing values from data frame*

Description

Reduce dimensionality after processing missing values from data frame

Usage

```
reduceDimensionality(data, type = c("pca", "ica"), center = TRUE,
  scale. = FALSE, naTolerance = NULL, missingValues = round(0.05 *
  ncol(data)), ...)
```

Arguments

data	Data frame: data
type	Character: dimensionality reduction technique (pca or ica)
center	either a logical value or a numeric vector of length equal to the number of columns of x.
scale.	Boolean: scale variables?
naTolerance	Integer: percentage of tolerated missing values per column (deprecated)
missingValues	Integer: number of tolerated missing values per column to be replaced with the mean of the values of that same column (5 rows by default)
...	Extra parameters passed to FUN

Value

PCA result in a prcomp object or ICA result object

---

renameDuplicated *Rename vector to avoid duplicated values with another vector*

Description

Renames values by adding an index to the end of duplicates. This allows to prepare unique values in two vectors before a merge, for instance.

Usage

```
renameDuplicated(check, comp)
```

Arguments

check	Character: values to rename if duplicated
comp	Character: values to compare with

Value

Character vector with renamed values if duplicated; else, it returns the usual values. It does not return the comparator values.

Examples

```
psichomics:::renameDuplicated(check = c("blue", "red"), comp = c("green",
                                                                 "blue"))
```

renameGroups *Rename duplicated names from a new group*

Description

Rename duplicated names from a new group

Usage

```
renameGroups(new, old)
```

Arguments

new	Matrix: new groups
old	Matrix: pre-existing groups

Value

Character with no duplicated group names

Note

The names of pre-existing groups are not modified.

renderDataTableSparklines *Render a data table with sparkline HTML elements*

Description

Render a data table with sparkline HTML elements

Usage

```
renderDataTableSparklines(..., options = NULL)
```

Arguments

...	Arguments passed on to `shiny::renderDataTable` **expr** An expression that returns a data frame or a matrix. **searchDelay** The delay for searching, in milliseconds (to avoid too frequent search requests). **callback** A JavaScript function to be applied to the DataTable object. This is useful for DataTables plug-ins, which often require the DataTable instance to be available (http://datatables.net/extensions/). **quoted** Is `expr` a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable. **outputArgs** A list of arguments to be passed through to the implicit call to `dataTableOutput` when `renderDataTable` is used in an interactive R Markdown document.
options	List of options to pass to `renderDataTable`

Details

This slightly modified version of `renderDataTable` calls a JavaScript function to convert the sparkline HTML elements to interactive Highcharts

Value

NULL (this function is used to modify the Shiny session's state)

`renderGeneticInfo` *Render genetic information*

Description

Render genetic information

Usage

```
renderGeneticInfo(output, ns, info, species = NULL, assembly = NULL,
  grch37 = FALSE)
```

Arguments

`output`	Shiny output
`ns`	Namespace function
`info`	Information as retrieved from Ensembl
`species`	Character: species name (NULL by default)
`assembly`	Character: assembly version (NULL by default)
`grch37`	Boolean: use version GRCh37 of the genome? FALSE by default

Value

HTML elements to render gene, protein and transcript annotation

---

`renderGroupInterface` *Render group interface*

Description

Render group interface

Usage

```
renderGroupInterface(ns, multiFisherTests = TRUE)
```

Arguments

`ns`	Namespace function
`multiFisherTests`	Boolean: allow to perform multiple Fisher exact test between groups

Value

HTML elements

`renderProteinInfo` *Render protein information*

Description

Render protein information

Usage

```
renderProteinInfo(protein, transcript, species, assembly)
```

Arguments

`protein`	Character: protein identifier
`transcript`	Character: Ensembl identifier of the protein's respective transcript
`species`	Character: species
`assembly`	Character: assembly

