Package ‘psichomics’

May 30, 2024

Title  Graphical Interface for Alternative Splicing Quantification, Analysis and Visualisation

Version  1.30.0

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 based on The Cancer Genome Atlas (TCGA), the Genotype-Tissue Expression project (GTEx), Sequence Read Archive (SRA) and user-provided 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 (>= 4.0), shiny (>= 1.7.0), shinyBS

License  MIT + file LICENSE

LazyData  true

RoxygenNote  7.3.1

Imports  AnnotationDbi, AnnotationHub, BiocFileCache, cluster, colourpicker, data.table, digest, dplyr, DT (>= 0.2), edgeR, fastICA, fastmatch, ggplot2, ggrepel, graphics, grDevices, highcharter (>= 0.5.0), htmltools, httr, jsonlite, limma, pairsD3, plyr, purrr, Rcpp (>= 0.12.14), recount, Rfast, R.utils, reshape2, shinyjs, stringr, stats, SummarizedExperiment, survival, tools, utils, XML, xtable, methods

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

LinkingTo  Rcpp

VignetteBuilder  knitr

Collate  'RcppExports.R' 'utils.R' 'globalAccess.R' 'app.R' 'analysis.R' 'analysis_correlation.R'
## Contents

**Repository**  Bioconductor 3.19  
**Date/Publication**  2024-05-29  
**Author**  Nuno Saraiva-Agostinho [aut, cre]  
  (<https://orcid.org/0000-0002-5549-105X>),  
  Nuno Luís Barbosa-Morais [aut, led, ths]  
  (<https://orcid.org/0000-0002-1215-0538>),  
  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>  

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>.onAttach</td>
<td>9</td>
</tr>
<tr>
<td>addObjectAttrs</td>
<td>9</td>
</tr>
<tr>
<td>addTCGAdata</td>
<td>10</td>
</tr>
<tr>
<td>analysesTableSet</td>
<td>10</td>
</tr>
<tr>
<td>appendNewGroups</td>
<td>12</td>
</tr>
<tr>
<td>appServer</td>
<td>12</td>
</tr>
<tr>
<td>appUI</td>
<td>14</td>
</tr>
<tr>
<td>areSplicingEvents</td>
<td>15</td>
</tr>
<tr>
<td>articleUI</td>
<td>16</td>
</tr>
<tr>
<td>assignColours</td>
<td>16</td>
</tr>
<tr>
<td>assignValuePerSubject</td>
<td>17</td>
</tr>
<tr>
<td>basicStats</td>
<td>18</td>
</tr>
<tr>
<td>blendColours</td>
<td>18</td>
</tr>
<tr>
<td>browseDownloadFolderInput</td>
<td>19</td>
</tr>
<tr>
<td>browserHistory</td>
<td>19</td>
</tr>
<tr>
<td>calculateInclusionLevels</td>
<td>20</td>
</tr>
<tr>
<td>calculateLoadingsContribution</td>
<td>21</td>
</tr>
<tr>
<td>checkFileFormat</td>
<td>22</td>
</tr>
<tr>
<td>checkFirebrowse</td>
<td>22</td>
</tr>
<tr>
<td>checkGroupType</td>
<td>23</td>
</tr>
<tr>
<td>checkIntegrity</td>
<td>23</td>
</tr>
<tr>
<td>checkSurvivalInput</td>
<td>24</td>
</tr>
<tr>
<td>clusterICAset</td>
<td>24</td>
</tr>
<tr>
<td>clusterSet</td>
<td>25</td>
</tr>
<tr>
<td>colourInputMod</td>
<td>25</td>
</tr>
<tr>
<td>colSums,EList-method</td>
<td>26</td>
</tr>
<tr>
<td>convertGeneIdentifiers</td>
<td>27</td>
</tr>
<tr>
<td>correlateGEandAS</td>
<td>28</td>
</tr>
<tr>
<td>createDataTab</td>
<td>29</td>
</tr>
<tr>
<td>createDensitySparklines</td>
<td>29</td>
</tr>
<tr>
<td>Contents</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>createEventPlotting</td>
<td>30</td>
</tr>
<tr>
<td>createGroup</td>
<td>31</td>
</tr>
<tr>
<td>createGroupById</td>
<td>33</td>
</tr>
<tr>
<td>createGroupByAttribute</td>
<td>32</td>
</tr>
<tr>
<td>createGroupFromInput</td>
<td>33</td>
</tr>
<tr>
<td>createJunctionsTemplate</td>
<td>34</td>
</tr>
<tr>
<td>createOptimalSurvData</td>
<td>35</td>
</tr>
<tr>
<td>createSparklines</td>
<td>36</td>
</tr>
<tr>
<td>customRowMeans</td>
<td>37</td>
</tr>
<tr>
<td>diagramSplicingEvent</td>
<td>38</td>
</tr>
<tr>
<td>diffAnalyses</td>
<td>39</td>
</tr>
<tr>
<td>diffExpressionSet</td>
<td>41</td>
</tr>
<tr>
<td>diffSplicingSet</td>
<td>41</td>
</tr>
<tr>
<td>disableTab</td>
<td>42</td>
</tr>
<tr>
<td>discardLowCoveragePSIvalues</td>
<td>42</td>
</tr>
<tr>
<td>discardOutsideSamplesFromGroups</td>
<td>43</td>
</tr>
<tr>
<td>display</td>
<td>43</td>
</tr>
<tr>
<td>downloadFiles</td>
<td>44</td>
</tr>
<tr>
<td>ensemblToUniprot</td>
<td>44</td>
</tr>
<tr>
<td>escape</td>
<td>45</td>
</tr>
<tr>
<td>eventPlotOptions</td>
<td>46</td>
</tr>
<tr>
<td>exportGroupsToFile</td>
<td>46</td>
</tr>
<tr>
<td>export_highcharts</td>
<td>47</td>
</tr>
<tr>
<td>fileBrowser</td>
<td>47</td>
</tr>
<tr>
<td>fileBrowserInfoInput</td>
<td>48</td>
</tr>
<tr>
<td>filterGeneExpr</td>
<td>49</td>
</tr>
<tr>
<td>filterGroups</td>
<td>51</td>
</tr>
<tr>
<td>filterPSI</td>
<td>51</td>
</tr>
<tr>
<td>findASeventsFromGene</td>
<td>53</td>
</tr>
<tr>
<td>eventData</td>
<td>53</td>
</tr>
<tr>
<td>geneExprFileInput</td>
<td>54</td>
</tr>
<tr>
<td>geneExprSurvSet</td>
<td>54</td>
</tr>
<tr>
<td>getNormalisationFilteringInterface</td>
<td>55</td>
</tr>
<tr>
<td>getAttributesTime</td>
<td>55</td>
</tr>
<tr>
<td>getClinicalDataForSurvival</td>
<td>56</td>
</tr>
<tr>
<td>getClinicalMatchFrom</td>
<td>57</td>
</tr>
<tr>
<td>getData</td>
<td>57</td>
</tr>
<tr>
<td>getDataRows</td>
<td>58</td>
</tr>
<tr>
<td>getDifferentialExpression</td>
<td>58</td>
</tr>
<tr>
<td>getDifferentialSplicing</td>
<td>59</td>
</tr>
<tr>
<td>getDownloadsFolder</td>
<td>61</td>
</tr>
<tr>
<td>getFirebrowseDateFormat</td>
<td>61</td>
</tr>
<tr>
<td>getGeneList</td>
<td>53</td>
</tr>
<tr>
<td>getGlobal</td>
<td>63</td>
</tr>
<tr>
<td>getGroups</td>
<td>66</td>
</tr>
<tr>
<td>getGtexDataTypes</td>
<td>67</td>
</tr>
<tr>
<td>getGtexDataURL</td>
<td>68</td>
</tr>
<tr>
<td>getGtexTissues</td>
<td>68</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>getHidden</td>
<td>69</td>
</tr>
<tr>
<td>getHighlightedPoints</td>
<td>69</td>
</tr>
<tr>
<td>getNumerics</td>
<td>70</td>
</tr>
<tr>
<td>getSampleFromSubject</td>
<td>71</td>
</tr>
<tr>
<td>getSelectedDataPanel</td>
<td>72</td>
</tr>
<tr>
<td>getServerFunctions</td>
<td>73</td>
</tr>
<tr>
<td>getSplicingEventCoordinates</td>
<td>73</td>
</tr>
<tr>
<td>getSplicingEventData</td>
<td>74</td>
</tr>
<tr>
<td>getSplicingEventFromGenes</td>
<td>74</td>
</tr>
<tr>
<td>getSplicingEventTypes</td>
<td>75</td>
</tr>
<tr>
<td>getSubjectFromSample</td>
<td>76</td>
</tr>
<tr>
<td>getTCGAdataTypes</td>
<td>77</td>
</tr>
<tr>
<td>getUiFunctions</td>
<td>78</td>
</tr>
<tr>
<td>getValidEvents</td>
<td>78</td>
</tr>
<tr>
<td>ggplotServer</td>
<td>79</td>
</tr>
<tr>
<td>ggplotTooltip</td>
<td>80</td>
</tr>
<tr>
<td>ggplotUI</td>
<td>81</td>
</tr>
<tr>
<td>globalSelectize</td>
<td>81</td>
</tr>
<tr>
<td>groupByAttribute</td>
<td>82</td>
</tr>
<tr>
<td>groupManipulation</td>
<td>82</td>
</tr>
<tr>
<td>groupManipulationInput</td>
<td>83</td>
</tr>
<tr>
<td>groupPerElem</td>
<td>83</td>
</tr>
<tr>
<td>groupsServerOnce</td>
<td>84</td>
</tr>
<tr>
<td>hchart_survfit</td>
<td>85</td>
</tr>
<tr>
<td>hc_scatter</td>
<td>86</td>
</tr>
<tr>
<td>HTMLfast</td>
<td>87</td>
</tr>
<tr>
<td>importGroupsFrom</td>
<td>87</td>
</tr>
<tr>
<td>inclusionLevelsFilterInterface</td>
<td>88</td>
</tr>
<tr>
<td>inclusionLevelsInterface</td>
<td>88</td>
</tr>
<tr>
<td>inlineDialog</td>
<td>89</td>
</tr>
<tr>
<td>insideFile</td>
<td>90</td>
</tr>
<tr>
<td>is.whole</td>
<td>90</td>
</tr>
<tr>
<td>isFile</td>
<td>91</td>
</tr>
<tr>
<td>isFirebrowseUp</td>
<td>91</td>
</tr>
<tr>
<td>isRStudioServer</td>
<td>92</td>
</tr>
<tr>
<td>joinEventsPerType</td>
<td>92</td>
</tr>
<tr>
<td>junctionString</td>
<td>93</td>
</tr>
<tr>
<td>labelBasedOnCutoff</td>
<td>93</td>
</tr>
<tr>
<td>leveneTest</td>
<td>94</td>
</tr>
<tr>
<td>linkToArticles</td>
<td>95</td>
</tr>
<tr>
<td>linkToRunJS</td>
<td>95</td>
</tr>
<tr>
<td>listAllAnnotations</td>
<td>96</td>
</tr>
<tr>
<td>listSplicingAnnotations</td>
<td>96</td>
</tr>
<tr>
<td>loadAnnotation</td>
<td>97</td>
</tr>
<tr>
<td>loadAnnotationHub</td>
<td>98</td>
</tr>
<tr>
<td>loadBy</td>
<td>99</td>
</tr>
<tr>
<td>loadCustomSplicingAnnotationSet</td>
<td>99</td>
</tr>
<tr>
<td>loadedDataModal</td>
<td>100</td>
</tr>
</tbody>
</table>
plotLibrarySize .......................................................... 145
plotPCA ................................................................. 146
plotPCAvariance ........................................................ 147
plotPointsStyle ........................................................ 148
plotProtein .............................................................. 148
plotRowStats ............................................................ 149
plotSingleICA .......................................................... 151
plotSplicingEvent ...................................................... 152
plotSurvivalCurves ..................................................... 153
plotSurvivalPvaluesByCutoff ........................................ 154
plottableXranges ....................................................... 156
plotTranscripts ........................................................ 157
prepareAnnotationFromEvents ....................................... 158
prepareEventPlotOptions ............................................. 159
prepareFileBrowser ................................................... 160
prepareFirebrowseArchives ......................................... 160
prepareGenePresentation .............................................. 161
prepareJunctionQuantSTAR .......................................... 162
preparePreMadeGroupForSelection ................................ 163
prepareSRAmetadata ................................................... 163
prepareWordBreak ..................................................... 164
preserveAttributes ..................................................... 165
processButton .......................................................... 165
processDatasetNames .................................................. 166
processSRAdata ........................................................ 166
processSurvData ........................................................ 167
processSurvival ........................................................ 168
processSurvTerms ...................................................... 168
psichomics ............................................................... 170
pubmedUI ................................................................. 172
quantifySplicing ........................................................ 172
quantifySplicingSet ..................................................... 173
queryEnsembl .......................................................... 174
queryEnsemblByGene ................................................. 174
queryFirebrowseData .................................................. 175
queryPubMed ............................................................ 176
queryUniprot ............................................................ 177
readAnnot ............................................................... 178
readFile ................................................................. 178
reduceDimensionality ................................................... 179
renameDuplicated ....................................................... 180
renameGroups ........................................................... 180
renderBoxplot .......................................................... 181
renderDataTableSparklines .......................................... 182
renderGeneticInfo ...................................................... 183
renderGroupInterface ................................................ 183
renderProteinInfo ...................................................... 184
replaceStrInList ......................................................... 184
Contents

rm.null ......................................................... 185
roundDigits .................................................. 185
roundMinDown ................................................. 186
saveProcessedSRAData ....................................... 186
selectGroupsUI ................................................ 187
selectizeGeneInput .......................................... 188
selectPreMadeGroup ......................................... 189
setFirebrowseData ........................................... 189
setLocalData .................................................. 190
setOperation .................................................. 190
setOperationIcon ............................................. 191
showAlert ...................................................... 192
showGroupsTable ............................................. 193
sidebar ......................................................... 194
signifDigits .................................................... 194
singleDiffAnalyses .......................................... 195
sortCoordinates .............................................. 196
startProcess ................................................... 196
startProgress .................................................. 197
styleModal ...................................................... 198
subjectMultiMatchWarning ................................. 199
subsetGeneExpressionFromMatchingGenes .................. 200
survdiffTerms .................................................. 200
survfit.survTerms ............................................ 202
t.sticky ......................................................... 202
tabDataset ...................................................... 203
table2html ..................................................... 204
tableRow ......................................................... 206
testGroupIndependence ....................................... 206
testSingleIndependence ...................................... 208
testSurvival .................................................... 209
testSurvivalCutoff ........................................... 210
textSuggestions ............................................... 211
toJSarray ......................................................... 212
traceInList .................................................... 212
transformData .................................................. 212
transformOptions ............................................. 213
transformValues .............................................. 213
trimWhitespace ................................................. 214
uniqueBy ......................................................... 215
updateClinicalParams ....................................... 215
updateFileBrowserInput ..................................... 216
vennEvents ..................................................... 216
wilcox .......................................................... 217
[.GEandAScorrelation ........................................ 218

Index 221
Description
Print startup message

Usage
.onAttach(libname, pkgname)

Arguments

libname Character: library name
pkgname Character: package name

Value
Startup message

addObjectAttrs Set attributes to an object

Description
Set attributes to an object

Usage
addObjectAttrs(object, ..., replace = TRUE)

Arguments

object Object
...	Named parameters to convert to attributes
replace Boolean: replace an attribute if already set?

Value
Object with attributes set

Examples
11 <- list(a="hey", b="there")
psychomics:::addObjectAttrs(11, "words"=2, "language"="English")
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

analysesTableSet

Set of functions to render differential analyses (plot and table)

Description

Set of functions to render differential analyses (plot and table)
Set up environment and redirect user to a page based on click information

Usage

analysesTableSet(
    session,  
    input,   
    output, 
    analysesType,  
    analysesID, 
    getAnalysesData, 
    getAnalysesFiltered, 
    setAnalysesFiltered, 
    getAnalysesSurvival, 
    getAnalysesColumns, 
    setAnalysesColumns,  
    getResetPaging,  
    setResetPaging
)
processClickRedirection(click, psi = NULL, survival = FALSE)

analysesPlotSet(
  session,
  input,
  output,
  analysesType,
  analysesID,
  getAnalysesData,
  getAnalysesFiltered,
  getAnalysesSurvival
)

**Arguments**

- `session`: Shiny session
- `input`: Shiny input
- `output`: Shiny output
- `analysesType`: Character: type of analyses (GE or PSI)
- `analysesID`: Character: identifier
- `getAnalysesData`: Function: get analyses data
- `getAnalysesFiltered`: Function: get filtered analyses data
- `setAnalysesFiltered`: Function: set filtered analyses data
- `getAnalysesSurvival`: Function: get survival data
- `getAnalysesColumns`: Function: get columns
- `setAnalysesColumns`: Function: set columns
- `getResetPaging`: Function: get toggle of reset paging
- `setResetPaging`: Function: set toggle of reset paging
- `click`: List: click information
- `psi`: Data frame or matrix: alternative splicing quantification
- `survival`: Boolean: redirect to survival page?