Value

HTML elements

`rm.null` *Filter NULL elements from vector or list*

Description

Filter NULL elements from vector or list

Usage

```
rm.null(v)
```

Arguments

`v`	Vector or list

Value

Filtered vector or list with no NULL elements; if the input is a vector composed of only NULL elements, it returns a NULL (note that it will returns an empty list if the input is a list with only NULL elements)

roundDigits *Round by the given number of digits*

Description

Round by the given number of digits

Usage

```
roundDigits(n)
```

Arguments

n	Numeric: number to round

Value

Formatted number with a given numeric precision

roundMinDown *Round down/up the minimum/maximum value*

Description

Round down/up the minimum/maximum value

Usage

```
roundMinDown(x, digits = 0)

roundMaxUp(x, digits = 0)
```

Arguments

x	Numeric: values
digits	Numeric: number of maximum digits

Value

Rounded numeric value

rowMeans *Calculate mean for each row of a matrix*

Description

Calculate mean for each row of a matrix

Usage

```
rowMeans(mat, na.rm = FALSE)
```

Arguments

mat	Matrix
na.rm	Boolean: remove NAs?

Value

Vector of means

Examples

```
df <- rbind("Gene 1"=c(3, 5, 7), "Gene 2"=c(8, 2, 4), "Gene 3"=c(9:11))
rowMeans(df)
```

rowVars *Calculate variance for each row of a matrix*

Description

Calculate variance for each row of a matrix

Usage

```
rowVars(mat, na.rm = FALSE)
```

Arguments

mat	Matrix
na.rm	Boolean: remove NAs?

Value

Vector of variances

Examples

```
df <- rbind("Gene 1"=c(3, 5, 7), "Gene 2"=c(8, 2, 4), "Gene 3"=c(9:11))
rowVars(df)
```

`selectGroupsUI` *Group selection*

Description

Group selection interface and logic

Usage

```
selectGroupsUI(id, label,
  placeholder = "Click on 'Groups' to create or edit groups",
  noGroupsLabel = NULL, groupsLabel = NULL, maxItems = NULL)

selectGroupsServer(session, id, type, preference = NULL)

getSelectedGroups(input, id, type, filter = NULL)
```

Arguments

`id`	Character: identifier
`label`	Character: selectize label
`placeholder`	Character: selectize placeholder
`noGroupsLabel`	Character: label to show when no groups may be selected (if NULL, the option to show no groups will not be shown)
`groupsLabel`	Character: label to show to the option of using groups when no groups may be selected
`maxItems`	Numeric: maximum number of selected items
`session`	Shiny session
`type`	Character: type of groups (either "Patients", "Samples", "ASevents" or "Genes")
`preference`	Character: name of groups to pre-select, when available (if NULL, all groups will be pre-selected)
`input`	Shiny input
`filter`	Character: get groups only if they are present in this argument (if TCGA-styled gene symbols, they will be "converted" to gene symbols alone)

Value

`selectGroupsUI`: Interface for group selection

`selectGroupsServer`: Server logic for group selection

`getSelectedGroups`: List with selected groups (or NULL if no groups were selected)

Note

To allow the user to (explicitly) select no groups, pass the `noGroupsLabel` and `groupsLabel` arguments.

selectizeGeneInput *Create input to select a gene*

Description

Create input to select a gene

Usage

```
selectizeGeneInput(id, label = "Gene", choices = NULL, multiple = FALSE)
```

Arguments

id	Character: identifier
label	Display label for the control, or NULL for no label.
choices	List of values to select from. If elements of the list are named, then that name rather than the value is displayed to the user. This can also be a named list whose elements are (either named or unnamed) lists or vectors. If this is the case, the outermost names will be used as the "optgroup" label for the elements in the respective sublist. This allows you to group and label similar choices. See the example section for a small demo of this feature.
multiple	Is selection of multiple items allowed?

Value

HTML elements

setASevent *Get or set globally accessible elements*

Description

Get or set globally accessible elements

Usage

```
setASevent(event)

getEvent()

setEvent(event)

getCategories()

getCategory()

setCategory(category)

getCategoryData()
```

```
getActiveDataset()

setActiveDataset(dataset)

getClinicalData(attrs = NULL)

getPatientId()

getPatientAttributes()

getSampleInfo()

setSampleInfo(value, category = getCategory())

getSampleId()

getSampleAttributes()

getJunctionQuantification(category = getCategory())

getGeneExpression(category = getCategory())

setNormalisedGeneExpression(geneExpr, category = getCategory())

getInclusionLevels()

setInclusionLevels(incLevels, category = getCategory())

getPCA(category = getCategory())

setPCA(pca, category = getCategory())

getICA(category = getCategory())

setICA(ica, category = getCategory())

getGroupIndependenceTesting(category = getCategory())

setGroupIndependenceTesting(groupIndependenceTesting,
  category = getCategory())

getSpecies(category = getCategory())

setSpecies(species, category = getCategory())

getAssemblyVersion(category = getCategory())

setAssemblyVersion(assembly, category = getCategory())

getAnnotationName(category = getCategory())
```

```
setAnnotationName(annotName, category = getCategory())

getURLtoDownload()

setURLtoDownload(url)
```

Arguments

event	Character: alternative splicing event
category	Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category
dataset	Character: dataset name
attrs	Character: name of attributes to retrieve (if NULL, the whole dataset is returned)
value	Value to attribute to an element
geneExpr	Data frame or matrix: normalised gene expression
incLevels	Data frame or matrix: inclusion levels
pca	prcomp object (principal component analysis)
ica	Object containing independent component analysis
groupIndependenceTesting	Object containing group independence testing results
species	Character: species
assembly	Character: assembly version
annotName	Character: annotation name
url	Character: URL links to download

Value

Getters return globally accessible data, whereas setters return NULL as they are only used to modify the Shiny session's state

See Also

getEvent, getClinicalMatchFrom, getGroups and getDifferentialAnalyses

---

setFirebrowseData *Set data from Firebrowse*

Description

Set data from Firebrowse

Usage

```
setFirebrowseData(input, output, session, replace = TRUE)
```

Arguments

input	Shiny input
output	Shiny output
session	Shiny session
replace	Boolean: replace loaded data? TRUE by default

Value

NULL (this function is used to modify the Shiny session's state)

---

setLocalData *Load local files*

Description

Load local files

Usage

```
setLocalData(input, output, session, replace = TRUE)

setMultipleFilesData(input, output, session, replace = TRUE)
```

Arguments

input	Shiny input
output	Shiny output
session	Shiny session
replace	Boolean: replace loaded data? TRUE by default

Value

NULL (this function is used to modify the Shiny session's state)

---

setOperation *Perform set operations on selected groups*

Description

Perform set operations on selected groups

Usage

```
setOperation(operation, groups, selected, symbol = " ", groupName = NULL,
  first = NULL, second = NULL, matches = NULL, type = "Samples",
  assignColoursToGroups = FALSE)
```

Arguments

operation	Character: set operation
groups	Matrix: groups
selected	Integer: index of rows regarding selected groups
symbol	Character: Unicode symbol to visually indicate the operation performed (" " by default)
groupName	Character: group name (automatically created if NULL or "")
first	Character: identifiers of the first element (required when performing the complement operation)
second	Character: identifiers of the second element (required when performing the complement operation)
matches	Character: match between samples (as names) and patients (as values)
type	Character: type of group where set operations are to be performed
assignColoursToGroups	Boolean: assign colours to new groups? FALSE by default

Value

Matrix containing groups (new group is in the first row)

---

setOperationIcon *Create an icon based on set operations*

Description

Based on the icon function

Usage

```
setOperationIcon(name, class = NULL, ...)
```

Arguments

name	Character: icon name
class	Character: additional classes to customise the icon element
...	Extra arguments for the icon HTML element

Value

Icon element

showAlert *Show or remove an alert*

Description

You can also use `errorAlert` and `warningAlert` to use template alerts already stylised to show errors and warnings respectively.