**Value**

NULL (function is only used to modify the Shiny session’s state or internal variables)
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**

```r
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` (function is only used to modify the Shiny session’s state or internal variables)

---

**appServer**  

**Server logic**

**Description**

Instructions to build the Shiny app.

**Usage**

```r
appServer(input, output, session)
analysesServer(input, output, session)
diffEventServer(ns, input, output, session, psi)
correlationServer(input, output, session)
diffExpressionServer(input, output, session)
diffExpressionEventServer(input, output, session)
diffExpressionTableServer(input, output, session)
```
appServer

    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)
    inclusionLevelsFilterServer(input, output, session)
    localDataServer(input, output, session)
    recountDataServer(input, output, session)
    groupsServer(input, output, session)
    helpServer(input, output, session)

Arguments

    input          Shiny input
    output         Shiny output
    session        Shiny session

Value

    NULL (function is only used to modify the Shiny session’s state or internal variables)
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

```r
appUI()

analysesUI(id, tab)

diffEventUI(id, ns, psi = TRUE)

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)
```
areSplicingEvents

Arguments

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

Value

HTML elements

Description
Check if string identifies splicing events

Usage

areSplicingEvents(char, data = NULL, num = 6)

Arguments

char Character vector
data Object containing event data
num Integer: number of elements to check

Value

TRUE if first elements are splicing events; FALSE, otherwise
### articleUI

*Return the interface to display an article*

<table>
<thead>
<tr>
<th>Description</th>
<th>Return the interface to display an article</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>articleUI(article)</td>
</tr>
<tr>
<td>Arguments</td>
<td>article: PubMed article</td>
</tr>
<tr>
<td>Value</td>
<td>HTML to render an article's interface</td>
</tr>
</tbody>
</table>

### assignColours

*Assign colours to groups*

<table>
<thead>
<tr>
<th>Description</th>
<th>Assign colours to groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>assignColours(new, groups = NULL)</td>
</tr>
<tr>
<td>Arguments</td>
<td>new: Matrix: groups to which colours will be assigned</td>
</tr>
<tr>
<td></td>
<td>groups: Matrix: groups to check which colours are already assigned</td>
</tr>
<tr>
<td>Value</td>
<td>Groups with an added column to state the colour</td>
</tr>
</tbody>
</table>
assignValuePerSubject  Assign average sample values to their corresponding subjects

Description
Assign average sample values to their corresponding subjects

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

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

Value
Values per subject

See Also
Other functions to analyse survival: getAttributesTime(), labelBasedOnCutoff(), optimalSurvivalCutoff(), plotSurvivalCurves(), plotSurvivalPvaluesByCutoff(), processSurvTerms(), survdiffTerms(), survfit.survTerms(), testSurvival()

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"))

# Match between subjects and samples
```r
class <- rep(paste("Subject", 1:3), 2)
names(class) <- colnames(psi)

# Assign PSI values to each subject based on the PSI of their samples
assignValuePerSubject(psi[3, ], class)
```

## 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

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

### Arguments

- **data**
  - Numeric, data frame or matrix: gene expression data or alternative splicing event quantification values (sample names are based on their `names` or `colnames`)
- **groups**
  - List of sample names or vector containing the group name per data value (read Details); if NULL or a character vector of length 1, data values are considered from the same group

### Value

HTML elements

## blendColours

**Blend two HEX colours**

### Description

Blend two HEX colours

### Usage

```r
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
**browseDownloadFolderInput**

**Value**

Character representing an HEX colour

**Source**

Code modified from [https://stackoverflow.com/questions/5560248](https://stackoverflow.com/questions/5560248)

**Examples**

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

---

**Description**

Browse download folder input

**Usage**

```r
browseDownloadFolderInput(id)
```

**Arguments**

- `id`  
  Character: element identifier

**Value**

HTML element in character

---

**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](https://github.com/daattali/advanced-shiny/blob/master/navigate-history)

**Usage**

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

```r
calculateInclusionLevels(
  eventType,
  junctionQuant,
  annotation,
  minReads = 10,
  onlyReturnASeventNames = FALSE
)
```

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

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 \( \frac{(C_x \times E_x) + (C_y \times E_y)}{E_x + E_y} \), where:

- \( C_x \) and \( C_y \) are the contributions of a variable to principal components \( x \) and \( y \)
- \( E_x \) and \( E_y \) 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


See Also

Other functions to analyse principal components: `performPCA()`, `plotPCA()`, `plotPCAvariance()`

Examples

```r
pca <- performPCA(USArrests)
calculateLoadingsContribution(pca)
```
**checkFileFormat**

*Checks the format of a file*

**Description**
Checks the format of a file

**Usage**

```r
ccheckFileFormat(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 matches the given format’s attributes

---

**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**

```r
ccheckFirebrowse(ns)
```

**Arguments**

- `ns`: Namespace function

**Value**

HTML elements
**checkGroupType**

Check type of groups within file

**Description**

Check type of groups within file

**Usage**

`checkGroupType(file)`

**Arguments**

- **file**: Character: file path

**Value**

Type of group: Samples, ASevents or NULL

---

**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**
Prepares 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?

**Value**
NULL (function is only used to modify the Shiny session’s state or internal variables)

---

**clusterICAsset**  
*Server logic for clustering ICA data*

**Description**
Server logic for clustering ICA data.

**Usage**
```
clusterICAsset(session, input, output)
```

**Arguments**
- `session` Shiny session
- `input` Shiny input
- `output` Shiny output

**Value**
NULL (function is only used to modify the Shiny session’s state or internal variables)
### clusterSet

**Description**

Server logic for clustering PCA data

**Usage**

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

**Arguments**

- `session`  
  Shiny session
- `input`  
  Shiny input
- `output`  
  Shiny output

**Value**

NULL (function is only used to modify the Shiny session’s state or internal variables)

### colourInputMod

**Description**

Modified colour input with 100% width

**Usage**

```r
colourInputMod(...)
```

**Arguments**

- `...`  
  Arguments passed on to `colourpicker::colourInput`
- `inputId`  
  The input slot that will be used to access the value.
- `label`  
  Display label for the control, or ‘NULL for no label.
- `value`  
  Initial value (can be a colour name or HEX code)
- `showColour`  
  Whether to show the chosen colour as text inside the input, as the background colour of the input, or both (default).
- `palette`  
  The type of colour palette to allow the user to select colours from. `square` (default) shows a square colour palette that allows the user to choose any colour, while `limited` only gives the user a predefined list of colours to choose from.
allowedCols A list of colours that the user can choose from. Only applicable when palette == "limited". The limited palette uses a default list of 40 colours if allowedCols is not defined. If the colour specified in value is not in the list, the default colour will revert to black.

allowTransparent If TRUE, enables a slider to choose an alpha (transparency) value for the colour. When a colour with opacity is chosen, the return value is an 8-digit HEX code.

returnName If TRUE, then return the name of an R colour instead of a HEX value when possible.

closeOnClick If TRUE, then the colour selection panel will close immediately after selecting a colour.

width The width of the input, e.g. "400px" or "100%"

Value

HTML elements

colSums,EList-method

Sum columns using an EList-class object

Description

Sum columns using an EList-class object

Usage

## S4 method for signature 'EList'
colSums(x, na.rm = FALSE, dims = 1)

Arguments

x an array of two or more dimensions, containing numeric, complex, integer or logical values, or a numeric data frame. For .colSums() etc, a numeric, integer or logical matrix (or vector of length m * n).

na.rm logical. Should missing values (including NaN) be omitted from the calculations?

dims integer: Which dimensions are regarded as ‘rows’ or ‘columns’ to sum over. For row*, the sum or mean is over dimensions dims+1, ...; for col* it is over dimensions 1:dims.

Value

Numeric vector with the sum of the columns
convertGeneIdentifiers

Convert gene identifiers

Description

Convert gene identifiers

Usage

convertGeneIdentifiers(
  annotation,
  genes,
  key = "ENSEMBL",
  target = "SYMBOL",
  ignoreDuplicatedTargets = TRUE
)

Arguments

annotation OrgDb with genome wide annotation for an organism or character with species
name to query OrgDb, e.g. "Homo sapiens"
genes Character: genes to be converted
key Character: type of identifier used, e.g. ENSEMBL; read ?AnnotationDbi::columns
target Character: type of identifier to convert to; read ?AnnotationDbi::columns
ignoreDuplicatedTargets Boolean: if TRUE, identifiers that share targets with other identifiers will not be
converted

Value

Character vector of the respective targets of gene identifiers. The previous identifiers remain other
identifiers have the same target (in case ignoreDuplicatedTargets = TRUE) or if no target was
found.

See Also

Other functions for gene expression pre-processing: filterGeneExpr(), normaliseGeneExpression(),
plotGeneExprPerSample(), plotLibrarySize(), plotRowStats()

Examples

# Use species name to automatically look for a OrgDb database
sp <- "Homo sapiens"
genes <- c("ENSG00000012048", "ENSG00000083093", "ENSG00000141510",
           "ENSG000000051180")
convertGeneIdentifiers(sp, genes)
convertGeneIdentifiers(sp, genes, key="ENSEMBL", target="UNIPROT")

# Alternatively, set the annotation database directly
ah <- AnnotationHub::AnnotationHub()
sp <- AnnotationHub::query(ah, c("OrgDb", "Homo sapiens"))[1]
columns(sp) # these attributes can be used to change the attributes

convertGeneIdentifiers(sp, genes)
convertGeneIdentifiers(sp, genes, key="ENSEMBL", target="UNIPROT")

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)
... Extra parameters passed to cor.test

Value

List of correlations where each element contains:

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

See Also

Other functions to correlate gene expression and alternative splicing: [.GEandAScorrelation()]
Examples

```r
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

```r
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 (function is only used to modify the Shiny session’s state or internal variables)

createDensitySparklines

Create density sparklines for inclusion levels

Description

Create density sparklines for inclusion levels
Usage
createDensitySparklines(
  data,
  events,
  areSplicingEvents = TRUE,
  groups = NULL,
  geneExpr = NULL,
  inputID = "sparklineInput"
)

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
- inputID: Character, identifier of input to get attributes of clicked event (Shiny only)

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
)
createGroup

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>Data frame</td>
</tr>
<tr>
<td>x</td>
<td>Character: name of the variable used for the X axis</td>
</tr>
<tr>
<td>y</td>
<td>Character: name of the variable used for the Y axis</td>
</tr>
<tr>
<td>params</td>
<td>List of parameters to pass to <code>geom_point()</code> related to most points</td>
</tr>
<tr>
<td>highlightX</td>
<td>Integer: region of points in X axis to highlight</td>
</tr>
<tr>
<td>highlightY</td>
<td>Integer: region of points in Y axis to highlight</td>
</tr>
<tr>
<td>highlightParams</td>
<td>List of parameters to pass to <code>geom_point()</code> related to highlighted points</td>
</tr>
<tr>
<td>selected</td>
<td>Integer: index of rows/points to be coloured</td>
</tr>
<tr>
<td>selectedParams</td>
<td>List of parameters to pass to <code>geom_point()</code> related to selected points</td>
</tr>
<tr>
<td>labelled</td>
<td>Integer: index of rows/points to be labelled</td>
</tr>
<tr>
<td>labelledParams</td>
<td>List of parameters to pass to <code>ggrepel::geom_label_repel</code> related to labelled points</td>
</tr>
<tr>
<td>xlim</td>
<td>Numeric: limits of X axis</td>
</tr>
<tr>
<td>ylim</td>
<td>Numeric: limits of Y axis</td>
</tr>
</tbody>
</table>

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

```r
createGroup(
  session,       
  input,         
  output,        
  id,            
  type,          
  selected = NULL,
  expr = NULL,   
  groupNames = NULL
)
```
createGroupByAttribute

**Arguments**

- **session**: Shiny session
- **input**: Shiny input
- **output**: Shiny output
- **id**: Character: identifier of the group selection
- **type**: Character: type of group to create
- **selected**: Character: selected item
- **expr**: Character: expression
- **groupNames**: Character: group names

**Value**

NULL (function is only used to modify the Shiny session’s state or internal variables)

---

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

---

**Description**

Elements are identified by their respective row name.

**Usage**

`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

**See Also**

Other functions for data grouping: `getGeneList()`, `getSampleFromSubject()`, `getSubjectFromSample()`, `groupPerElem()`, `plotGroupIndependence()`, `testGroupIndependence()`

**Examples**

```r
df <- data.frame(gender=c("male", "female"),
                 stage=paste("stage", c(1, 3, 1, 4, 2, 3, 2, 2)))
rownames(df) <- paste0("subject-", 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, 
    selected = NULL, 
    expr = NULL, 
    groupNames = NULL 
)
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
- **selected**: Character: selected item
- **expr**: Character: expression
- **groupNames**: Character: group names

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

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

Arguments

- **nrow**: Integer: row number
- **program**: Character: program used to get the junctions
- **event.type**: Character: event type
- **chromosome**: Character: chromosome
- **strand**: Character: positive-sense (+) or negative-sense (−) strand
- **id**: Character: event identifiers
createOptimalSurvData

Value
A data frame with the junctions coordinate names pre-filled with NA

Examples
psychomics::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
  timeStop Character: name of column containing ending time of the interval (only relevant
                   for interval censoring)
  match Matrix: match between samples and subjects
  patients Character: subject identifiers (only required if the clinical argument is not
                     handed)
  samples Character: samples to use when assigning values per subject (if NULL, all sam-
                   ples will be used)
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,
  groups = NULL,
  geneExpr = NULL,
  inputID = "sparklineInput",
  ...
)

Arguments

hc highchart object
data Character: HTML-formatted data series of interest
events Character: event identifiers
groups Character: name of the groups used for differential analyses
geneExpr Character: name of the gene expression dataset
inputID Character: identifier of input to get attributes of clicked event (Shiny only)
id Character: Shiny input identifier

Value

HTML element with sparkline data
customRowMeans

Calculate statistics for each row or column of a matrix

Description

Calculate statistics for each row or column of a matrix

Usage

customRowMeans(mat, na.rm = FALSE, fast = FALSE)
customRowMedians(mat, na.rm = FALSE, fast = FALSE)
customRowVars(mat, na.rm = FALSE, fast = FALSE)
customRowMins(mat, na.rm = FALSE, fast = FALSE)
customRowMaxs(mat, na.rm = FALSE, fast = FALSE)
customRowRanges(mat, na.rm = FALSE, fast = FALSE)
customColMedians(mat, na.rm = FALSE, fast = FALSE)

Arguments

mat Matrix
na.rm Boolean: remove missing values (NA)?
fast Boolean: use Rfast functions? They may return different results from R built-in functions

Value

Vector of selected statistic

Examples

df <- rbind("Gene 1"=c(3, 5, 7), "Gene 2"=c(8, 2, 4), "Gene 3"=c(9:11))
psichomics:::customRowMeans(df)
psichomics:::customRowVars(df, fast=TRUE)
diagramSplicingEvent  Prepare SVG diagram of alternative splicing events

**Description**

Prepare SVG diagram of alternative splicing events

**Usage**

```r
diagramSplicingEvent(  
  parsed,  
  type,  
  class = "pull-right",  
  style = NULL,  
  showText = TRUE,  
  showPath = TRUE,  
  showAlternative1 = TRUE,  
  showAlternative2 = TRUE,  
  constitutiveWidth = NULL,  
  alternativeWidth = NULL,  
  intronWidth = NULL,  
  constitutiveFill = "lightgray",  
  constitutiveStroke = "darkgray",  
  alternative1Fill = "#ffdb15",  
  alternative1Stroke = "#faa000",  
  alternative2Fill = "#caa06c",  
  alternative2Stroke = "#9d7039"
)
```

**Arguments**

- `parsed`  
  Alternative splicing event

- `type`  
  Character: alternative splicing event type

- `class`  
  Character: class of SVG parent tag

- `style`  
  Character: style of SVG parent tag

- `showText`  
  Boolean: display coordinates and length (if available)

- `showPath`  
  Boolean: display alternative splicing junctions

- `showAlternative1`  
  Boolean: show alternative exon 1 and respective splicing junctions and text?

- `showAlternative2`  
  Boolean: show alternative exon 2 and respective splicing junctions and text?  
  (only related with mutually exclusive exons)

- `constitutiveWidth`  
  Numeric: width of constitutive exon(s)
diffAnalyses

alternativeWidth  Numeric: width of alternative exon(s)
intronWidth       Numeric: width of intron’s representation
constitutiveFill  Character: fill colour of constitutive exons
constitutiveStroke Character: stroke colour of constitutive exons
alternative1Fill  Character: fill colour of alternative exon 1
alternative1Stroke Character: stroke colour of alternative exon 1
alternative2Fill  Character: fill colour of alternative exon 2
alternative2Stroke Character: stroke colour of alternative exon 2

Value

Diagrams per alternative splicing event in SVG

---

diffAnalyses  \hspace{1cm} \textit{Perform statistical analyses}

Description

Perform statistical analyses

Usage

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

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)

inputID    Character: identifier of input to get attributes of clicked event (Shiny only)

Details

The following statistical analyses may be performed simultaneously via 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 p-value adjustment methods are supported via 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

See Also

Other functions to perform and plot differential analyses: `plotDistribution()`

Examples

```r
# 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)
```
diffExpressionSet

Set of functions to perform differential analyses

Usage

diffExpressionSet(session, input, output)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>Shiny session</td>
</tr>
<tr>
<td>input</td>
<td>Shiny input</td>
</tr>
<tr>
<td>output</td>
<td>Shiny output</td>
</tr>
</tbody>
</table>

Value

NULL (function is only used to modify the Shiny session’s state or internal variables)

diffSplicingSet

Set of functions to perform differential analyses

Usage

diffSplicingSet(session, input, output)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>Shiny session</td>
</tr>
<tr>
<td>input</td>
<td>Shiny input</td>
</tr>
<tr>
<td>output</td>
<td>Shiny output</td>
</tr>
</tbody>
</table>

Value

NULL (function is only used to modify the Shiny session’s state or internal variables)
**disableTab**  
*Enable or disable a tab from the navbar*

**Description**
Enable or disable a tab from the navbar

**Usage**
```r
disableTab(tab)
```
```r
enableTab(tab)
```

**Arguments**
- `tab` Character: tab

**Value**
`NULL` (function is only used to modify the Shiny session’s state or internal variables)

---

**discardLowCoveragePSIvalues**  
*Remove alternative splicing quantification values based on coverage*

**Description**
Remove alternative splicing quantification values based on coverage

**Usage**
```r
discardLowCoveragePSIvalues(
  psi,
  minReads = 10,
  vasttoolsScoresToDiscard = c("VLOW", "N")
)
```

**Arguments**
- `psi` Data frame or matrix: alternative splicing quantification
- `minReads` Currently this argument does nothing
- `vasttoolsScoresToDiscard` Character: if you are using inclusion levels from VAST-TOOLS, filter the data based on quality scores for read coverage, e.g. use `vasttoolsScoresToDiscard = c("SOK", "OK", "LOW")` to only keep events with good read coverage (by default, events are not filtered based on quality scores); read [https://github.com/vastgroup/vast-tools](https://github.com/vastgroup/vast-tools) for more information on VAST-TOOLS quality scores
### discardOutsideSamplesFromGroups

**Description**

Discard grouped samples if not within a sample vector

**Usage**

```r
discardOutsideSamplesFromGroups(groups, samples, clean = FALSE)
```

**Arguments**

- `groups` Named list of samples
- `samples` Character: vector with all available samples
- `clean` Boolean: clean results?

**Value**

Groups without samples not found in samples

### display

**Description**

Display characters in the command-line

**Usage**

```r
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**

```r
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**

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

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

## End(Not run)
```

ensemblToUniprot  
*Convert from Ensembl to UniProt identifier*

**Description**

Convert from Ensembl to UniProt identifier

**Usage**

```r
ensemblToUniprot(protein)
```
escape

Arguments

protein  Character: Ensembl identifier

Value

UniProt protein identifier

See Also

Other functions to retrieve external information: `plotProtein()`, `plotTranscripts()`, `queryEnsemblByGene()`

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

---

**exportGroupsToFile**  
*Export groups to a file*

**Description**
Export groups to a file

**Usage**

```
exportGroupsToFile(groups, file, match = NULL)
```

**Arguments**

- **groups**: Matrix with groups
- **file**: Character: path to output file
- **match**: Match between elements within groups

**Value**
Saves groups to file
**export_highcharts**

Add an exporting feature to a highcharts object

**Description**

Add an exporting feature to a highcharts object

**Usage**

```r
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**

```r
fileBrowser(
  default = NULL,
  caption = NULL,
  multiple = FALSE,
  directory = FALSE
)
```

**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?
Details

Platform-dependent implementation:

- **Windows**: calls the `utils::choose.files` R function.
- **macOS**: uses AppleScript to display a folder selection dialogue. If `default = NA`, folder selection falls back to the default behaviour of the `choose folder` AppleScript command. Otherwise, paths are expanded with `path.expand()`.
- **Linux**: calls the `zenity` system command.