Usage

```
showAlert(session, ..., title = NULL, style = NULL, dismissible = TRUE,
  alertId = "alert")

errorAlert(session, ..., title = NULL, dismissible = TRUE,
  alertId = "alert")

warningAlert(session, ..., title = NULL, dismissible = TRUE,
  alertId = "alert")

removeAlert(output, alertId = "alert")
```

Arguments

`session`	Shiny session
`...`	Arguments to render as elements of alert
`title`	Character: title of the alert (optional)
`style`	Character: style of the alert ("alert-danger", "alert-warning" or NULL)
`dismissible`	Boolean: is the alert dismissible? TRUE by default
`alertId`	Character: alert identifier
`output`	Shiny output

Value

NULL (this function is used to modify the Shiny session's state)

See Also

`showModal`

showGroupsTable *Present groups table*

Description

Present groups table

Usage

```
showGroupsTable(type)
```

Arguments

type	Character: type of groups (either "Patients", "Samples", "ASevents" or "Genes")

Value

Matrix with groups ordered (or NULL if no groups exist)

sidebar *Sidebar without a well*

Description

Modified version of `shiny::sidebarPanel` without a well

Usage

```
sidebar(..., width = 4)
```

Arguments

...	UI elements to include on the sidebar
width	The width of the sidebar. For fluid layouts this is out of 12 total units; for fixed layouts it is out of whatever the width of the sidebar's parent column is.

Value

A sidebar that can be passed to `sidebarLayout`

signifDigits *Get number of significant digits*

Description

Get number of significant digits

Usage

```
signifDigits(n)
```

Arguments

n	Numeric: number to round

Value

Formatted number with a given number of significant digits

singleDiffAnalyses *Perform statistical analysis on a given splicing event*

Description

Perform statistical analyses on a given vector containing elements from different groups

Usage

```
singleDiffAnalyses(vector, group, threshold = 1, step = 100,
  analyses = c("wilcoxRankSum", "ttest", "kruskal", "levene", "fligner"))
```

Arguments

vector	Numeric
group	Character: group of each element in the vector
threshold	Integer: minimum number of data points to perform analysis in a group (default is 1)
step	Numeric: number of events before the progress bar is updated (a bigger number allows for a faster execution)
analyses	Character: analyses to perform (see "Details")

Details

The following statistical analyses may be performed by including the respective string in the `analysis` argument:

- `ttest` - Unpaired t-test (2 groups)
- `wilcoxRankSum` - Wilcoxon Rank Sum test (2 groups)
- `kruskal` - Kruskal test (2 or more groups)
- `levene` - Levene's test (2 or more groups)
- `fligner` - Fligner-Killeen test (2 or more groups)

Value

A row from a data frame with the results

`sortCoordinates` *Sort coordinates for some event types*

Description

Some programs sort the coordinates of specific event types differently. To make them all comparable across programs, the coordinates are ordered by increasing (plus strand) or decreasing order (minus strand)

Usage

```
sortCoordinates(events)
```

Arguments

`events`	List of data frames with alternative splicing events for a given program

Value

List of data frames with alternative splicing events for a given program

---

`spearman` *Perform Spearman's test and return interface to show the results*

Description

Perform Spearman's test and return interface to show the results

Usage

```
spearman(data, groups)
```

Arguments

`data`	Numeric, data frame or matrix: data for one gene or alternative splicing event
`groups`	List of characters (list of groups containing data identifiers) or character vector (group of each value in `data`)

Value

HTML elements

startProcess *Signal the program that a process is starting*

Description

Style button to show processing is in progress

Usage

```
startProcess(id)
```

Arguments

id	Character: button identifier

Value

Start time of the process

startProgress *Create a progress object*

Description

Create a progress object

Usage

```
startProgress(message, divisions, global = if (isRunning()) sharedData else
  getHidden())
```

Arguments

message	Character: progress message
divisions	Integer: number of divisions in the progress bar
global	Shiny's global variable

Value

NULL (this function is used to modify the Shiny session's state or internal hidden variables)

styleModal *Style and show a modal*

Description

You can also use `errorModal` and `warningModal` to use a template modal already stylised to show errors and warnings, respectively.

Usage

```
styleModal(session, title, ..., style = NULL,
  iconName = "exclamation-circle", footer = NULL, echo = FALSE,
  size = "medium", dismissButton = TRUE)

errorModal(session, title, ..., size = "small", footer = NULL)

warningModal(session, title, ..., size = "small", footer = NULL)

infoModal(session, title, ..., size = "small", footer = NULL)
```

Arguments

`session`	Current Shiny session
`title`	Character: modal title
`...`	Extra arguments to pass to `shiny::modalDialog`
`style`	Character: style of the modal (NULL, "warning", "error" or "info"; NULL by default)
`iconName`	Character: FontAwesome icon name to appear with the title
`footer`	HTML elements to use in footer
`echo`	Boolean: print to console? FALSE by default
`size`	Character: size of the modal - "medium" (default), "small" or "large"
`dismissButton`	Boolean: show dismiss button in footer? TRUE by default

Value

NULL (this function is used to modify the Shiny session's state)

See Also

`showAlert`

`survdiff.survTerms` *Test differences between survival curves*

Description

Test differences between survival curves

Usage

```
survdiff.survTerms(survTerms, ...)
```

Arguments

`survTerms`	survTerms object: processed survival terms
`...`	Arguments passed on to `survival::survdiff`
	subset expression indicating which subset of the rows of data should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), a numeric vector indicating which observation numbers are to be included (or excluded if negative), or a character vector of row names to be included. All observations are included by default.
	na.action a missing-data filter function. This is applied to the `model.frame` after any subset argument has been used. Default is `options()$na.action`.
	rho a scalar parameter that controls the type of test.

Value

an object of class "survfit". See survfit.object for details. Methods defined for survfit objects are print, plot, lines, and points.

Examples

```
clinical <- read.table(text = "2549   NA ii  female
                                840   NA i   female
                                 NA 1204 iv    male
                                 NA  383 iv  female
                               1293   NA iii   male
                                 NA 1355 ii    male")
names(clinical) <- c("patient.days_to_last_followup",
                     "patient.days_to_death",
                     "patient.stage_event.pathologic_stage",
                     "patient.gender")
timeStart  <- "days_to_death"
event      <- "days_to_death"
formulaStr <- "patient.stage_event.pathologic_stage + patient.gender"
survTerms  <- processSurvTerms(clinical, censoring="right", event, timeStart,
                               formulaStr=formulaStr)
survdiff.survTerms(survTerms)
```

`survfit.survTerms` *Compute estimates of survival curves*

Description

Compute estimates of survival curves

Usage

```
## S3 method for class 'survTerms'
survfit(survTerms, ...)
```

Arguments

`survTerms`	`survTerms` object: processed survival terms
`...`	Arguments passed on to `survival::survfit.formula`

weights The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous, compared to use of the `subset` argument.

subset expression saying that only a subset of the rows of the data should be used in the fit.

na.action a missing-data filter function, applied to the model frame, after any `subset` argument has been used. Default is `options()$na.action`.

etype a variable giving the type of event. This has been superseded by multi-state Surv objects; see example below.

id identifies individual subjects, when a given person can have multiple lines of data.

istate for multi-state models, identifies the initial state of each subject

timefix process times through the `aeqSurv` function to eliminate potential round-off issues.

Value

`survfit` object. See `survfit.object` for details. Methods defined for survfit objects are `print`, `plot`, `lines`, and `points`.