Value

A length one character vector, character NA if ‘Cancel’ was selected

Source

[https://github.com/wleepang/shiny-directory-input](https://github.com/wleepang/shiny-directory-input)
Arguments

- **id**: Character: input identifier
- **label**: Character: input label (if NULL, no labels are displayed)
- **infoContent**: Character: text to show as content of information
- **clearable**: Boolean: allow to clear selected file or directory?
- **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 pop-overs
- **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

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

HTML elements for a file browser input

Source

[https://github.com/wleepang/shiny-directory-input](https://github.com/wleepang/shiny-directory-input)

See Also

`updateFileBrowserInput()` and `prepareFileBrowser()`

---

`filterGeneExpr` *Filter genes based on their expression*

Description

Uses `filterByExpr` to determine genes with sufficiently large counts to retain for statistical analysis.
Usage

```
filterGeneExpr(
    geneExpr,
    minMean = 0,
    maxMean = Inf,
    minVar = 0,
    maxVar = Inf,
    minCounts = 10,
    minTotalCounts = 15
)
```

Arguments

- **geneExpr**: Data frame or matrix: gene expression
- **minMean**: Numeric: minimum of read count mean per gene
- **maxMean**: Numeric: maximum of read count mean per gene
- **minVar**: Numeric: minimum of read count variance per gene
- **maxVar**: Numeric: maximum of read count variance per gene
- **minCounts**: Numeric: minimum number of read counts per gene for a worthwhile number of samples (check `filterByExpr` for more information)
- **minTotalCounts**: Numeric: minimum total number of read counts per gene

Value

Boolean vector indicating which genes have sufficiently large counts

See Also

Other functions for gene expression pre-processing: `convertGeneIdentifiers()`, `normaliseGeneExpression()`, `plotGeneExprPerSample()`, `plotLibrarySize()`, `plotRowStats()`

Examples

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

# Add some genes with low expression
geneExpr <- rbind(geneExpr,
    lowReadGene1=c(rep(4:5, 10)),
    lowReadGene2=c(rep(5:1, 10)),
    lowReadGene3=c(rep(10:1, 10)),
    lowReadGene4=c(rep(7:8, 10)))

# Filter out genes with low reads across samples
geneExpr[filterGeneExpr(geneExpr), ]
```
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**
```r
filterGroups(vector, group, threshold = 1)
```

**Arguments**
- `vector`: Character: elements
- `group`: Character: respective group of each 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 as an attribute.

**Examples**
```r
# Removes groups with less than two elements
vec <- 1:6	names(vec) <- paste("sample", letters[1:6])
filterGroups(vec, c("A", "B", "B", "C", "D", "D"), threshold=2)
```

---

filterPSI  
*Filter alternative splicing quantification*

**Description**
Filter alternative splicing quantification

**Usage**
```r
filterPSI(
    psi,
    eventType = NULL,
    eventSubtype = NULL,
    minPSI = -Inf,
    maxPSI = Inf,
    minMedian = -Inf,
)```
filterPSI

Arguments

psi Data frame or matrix: alternative splicing quantification
eventType Character: filter data based on event type; check all event types available by using getSplicingEventTypes(psi), where psi is the alternative splicing quantification data; if eventType = NULL, events are not filtered by event type
eventSubtype Character: filter data based on event subtype; check all event subtypes available in your data by using unique(getSplicingEventData(psi)$subtype), where psi is the alternative splicing quantification data; if eventSubtype = NULL, events are not filtered by event subtype
minPSI Numeric: minimum PSI value
maxPSI Numeric: maximum PSI value
minMedian Numeric: minimum median PSI per splicing event
maxMedian Numeric: maximum median PSI per splicing event
minLogVar Numeric: minimum log10(PSI variance) per splicing event
maxLogVar Numeric: maximum log10(PSI variance) per splicing event
minRange Numeric: minimum PSI range across samples per splicing event
maxRange Numeric: maximum PSI range across samples per splicing event

Value

Boolean vector indicating which splicing events pass the thresholds

See Also

Other functions for PSI quantification: getSplicingEventTypes(), listSplicingAnnotations(), loadAnnotation(), plotRowStats(), quantifySplicing()

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"))
# Filter PSI
psi[filterPSI(psi, minMedian=0.05, maxMedian=0.95, minRange=0.15), ]
findASeventsFromGene  
_Find splicing events based on given genes_

**Description**

Find splicing events based on given genes

**Usage**

```r
findASeventsFromGene(psi, gene)
```

**Arguments**

- `psi`  
  Data frame or matrix: alternative splicing quantification
- `gene`  
  Character: gene

**Value**

Character vector containing alternative splicing events

---

findEventData  
_Look for event data in input_

**Description**

Check if event data can be found in `data` and then `event`. Event data has to be an object of class `eventData`

**Usage**

```r
findEventData(event = NULL, data = NULL)
```

**Arguments**

- `event`  
  Character: AS event that may contain event data in its attribute `eventData`
- `data`  
  Data frame or matrix: either event data or data containing event data in its attributes `rowData` or `eventData`

**Value**

Event data (or NULL if not found)
**Description**

File input for molecular data

**Usage**

geneExprFileInput(id, clearable = FALSE)

ASquantFileInput(id, clearable = FALSE)

junctionQuantFileInput(id, clearable = FALSE)

sampleInfoFileInput(id, clearable = FALSE)

subjectInfoFileInput(id, clearable = FALSE)

**Arguments**

- **id**  
  Character: identifier for gene expression input

- **clearable**  
  Boolean: allow to clear selected file or directory?

**Value**

HTML elements

---

**geneExprSurvSet  Logic set to perform survival analysis based on gene expression cutoffs**

**Description**

Logic set to perform survival analysis based on gene expression cutoffs

**Usage**

geneExprSurvSet(session, input, output)

**Arguments**

- **session**  
  Shiny session

- **input**  
  Shiny input

- **output**  
  Shiny output
geNormalisationFilteringInterface

*Interface to normalise and filter gene expression*

### Value

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

### getAttributesTime

*Get time values for given columns in a clinical dataset*

### Description

Get time values for given columns in a clinical dataset

### Usage

```r
getAttributesTime(
  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

See Also

Other functions to analyse survival: `assignValuePerSubject()`, `labelBasedOnCutoff()`, `optimalSurvivalCutoff()`, `plotSurvivalCurves()`, `plotSurvivalPvaluesByCutoff()`, `processSurvTerms()`, `survdiffTerms()`, `survfit survTerms()`, `testSurvival()`

Examples

```r
df <- data.frame(followup=c(200, 300, 400), death=c(NA, 300, NA))
rownames(df) <- paste("subject", 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

```r
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset</td>
<td>Character: data set name</td>
</tr>
<tr>
<td>category</td>
<td>Character: data category</td>
</tr>
<tr>
<td>matches</td>
<td>Vector of integers: clinical matches of dataset</td>
</tr>
</tbody>
</table>

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
Other functions to get and set global variables: getDifferentialExpression(), getDifferentialSplicing(), getGlobal(), getGroups(), getHighlightedPoints(), getSelectedDataPanel()

data
Get global data

Description
Get global data

Usage
data()
getDifferentialExpression

Get or set differential expression’ elements for a data category

Description

Get or set differential expression’ elements for a data category

Usage

generateExpression(category = getCategory())

generateExpressionFiltered(category = getCategory())

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

Value

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

getDifferentialExpressionSurvival(category = getCategory())

setDifferentialExpressionSurvival(survival, category = getCategory())

getDifferentialExpressionResetPaging(category = getCategory())

setDifferentialExpressionResetPaging(reset, category = getCategory())

getDifferentialExpressionColumns(category = getCategory())

setDifferentialExpressionColumns(columns, category = getCategory())

Arguments

category  Character: 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

Note

Needs to be called inside a reactive function

See Also

Other functions to get and set global variables: getClinicalMatchFrom(), getDifferentialSplicing(), getGlobal(), getGroups(), getHighlightedPoints(), getSelectedDataPanel()
Usage

getDifferentialSplicing(category = getCategory())

setDifferentialSplicing(differential, category = getCategory())

getDifferentialSplicingFiltered(category = getCategory())

setDifferentialSplicingFiltered(differential, category = getCategory())

getDifferentialSplicingSurvival(category = getCategory())

setDifferentialSplicingSurvival(survival, category = getCategory())

getDifferentialSplicingResetPaging(category = getCategory())

setDifferentialSplicingResetPaging(reset, category = getCategory())

getDifferentialSplicingColumns(category = getCategory())

setDifferentialSplicingColumns(columns, category = getCategory())

Arguments

category Character: 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

Note

Needs to be called inside a reactive function

See Also

Other functions to get and set global variables: getClinicalMatchFrom(). getDifferentialExpression(). getGlobal(). getGroups(). getHighlightedPoints(). getSelectedDataPanel()
getDownloadsFolder

Get the path to the Downloads folder

Description
Get the path to the Downloads folder

Usage
getDownloadsFolder()

Value
Path to Downloads folder

See Also
Other functions associated with TCGA data retrieval: getTCGAdataTypes(), isFirebrowseUp(), loadTCGAdata(), parseTCGAsampleTypes()
Other functions associated with GTEx data retrieval: getGtexDataTypes(), getGtexTissues(), loadGtexData()
Other functions associated with SRA data retrieval: loadSRAproject()

Examples
getDownloadsFolder()

getFirebrowseDateFormat
Returns the date format used by the FireBrowse API

Description
Returns the date format used by the FireBrowse API

Usage
getFirebrowseDateFormat()

Value
Named list with date formats from FireBrowse API
Examples

```r
format <- psychomics:::getFirebrowseDateFormat()
# date format to use in a query to FireBrowse API
format$query
# date format to parse a date in a response from FireBrowse API
format$response
```

geneList <- Get curated, literature-based gene lists

Description

Available gene lists:

- **Sebestyen et al., 2016**: 1350 genes encoding RNA-binding proteins, 167 of which are splicing factors

Usage

```r
getGeneList(genes = NULL)
```

Arguments

- `genes` Vector of characters: intersect lists with given genes (lists with no matching genes will not be returned)

Value

List of genes

See Also

Other functions for data grouping: `createGroupByAttribute()`, `getSampleFromSubject()`, `getSubjectFromSample()`, `groupPerElem()`, `plotGroupIndependence()`, `testGroupIndependence()`

Examples

```r
getGeneList()
```
**getGlobal**

Describe 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)
getASevents()
getAnnotationHub()
setAnnotationHub(ah)
getASevent()
setASevent(event, data = NULL)
getEvent()
setEvent(event, data = NULL)
```
getGenes()
getCategories()
getCategory()
setCategory(category)
getCategoryData()
getActiveDataset()
setActiveDataset(dataset)
getClinicalData(attrs = NULL)
getSubjectId()
getSubjectAttributes()
getSampleInfo()
setSampleInfo(value, category = getCategory())
getSampleId()
getSampleAttributes()
getJunctionQuantification(category = getCategory())
getGeneExpression(item = NULL, category = getCategory(), EList = FALSE)
setNormalisedGeneExpression(geneExpr, category = getCategory())
getInclusionLevels()
setInclusionLevels(incLevels, category = getCategory())
getInclusionLevelsSummaryStatsCache(category = getCategory())
setInclusionLevelsSummaryStatsCache(cache, category = getCategory())
getPCA(category = getCategory())
setPCA(pca, category = getCategory())
getICA(category = getCategory())
```r
setICA(ica, category = getCategory())
getCorrelation(category = getCategory())
setCorrelation(correlation, 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**

- `category` Character: data category
- `...` Arguments to identify a variable
- `sep` Character to separate identifiers
- `value` Value to attribute to an element
- `data` Matrix or data frame: alternative splicing information
- `name` Character: data table name
- `auto` Boolean: enable automatic navigation of browser history?
- `integer` Integer: value of the setting
- `ah` AnnotationHub
- `event` Character: alternative splicing event
- `dataset` Character: dataset name
- `attrs` Character: name of attributes to retrieve (if NULL, the whole dataset is returned)
- `item` Character: name of specific item to retrieve (if NULL, the whole list is returned)
- `EList` Boolean: return gene expression datasets as EList if possible or as data frames?
- `geneExpr` Data frame or matrix: normalised gene expression
- `incLevels` Data frame or matrix: inclusion levels
getGroups

Description
Get or set groups

Usage

```r
cache List of summary statistics
pca prcomp object (principal component analysis)
cia Object containing independent component analysis
correlation prcomp object (correlation analyses)
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

Note
Needs to be called inside a reactive function

See Also
Other functions to get and set global variables: `getClinicalMatchFrom()`, `getDifferentialExpression()`, `getDifferentialSplicing()`.

```
getGtexDataTypes

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)?
- **category**: Character: 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

Note

Needs to be called inside a reactive function

See Also

Other functions to get and set global variables: getClinicalMatchFrom(), getDifferentialExpression(), getDifferentialSplicing(), getGlobal(), getHighlightedPoints(), getSelectedDataPanel()

getGtxDataTypes 
Get GTEx data information

Description

Get GTEx data information

Usage

getGtxDataTypes()
getGtxReleases()

Value

GTEx data information

See Also

Other functions associated with GTEx retrieval: getDownloadsFolder(), getGtxTissues(), loadGtxData()

Examples

getGtxDataTypes()
getGtxReleases()
getGtexDataURL  
*Get links to download GTEx data*

**Description**

Get links to download GTEx data

**Usage**

```r
getGtexDataURL(
  release,
  domain = "https://storage.googleapis.com",
  offline = FALSE
)
```

**Arguments**

- `release`: Numeric: GTEx data release
- `domain`: Character: GTEx data storage domain
- `offline`: Boolean: simulate offline behaviour

**Value**

Character with URLs to download GTEx data

---

getGtexTissues  
*Get GTEx tissues from given GTEx sample attributes*

**Description**

Get GTEx tissues from given GTEx sample attributes

**Usage**

```r
getGtexTissues(folder = getDownloadsFolder(), release = getGtexReleases()[[1]])
```

**Arguments**

- `folder`: Character: folder containing data
- `release`: Numeric: GTEx data release to load

**Value**

Character: available tissues
getHidden

See Also
Other functions associated with GTEx data retrieval: `getDownloadsFolder()`, `getGtexDataTypes()`, `loadGtexData()`

Examples

```r
## Not run:
getGtexTissues()

## End(Not run)
```

---

**getHidden**

Get or set hidden globally accessible elements

---

### Description

Get or set hidden globally accessible elements

### Usage

```r
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
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

Note

Needs to be called inside a reactive function

See Also

Other functions to get and set global variables: getClinicalMatchFrom(), getDifferentialExpression(), getDifferentialSplicing(), getGlobal(), getGroups(), getSelectedDataPanel()

generateColumn = function(test, id)
{
  if (!is.null(test[[id]]))
    return(test[[id]])
  else
    return(NULL)
}

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
getSampleFromSubject

Get samples matching the given subjects

Usage

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

Description

Get samples matching the given subjects

Usage

getSampleFromSubject(
  table, by = NULL, toNumeric = FALSE)

Arguments

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

Value

Processed data matrix

Examples

event <- read.table(text = "ABC123 + 250 300 350
DEF456 - 900 800 700")

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

event <- psychomics::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"])

**getSelectedDataPanel**

Get or set selected panel in data section

**Description**

Get or set selected panel in data section

**Usage**

getSelectedDataPanel()

setSelectedDataPanel(id)

**Value**

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

**Arguments**

- **patients**: Character or list of characters: subject identifiers
- **samples**: Character: sample identifiers
- **clinical**: Data frame or matrix: clinical dataset
- **rm.NA**: Boolean: remove missing values?
- **match**: Integer: vector of subject index with the sample identifiers as name to save time (optional)
- **showMatch**: Boolean: show matching subject index?

**Value**

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

**See Also**

Other functions for data grouping: `createGroupByAttribute()`, `getGeneList()`, `getSubjectFromSample()`, `groupPerElem()`, `plotGroupIndependence()`, `testGroupIndependence()`

**Examples**

```r
subjects <- c("GTEX-ABC", "GTEX-DEF", "GTEX-GHI", "GTEX-JKL", "GTEX-MNO")
samples <- paste(subjects, "-sample")
clinical <- data.frame(samples=samples)
rownames(clinical) <- subjects
getSampleFromSubject(subjects[c(1, 4)], samples, clinical)
```
getServerFunctions

Note

Needs to be called inside a reactive function

See Also

Other functions to get and set global variables: `getClinicalMatchFrom()`, `getDifferentialExpression()`, `getDifferentialSplicing()`, `getGlobal()`, `getGroups()`, `getHighlightedPoints()`

---

getServerFunctions  Matches server functions from a given loader

Description

Matches server functions from a given loader

Usage

getAddressFunctions(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 and finally the 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

getAddressSplicingEventCoordinates(type, sorting = FALSE)
**getSplicingEventFromGenes**

**Arguments**
- **type** Character: alternative splicing event type
- **sorting** Boolean: get coordinates used for sorting and comparison between different programs?

**Value**
Coordinates of interest according to the alternative splicing event type

---

**getSplicingEventData**  
*Get splicing event information for given alternative splicing quantification data*

**Description**
Get splicing event information for given alternative splicing quantification data

**Usage**

```r
getSplicingEventData(psi)
```

**Arguments**
- **psi** Matrix or data frame: alternative splicing quantification data

**Value**
Matrix or data frame containing splicing event information for alternative splicing events in psi (if available)

---

**getSplicingEventFromGenes**  
*Get alternative splicing events from genes or vice-versa*

**Description**
Get alternative splicing events from genes or vice-versa

**Usage**

```r
getSplicingEventFromGenes(genes, ASevents, data = NULL)
getGenesFromSplicingEvents(ASevents, data = NULL)
```
getSplicingEventTypes

Arguments
- genes: Character: gene symbols (or TCGA-styled gene symbols)
- ASevents: Character: alternative splicing events
- data: Matrix or data frame: alternative splicing information

Details
A list of alternative splicing events is required to run getSplicingEventFromGenes

Value
Named character containing alternative splicing events or genes and their respective genes or alternative splicing events as names (depending on the function in use)

Examples
```r
ASevents <- c("SE_1+_201763003_201763300_20176374_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")

# Get splicing events from genes
matchedASevents <- getSplicingEventFromGenes(genes, ASevents)

# Names of matched events are the matching input genes
names(matchedASevents)

# Get genes from splicing events
matchedGenes <- getGenesFromSplicingEvents (ASevents)

# Names of matched genes are the matching input alternative splicing events
names(matchedGenes)
```

getSplicingEventTypes (Get supported splicing event types)

Description
Get supported splicing event types

Usage
getSplicingEventTypes(psi = NULL, acronymsAsNames = FALSE)
Arguments

psi  Data frame or matrix: alternative splicing quantification data
acronymsAsNames  Boolean: return acronyms as names?

Value

Named character vector with splicing event types

See Also

Other functions for PSI quantification: filterPSI(), listSplicingAnnotations(), loadAnnotation(), plotRowStats(), quantifySplicing()

Examples

getSplicingEventTypes()

gSubjectFromSample  Get subjects from given samples

Description

Get subjects from given samples

Usage

gSubjectFromSample(sampleId, patientId = NULL, na = FALSE, sampleInfo = NULL)

Arguments

sampleId  Character: sample identifiers
patientId  Character: subject identifiers to filter by (optional; if a matrix or data frame is given, its rownames will be used to infer the subject identifiers)
na  Boolean: return NA for samples with no matching subjects
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: subject identifiers corresponding to the given samples

See Also

Other functions for data grouping: createGroupByAttribute(), getGeneList(), getSampleFromSubject(), groupPerElem(), plotGroupIndependence(), testGroupIndependence()
getTCGAdatasTypes

Examples

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

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

---

**getTCGAdatasTypes**  
*Get available parameters for TCGA data*

**Description**

Parameters obtained via FireBrowse

**Usage**

```r
getTCGAdatasTypes()
getTCGAdates()
getTCGAcohorts(cohort = NULL)
```

**Arguments**

- **cohort**: Character: filter results by cohorts (optional)

**Value**

Parsed response

**See Also**

Other functions associated with TCGA data retrieval:  
`getDownloadsFolder()`, `isFirebrowseUp()`,  
`loadTCGAdatas()`, `parseTCGAsampleTypes()`

**Examples**

```r
getTCGAdatasTypes()
if (isFirebrowseUp()) getTCGAdates()
if (isFirebrowseUp()) getTCGAcohorts()
```
**getUiFunctions**

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

**Description**

Matches user interface (UI) functions from a given loader

**Usage**

```r
getiFunctions(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 and finally the 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**

```r
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)
areMultipleExonsValid allows to consider runs of exons (i.e. sequences where exon occurs consecutively) as valid when comparing based on the validator. For example, if validator = c("gene", "mRNA", "exon") and areMultipleExonsValid = FALSE, the event c("gene", "mRNA", "exon", "exon") is not valid as it has one additional exon. If areMultipleExonsValid = TRUE, the same event would be valid.

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 mRNA 17233 18061 . - .
chr1 SE exon 17233 17368 . - .
chr1 SE exon 17915 18061 . - .
chr1 SE mRNA 17233 18061 . - .
chr1 SE exon 17233 17368 . - .
chr1 SE exon 17915 18061 . - .
"
psichomics:::getValidEvents(event, validator)
ggplotTooltip

Create the interface for the tooltip of a plot

df = NULL,
x = NULL,
y = NULL,
eventData = NULL
)

Arguments

input  Shiny input
output  Shiny output
id  Character: identifier
plot  Character: plot expression (if NULL, no plots are rendered)
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
eventData  Alternative splicing event information (if available)

Value
NULL (function is only used to modify the Shiny session’s state or internal variables)

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

Description
Create the interface for the tooltip of a plot

Usage
ggplotTooltip(df, hover, x, y, eventData = NULL)

Arguments

df  Data frame
hover  Mouse hover information for a given plot as retrieved from hoverOps
text
x  Character: name of the variable used for the X axis
y  Character: name of the variable used for the Y axis
eventData  Alternative splicing event information (if available)
### ggplotUI

**Value**

HTML elements

---

**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, AEvent = FALSE)`

**Arguments**

- **id**  
  Character: input identifier
- **placeholder**  
  Character: input placeholder
- **AEvent**  
  Boolean: select alternative splicing events?

**Value**

HTML element for a global selectize input
groupByAttribute  

Data grouping interface

Description

Data grouping interface

Usage

groupByAttribute(ns, cols, id, example)
groupByPreMadeList(ns, data, id)
groupById(ns, id)
groupByExpression(ns, id)
groupByGrep(ns, cols, id)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns</td>
<td>Namespace function</td>
</tr>
<tr>
<td>cols</td>
<td>Character or list: name of columns to show</td>
</tr>
<tr>
<td>id</td>
<td>Character: identifier</td>
</tr>
<tr>
<td>example</td>
<td>Character: text to show as an example</td>
</tr>
<tr>
<td>data</td>
<td>List: list of groups with elements</td>
</tr>
</tbody>
</table>

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**: Shiny input
- **output**: Shiny output
- **session**: Shiny session
- **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**

```r
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**

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

- **groups**: List of integers: groups of elements
- **elem**: Character: all elements available
- **outerGroupName**: Character: name to give to outer group (if NULL, only show elements matched to their respective groups)

Value

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

See Also

Other functions for data grouping: `createGroupByAttribute()`, `getGeneList()`, `getSampleFromSubject()`, `getSubjectFromSample()`, `plotGroupIndependence()`, `testGroupIndependence()`

Examples

```r
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

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

Arguments

- **input**: Shiny input
- **output**: Shiny output
- **session**: Shiny session

Value

NULL (function is only used to modify the Shiny session’s state or internal variables)
**hchart.survfit**  
*Plot survival curves*

**Description**

Plot survival curves

**Usage**

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

**Arguments**

- `object`: survfit object as returned from `survfit.survTerms()` 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?
- `symbol`: Symbol to use as marker
- `markerColor`: Colour of the marker; if `NULL`, the respective colour of each series are used
- `ranges`: Plot interval ranges?
- `rangesOpacity`: Opacity of the interval ranges

**Value**

`highchart` 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,
  color = NULL,
  ...
)

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?
color          Character: series colour
...            Arguments passed on to highcharter::hc_add_series
**HTMLfast**

Faster version of shiny::HTML

**Description**

Faster version of shiny::HTML

**Usage**

```r
text

HTMLfast(text)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Character: text</td>
</tr>
</tbody>
</table>

**Value**

HTML element

---

**importGroupsFrom**

Import groups from a file

**Description**

Import groups from a file

**Usage**

```r
coded_example

importGroupsFrom(file, uniqueElems = NULL, matchingElems = NULL, match = NULL, type = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>Character: path to file</td>
</tr>
<tr>
<td>uniqueElems</td>
<td>Character: vector of unique elements (samples or alternative splicing events)</td>
</tr>
<tr>
<td>matchingElems</td>
<td>Character: vector of matching elements (subjects or genes)</td>
</tr>
<tr>
<td>match</td>
<td>Match between elements within groups</td>
</tr>
</tbody>
</table>
inclusionLevelsInterface

Value
Matrix with groups

inclusionLevelsFilterInterface

Interface to filter alternative splicing

Description
Interface to filter alternative splicing

Usage
inclusionLevelsFilterInterface(ns)

Arguments
ns Namespace function

Value
HTML elements

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
buttonIcon Character: button icon
buttonId Character: button identifier
id Character: identifier
type Character: type of alert (error or warning)
bigger Boolean: wrap the description in a h4 tag?

Value

HTML elements
insideFile

Get psychomics file inside a given directory

Description

Get psychomics 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
isFile

Check if files exist

Description
Check if files exist

Usage
isFile(files)

Arguments
files Character: vector of filepaths to check

Value
Boolean vector stating whether each file exists or not

isFirebrowseUp

Check if FireBrowse API is running

Description
Check if FireBrowse API is running

Usage
isFirebrowseUp()

Value
Invisible TRUE if the FireBrowse API is working; otherwise, raises a warning with the status code and a brief explanation.

See Also
Other functions associated with TCGA data retrieval: getDownloadsFolder(), getTCGADATATypes(), loadTCGADATA(), parseTCGASampleTypes()

Examples
isFirebrowseUp()
### isRStudioServer

**Check if running in RStudio Server**

**Description**

Check if running in RStudio Server

**Usage**

```r
isRStudioServer()
```

**Value**

Boolean stating whether running in RStudio Server

### joinEventsPerType

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

**Description**

Full outer join all given events based on select columns

**Usage**

```r
joinEventsPerType(events, types = NULL)
```

**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

---

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
- `gte`: Boolean: test using greater than or equal than cutoff (TRUE) or less than or equal than cutoff (FALSE)?
leveneTest

Description

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

Usage

leveneTest(x, g, centers = median)

Arguments

x Numeric vector or list of numeric vectors: non-numeric elements of a list will be coerced with a warning
g Vector or factor: groups of elements in x (ignored with a warning if x is a list)
centers Function used to calculate how much values spread; for instance, median (default) or mean

Details

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

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)
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)
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

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

Arguments

- ...: Custom annotation loaded

Value

Named character vector with splicing annotation files available

Examples

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

---

**listSplicingAnnotations**

*List alternative splicing annotations*

---

Description

List alternative splicing annotations
Usage

```r
listSplicingAnnotations(
    species = NULL,
    assembly = NULL,
    date = NULL,
    cache = getAnnotationHubOption("CACHE"),
    group = FALSE
)
```

Arguments

- **species**: Character: filter results by species (regular expression)
- **assembly**: Character: filter results by assembly (regular expression)
- **date**: Character: filter results by date (regular expression)
- **cache**: Character: path to AnnotationHub cache (used to load alternative splicing event annotation)
- **group**: Boolean: group values based on data provider?

Value

Named character vector with splicing annotation names

See Also

Other functions for PSI quantification: `filterPSI()`, `getSplicingEventTypes()`, `loadAnnotation()`, `plotRowStats()`, `quantifySplicing()`

Examples

```r
listSplicingAnnotations() # Return all alternative splicing annotations
listSplicingAnnotations(assembly="hg19") # Search for hg19 annotation
listSplicingAnnotations(assembly="hg38") # Search for hg38 annotation
listSplicingAnnotations(date="201(7|8)") # Search for 2017 or 2018 annotation
```

loadAnnotation

Load alternative splicing annotation from AnnotationHub

Description

Load alternative splicing annotation from AnnotationHub

Usage

```r
loadAnnotation(annotation, cache = getAnnotationHubOption("CACHE"))
```
**Arguments**

- `annotation` Character: annotation to load
- `cache` Character: path to AnnotationHub cache (used to load alternative splicing event annotation)

**Value**

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

**See Also**

Other functions for PSI quantification: `filterPSI()`, `getSplicingEventTypes()`, `listSplicingAnnotations()`, `plotRowStats()`, `quantifySplicing()`

**Examples**

```r
human <- listSplicingAnnotations(species="Homo sapiens")[[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**

\[
\text{loadBy}(\text{loader}, \text{FUN})
\]

**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**

Instructions to build the Shiny app

**Usage**

\[
\text{loadCustomSplicingAnnotationSet}(\text{session}, \text{input}, \text{output})
\]

**Arguments**

- **session**: Shiny session
- **input**: Shiny input
- **output**: Shiny output

**Value**

\text{NULL} (function is only used to modify the Shiny session’s state or internal variables)
###.loadedDataModal

**Warn user about loaded data**

**Description**

Warn user about loaded data

**Usage**

```r
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

**Load file based on its format**

**Description**

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

**Usage**

```r
loadFile(
  file,
  formats = loadFileFormats(),
  ..., 
  verbose = FALSE,
  multiple = FALSE 
)
```

**Arguments**

- `file`: Character: file to parse
- `formats`: List of file formats to check
- `...`: Extra parameters passed to `fread`
- `verbose`: Boolean: detail steps while parsing
- `multiple`: Boolean: expect more than one file?
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.

loadFileFormats  Load supported file formats

Description
Load supported file formats.

Usage
loadFileFormats()

Value
Supported file formats.

loadFirebrowseFolders  Load FireBrowse folders

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

Usage
loadFirebrowseFolders(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**

Instructions to build the Shiny app

**Usage**

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

**Arguments**

- `session` Shiny session
- `input` Shiny input
- `output` Shiny output

**Value**

`NULL` (function is only used to modify the Shiny session’s state or internal variables)

---

loadGtexData  
*Download and load GTEx data*

**Description**

Download and load GTEx data

**Usage**

```r
loadGtexData(
  folder = getDownloadsFolder(),
  data = getGtexDataTypes(),
  tissue = NULL,
  release = getGtexReleases()[[1]],
  progress = TRUE
)
```

**Arguments**

- `folder` Character: folder containing data
- `data` Character: data types to load (see getGtexDataTypes)
- `tissue` Character: tissues to load (if NULL, load all); tissue selection may speed up data loading
- `release` Numeric: GTEx data release to load
- `progress` Boolean: display progress?
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?

Value

NULL (function is only used to modify the Shiny session’s state or internal variables)

Examples

## Not run:
# Download and load all available GTEx data
data <- loadGtexData()

# Download and load only junction quantification and sample info from GTEx
goGtexDataTypes()
data <- loadGtexData(data=c("sampleInfo", "junctionQuant"))

# Download and load only data for specific tissues
goGtexTissues()
data <- loadGtexData(tissue=c("Stomach", "Small Intestine"))

# Download and load data from a specific GTEx data release
data <- loadGtexData(tissue=c("Stomach", "Small Intestine"), release=7)

## End(Not run)
loadGtexFile  

Load GTEx file

Description
Load GTEx file

Usage
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",
  verbose = FALSE
)

Arguments
- folder: Character: path to folder or ZIP archive
- ignore: Character: skip folders and filenames that match the expression
- name: Character: name
- verbose: Boolean: print steps?
loadRequiredData

Value
List of data frames from valid files

See Also
Other functions to load data: loadGtexData(), loadSRAproject(), loadTCGAdataset()

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 (function is only used to modify the Shiny session's state or internal variables)
Examples

```r
## Not run:
if (shiny::isRunning()) {
  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

Instructions to build the Shiny app

Usage

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

Arguments

- `session`: Shiny session
- `input`: Shiny input
- `output`: Shiny output

Value

`NULL` (function is only used to modify the Shiny session’s state or internal variables)

loadSRAproject

*Download and load SRA projects via recount2*

Description

Download and load SRA projects via [*recount2*](https://jhubiostatistics.shinyapps.io/recount/recount2)

Usage

```r
loadSRAproject(project, outdir = getDownloadsFolder())
```
loadTCGAdata

Arguments

project Character: SRA project identifiers (check \texttt{recount\_abstract})
outdir Character: directory to store the downloaded files

Value

List with loaded projects

See Also

Other functions associated with SRA data retrieval: \texttt{getDownloadsFolder()}  
Other functions to load data: \texttt{loadGtexData()}, \texttt{loadLocalFiles()}, \texttt{loadTCGAdata()}

Examples

```r
## Not run:
View(recount::recount\_abstract)
sra <- loadSRAProject("SRP053101")
names(sra)
names(sra[[1]])
## End(Not run)
```

loadTCGAdata  \hspace{1cm} Download and process TCGA data

Description

TCGA data obtained via \texttt{FireBrowse}

Usage

```r
loadTCGAdata(
  folder = getDownloadsFolder(),
  data = c("clinical", "junction\_quantification", "RSEM\_genes"),
  exclude = c(".aux.", ".mage\_tab.", "MANIFEST.txt"),
  ...,
  download = TRUE
)
```

Arguments

folder Character: directory to store the downloaded archives (by default, saves to \texttt{getDownloadsFolder()})
data Character: data to load (see \texttt{getTCGADataTypes()})
exclude Character: files and folders to exclude from downloading and from loading into \texttt{R} (by default, exclude files containing \texttt{.aux.}, \texttt{.mage\_tab.} and \texttt{MANIFEST.TXT})
Arguments passed on to `queryFirebrowseData`

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 (optional)

casecontrol String: column used to sort the data (by default, sort by cohort)

download Boolean: download missing files

Value

A list with the loaded data, unless required files are unavailable and `download = FALSE` (if so, it returns the URL of files to download)

See Also

Other functions associated with TCGA data retrieval: `getDownloadsFolder()`, `getTCGAdatatypes()`, `isFirebrowseUp()`, `parseTCGAsampleTypes()`

Other functions to load data: `loadGtexData()`, `loadLocalFiles()`, `loadSRAproject()`

Examples

```r
getTCGAcohorts()
getTCGAdatatypes()
## Not run:
loadTCGAdata(cohort = "ACC", data_type = "Clinical")

## End(Not run)
```

---

loadTCGAsampleMetadata

Prepare TCGA sample metadata from loaded datasets

Description

If no TCGA datasets apply, the input is returned
matchGroupASeventsAndGenes

**Usage**

loadTCGAsampleMetadata(data)

**Arguments**

data List of list of data frames

**Value**

List of list of data frames

---

matchGroupASeventsAndGenes

*Match AS events and genes in a group*

**Description**

Match AS events and genes in a group

**Usage**

matchGroupASeventsAndGenes(id, group, ASevents)

**Arguments**

id Character: identifier

group Data frame: group

**Value**

Data frame with groups containing matching elements

---

matchGroupSubjectsAndSamples

*Match subjects and samples in a group*

**Description**

Match subjects and samples in a group

**Usage**

matchGroupSubjectsAndSamples(id, group)
Arguments

id Character: identifier

Value
Data frame with groups containing matching elements

matchSplicingEventsWithGenes

Match splicing events with respective genes

Description
Match splicing events with respective genes

Usage
matchSplicingEventsWithGenes(ASevents, data = NULL)

Arguments

ASevents Character: alternative splicing events to be matched
data Matrix or data frame: alternative splicing information

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?
**navSelectize**

**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**

```r
callNavSelectize(id, label, placeholder = label, ASevent = FALSE)
```

**Arguments**

- `id` Character: input identifier
- `label` Character: input label
- `placeholder` Character: input placeholder
- `ASevent` Boolean: select alternative splicing events?

**Value**

HTML element to be included in a navigation bar

---

**normaliseGeneExpression**

*Filter and normalise gene expression*

**Description**

Gene expression is filtered and normalised in the following steps:

- Filter gene expression;
- Normalise gene expression with `calcNormFactors`;
- If `performVoom = FALSE`, compute counts per million (CPM) using `cpm` and log2-transform values if `log2transform = TRUE`;
- If `performVoom = TRUE`, use `voom` to compute log2-CPM, quantile-normalise (if `method = "quantile"`) and estimate mean-variance relationship to calculate observation-level weights.
normaliseGeneExpression

Usage

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

normalizeGeneExpression(
  geneExpr,
  geneFilter = NULL,
  method = "TMM",
  p = 0.75,
  log2transform = TRUE,
  priorCount = 0.25,
  performVoom = FALSE
)

Arguments

geneExpr: Matrix or data frame: gene expression
geneFilter: Boolean: filtered genes (if NULL, skip filtering)
method: Character: normalisation method, including TMM, RLE, upperquartile, none or quantile (see Details)
p: numeric value between 0 and 1 specifying which quantile of the counts should be used by method="upperquartile".
log2transform: Boolean: perform log2-transformation?
priorCount: Average count to add to each observation to avoid zeroes after log-transformation
performVoom: Boolean: perform mean-variance modelling (using voom)?

Details

edgeR::calcNormFactors will be used to normalise gene expression if method is TMM, RLE, upperquartile or none. If performVoom = TRUE, voom will only normalise if method = "quantile".

Available normalisation methods:

• **TMM** is recommended for most RNA-seq data where more than half of the genes are believed not differentially expressed between any pair of samples;
• **RLE** calculates the median library from the geometric mean of all columns and the median ratio of each sample to the median library is taken as the scale factor;
• **upperquartile** calculates the scale factors from a given quantile of the counts for each library, after removing genes with zero counts in all libraries;
• **quantile** forces the entire empirical distribution of each column to be identical (only performed if performVoom = TRUE).
operateOnGroups

Value

Filtered and normalised gene expression

See Also

Other functions for gene expression pre-processing: `convertGeneIdentifiers()`, `filterGeneExpr()`, `plotGeneExprPerSample()`, `plotLibrarySize()`, `plotRowStats()`

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
type Character: type of group where set operations are to be performed
sharedData Shiny app’s global variable

Value

NULL (function is only used to modify the Shiny session’s state or internal variables)
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 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 = NULL,  # Character: name of column containing ending time of the interval (only relevant for interval censoring)
  followup = "days_to_last_followup",  # Character: name of column containing follow up time
  session,    # Shiny session (only used for the visual interface)
  filter = TRUE,  # Boolean or numeric: elements to use (all are used by default)
  survTime = NULL,  # survTime object: times to follow up, time start, time stop and event (optional)
  lower = NULL,  # Bounds in which to search (if NULL, bounds are set to lower = 0 and upper = 1 if all data values are within that interval; otherwise, lower = min(data, na.rm = TRUE) and upper = max(data, na.rm = TRUE))
  upper = 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 are used 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, bounds are set to lower = 0 and upper = 1 if all data values are within that interval; otherwise, lower = min(data, na.rm = TRUE) and upper = max(data, na.rm = TRUE)).
Value

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

See Also

Other functions to analyse survival: `assignValuePerSubject()`, `getAttributesTime()`, `labelBasedOnCutoff()`, `plotSurvivalCurves()`, `plotSurvivalPvaluesByCutoff()`, `processSurvTerms()`, `survdiffTerms()`, `survfit.survTerms()`, `testSurvival()`

Examples

```r
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)
```

See Also

Other functions to analyse survival: `assignValuePerSubject()`, `getAttributesTime()`, `labelBasedOnCutoff()`, `plotSurvivalCurves()`, `plotSurvivalPvaluesByCutoff()`, `processSurvTerms()`, `survdiffTerms()`, `survfit.survTerms()`, `testSurvival()`

Examples

```r
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)
```
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("subject", 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
Parse file according to its format

Description
Parse file according to its format

Usage
parseFile(format, file, ..., verbose = FALSE)

Arguments
format Environment: format of the file
file Character: file to load
... Extra parameters passed to fread
verbose Boolean: detail step while parsing?

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

Value
Data frame with the loaded file

Query the FireBrowse API for metadata

Description
Query the FireBrowse API for metadata

Usage
parseFirebrowseMetadata(type, ...)

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

Value
List with parsed response
Examples

```r
psichomics::parseFirebrowseMetadata("Dates")
pseudomics::parseFirebrowseMetadata("Centers")
pseudomics::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**

```r
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**

```r
# MATS event (alternative 3' splice site)
event <- read.table(text = "
  2 ENSG000001166012 TAF1D chr11 - 93466515 93466671 93466515 93466563 93467790 93467826
  5 ENSG000001166012 TAF1D chr11 - 93466515 93466671 93466515 93466585 93467790 93467826
  6 ENSG000001166012 TAF1D chr11 - 93466515 93466585 93466515 93466563 93467790 93467826
"
)
pseudomics::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**

```r
parseMatsGeneric(junctions, strand, coords, plus_pos, minus_pos)
parsMatsSE(junctions, strand)
parsMatsMXE(junctions, strand)
parsMatsRI(junctions, strand)
parsMatsA3SS(junctions, strand)
parsMatsA5SS(junctions, strand)
parsMatsAFE(junctions, strand)
parsMatsALE(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** (retained intron)
- **A5SS** (alternative 5’ splice site)
- **A3SS** (alternative 3’ splice site)
- **AFE** (alternative first exon)
- **ALE** (alternative last exon)

You can use `parsMatsGeneric` 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")
plus <- c(1:6)
minus <- c(2:1, 6:3)
psychomics:::parseMatsGeneric(junctions, strand = "+", coords, plus, minus)

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

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

# Parse retained intron event
junctions <- read.table(text = "15929853 15932100 15929853 15930016 15930687 15932100")
pseudochromatic:::parseMatsRI(junctions, strand = "+")

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

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

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

# Parse alternative last exon event
junctions <- read.table(text = "111858645 111858828 111851063 111851921 111850441 111850543")
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 . - .
"
)
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 matches)

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
validator
eventType
coord
plusIndex
minusIndex
strand

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

Details

The following event types are available to be parsed:

- **SE** (exon skipping)
- **MXE** (mutually exclusive exon)
- **RI** (retained intron)
• **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**