Examples

```
clinical <- read.table(text = "2549   NA ii  female
                                840   NA i   female
                                 NA 1204 iv    male
                                 NA  383 iv  female
                               1293   NA iii   male
                                 NA 1355 ii    male")
names(clinical) <- c("patient.days_to_last_followup",
                     "patient.days_to_death",
                     "patient.stage_event.pathologic_stage",
                     "patient.gender")
timeStart  <- "days_to_death"
event      <- "days_to_death"
formulaStr <- "patient.stage_event.pathologic_stage + patient.gender"
```

```
survTerms  <- processSurvTerms(clinical, censoring="right", event, timeStart,
                               formulaStr=formulaStr)
require("survival")
survfit(survTerms)
```

`tabDataset` *Creates a* `tabPanel` *template for a* `datatable` *with a title and description*

Description

Creates a `tabPanel` template for a `datatable` with a title and description

Usage

```
tabDataset(ns, title, tableId, columns, visCols, data, description = NULL)
```

Arguments

`ns`	Namespace function
`title`	Character: tab title
`tableId`	Character: id of the `datatable`
`columns`	Character: column names of the `datatable`
`visCols`	Boolean: visible columns
`data`	Data frame: dataset of interest
`description`	Character: description of the table (optional)

Value

HTML elements

`table2html` *Create HTML table from data frame or matrix*

Description

Create HTML table from data frame or matrix

Usage

```
table2html(data, rownames = TRUE, colnames = TRUE, class = NULL)
```

Arguments

`data`	Data frame or matrix
`rownames`	Boolean: print row names?
`colnames`	Boolean: print column names?
`class`	Character: table class

Value

HTML elements

tableRow *Create a row for a HTML table*

Description

Create a row for a HTML table

Usage

```
tableRow(..., th = FALSE)
```

Arguments

...	Elements to include in the row
th	Boolean: is this row the table head?

Value

HTML elements

testGroupIndependence *Multiple independence tests between reference groups and list of groups*

Description

Test multiple contigency tables comprised by two groups (one reference group and another containing remaing elements) and provided groups.

Usage

```
testGroupIndependence(ref, groups, elements, pvalueAdjust = "BH")
```

Arguments

ref	List of character: list of groups where each element contains the identifiers of respective elements
groups	List of characters: list of groups where each element contains the identifiers of respective elements
elements	Character: all available elements (if a data frame is given, its rownames will be used)
pvalueAdjust	Character: method used to adjust p-values (see Details)

Details

The following methods for p-value adjustment are supported by using the respective string in the `pvalueAdjust` argument:

- `none`: Do not adjust p-values
- `BH`: Benjamini-Hochberg's method (false discovery rate)
- `BY`: Benjamini-Yekutieli's method (false discovery rate)
- `bonferroni`: Bonferroni correction (family-wise error rate)
- `holm`: Holm's method (family-wise error rate)
- `hochberg`: Hochberg's method (family-wise error rate)
- `hommel`: Hommel's method (family-wise error rate)

Value

`multiGroupIndependenceTest` object, a data frame containing:

`attribute`	Name of the original groups compared against the reference groups
`table`	Contigency table used for testing
`pvalue`	Fisher's exact test's p-value

See Also

`parseCategoricalGroups` and `plotGroupIndependence`

Examples

```
elements <- paste("patients", 1:10)
ref      <- elements[5:10]
groups   <- list(race=list(asian=elements[1:3],
                           white=elements[4:7],
                           black=elements[8:10]),
                 region=list(european=elements[c(4, 5, 9)],
                             african=elements[c(6:8, 10)]))
groupTesting <- testGroupIndependence(ref, groups, elements)
# View(groupTesting)
```

`testSingleIndependence` *Multiple independence tests between a reference group and list of groups*

Description

Uses Fisher's exact test.

Usage

```
testSingleIndependence(ref, groups, elements, pvalueAdjust = "BH")
```

Arguments

ref	Character: identifier of elements in reference group
groups	List of characters: list of groups where each element contains the identifiers of respective elements
elements	Character: all patient identifiers
pvalueAdjust	Character: method used to adjust p-values (see Details)

Details

The following methods for p-value adjustment are supported by using the respective string in the pvalueAdjust argument:

- none: Do not adjust p-values
- BH: Benjamini-Hochberg's method (false discovery rate)
- BY: Benjamini-Yekutieli's method (false discovery rate)
- bonferroni: Bonferroni correction (family-wise error rate)
- holm: Holm's method (family-wise error rate)
- hochberg: Hochberg's method (family-wise error rate)
- hommel: Hommel's method (family-wise error rate)