```r
# 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 exon 16854 18061 . - .
  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)

# retained intron (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 17742 . - .
  chr1 RI exon 17601 17742 . - .
)
psichomics::parseMisoRI(event)
```
# alternative 5' splice site (A5SS) event
```r
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
```r
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
```r
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 10664423 10664625 . - ."
) psichomics:::parseMisoTandemUTR(event)
```

# alternative first exon (AFE) event
```r
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
```r
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**

```r
parseMisoId(id)
```

**Arguments**

- **id** Character: MISO alternative splicing event identifier

**Value**

Character with the parsed ID

**Examples**

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

**parseSplicingEvent** *Parse alternative splicing event identifier*

**Description**

Parse alternative splicing event identifier

**Usage**

```r
parseSplicingEvent(
  event,
  char = FALSE,
  pretty = FALSE,
  extra = NULL,
  coords = FALSE,
  data = NULL
)
```
parseSuppaAnnotation

Arguments

event Character: event identifier
char Boolean: return character vector instead of list with parsed values?
pretty Boolean: return a prettier name of the event identifier?
extra Character: extra information to add (such as species and assembly version); only used if pretty = TRUE and char = TRUE
coords Boolean: display extra coordinates regarding the alternative and constitutive regions of alternative splicing events? Only used if char = FALSE
data Matrix or data frame: alternative splicing information

Value

Data.frame containing type of event, chromosome, strand, gene and position of alternative splicing events or character with that same information (depending on what is available)

Examples

events <- c("A3SS_15_+_63353138_63353912_63353397_TPM1",
"A3SS_11_-61118463_61117115_61117894_CYB561A3",
"A5SS_21_+_48055375_48056459_48056808_PRMT2",
"A5SS_1_-1274742_1274667_1274033_DVL1",
"AFE_9_+_131902430_131901928_131904724_PPP2R4",
"AFE_5_-134686513_134688636_134681747_H2AFY",
"ALE_12_+_56554104_56554410_56555171_MYL6",
"ALE_8_-38314874_38287466_38285953_FGFR1",
"SE_9_+_6486925_6492303_6492401_6493826_UHRF2",
"SE_19_-5218431_5216778_5216731_5215606_PTPRS",
"MXE_15_+_63335142_63335905_63336030_63336226_63336351_63349184_TPM1",
"MXE_17_-74090495_74087316_74087224_74086478_74086410_74085401_EXOC7")
parseSplicingEvent(events)

Description

Parse events from alternative splicing annotation

Usage

parseSuppaAnnotation(
  folder,
  types = c("SE", "AF", "AL", "MX", "A5", "A3", "RI"),
  genome = "hg19"
)
parseSuppaAnnotation(  
  folder,  
  types = c("ALT3", "ALT5", "COMBI", "IR", "MERGE3m", "MIC", "EXSK", "MULTI"),  
  genome = "Hsa",  
  complexEvents = FALSE  
)

parseMisoAnnotation(  
  folder,  
  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?
novelEvents Boolean: parse events detected due to novel splice sites

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
See Also

Other functions to prepare alternative splicing annotations: `prepareAnnotationFromEvents()`

Examples

```r
# Load sample files
folder <- "extdata/eventsAnnotSample/suppa_output/suppaEvents"
suppaOutput <- system.file(folder, package="psichomics")
suppa <- parseSuppaAnnotation(suppaOutput)
# Load sample files
caller <- "extdata/eventsAnnotSample/VASTDB/Hsa/TEMPLATES"
vastToolsOutput <- system.file(caller, package="psichomics")
vast <- parseVastToolsAnnotation(vastToolsOutput)
# Load sample files
caller <- "extdata/eventsAnnotSample/miso_annotation"
misoOutput <- system.file(caller, package="psichomics")
miso <- parseMisoAnnotation(misoOutput)
# Load sample files
caller <- "extdata/eventsAnnotSample/mats_output/ASEvents"
matsOutput <- system.file(caller, 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**

```r
parseSuppaEvent(event)
```

**Arguments**

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

**Details**

More information about SUPPA available at [https://bitbucket.org/regulatorygenomicsupf/suppa](https://bitbucket.org/regulatorygenomicsupf/suppa)

The following event types are available to be parsed:
parseSuppaGeneric

- **SE** (skipped exon)
- **RI** (retained intron)
- **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**

```r
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**

```r
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

- **juncions**: List of integers: exon-exon junctions of an event
- **strand**: Character: positive-sense (+) or negative-sense (−) 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** (retained intron)
- **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

```r
# Parse generic event (in this case, an exon skipping event)
junctions <- read.table(text = "169768099 169770024 169770112 169771762")
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")
```
parseTCGAsampleTypes

Parse sample information from TCGA sample identifiers

parseTCGAsampleTypes

Parse sample information from TCGA sample identifiers

parseTCGAsampleTypes(samples,
  filename = system.file("extdata", "TCGAsampleType.RDS", package = "psichomics")
)

parseTCGAsampleInfo(samples, match = NULL)

Arguments

samples Character: sample identifiers
filename Character: path to RDS file containing corresponding types
match Integer: match between samples and subjects (NULL by default; performs the match)

Value

Metadata associated with each TCGA sample

See Also

Other functions associated with TCGA data retrieval: getDownloadsFolder(), getTCGADATATypes(), isFirebrowseUp(), loadTCGADATA()
Examples

```r
c# Parse sample types from TCGA dataset
parseTCGASampleTypes(c("TCGA-01A-Tumour", "TCGA-10B-Normal"))
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 REST service

Description

Parse XML from UniProt REST service

Usage

```r
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

```r
parseUrlsFromFirebrowseResponse(res)
```

Arguments

- `res` Response from `httr::GET` to a FireBrowse data query

Value

- Named character with URLs
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** (retained intron)
- **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")
performICA

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.
Perform principal component analysis after processing missing values

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

Description
Perform principal component analysis after processing missing values
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
- **...**: Arguments passed on to `stats::prcomp`

Value

PCA result in a `prcomp` object

See Also

Other functions to analyse principal components: `calculateLoadingsContribution()`, `plotPCA()`, `plotPCAvariance()`

Examples

```r
performPCA(USArrests)
```

---

**plotClusters**

*Add clusters to highchart object*

Description

Clusters are added as coloured polygons.

Usage

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

Arguments

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

Value

*highcharter object*
plotDistribution

Plot sample distribution

Description

The tooltip shows the median, variance, maximum, minimum and number of non-NA samples of each data series, as well as sample names if available.

Usage

plotDistribution(
  data,
  groups = NULL,
  rug = length(data) < 500,
  vLine = TRUE,
  ...,
  title = NULL,
  subtitle = NULL,
  type = c("density", "boxplot", "violin"),
  invertAxes = FALSE,
  psi = NULL,
  rugLabels = FALSE,
  rugLabelsRotation = 0,
  legend = TRUE,
  valueLabel = NULL
)

Arguments

data  Numeric, data frame or matrix: gene expression data or alternative splicing event quantification values (sample names are based on their names or colnames)
groups List of sample names or vector containing the group name per data value (read Details); if NULL or a character vector of length 1, data values are considered from the same group
rug Boolean: show rug plot?
vLine Boolean: plot vertical lines (including descriptive statistics for each group)?
... Arguments passed on to stats::density.default
bw  the smoothing bandwidth to be used. The kernels are scaled such that this is the standard deviation of the smoothing kernel. (Note this differs from the reference books cited below, and from S-PLUS.)
bw can also be a character string giving a rule to choose the bandwidth. See bw.nrd.
The default, "nrd0", has remained the default for historical and compatibility reasons, rather than as a general recommendation, where e.g., "SJ" would rather fit, see also Venables and Ripley (2002).
The specified (or computed) value of bw is multiplied by adjust.
adjust. The bandwidth used is actually \( \text{adjust} \times \text{bw} \). This makes it easy to specify values like ‘half the default’ bandwidth.

kernel, window. A character string giving the smoothing kernel to be used. This must partially match one of “gaussian", "rectangular", "triangular", "epanechnikov", "biweight", “cosine" or "optcosine", with default "gaussian", and may be abbreviated to a unique prefix (single letter). "cosine" is smoother than "optcosine", which is the usual ‘cosine’ kernel in the literature and almost MSE-efficient. However, "cosine" is the version used by S.

weights. Numeric vector of non-negative observation weights, hence of same length as \( x \). The default NULL is equivalent to \( \text{weights} = \text{rep}(1/\text{nx}, \text{nx}) \) where \( \text{nx} \) is the length of (the finite entries of) \( x[] \). If \( \text{na.rm} = \text{TRUE} \) and there are NA’s in \( x \), they and the corresponding weights are removed before computations. In that case, when the original weights have summed to one, they are re-scaled to keep doing so.

Note that weights are not taken into account for automatic bandwidth rules, i.e., when \( \text{bw} \) is a string. When the weights are proportional to true counts \( \text{cn} \), \( \text{density}(x = \text{rep}(x, \text{cn})) \) may be used instead of weights.

width. This exists for compatibility with S; if given, and \( \text{bw} \) is not, will set \( \text{bw} \) to width if this is a character string, or to a kernel-dependent multiple of width if this is numeric.

give.Rkern logical; if true, no density is estimated, and the ‘canonical bandwidth’ of the chosen kernel is returned instead.

subdensity used only when weights are specified which do not sum to one.

When true, it indicates that a “sub-density” is desired and no warning should be signalled. By default, when false, a warning is signalled when the weights do not sum to one.

warnWbw logical, used only when weights are specified and \( \text{bw} \) is character, i.e., automatic bandwidth selection is chosen (as by default). When true (as by default), a warning is signalled to alert the user that automatic bandwidth selection will not take the weights into account and hence may be suboptimal.

n. The number of equally spaced points at which the density is to be estimated. When \( n > 512 \), it is rounded up to a power of 2 during the calculations (as \text{fft} is used) and the final result is interpolated by \text{approx}. So it almost always makes sense to specify \( n \) as a power of two.

from, to. The left and right-most points of the grid at which the density is to be estimated; the defaults are \( \text{cut} \times \text{bw} \) outside of \text{range}(x).

cut. By default, the values of \( \text{from} \) and \( \text{to} \) are cut bandwidths beyond the extremes of the data. This allows the estimated density to drop to approximately zero at the extremes.

title. Character: plot title

subtitle. Character: plot subtitle

type. Character: density, boxplot or violin plot

invertAxes. Boolean: plot X axis as Y and vice-versa?
plotGeneExprPerSample

psi
Booleans: are data composed of PSI values? If NULL, psi = TRUE if all data values are between 0 and 1

rugLabels
Boolean: plot sample names in the rug?

rugLabelsRotation
Numeric: rotation (in degrees) of rug labels; this may present issues at different zoom levels and depending on the proximity of data values

legend
Boolean: show legend?

valueLabel
Character: label for the value (by default, either Inclusion levels or Gene expression)

Details
Argument groups can be either:

- a list of sample names, e.g. list("Group 1"=c("Sample A", "Sample B"), "Group 2"=c("Sample C"))
- a character vector with the same length as data, e.g. c("Sample A", "Sample C", "Sample B").

Value
highchart object with density plot

See Also
Other functions to perform and plot differential analyses: diffAnalyses()

Examples

data  <- sample(20, rep=TRUE)/20
groups <- paste("Group", c(rep("A", 10), rep("B", 10)))
names(data) <- paste("Sample", seq(data))
plotDistribution(data, groups)

# Using colours
attr(groups, "Colour") <- c("Group A"="pink", "Group B"="orange")
plotDistribution(data, groups)

plotGeneExprPerSample  Plot distribution of gene expression per sample

Description
Plot distribution of gene expression per sample

Usage
plotGeneExprPerSample(geneExpr, ...)

## plotGroupIndependence

### Arguments

- `geneExpr`: Data frame or matrix: gene expression
- `...`: Arguments passed on to `renderBoxplot`
- `data`: Data frame or matrix
- `outliers`: Boolean: draw outliers?
- `sortByMedian`: Boolean: sort box plots based on ascending median?
- `showXlabels`: Boolean: show labels in X axis?

### Value

Gene expression distribution plots

### See Also

Other functions for gene expression pre-processing: `convertGeneIdentifiers()`, `filterGeneExpr()`, `normaliseGeneExpression()`, `plotLibrarySize()`, `plotRowStats()`

### Examples

```r
df <- data.frame(geneA=c(2, 4, 5),
                 geneB=c(20, 3, 5),
                 geneC=c(5, 10, 21))
colnames(df) <- paste("Sample", 1:3)
plotGeneExprPerSample(df)
```

### Description

Plot $-\log_{10}(p\text{-values})$ of the results obtained after multiple group independence testing

### Usage

```r
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()`

Other functions for data grouping: `createGroupByAttribute()`, `getGeneList()`, `getSampleFromSubject()`, `getSubjectFromSample()`, `groupPerElem()`, `testGroupIndependence()`

Examples

```r
elements <- paste("subjects", 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)
```

### Description

Create multiple scatterplots from ICA

#### Usage

```r
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 (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

**diag**
logical, whether or not the main diagonal is plotted (scatter plot of variables against themselves).

Value

Multiple scatterplots as a `pairsD3` object

See Also

Other functions to analyse independent components: `performICA()`

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)

plotLibrarySize(data, log10 = TRUE,
title = "Library size distribution across samples",
subtitle = "Library size: total number of mapped reads",
colour = "orange")

Arguments
- data: Data frame or matrix: gene expression
- log10: Boolean: log10-transform data?
- title: Character: plot title
- subtitle: Character: plot subtitle
- colour: Character: data colour

Value
Library size distribution

See Also
Other functions for gene expression pre-processing: convertGeneIdentifiers(), filterGeneExpr(),
normaliseGeneExpression(), plotGeneExprPerSample(), plotRowStats()

Examples

```r
df <- data.frame(geneA=c(2, 4, 5),
  geneB=c(20, 3, 5),
  geneC=c(5, 10, 21))
colnames(df) <- paste("Sample", 1:3)
plotLibrarySize(df)
```
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

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

Value

Scatterplot as an highchart object

See Also

Other functions to analyse principal components: calculateLoadingsContribution(), performPCA(), plotPCAvariance()
Examples

```r
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)

# Only plot loadings
plotPCA(pca, pcX=2, pcY=3, loadings=TRUE, individuals=FALSE)
```

---

plotPCAvariance

Create the explained variance plot from a PCA

Description

Create the explained variance plot from a PCA

Usage

```r
plotPCAvariance(pca)
```

Arguments

- `pca`: `prcomp` object

Value

Plot variance as an highchart object

See Also

Other functions to analyse principal components: `calculateLoadingsContribution()`, `performPCA()`, `plotPCA()`

Examples

```r
pca <- prcomp(USArrests)
plotPCAvariance(pca)
```
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
colour  Character: default colour
alpha  Numeric: default transparency value

Value
HTML elements

plotProtein  
Plot protein features

Description
Plot protein features

Usage
plotProtein(molecule)
plotRowStats

Arguments

molecule Character: UniProt protein or Ensembl transcript identifier

Value

highcharter object

See Also

Other functions to retrieve external information: ensemblToUniprot(), plotTranscripts(), queryEnsemblByGene()

Examples

protein <- "P38398"
plotProtein(protein)

transcript <- "ENST00000488540"
plotProtein(transcript)

plotRowStats() Plot row-wise statistics

Description

Scatter plot to compare between the row-wise mean, median, variance or range from a data frame or matrix. Also supports transformations of those variables, such as log10(mean). If y = NULL, a density plot is rendered instead.

Usage

plotRowStats(
  data,
  x,
  y = NULL,
  subset = NULL,
  xmin = NULL,
  xmax = NULL,
  ymin = NULL,
  ymax = NULL,
  xlim = NULL,
  ylim = NULL,
  cache = NULL,
  verbose = FALSE,
  data2 = NULL,
  legend = FALSE,
  legendLabels = c("Original", "Highlighted")
)
Arguments

data Data frame or matrix containing samples per column and, for instance, gene or alternative splicing event per row

x, y Character: statistic to calculate and display in the plot per row; choose between mean, median, var or range (or transformations of those variables, e.g. log10(var)); if y = NULL, the density of x will be plot instead

subset Boolean or integer: data points to highlight

xmin, xmax, ymin, ymax Numeric: minimum and maximum X and Y values to draw in the plot

xlim, ylim Numeric: X and Y axis range

cache List of summary statistics for data previously calculated to avoid repeating calculations (output also returns cache in attribute named cache with appropriate data)

verbose Boolean: print messages of the steps performed

data2 Same as data argument but points in data2 are highlighted (unless data2 = NULL)

legend Boolean: show legend?

legendLabels Character: legend labels

Value

Plot of data

See Also

Other functions for gene expression pre-processing: convertGeneIdentifiers(), filterGeneExpr(), normaliseGeneExpression(), plotGeneExprPerSample(), plotLibrarySize()

Other functions for PSI quantification: filterPSI(), getSplicingEventTypes(), listSplicingAnnotations(), loadAnnotation(), quantifySplicing()

Examples

library(ggplot2)

# Plotting gene expression data
geneExpr <- readFile("ex_gene_expression.RDS")
plotRowStats(geneExpr, "mean", "var^((1/4))") +
  ggtitle("Mean-variance plot") +
  labs(y="Square Root of the Standard Deviation")

# Plotting alternative splicing quantification
annot <- readFile("ex_splicing_annotation.RDS")
junctionQuant <- readFile("ex_junctionQuant.RDS")
psi <- quantifySplicing(annot, junctionQuant, eventType=c("SE", "MXE")))

medianVar <- plotRowStats(psi, x="median", y="var", xlim=c(0, 1)) +
  labs(x="Median PSI", y="PSI variance")
medianVar

rangeVar <- plotRowStats(psi, x="range", y="log10(var)", xlim=c(0, 1)) +
labs(x="PSI range", y="log10(PSI variance)"
rangeVar

---

**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)
psychomics::plotSingleICA(ica)
psychomics::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")
psychomics::plotSingleICA(ica, groups=groups)
plotSplicingEvent  
*Plot diagram of alternative splicing events*

**Description**

Plot diagram of alternative splicing events

**Usage**

```r
plotSplicingEvent(
  ASevent, 
  data = NULL, 
  showText = TRUE, 
  showPath = TRUE, 
  showAlternative1 = TRUE, 
  showAlternative2 = TRUE, 
  constitutiveWidth = NULL, 
  alternativeWidth = NULL, 
  intronWidth = NULL, 
  constitutiveFill = "lightgray", 
  constitutiveStroke = "darkgray", 
  alternative1Fill = "#ff8b63", 
  alternative1Stroke = "#faa000", 
  alternative2Fill = "#ca06c", 
  alternative2Stroke = "#9d7039", 
  class = NULL, 
  style = NULL
)
```

**Arguments**

- **ASevent**: Character: alternative splicing event identifiers
- **data**: Matrix or data frame: alternative splicing information
- **showText**: Boolean: display coordinates and length (if available)
- **showPath**: Boolean: display alternative splicing junctions
- **showAlternative1**: Boolean: show alternative exon 1 and respective splicing junctions and text?
- **showAlternative2**: Boolean: show alternative exon 2 and respective splicing junctions and text? (only related with mutually exclusive exons)
- **constitutiveWidth**: Numeric: width of constitutive exon(s)
- **alternativeWidth**: Numeric: width of alternative exon(s)
- **intronWidth**: Numeric: width of intron’s representation
constitutiveFill
Character: fill colour of constitutive exons
constitutiveStroke
Character: stroke colour of constitutive exons
alternative1Fill
Character: fill colour of alternative exon 1
alternative1Stroke
Character: stroke colour of alternative exon 1
alternative2Fill
Character: fill colour of alternative exon 2
alternative2Stroke
Character: stroke colour of alternative exon 2
class
Character: class of SVG parent tag
style
Character: style of SVG parent tag

Value
List of SVG (one for each alternative splicing event)

Examples

```r
events <- c(
  "A3SS_15_+_63353138_63353912_63353397_TPM1",
  "A3SS_11_-_61118463_61117115_61117894_CYB561A3",
  "A5SS_21_+_48055675_48056459_48056808_PRMT2",
  "A5SS_1_-_1274742_1274667_1274033_DVL1",
  "AFE_9_+_131902430_131901928_131904724_PPP2R4",
  "AFE_5_-_134686513_134688636_134681747_H2AFY",
  "ALE_12_+_56554104_56554410_56555171_MYL6",
  "ALE_8_+_38314874_38287466_38285953_FGFR1",
  "SE_9_+_6486925_6492303_6492401_UHRF2",
  "SE_19_+_5218431_5216778_5216731_5215606_PTPRS",
  "MXE_15_+_63335142_63335905_63336351_63336226_63349184_TPM1",
  "MXE_17_+_74090495_74087316_74087224_74086478_74086410_EXOC7")
diagram <- plotSplicingEvent(events)
```

```r
## Not run:
diagram[["A3SS_3_-_145796903_145794682_145795711_PLOD2"]]
diagram[[6]]
diagram
```

## End(Not run)

---

**plotSurvivalCurves**

*Plot survival curves*

**Description**
Plot survival curves
plotSurvivalCurves

Usage

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

Arguments

surv  Survival object
mark  Boolean: mark times?
interval  Boolean: show interval ranges?
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)?

Value

Plot of survival curves

See Also

Other functions to analyse survival: assignValuePerSubject(), getAttributesTime(), labelBasedOnCutoff(), optimalSurvivalCutoff(), plotSurvivalPvaluesByCutoff(), processSurvTerms(), survdiffTerms(), survfit.survTerms(), testSurvival()

Examples

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

plotSurvivalPvaluesByCutoff

Plot p-values of survival difference between groups based on multiple cutoffs

Description

Plot p-values of survival difference between groups based on multiple cutoffs
**plotSurvivalPvaluesByCutoff**

**Usage**

```r
plotSurvivalPvaluesByCutoff(
  clinical,  # Data frame: clinical data
  data,     # Numeric: elements of interest to test against the cutoff
  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 = NULL, # Character: name of column containing ending time of the interval (only relevant for interval censoring)
  followup = "days_to_last_followup", # Character: name of column containing follow up time
  significance = 0.05, # Numeric: significance threshold
  cutoffs = seq(0, 0.99, 0.01)  # Numeric: cutoffs to test
)
```

**Arguments**

- **clinical**: Data frame: clinical data
- **data**: Numeric: elements of interest to test against the cutoff
- **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
- **significance**: Numeric: significance threshold
- **cutoffs**: Numeric: cutoffs to test

**Value**

p-value plot

**See Also**

Other functions to analyse survival: `assignValuePerSubject()`, `getAttributeTime()`, `labelBasedOnCutoff()`, `optimalSurvivalCutoff()`, `plotSurvivalCurves()`, `processSurvTerms()`, `survdiffTerms()`, `survfit.survTerms()`, `testSurvival()`

**Examples**

```r
clinical <- read.table(text = "2549 NA ii female
840 NA i female
NA 1204 iv male
NA 383 iv female
1293 NA iii male")

names(clinical) <- c("patient.days_to_last_followup",
  "patient.days_to_death",
  "patient.stage_event.pathologic_stage",
```

```r
```
clinical <- do.call(rbind, rep(list(clinical), 5))
rownames(clinical) <- paste("Subject", seq(nrow(clinical)))

# 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"))

# Match between subjects and samples
match <- c("Cancer 1"="Subject 3",
           "Cancer 2"="Subject 17",
           "Cancer 3"="Subject 21")

eventData <- assignValuePerSubject(psi[, ], match)

event <- "days_to_death"
timeStart <- "days_to_death"
plotSurvivalPvaluesByCutoff(clinical, eventData, censoring="right",
                           event=event, timeStart=timeStart)

---

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?

**Value**

HTML elements
plotTranscripts

Description

Plot transcripts

Usage

plotTranscripts(
  info,
  eventPosition = NULL,
  event = NULL,
  eventData = NULL,
  shiny = FALSE
)

Arguments

info Information retrieved from Ensembl
eventPosition Numeric: coordinates of the alternative splicing event (ignored if event is set)
event Character: identifier of the alternative splicing event to plot
eventData Object containing event information to be parsed
shiny Boolean: is the function running in a Shiny session?

Value

NULL (function is only used to modify the Shiny session’s state or internal variables)

See Also

Other functions to retrieve external information: ensemblToUniprot(), plotProtein(), queryEnsemblByGene()

Examples

event <- "SE_12_-_7985318_7984360_7984200_7982602_SLC2A14"
info <- queryEnsemblByEvent(event, species="human", assembly="hg19")
## Not run:
plotTranscripts(info, event=event)

## End(Not run)
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.

See Also

Other functions to prepare alternative splicing annotations: parseSuppaAnnotation()
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, modalId = "modal", ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>Shiny session</td>
</tr>
<tr>
<td>input</td>
<td>Shiny input</td>
</tr>
<tr>
<td>id</td>
<td>Character: input identifier</td>
</tr>
<tr>
<td>modalId</td>
<td>Character: modal window identifier</td>
</tr>
<tr>
<td>...</td>
<td>Arguments passed on to fileBrowser</td>
</tr>
<tr>
<td>default</td>
<td>Character: path to initial folder</td>
</tr>
<tr>
<td>caption</td>
<td>Character: caption on the selection dialogue</td>
</tr>
<tr>
<td>multiple</td>
<td>Boolean: allow to select multiple files?</td>
</tr>
<tr>
<td>directory</td>
<td>Boolean: allow to select directories instead of files?</td>
</tr>
</tbody>
</table>

Value

NULL (function is only used to modify the Shiny session’s state or internal variables)

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)
**prepareGenePresentation**

Prepare presentation of multiple genes for the same splicing event

**Description**

Prepare presentation of multiple genes for the same splicing event

**Usage**

```r
prepareGenePresentation(gene, collapse = "/")
```

**Arguments**

- **gene**: Character: gene
- **collapse**: Character: character string to separate in case of more than one gene

**Value**

Same object with items collapsed

### 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**

```r
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)
```
### Description

Prepare user-provided files to be loaded into psichomics

### Usage

```r
prepareJunctionQuantSTAR(..., startOffset = -1, endOffset = +1)
prepareGeneQuantSTAR(...,
  strandedness = c("unstranded", "stranded", "stranded (reverse)")
)
```

### Arguments

- `...`  Character: path of (optionally named) input files (see Examples)
- `startOffset`  Numeric: value to offset start position
- `endOffset`  Numeric: value to offset end position
- `strandedness`  Character: strandedness of RNA-seq protocol; may be one of the following: unstraded, stranded or stranded (reverse)

### Value

Prepared file (if output != NULL) and object

### Examples

```r
## Not run:
prepareJunctionQuant("Control rep1"=junctionFile1,
  "Control rep2"=junctionFile2,
  "KD rep1"=junctionFile3,
  "KD rep2"=junctionFile4)
## End(Not run)

## Not run:
prepareGeneQuant("Control rep1"=geneCountFile1,
  "Control rep2"=geneCountFile2,
  "KD rep1"=geneCountFile3,
  "KD rep2"=geneCountFile4)
## End(Not run)
```
preparePreMadeGroupForSelection

Prepare list of pre-made groups for a selectize element

Description
Prepare list of pre-made groups for a selectize element

Usage
preparePreMadeGroupForSelection(groups)

Arguments

- **groups** List of list of characters

Value
List

prepareSRAmetadata

Prepare user-provided files to be loaded into psichomics

Description
Prepare user-provided files to be loaded into psichomics

Usage
prepareSRAmetadata(file, output = "psichomics_metadata.txt")

prepareJunctionQuant(...,
output = "psichomics_junctions.txt",
startOffset = NULL,
endOffset = NULL)

prepareGeneQuant(...,
output = "psichomics_gene_counts.txt",
strandedness = c("unstranded", "stranded", "stranded (reverse)"))
prepareWordBreak

Create word break opportunities (for HTML) using given characters

Arguments

- `file` Character: path to file
- `output` Character: path of output file (if NULL, only returns the data without saving it to a file)
- ... Character: path of (optionally named) input files (see Examples)
- `startOffset` Numeric: value to offset start position
- `endOffset` Numeric: value to offset end position
- `strandedness` Character: strandedness of RNA-seq protocol; may be one of the following: unstraded, stranded or stranded (reverse)

Value

Prepared file (if `output` != NULL) and object

Examples

```r
# Not run:
prepareJunctionQuant("Control rep1"=junctionFile1,
    "Control rep2"=junctionFile2,
    "KD rep1"=junctionFile3,
    "KD rep2"=junctionFile4)

# End(Not run)

# Not run:
prepareGeneQuant("Control rep1"=geneCountFile1,
    "Control rep2"=geneCountFile2,
    "KD rep1"=geneCountFile3,
    "KD rep2"=geneCountFile4)

# End(Not run)
```

prepareWordBreak

Create word break opportunities (for HTML) using given characters

Description

Create word break opportunities (for HTML) using given characters

Usage

```r
prepareWordBreak(
    str,
    pattern = c(".", "-", "\", "/", ",", ",", "+", "="),
    html = TRUE
)
```
Arguments

- **str**: Character: text
- **pattern**: Character: pattern(s) of interest to be used as word break opportunities
- **html**: Boolean: convert to HTML?

Value

String containing HTML elements

---

**preserveAttributes**

Preserve attributes when extracting values

Description

Add object to class `sticky`

Usage

`preserveAttributes(x)`

Arguments

- **x**: Object

Value

Object with class `sticky`

---

**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
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

processSRAdataset  Process SRA quantification data

Description
Process SRA quantification data

Usage
processSRAdataset(files, data, IDcolname)

Arguments
- files: Character: path to SRA quantification files
- data: Data frame: processed quantification data
- IDcolname: Character: name of the column containing the identifiers

Value
Process file
**processSurvData**

*Process survival data to calculate survival curves*

**Description**

Process survival data to calculate survival curves

**Usage**

```r
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 subject
- `clinical`: Data frame: clinical data
- `survTime`: `survTime` object: Times to follow up, time start, time stop and event (optional)

**Details**

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

If `survTime = NULL`, survival times are obtained from the clinical dataset according to the names given in `timeStart`, `timeStop`, `event` and `followup`. This may become quite slow when used in a loop. If the aforementioned variables are constant, consider running `getAttributesTime()` outside the loop and using its output via the `survTime` argument of this function (see Examples).

**Value**

Data frame with terms needed to calculate survival curves
processSurvTerms

Process survival curves terms to calculate survival curves

Description

Process survival curves terms to calculate survival curves

processSurvTerms

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?
survTime survTime object: times to follow up, time start, time stop and event (optional)
group Character: group relative to each subject
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

List with survival analysis results
Usage

```
processSurvTerms(
  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 = NULL,  # Character: name of column containing ending time of the interval (only relevant for interval censoring)
  group = NULL,  # Character: group relative to each subject
  formulaStr = NULL,  # Character: formula to use
  coxph = FALSE,  # Boolean: fit a Cox proportional hazards regression model?
  scale = "days",  # Character: rescale the survival time to days, weeks, months or years
  followup = "days_to_last_followup",  # Character: name of column containing follow up time
  survTime = NULL  # survTime object: times to follow up, time start, time stop and event (optional)
)
```

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 subject
- `formulaStr`: Character: formula to use
- `coxph`: Boolean: fit a Cox proportional hazards regression model?
- `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

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

If `survTime = NULL`, survival times are obtained from the clinical dataset according to the names given in `timeStart`, `timeStop`, `event` and `followup`. This may become quite slow when used in a loop. If the aforementioned variables are constant, consider running `getAttributeTime()` outside the loop and using its output via the `survTime` argument of this function (see Examples).

Value

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

Other functions to analyse survival: `assignValuePerSubject()`, `getAttributesTime()`, `labelBasedOnCutoff()`, `optimalSurvivalCutoff()`, `plotSurvivalCurves()`, `plotSurvivalPvaluesByCutoff()`, `survdiffTerms()`, `survfit.survTerms()`, `testSurvival()`

Examples

```r
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)

# If running multiple times, consider calculating survTime only once
survTime <- getAttributesTime(clinical, event, timeStart)
for (i in seq(5)) {
  survTerms <- processSurvTerms(clinical, censoring="right", event, timeStart, formulaStr=formulaStr,
survTime=survTime)
}
```

---

**psychomics**

Start graphical interface of psychomics

Description

Start graphical interface of psychomics

Usage

```r
psychomics(

  ..., 
  launch.browser = TRUE,
  shinyproxy = FALSE,
  testData = FALSE,
  cache = getAnnotationHubOption("CACHE")
)
```
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 between 3000:8000, excluding ports that are blocked by Google Chrome for being considered unsafe: 3659, 4045, 5060, 5061, 6000, 6566, 6665:6669 and 6697. Up to twenty random ports will be tried.

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.

launch.browser If true, the system’s default web browser will be launched automatically after the app is started. Defaults to true in interactive sessions only. The value of this parameter can also be a function to call with the application’s URL.

shinyproxy Boolean: prepare visual interface to run in Shinyproxy?

testData Boolean: load with test data

Value

NULL (function is only used to modify the Shiny session’s state or internal variables)

Examples

```r
## Not run:
psichomics()
```

```r
## 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(ns, gene, ...)

Arguments

ns
  Namespace function
gene
  Character: gene
...  Arguments passed on to queryPubMed
top  Numeric: number of articles to retrieve
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
)
quantifySplicingSet

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: values whose number of total supporting read counts is below minReads
                      are returned as NA
   genes             Character: gene symbols for which to quantify splicing events (if NULL, events
                      from all genes are quantified)

Value

   Data frame with the quantification of the alternative splicing events

See Also

   Other functions for PSI quantification: filterPSI(), getSplicingEventTypes(), listSplicingAnnotations(),
   loadAnnotation(), plotRowStats()

Examples

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

   quantifySplicing(annot, junctionQuant, eventType=c("SE", "MXE"))
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? if FALSE, query the most recent API

Value
Parsed response or NULL if no response

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

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

queryEnsemblByGene

Query information from Ensembl

Description
Query information from Ensembl

Usage
queryEnsemblByGene(gene, species = NULL, assembly = NULL)
queryEnsemblByEvent(event, species = NULL, assembly = NULL, data = NULL)
queryFirebrowseData

Query the FireBrowse API for TCGA data

Description

Query the FireBrowse 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
)
queryPubMed

Query the PubMed REST API

Description
Query the PubMed REST API

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

Arguments

- **primary**: Character: primary search term
- ... Character: other relevant search terms
- **top**: Numeric: number of articles to retrieve
- **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

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

---

Description

Query the UniProt REST API

Usage

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

Arguments

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

Value

Parsed response

Examples

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

transcript <- "ENST00000488540"
format <- "xml"
psichomics:::queryUniprot(transcript, format)
```
**readAnnot**  
*Read custom or remote annotation*

**Description**
Instructions to build the Shiny app

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

**Arguments**
- `session`  
  Shiny session
- `annotation`  
  Character: chosen annotation
- `showProgress`  
  Boolean: show progress?

**Value**
NULL (function is only used to modify the Shiny session’s state or internal variables)

---

**readFile**  
*Load psichomics-specific file*

**Description**
Load psichomics-specific 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 numeric-alike vector of length equal to the number of columns of x, where 'numeric-alike' means that as.numeric(.) will be applied successfully if is.numeric(.) is not true.

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

...  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**

```r
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
renderBoxplot

Note
The names of pre-existing groups are not modified.

Description
Render boxplot

Usage
renderBoxplot(
  data,
  outliers = FALSE,
  sortByMedian = TRUE,
  showXlabels = TRUE,
  title = NULL,
  seriesName = "Gene expression"
)

Arguments
data Data frame or matrix
outliers Boolean: draw outliers?
sortByMedian Boolean: sort box plots based on ascending median?
showXlabels Boolean: show labels in X axis?

Value
Box plot

Examples
psychomics::renderBoxplot(data.frame(a=1:10, b=10:19, c=45:54))
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.
quoted If it is TRUE, then the quote()ed value of expr will be used when expr
is evaluated. If expr is a quosure and you would like to use its expression
as a value for expr, then you must set quoted to TRUE.
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 an interactive highchart object

Value

NULL (function is only used to modify the Shiny session’s state or internal variables)
renderGeneticInfo  Render genetic information

Description
Render genetic information

Usage
renderGeneticInfo(
  output,
  info,
  species = NULL,
  assembly = NULL,
  grch37 = FALSE,
  eventDiagram = NULL,
  gene = NULL
)

Arguments
output  Shiny output
info    Information as retrieved from Ensembl
species Character: species name
assembly Character: assembly version
grch37   Boolean: use version GRCh37 of the genome?
eventDiagram Diagram of selected alternative splicing event
ns       Namespace function

Value
HTML elements to render gene, protein and transcript annotation

renderGroupInterface  Render group interface

Description
Render group interface

Usage
renderGroupInterface(ns, multiFisherTests = TRUE)
replaceStrInList

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

replaceStrInList Replace a string with another in a list

Description
Replace a string with another in a list

Usage
replaceStrInList(tag, old, new)
**rm.null**

Filter NULL elements from a vector or a list

**Description**
Filter NULL elements from a vector or a list

**Usage**

\( \text{rm.null}(v) \)

**Arguments**

- \( v \):
  - Vector or list

**Value**
Filtered vector or list with no NULL elements; if \( v \) is a vector composed of NULL elements, returns a NULL; if \( v \) is a list of NULL elements, returns an empty list

**roundDigits**

Round by the given number of digits

**Description**
Round by the given number of digits

**Usage**

\( \text{roundDigits}(n) \)

**Arguments**

- \( n \):
  - Numeric: number to round

**Value**
Formatted number with a given numeric precision
### roundMinDown

**Description**

Round down/up the minimum/maximum value

**Usage**

```r
roundMinDown(x, digits = 0)
roundMaxUp(x, digits = 0)
```

**Arguments**

- `x` Numeric: values
- `digits` Numeric: number of maximum digits

**Value**

Rounded numeric value

---

### saveProcessedSRAdata

**Description**

Save processed SRA data in file

**Usage**

```r
saveProcessedSRAdata(data, output = NULL)
```

**Arguments**

- `data` Object to save
- `output` Character: output filename (if NULL, no file is saved)

**Value**

If `output = NULL`, save input to a file and return it as invisible; otherwise, just return the input
**selectGroupsUI**

*Group selection*

**Description**

Group selection interface and logic

**Usage**

```r
selectGroupsUI(
  id,
  label,
  type,
  placeholder = "Type to search groups",
  noGroupsLabel = NULL,
  groupsLabel = NULL,
  maxItems = NULL,
  returnAllDataLabel = NULL,
  returnAllDataValue = FALSE
)
```

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

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

**Arguments**

- **id**  
  Character: identifier

- **label**  
  Character: selectize label

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

- **placeholder**  
  Character: selectize placeholder

- **noGroupsLabel**  
  Character: label to explicitly allow to select no groups (if NULL, this option is not displayed to the user)

- **groupsLabel**  
  Character: label to explicitly allow to select groups (only required if noGroupsLabel is not NULL)