Value

Returns a groupIndependenceTest object: a list where each element is a list containing:

attribute	Name of the original groups compared against the reference groups
table	Contigency table used for testing
pvalue	Fisher's exact test's p-value

---

testSurvival *Test the survival difference between groups of patients*

Description

Test the survival difference between groups of patients

Usage

```
testSurvival(survTerms, ...)
```

Arguments

survTerms	survTerms object: processed survival terms
...	Arguments passed on to survival::survdiff

subset expression indicating which subset of the rows of data should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), a numeric vector indicating which observation numbers are to be included (or excluded if negative), or a character vector of row names to be included. All observations are included by default.

na.action a missing-data filter function. This is applied to the `model.frame` after any subset argument has been used. Default is `options()$na.action`.

rho a scalar parameter that controls the type of test.

Value

p-value of the survival difference or `NA`

Note

Instead of raising errors, an `NA` is returned

Examples

```
require("survival")
data <- aml
timeStart  <- "event"
event      <- "event"
followup   <- "time"
data$event <- NA
data$event[aml$status == 1] <- aml$time[aml$status == 1]
censoring  <- "right"
formulaStr <- "x"
survTerms <- processSurvTerms(data, censoring=censoring, event=event,
                              timeStart=timeStart, followup=followup,
                              formulaStr=formulaStr)
testSurvival(survTerms)
```

---

testSurvivalCutoff *Test the survival difference between two survival groups given a cutoff*

Description

Test the survival difference between two survival groups given a cutoff

Usage

```
testSurvivalCutoff(cutoff, data, filter = TRUE, clinical, ...,
  session = NULL, survivalInfo = FALSE)
```

Arguments

`cutoff`	Numeric: Cutoff of interest
`data`	Numeric: elements of interest to test against the cutoff
`filter`	Boolean or numeric: elements to use (all by default)
`clinical`	Data frame: clinical data
`...`	Arguments passed on to `processSurvTerms` **censoring** Character: censor using "left", "right", "interval" or "interval2" **scale** Character: rescale the survival time to "days", "weeks", "months" or "years" **formulaStr** Character: formula to use

coxph Boolean: fit a Cox proportional hazards regression model? FALSE by default

survTime survTime object: times to follow up, time start, time stop and event (optional)

event Character: name of column containing time of the event of interest

timeStart Character: name of column containing starting time of the interval or follow up time

timeStop Character: name of column containing ending time of the interval (only relevant for interval censoring)

followup Character: name of column containing follow up time

session	Shiny session
survivalInfo	Boolean: return extra survival information

Value

p-value of the survival difference

`textSuggestions` *Create script for auto-completion of text input*

Description

Uses the JavaScript library `jquery.textcomplete`

Usage

```
textSuggestions(id, words, novalue = "No matching value", char = " ")
```

Arguments

id	Character: input ID
words	Character: words to suggest
novalue	Character: string when there's no matching values
char	Character to succeed accepted word

Value

HTML string with the JavaScript script prepared to run

Examples

```
words <- c("tumor_stage", "age", "gender")
psichomics:::textSuggestions("textareaid", words)
```

toJSarray *Convert vector of values to JavaScript array*

Description

Convert vector of values to JavaScript array

Usage

```
toJSarray(values)
```

Arguments

values	Character vector

Value

Character with valid JavaScript array

transformData *Transform data in data frame*

Description

Transform data in data frame

Usage

```
transformData(input, df, x, y)
```

Arguments

input	Shiny input
df	Data frame
x	Character: column name
y	Character: column name

Value

Data frame with transformed data in new columns and respective name of created columns

transformOptions *Show variable transformation(s)*

Description

Show variable transformation(s)

Usage

```
transformOptions(label, type = NULL)
```

Arguments

label	Character: label to display
type	Character: show the variable transformation for the chosen type; NULL (by default) to show all variable transformations