- **maxItems**  
  Numeric: maximum number of groups to select

- **returnAllDataLabel**  
  Character: label to allow to return data outside selected groups as belonging to an outside group (if NULL, this option is not displayed to the user)

- **returnAllDataValue**  
  Boolean: default value to whether return all data or not (only required if returnAllDataLabel is not NULL)

- **session**  
  Shiny session
**Description**

Create input to select a gene

**Usage**

```r
selectizeGeneInput(
  id, 
  label = "Gene", 
  choices = NULL, 
  multiple = FALSE, 
  ..., 
  placeholder = "Type to search for a gene..."
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Character: identifier</td>
</tr>
<tr>
<td>label</td>
<td>Display label for the control, or NULL for no label.</td>
</tr>
<tr>
<td>choices</td>
<td>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. It’s also possible to group related inputs by providing a named list whose elements are (either named or unnamed) lists, vectors, or factors. In this case, the outermost names will be used as the group labels (leveraging the <code>&lt;optgroup&gt;</code> HTML tag) for the elements in the respective sublist. See the example section for a small demo of this feature.</td>
</tr>
</tbody>
</table>
selectPreMadeGroup

Is selection of multiple items allowed?

... Arguments passed to the options list of selectizeInput()

placeholder Character: placeholder

Value

HTML elements

selectPreMadeGroup Select pre-made groups from a selected item

Description

Select pre-made groups from a selected item

Usage

selectPreMadeGroup(groups, selected, genes = NULL)

Arguments

groups List of list of characters

selected Character: selected item

Value

Elements of selected item

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?
Value

NULL (function is only used to modify the Shiny session’s state or internal variables)

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?

Value

NULL (function is only used to modify the Shiny session’s state or internal variables)

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
)
**setOperationIcon**

**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
- **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 subjects (as values)
- **type**  
  Character: type of group where set operations are to be performed
- **assignColoursToGroups**  
  Boolean: assign colours to new groups?

**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**

Show or remove an alert

**Usage**

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

```
successAlert(
  session,
  ..., ...
  title = NULL,
  dismissible = TRUE,
  alertId = "success",
  caller = NULL
)
```

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

```
warningAlert(
  session,
  ..., ...
  title = NULL,
  dismissible = TRUE,
  alertId = "alert",
  caller = NULL
)
```
removeAlert(output, alertId = "alert")

**Arguments**

- **session**: Shiny session
- **...**: Arguments to render as elements of alert
- **title**: Character: title
- **style**: Character: style (error, warning or NULL)
- **dismissible**: Boolean: is the alert dismissible?
- **alertId**: Character: identifier
- **iconName**: Character: icon name
- **caller**: Character: caller module identifier
- **output**: Shiny output

**Value**

NULL (function is only used to modify the Shiny session’s state or internal variables)

**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 there are no groups)
**sidebar**  
*Sidebar without a well*

**Description**
Modified version of shiny::sidebarPanel without a well

**Usage**
```r
sidebar(..., width = 4)
```

**Arguments**
- `...`  
  Output elements to include in the sidebar/main panel.
- `width`  
  The width of the sidebar and main panel. By default, the sidebar takes up 1/3 of the width, and the main panel 2/3. The total width must be 12 or less.

**Value**
HTML elements

---

**signifDigits**  
*Get number of significant digits*

**Description**
Get number of significant digits

**Usage**
```r
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**

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

**Arguments**

- **vector**: Numeric
- **group**: Character: group of each element in the vector
- **threshold**: Integer: minimum number of values per group
- **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 `analyses` argument:

- **ttest** - Unpaired t-test (2 groups)
- **wilcoxonRankSum** - 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  

**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

---

*startProcess*

**Description**

- `startProcess` : Style button to show a process is in progress
- `endProcess` : Style button to show a process finished; also, closes the progress bar (if `closeProgressBar = TRUE`) and prints the difference between the current time and time.

**Usage**

```
startProcess(id)
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?
**Value**

`startProgress` returns the start time of the process (may be used as the time argument to `endProcess`), whereas `endProcess` returns the difference between current time and time (or NULL if time is not specified)

---

**startProgress**  
Create, set and terminate a progress object

---

**Description**

Create, set and terminate a progress object

**Usage**

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

updateProgress(
  message = "Loading...",
  value = NULL,
  max = NULL,
  detail = NULL,
  divisions = NULL,
  global = if (isRunning()) sharedData else getHidden(),
  console = TRUE
)

closeProgress(
  message = NULL,
  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` Integer: current progress value
- `max` Integer: maximum progress value
- `detail` Character: detailed message
- `console` Boolean: print message to console?
Details

If `divisions` is not `NULL`, a progress bar starts with the given divisions. If `value` = `NULL`, the progress bar increments one unit; otherwise, the progress bar increments `value`.

Value

`NULL` (function is only used to modify the Shiny session’s state or internal variables)

---

**styleModal**

Create a modal window

---

Description

Create a modal window

Usage

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

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

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

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>session</code></td>
<td>Shiny session</td>
</tr>
<tr>
<td><code>title</code></td>
<td>Character: title</td>
</tr>
<tr>
<td><code>...</code></td>
<td>Arguments passed on to <code>shiny::modalDialog</code></td>
</tr>
<tr>
<td><code>easyClose</code></td>
<td>If TRUE, the modal dialog can be dismissed by clicking outside the dialog box, or by pressing the Escape key. If FALSE (the default), the modal dialog can’t be dismissed in those ways; instead it must be dismissed by clicking on a <code>modalButton()</code>, or from a call to <code>removeModal()</code> on the server.</td>
</tr>
</tbody>
</table>
subjectMultiMatchWarning

- **fade**: If FALSE, the modal dialog will have no fade-in animation (it will simply appear rather than fade in to view).

- **style**: Character: style (NULL, warning, error or info)

- **iconName**: Character: icon name

- **footer**: HTML elements to use in footer

- **echo**: Boolean: print to console?

- **size**: Character: size of the modal (small, medium or large)

- **dismissButton**: Boolean: show dismiss button in footer?

- **caller**: Character: caller module identifier

**Value**

NULL (function is only used to modify the Shiny session’s state or internal variables)

**See Also**

- `showAlert()`

---

subjectMultiMatchWarning

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

**Description**

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

**Usage**

subjectMultiMatchWarning()

**Value**

Character
subsetGeneExpressionFromMatchingGenes

*Subset gene expression based on (full or partial) matching genes*

**Description**

Subset gene expression based on (full or partial) matching genes

**Usage**

```r
subsetGeneExpressionFromMatchingGenes(geneExpr, gene)
```

**Arguments**

- `geneExpr`  
  Data frame or matrix: gene expression
- `gene`  
  Character: genes to look for

**Value**

Gene expression subset for the input genes

---

survdiffTerms

*Test Survival Curve Differences*

**Description**

Tests if there is a difference between two or more survival curves using the \( G^p \) family of tests, or for a single curve against a known alternative.

**Usage**

```r
survdiffTerms(survTerms, ...)
```

**Arguments**

- `survTerms`  
  `survTerms` object: survival terms obtained after running `processSurvTerms` (see examples)
- `...`  
  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.
timefix process times through the aeqSurv function to eliminate potential roundoff issues.

Value

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

Description

This function implements the G-rho family of Harrington and Fleming (1982), with weights on each death of $S(t)^\rho$, where $S(t)$ is the Kaplan-Meier estimate of survival. With $\rho = 0$ this is the log-rank or Mantel-Haenszel test, and with $\rho = 1$ it is equivalent to the Peto & Peto modification of the Gehan-Wilcoxon test.

Peto and Peto show that the Gehan-Wilcoxon test can be badly biased if the two groups have different censoring patterns, and proposed an alternative. Prentice and Marek later showed an actual example where this issue occurs. For most data sets the Gehan-Wilcoxon and Peto-Peto-Prentice variant will hardly differ, however.

If the right hand side of the formula consists only of an offset term, then a one sample test is done. To cause missing values in the predictors to be treated as a separate group, rather than being omitted, use the factor function with its exclude argument to recode the righ-hand-side covariate.

References


See Also

Other functions to analyse survival: assignValuePerSubject(), getAttributesTime(), labelBasedOnCutoff(), optimalSurvivalCutoff(), plotSurvivalCurves(), plotSurvivalPvaluesByCutoff(), processSurvTerms(), survfit.survTerms(), testSurvival()

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", "sex", "stage"
survfit.survTerms

Create survival curves

Description
Create survival curves

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

Arguments

formula survTerms object: survival terms obtained after running processSurvTerms
(see examples)

... 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 char-
acter 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.

timefix process times through the aeqSurv function to eliminate potential
roundoff issues.

Details
A survival curve is based on a tabulation of the number at risk and number of events at each unique
death time. When time is a floating point number the definition of "unique" is subject to interpre-
tation. The code uses factor() to define the set. For further details see the documentation for the
appropriate method, i.e., ?survfit.formula or ?survfit.coxph.

A survfit object may contain a single curve, a set of curves (vector), a matrix of curves, or even a
3 way array: dim(fit) will reveal the dimensions. Predicted curves from a coxph model have one
row for each stratum in the Cox model fit and one column for each specified covariate set. Curves from a multi-state model have one row for each stratum and a column for each state, the strata correspond to predictors on the right hand side of the equation. The default printing and plotting order for curves is by column, as with other matrices.

Value

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

See Also

Other functions to analyse survival: assignValuePerSubject(), getAttributesTime(), labelBasedOnCutoff(), optimalSurvivalCutoff(), plotSurvivalCurves(), plotSurvivalPvaluesByCutoff(), processSurvTerms(), survdiffTerms(), testSurvival()

Examples

```r
library("survival")
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)
survfit(survTerms)
```

t.sticky

Preserve attributes of sticky objects when extracting or transposing object

Description

Most attributes - with the exception of names, dim, dimnames, class and row.names - are preserved in simple transformations of objects from class sticky
Usage

```r
## S3 method for class 'sticky'
t(x)
```

```r
## S3 method for class 'sticky'
x[i, j, ...]
```

Arguments

- `x`: Object
- `i, j, ...`: Numeric or character: indices of elements to extract

Value

Transformed object with most attributes preserved

---

**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

```r
tabDataset(
  ns,
  title,
  tableId,
  columns,
  visCols,
  data,
  description = NULL,
  icon = 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
table2html

Description

Create HTML table from data frame or matrix

Usage

```r
table2html(
  data,
  rownames = TRUE,
  colnames = TRUE,
  class = NULL,
  style = NULL,
  thead = FALSE
)
```

Arguments

- `data` Data frame or matrix
- `rownames` Boolean: print row names?
- `colnames` Boolean: print column names?
- `class` Character: table class
- `style` Character: table style
- `thead` Boolean: add a thead tag to the first row?

Value

HTML elements
**tableRow**

*Create a row for a HTML table*

---

**Description**

Create a row for a HTML table

**Usage**

```r
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 contingency tables comprised by two groups (one reference group and another containing remaining elements) and provided groups.

**Usage**

```r
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`: Contingency table used for testing
- `pvalue`: Fisher’s exact test’s p-value

See Also

`parseCategoricalGroups()` and `plotGroupIndependence()`

Other functions for data grouping: `createGroupByAttribute()`, `getGeneList()`, `getSampleFromSubject()`, `getSubjectFromSample()`, `groupPerElem()`, `plotGroupIndependence()`

Examples

```r
elements <- paste("subjects", 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 subject 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**: Contingency table used for testing
- **pvalue**: Fisher’s exact test’s p-value
testSurvival

Test the survival difference between groups of subjects

Description
Test the survival difference between groups of subjects

Usage
testSurvival(survTerms, ...)

Arguments

| survTerms | survTerms object: survival terms obtained after running processSurvTerms (see examples) |
| ...       | 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. |
| timefix   | process times through the aeqSurv function to eliminate potential roundoff issues. |

Value
p-value of the survival difference or NA

Note
Instead of raising errors, returns NA

See Also
Other functions to analyse survival: assignValuePerSubject(), getAttributesTime(), labelBasedOnCutoff(), optimalSurvivalCutoff(), plotSurvivalCurves(), plotSurvivalPvaluesByCutoff(), processSurvTerms(), survdiffTerms(), survfit.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,
classical,
...,
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 are used by default)
classical 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?
textSuggestions

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

Text Suggestions

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

---

traceInList  

Find an item in list of lists and return its coordinates

**Description**

Find an item in list of lists and return its coordinates

**Usage**

```
traceInList(ll, item)
```

---

transformData  

Transform data in data frame

**Description**

Transform data in data frame

**Usage**

```
transformData(input, df, x, y)
```
transformOptions

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; if NULL, 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)
trimWhitespace

**Arguments**

- **val**  
  Integer: values to transform  
- **type**  
  Character: type of transformation  
- **avoidZero**  
  Boolean: add the smallest non-zero number available (.Machine$double.xmin) to avoid infinity values following log-transformation (may not be plotted); useful for p-values of 0

**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**

```r
psichomics:::trimWhitespace(" hey there ")
psichomics:::trimWhitespace(c("pineapple ", "one two three", " sunken ship "))
```
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: subject attributes |

Value
NULL (function is only used to modify the Shiny session’s state or internal variables)
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, ask = FALSE)

**Arguments**

- **session**: Shiny session
- **id**: Character: identifier
- **...**: Additional arguments passed to fileBrowser(). Only used if value = NULL.
- **value**: Character: file or directory path
- **ask**: Boolean: ask user to pick a file using file browser?

**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 (function is only used to modify the Shiny session’s state or internal variables)

**Source**

https://github.com/wleepang/shiny-directory-input

---

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 and display statistical analysis*

Description

Includes interface containing the results

Usage

wilcox(data, groups, stat = NULL)
ttest(data, groups, stat = NULL)
levene(data, groups, stat = NULL)
fligner(data, groups, stat = NULL)
kruskal(data, groups, stat = NULL)
fisher(data, groups)
spearman(data, groups)

Arguments

data  Numeric, data frame or matrix: gene expression data or alternative splicing event quantification values (sample names are based on their names or colnames)

groups  List of sample names or vector containing the group name per data value (read Details); if NULL or a character vector of length 1, data values are considered from the same group

stat  Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)
Details

- ttest: unpaired t-test
- wilcox: Wilcoxon test
- levene: Levene's test
- fligner: Fligner-Killeen test
- kruskal: Kruskal test
- fisher: Fisher's exact test
- spearman: Spearman's test

Value

HTML elements

---

### .GEandAScorrelation

Display results of correlation analyses

Description

Plot, print and display as table the results of gene expression and alternative splicing

Usage

```r
## S3 method for class 'GEandAScorrelation'
x[genes = NULL, ASevents = NULL]
## S3 method for class 'GEandAScorrelation'
plot(
x, 
   autoZoom = FALSE,
   loessSmooth = TRUE,
   loessFamily = c("gaussian", "symmetric"),
   colour = "black",
   alpha = 0.2,
   size = 1.5,
   loessColour = "red",
   loessAlpha = 1,
   loessWidth = 0.5,
   fontSize = 12,
   ...,
   colourGroups = NULL,
   legend = FALSE,
   showAllData = TRUE,
   density = FALSE,
   densityColour = "blue",
   densityWidth = 0.5
```
## S3 method for class 'GEandAScorrelation'
print(x, ...)

## S3 method for class 'GEandAScorrelation'
as.table(x, pvalueAdjust = "BH", ...)

### Arguments

- **x**: `GEandAScorrelation` object obtained after running `correlateGEandAS()`
- **genes**: Character: genes
- **ASEvents**: Character: AS events
- **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
- **fontSize**: Numeric: plot font size
- **...**: 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.
- **colourGroups**: List of characters: sample colouring by group
- **legend**: Boolean: show legend for sample colouring?
- **showAllData**: Boolean: show data outside selected groups as a single group (coloured based on the colour argument)
- **density**: Boolean: contour plot of a density estimate
- **densityColour**: Character: line colour of contours
- **densityWidth**: Numeric: line width of contours
- **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

Plots, summary tables or results of correlation analyses

See Also

Other functions to correlate gene expression and alternative splicing: `correlateGEandAS()`

Examples

```r
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")