Value

Character labelling variable transformation(s)

transformValues *Transform values as per a given type of transformation*

Description

Transform values as per a given type of transformation

Usage

```
transformValues(val, type, avoidZero = TRUE)
```

Arguments

val	Integer: values to transform
type	Character: type of transformation
avoidZero	Boolean: add the smallest non-zero number available to zero values; avoids returning infinity values during Log transformation (which are not plotted); useful for preserving p-values of 0, for instance; TRUE by default

Value

Integer containing transformed values

trimWhitespace *Trims whitespace from a word*

Description

Trims whitespace from a word

Usage

```
trimWhitespace(word)
```

Arguments

word	Character to trim

Value

Character without whitespace

Examples

```
psichomics:::trimWhitespace("    hey   there     ")
psichomics:::trimWhitespace(c("pineapple    ", "one two three",
                              " sunken    ship   "))
```

ttest *Perform unpaired t-test analysis and return interface to show the results*

Description

Perform unpaired t-test analysis and return interface to show the results

Usage

```
ttest(data, groups, stat = NULL)
```

Arguments

data	Numeric, data frame or matrix: data for one gene or alternative splicing event
groups	List of characters (list of groups containing data identifiers) or character vector (group of each value in data)
stat	Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)

Value

HTML elements

uniqueBy *Check unique rows of a data frame based on a set of its columns*

Description

Check unique rows of a data frame based on a set of its columns

Usage

```
uniqueBy(data, ...)
```

Arguments

data	Data frame or matrix
...	Name of columns

Value

Data frame with unique values based on set of columns

updateClinicalParams *Update available clinical attributes when the clinical data changes*

Description

Update available clinical attributes when the clinical data changes

Usage

```
updateClinicalParams(session, attrs)
```

Arguments

session	Shiny session
attrs	Character: patient attributes

Value

NULL (this function is used to modify the Shiny session's state)

updateFileBrowserInput *Change the value of a fileBrowserInput on the client*

Description

Change the value of a fileBrowserInput on the client

Usage

```
updateFileBrowserInput(session, id, ..., value = NULL)
```

Arguments

session	Shiny session
id	Character: input identifier
...	Additional arguments passed to fileBrowser. Only used if value is NULL.
value	Character: file or directory path

Details

Sends a message to the client, telling it to change the value of the input object. For fileBrowserInput objects, this changes the value displayed in the text-field and triggers a client-side change event. A directory selection dialogue is not displayed.

Value

NULL (this function is used to modify the Shiny session's state)

Source

Original code by wleepang: https://github.com/wleepang/shiny-directory-input

updateProgress *Update a progress object*

Description

Update a progress object

Usage

```
updateProgress(message = "Loading", value = NULL, max = NULL,
  detail = NULL, divisions = NULL, global = if (isRunning()) sharedData
  else getHidden(), console = TRUE)
```

Arguments

message	Character: progress message
value	Integer: current progress value
max	Integer: maximum progress value
detail	Character: detailed message
divisions	Integer: number of divisions in the progress bar
global	Shiny's global variable
console	Boolean: print message to console? (TRUE by default)

Details

If divisions is not NULL, a progress bar is started with the given divisions. If value is NULL, the progress bar will be incremented by one; otherwise, the progress bar will be incremented by the integer given in value.

Value

NULL (this function is used to modify the Shiny session's state)

---

vennEvents *Compare the number of events from the different programs in a Venn diagram*

Description

Compare the number of events from the different programs in a Venn diagram

Usage

```
vennEvents(join, eventType)
```

Arguments

join	List of lists of data frame
eventType	Character: type of event

Value

Venn diagrams for a given event type

`wilcox` *Perform Wilcoxon analysis and return interface to show the results*

Description

Perform Wilcoxon analysis and return interface to show the results

Usage

```
wilcox(data, groups, stat = NULL)
```

Arguments

`data`	Numeric, data frame or matrix: data for one gene or alternative splicing event
`groups`	List of characters (list of groups containing data identifiers) or character vector (group of each value in `data`)
`stat`	Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)

Value

HTML elements

Index