# Quick display of the correlation results per splicing event and gene
print(corr)

# Table summarising the correlation analysis results
as.table(corr)

# Correlation analysis plots
colourGroups <- list(Normal=paste("Normal", 1:3),
                     Tumour=paste("Cancer", 1:3))
attr(colourGroups, "Colour") <- c(Normal="#00C65A", Tumour="#EEE273")
plot(corr, colourGroups=colourGroups, alpha=1)
```
Index

* functions associated with GTEx data retrieval
  getDownloadsFolder, 61
  getGtexDataTypes, 67
  getGtexTissues, 68
  loadGtexData, 102
* functions associated with SRA data retrieval
  getDownloadsFolder, 61
  loadSRAproject, 106
* functions associated with TCGA data retrieval
  getDownloadsFolder, 61
  getTCGAdataTypes, 77
  isFirebrowseUp, 91
  loadTCGAdata, 107
  parseTCGAsampleTypes, 132
* functions for PSI quantification
  filterPSI, 51
  getSplicingEventTypes, 75
  listSplicingAnnotations, 96
  loadAnnotation, 97
  plotRowStats, 149
  quantifySplicing, 172
* functions for data grouping
  createGroupByAttribute, 32
  getGeneList, 62
  getSampleFromSubject, 71
  getSubjectFromSample, 76
  groupPerElem, 83
  plotGroupIndependence, 142
  testGroupIndependence, 206
* functions for gene expression pre-processing
  convertGeneIdentifiers, 27
  filterGeneExpr, 49
  normaliseGeneExpression, 111
  plotGeneExprPerSample, 141
  plotLibrarySize, 145
  plotRowStats, 149
* functions to analyse independent components
  performICA, 136
  plotICA, 143
* functions to analyse principal components
  calculateLoadingsContribution, 21
  performPCA, 137
  plotPCA, 146
  plotPCAVariance, 147
* functions to analyse survival
  assignValuePerSubject, 17
  getAttributesTime, 55
  labelBasedOnCutoff, 93
  optimalSurvivalCutoff, 114
  plotSurvivalCurves, 153
  plotSurvivalPvaluesByCutoff, 154
  processSurvTerms, 168
  survdiffTerms, 200
  survfit.survTerms, 202
  testSurvival, 209
* functions to correlate gene expression and alternative splicing
  [.GEandAScorrelation, 218
  correlateGEandAS, 28
* functions to get and set global variables
  getClinicalMatchFrom, 57
  getDifferentialExpression, 58
  getDifferentialSplicing, 59
  getGlobal, 63
  getGroups, 66
  getHighlightedPoints, 69
  getSelectedDataPanel, 72
* functions to load data
  loadGtexData, 102
  loadLocalFiles, 104
  loadSRAproject, 106
  loadTCGAdata, 107
* functions to load local files
function to perform and plot differential analyses
  diffAnalyses, 39
  plotDistribution, 139

function to prepare alternative splicing annotations
  parseSuppaAnnotation, 127
  prepareAnnotationFromEvents, 158

function to retrieve external information
  ensemblToUniprot, 44
  plotProtein, 148
  plotTranscripts, 157
  queryEnsemblByGene, 174

internal
  addObjectAttrs, 9
  addTCGAdata, 10
  analysesTableSet, 10
  appendNewGroups, 12
  appServer, 12
  appUI, 14
  areSplicingEvents, 15
  articleUI, 16
  assignColours, 16
  basicStats, 18
  blendColours, 18
  browseDownloadFolderInput, 19
  browserHistory, 19
  calculateInclusionLevels, 20
  checkFileFormat, 22
  checkFirebrowse, 22
  checkGroupType, 23
  checkIntegrity, 23
  checkSurvivalInput, 24
  clusterICAset, 24
  clusterSet, 25
  colourInputMod, 25
  createDataTab, 29
  createDensitySparklines, 29
  createEventPlotting, 30
  createGroup, 31
  createGroupByid, 33
  createGroupFromInput, 33
  createJunctionsTemplate, 34
  createOptimalSurvData, 35
  createSparklines, 36
  customRowMeans, 37
  diagramSplicingEvent, 38
  diffExpressionSet, 41
  diffSplicingSet, 41
  disableTab, 42
  discardOutsideSamplesFromGroups, 43
  display, 43
  downloadFiles, 44
  escape, 45
  eventPlotOptions, 46
  export_highcharts, 47
  exportGroupsToFile, 46
  fileBrowser, 47
  fileBrowserInfoInput, 48
  findASeventsFromGene, 53
  findEventData, 53
  geneExprFileInput, 54
  geneExprSurvSet, 54
  getNormalisationFilteringInterface, 55
  getClinicalDataForSurvival, 56
  getClinicalMatchFrom, 57
  getData, 57
  getDataRows, 58
  getDifferentialExpression, 58
  getDifferentialSplicing, 59
  getFirebrowseDateFormat, 61
  getGlobal, 63
  getGroups, 66
  getGtexDataURL, 68
  getHidden, 69
  getHighlightedPoints, 69
  getNumerics, 70
  getSelectedDataPanel, 72
  getServerFunctions, 73
  getSplicingEventCoordinates, 73
  getUiFunctions, 78
  getValidEvents, 78
  ggplotServer, 79
  ggplotTooltip, 80
  ggplotUI, 81
  globalSelectize, 81
  groupByAttribute, 82
  groupManipulation, 82
  groupManipulationInput, 83
  groupsServerOnce, 84
  hc_scatter, 86
  hchart.survfit, 85
  HTMLfast, 87
importGroupsFrom, 87
inclusionLevelsFilterInterface, 88
inclusionLevelsInterface, 88
inlineDialog, 89
insideFile, 90
is.whole, 90
isFunction, 91
isRStudioServer, 92
joinEventsPerType, 92
junctionString, 93
leveneTest, 94
linkToArticles, 95
linkToRunJS, 95
listAllAnnotations, 96
loadAnnotationHub, 98
loadBy, 99
loadCustomSplicingAnnotationSet, 99
loadedDataModal, 100
loadFile, 100
loadFileFormats, 101
loadFirebrowseFolders, 101
loadGtexExpressionSet, 102
loadGtexDataShiny, 103
loadGtexFile, 104
loadRequiredData, 105
loadSplicingQuantificationSet, 106
loadTCGAsampleMetadata, 108
matchGroupASeventsAndGenes, 109
matchGroupSubjectsAndSamples, 109
matchSplicingEventsWithGenes, 110
modTabPanel, 110
navSelectize, 111
operateOnGroups, 113
optimSurvDiffSet, 115
parseDateResponse, 116
parseFile, 117
parseFirebrowseMetadata, 117
parseMatsEvent, 118
parseMatsGeneric, 119
parseMisoEvent, 121
parseMisoEventID, 122
parseMisoGeneric, 123
parseMisoId, 126
parseSuppaEvent, 129
parseSuppaGeneric, 130
parseUniprotXML, 133
parseUrlsFromFirebrowseResponse, 133
parseVastToolsEvent, 134
parseVastToolsSE, 135
plotClusters, 138
plotPointsStyle, 148
plotSingleICA, 151
plottableXranges, 156
prepareEventPlotOptions, 159
prepareFileBrowser, 160
prepareFirebrowseArchives, 160
prepareGenePresentation, 161
prepareJunctionQuantSTAR, 162
preparePreMadeGroupForSelection, 163
prepareWordBreak, 164
preserveAttributes, 165
processButton, 165
processDatasetNames, 166
processSRAdata, 166
processSurvData, 167
processSurvival, 168
pubmedUI, 172
quantifySplicingSet, 173
queryEnsembl, 174
queryFirebrowseData, 175
queryPubMed, 176
queryUniprot, 177
readAnnot, 178
reduceDimensionality, 179
renameDuplicated, 180
renameGroups, 180
renderBoxplot, 181
renderDataTableSparklines, 182
renderGeneticInfo, 183
renderGroupInterface, 183
renderProteinInfo, 184
replaceStrInList, 184
rm.null, 185
roundDigits, 185
roundMinDown, 186
saveProcessedSRAdata, 186
selectGroupsUI, 187
selectizeGeneInput, 188
selectPreMadeGroup, 189
setFirebrowseData, 189
setLocalData, 190
setOperation, 190
setOperationIcon, 191
showAlert, 192
showGroupsTable, 193
sidebar, 194
significantDigits, 194
singleDiffAnalyses, 195
sortCoordinates, 196
startProcess, 196
startProgress, 197
styleModal, 198
subjectMultiMatchWarning, 199
subsetGeneExpressionFromMatchingGenes, 200
tabDataset, 204
table2html, 205
tableRow, 206
testSingleIndependence, 208
testSurvivalCutoff, 210
textSuggestions, 211
toJSarray, 212
traceInList, 212
transformData, 212
transformOptions, 213
transformValues, 213
trimWhitespaces, 214
uniqueBy, 215
updateClinicalParams, 215
updateFileBrowserInput, 216
vennEvents, 216
wilcox, 217
.onAttach, 9
[.GEandAScorrelation, 28, 218
[.sticky (t.sticky), 203
addObjectAttrs, 9
addTCGAdata, 10
analysesPlotSet (analysesTableSet), 10
analysesServer (appServer), 12
analysesTableSet, 10
analysesUI (appUI), 14
appendNewGroups, 12
approx, 140
appServer, 12
appUI, 14
areSplicingEvents, 15
articleUI, 16
as.numeric, 179
as.table .GEandAScorrelation
  ([.GEandAScorrelation), 218
ASquantFileInput (geneExprFileInput), 54
assignColours, 16
assignValuePerPatient
  (assignValuePerSubject), 17
assignValuePerSubject, 17, 56, 94, 115,
  154, 155, 170, 201, 203, 209
basicStats, 18
blendColours, 18
browseDownloadFolderInput, 19
browserHistory, 19
bw.nrd, 139
calcNormFactors, 111
calculateInclusionLevels, 20
calculateLoadingsContribution, 21, 138,
  146, 147
checkFileFormat, 22
checkFirebrowse, 22
checkGroupType, 23
checkIntegrity, 23
checkSurvivalInput, 24
closeProgress (startProgress), 197
clusterICAset, 24
clusterSet, 25
colourInputMod, 25
colourpicker::colourInput, 25
colSums, elist-method, 26
customColMedians, 27, 50, 113, 142,
  145, 150
cor.test, 28
correlateGEandAS, 28, 219, 220
correlationServer (appServer), 12
correlationUI (appUI), 14
cpm, 111
createtabData, 29
createDensitySparklines, 29
createEventPlotting, 30
createGroup, 31
createGroupByAttribute, 32, 62, 72, 76, 84,
  143, 207
createGroupById, 33
createGroupFromInput, 33
createJunctionsTemplate, 34
createOptimalSurvData, 35
createSparklines, 36
customColMedians (customRowMeans), 37
customRowMaxs (customRowMeans), 37
customRowMeans, 37
customRowMedians (customRowMeans), 37
INDEX

customRowRanges (customRowMeans), 37

dataServer (appServer), 12
dataUI (appUI), 14
diffAnalyses, 39, 141
diffEventServer (appServer), 12
diffEventUI (appUI), 14
diffExpressionEventServer (appServer), 12
diffExpressionEventUI (appUI), 14
diffExpressionSet, 41
diffExpressionTableServer (appServer), 12
diffExpressionTableUI (appUI), 14
diffSplicingEventServer (appServer), 12
diffSplicingEventUI (appUI), 14
diffSplicingServer (appServer), 12
diffSplicingSet, 41
diffSplicingTableServer (appServer), 12
diffSplicingTableUI (appUI), 14
diffSplicingUI (appUI), 14
dimReductionServer (appServer), 12
dimReductionUI (appUI), 14
discardLowCoveragePSIvalues, 42
discardOutsideSamplesFromGroups, 43
display, 43
downloadFiles, 44

geneExprFileInput, 54
geneExprSurvSet, 54
geneNormalisationFilteringInterface, 55
geneNormalisationFilteringServer (appServer), 12
geneNormalisationFilteringUI (appUI), 14
gm_point, 31
getActiveDataset (getGlobal), 63
getAnnotationHub (getGlobal), 63
getAnnotationName (getGlobal), 63
getASevent (getGlobal), 63
getASevents (getGlobal), 63
getAddress (getGlobal), 63
getAttributesTime, 17, 55, 94, 115, 154, 155, 167, 169, 170, 201, 203, 209
getAutoNavigation (getGlobal), 63
getCategories (getGlobal), 63
getCategory (getGlobal), 63
getCategoryData (getGlobal), 63
getClinicalData (getGlobal), 63
getClinicalDataForSurvival, 56
getClinicalMatchFrom, 57, 59, 60, 66, 67, 70, 73
getCores (getGlobal), 63
getCorrelation (getGlobal), 63
data, 57
dataRows, 58
differentialExpression, 57, 58, 60, 66, 67, 70, 73
differentialExpressionColumns (differentialExpression), 58
differentialExpressionFiltered (differentialExpression), 58
getDifferentialExpressionResetPaging
  (getDifferentialExpression), 58
getDifferentialExpressionSurvival
  (getDifferentialExpression), 58
getDifferentialSplicing, 57, 59, 66, 67, 70, 73
getDifferentialSplicingColumns
  (getDifferentialSplicing), 59
getDifferentialSplicingFiltered
  (getDifferentialSplicing), 59
getDifferentialSplicingResetPaging
  (getDifferentialSplicing), 59
getDifferentialSplicingSurvival
  (getDifferentialSplicing), 59
dGetDownloadsFolder, 61, 67, 69, 77, 91, 103, 107, 108, 132
gEvent (getGlobal), 63
gFirebrowseCohorts
  (getTCGAdataTypes), 77
gFirebrowseDataTypes
  (getTCGAdataTypes), 77
gFirebrowseDateFormat, 61
gFirebrowseDates (getTCGAdataTypes), 77
gGeneExpression (getGlobal), 63
gGeneList, 32, 62, 72, 76, 84, 143, 207
gGenes (getGlobal), 63
gGenesFromSplicingEvents
  (getSplicingEventFromGenes), 74
gGlobal, 57, 59, 60, 63, 67, 70, 73
gGroupIndependenceTesting
  (getGlobal), 63
gGroups, 57, 59, 60, 66, 67, 70, 73
gGtexDataTypes, 61, 67, 69, 103
gGtexDataURL, 68
gGtexReleases (getGtexDataTypes), 67
gGtexTissues, 61, 67, 68, 103
gHidden, 69
gHighlightedPoints, 57, 59, 60, 66, 67, 70, 73
gICA (getGlobal), 63
gGetInclusionLevels (getGlobal), 63
gGetInclusionLevelsSummaryStatsCache
  (getGlobal), 63
gJunctionQuantification (getGlobal), 63
gLabelledPoints
  (getHighlightedPoints), 69
gMatchingSamples
  (getSampleFromSubject), 71
gNumerics, 70
gPatientFromSample
  (getSubjectFromSample), 76
gPCA (getGlobal), 63
gPrecision (getGlobal), 63
gSampleAttributes (getGlobal), 63
gSampleFromPatient
  (getSampleFromSubject), 71
gSampleFromSubject, 32, 62, 71, 76, 84, 143, 207
gSampleId (getGlobal), 63
gSampleInfo (getGlobal), 63
gSelectedDataPanel, 57, 59, 60, 66, 67, 70, 72
gSelectedGroups (selectGroupsUI), 187
gSelectedPoints
  (getHighlightedPoints), 69
gServerFunctions, 73
gSignificant (getGlobal), 63
gSpecies (getGlobal), 63
gSplicingEventCoordinates, 73
gSplicingEventData, 74
gSplicingEventFromGenes, 74
gSplicingEventTypes, 52, 75, 97, 98, 150, 173
gSubjectAttributes (getGlobal), 63
gSubjectFromSample, 32, 62, 72, 76, 84, 143, 207
gSubjectId (getGlobal), 63
gTCGAcohorts (getTCGAdataTypes), 77
gTCGAdataTypes, 61, 77, 91, 107, 108, 132
gTCGAdates (getTCGAdataTypes), 77
gUiFunctions, 78
gURLtoDownload (getGlobal), 63
gValidEvents, 78
gValuePerPatient
  (assignValuePerSubject), 17
gValuePerSubject
  (assignValuePerSubject), 17
gZoom (getHighlightedPoints), 69
ggplot, 79, 81
ggplotAuxServer (ggplotServer), 79
ggplotServer, 79
ggplotTooltip, 80
ggplotUI, 81
globalSelectize, 81
groupByAttribute, 82
groupByExpression (groupByAttribute), 82
groupByGrep (groupByAttribute), 82
groupByID (groupByAttribute), 82
groupByPreMadeList (groupByAttribute), 82
groupManipulation, 82
groupManipulationInput, 83
groupPerElem, 32, 62, 72, 76, 83, 143, 207
groupsServer (appServer), 12
groupsServerOnce, 84
groupsUI (appUI), 14
gtexDataServer (appServer), 12
gtexDataUI (appUI), 14
hc_scatter, 86
hchart.survfit, 85
helpServer (appServer), 12
helpUI (appUI), 14
highchart::hc_add_series, 85, 86
hoverOpts, 80
HTMLfast, 87
icaServer (appServer), 12
icaUI (appUI), 14
icon, 191
icon(), 165
importGroupsFrom, 87
inclusionLevelsFilterInterface, 88
inclusionLevelsFilterServer (appServer), 12
inclusionLevelsFilterUI (appUI), 14
inclusionLevelsInterface, 88
inclusionLevelsServer (appServer), 12
inclusionLevelsUI (appUI), 14
infoModal (styleModal), 198
infoServer (appServer), 12
infoUI (appUI), 14
inlineDialog, 89
insideFile, 90
is.numeric, 179
is.whole, 90
isFile, 91
isFirebrowseUp, 61, 77, 91, 108, 132
isRStudioServer, 92
joinEventsPerType, 92
junctionQuantFileInput (geneExprFileInput), 54
junctionString, 93
kruskal (wilcox), 217
labelBasedOnCutoff, 17, 56, 93, 115, 154, 155, 170, 201, 203, 209
levene (wilcox), 217
leveneTest, 94
linkToArticles, 95
linkToRunJS, 95
listAllAnnotations, 96
listSplicingAnnotations, 52, 76, 96, 98, 150, 173
loadAnnotation, 52, 76, 97, 97, 150, 173
loadAnnotationHub, 98
loadBy, 99
loadCustomSplicingAnnotationSet, 99
loadDataModal, 100
loadFile, 100
loadFileFormats, 101
loadFirebrowseData (loadTCGAdata), 107
loadFirebrowseFolders, 101
loadGeneExpressionSet, 102
loadGtexData, 61, 67, 69, 102, 105, 107, 108
loadGtexDataShiny, 103
loadGtexFile, 104
loadLocalFiles, 103, 104, 107, 108
loadRequiredData, 105
loadSplicingQuantificationSet, 106
loadSRAproject, 61, 103, 105, 106, 108
loadTCGAdata, 61, 77, 91, 103, 105, 107, 107, 132
loadTCGAsampleMetadata, 108
localDataServer (appServer), 12
localDataUI (appUI), 14
logical, 140
matchGroupASeventsAndGenes, 109
matchGroupSubjectsAndSamples, 109
matchSplicingEventsWithGenes, 110
missingDataGuide (loadRequiredData), 105
missingDataModal (loadRequiredData), 105
model.frame, 136, 138
modTabPanel, 110
navSelectize, 111
normaliseGeneExpression, 27, 50, 111, 142, 145, 150
normalizeGeneExpression (normaliseGeneExpression), 111
prepareAnnotationFromEvents, 129, 158
prepareEventPlotOptions, 159
prepareFileBrowser, 49, 160
prepareFirebrowseArchives, 160
prepareGenePresentation, 161
prepareGeneQuant (prepareSRAmetadata), 163
prepareGeneQuantSTAR (prepareJunctionQuantSTAR), 162
prepareJunctionQuant (prepareSRAmetadata), 163
prepareJunctionQuantSTAR, 162
preparePreMadeGroupForSelection, 163
prepareSRAmetadata, 163
prepareWordBreak, 164
print.GEandAScorrelation ([.GEandAScorrelation), 218
processButton, 165
processClickRedirection (analysesTableSet), 10
processDatasetNames, 166
processSRAData, 166
processSurvData, 167
processSurvival, 168
processSurvTerms, 17, 56, 94, 115, 154, 155, 168, 168, 201, 203, 209, 210
psichomics, 170
pubmedUI, 172
quantifySplicing, 52, 76, 97, 98, 150, 172
quantifySplicingSet, 173
queryEnsembl, 174
queryEnsemblByEvent (queryEnsemblByGene), 174
queryEnsemblByGene, 45, 149, 157, 174
queryFirebrowseData, 108, 175
queryPubMed, 172, 176
queryUniprot, 177
quote(), 182
readAnnot, 178
readFile, 178
recount_abstract, 107
recountDataServer (appServer), 12
recountDataUI (appUI), 14
reduceDimensionality, 179
removeAlert (showAlert), 192
removeModal(), 198
renameDuplicated, 180
renameGroups, 180
renderBoxplot, 142, 181
renderDataTable, 182
renderDataTableSparklines, 182
renderGeneticInfo, 183
renderGroupInterface, 183
renderProteinInfo, 184
replaceStrInList, 184
rm.null, 185
roundDigits, 185
roundMaxUp (roundMinDown), 186
roundMinDown, 186
sampleInfoFileInput (geneExprFileInput), 54
saveProcessedSRAData, 186
scale, 137, 138
selectGroupsServer (selectGroupsUI), 187
selectGroupsUI, 187
selectizeGeneInput, 188
selectPreMadeGroup, 189
setActiveGeneInput, 188
setDifferentialExpression (getDifferentialExpression), 58
setDifferentialExpression (getDifferentialExpressionColumns)
setDifferentialExpression (getDifferentialExpressionFiltered)
setDifferentialExpressionResetPaging (getDifferentialExpression)
setDifferentialExpressionSurvival (getDifferentialExpression)
setDifferentialExpression (getDifferentialExpressionSurvival)
setDifferentialExpressionSplicing (getDifferentialExpressionSplicing)
setDifferentialExpressionColumns (getDifferentialExpressionColumns)
setDifferentialSplicingFiltered (getDifferentialSplicing), 59
setDifferentialSplicingResetPaging (getDifferentialSplicing), 59
setDifferentialSplicingSurvival (getDifferentialSplicing), 59
setEvent (getGlobal), 63
setFirebrowseData, 189
setGlobal (getGlobal), 63
setGroupIndependenceTesting (getGlobal), 63
setGroups (getGroups), 66
setHidden (getHidden), 69
setHighlightedPoints (getHighlightedPoints), 69
setICA (getGlobal), 63
setInclusionLevels (getGlobal), 63
setInclusionLevelsSummaryStatsCache (getGlobal), 63
setLabelledPoints (getHighlightedPoints), 69
setLocalData, 190
setMultipleFilesData (setLocalData), 190
setNormalisedGeneExpression (getGlobal), 63
setOperation, 190
setOperationIcon, 191
setPCA (getGlobal), 63
setPrecision (getGlobal), 63
setSampleInfo (getGlobal), 63
setSelectedDataPanel (getSelectedDataPanel), 72
setSelectedPoints (getHighlightedPoints), 69
setSignificant (getGlobal), 63
setSpecies (getGlobal), 63
setURLtoDownload (getGlobal), 63
setZoom (getHighlightedPoints), 69
shiny::actionButton, 165
shiny::modalDialog, 198
shiny::renderDataTable, 182
shiny::runApp, 171
showAlert, 192, 199
showGroupsTable, 193
showModal, 193
sidebar, 194
signifDigits, 194
singleDiffAnalyses, 195
sortCoordinates, 196
spearman (wilcox), 217
startProcess, 196
startProgress, 197
stats::density.default, 139
stats::loess.smooth, 219
styleModal, 198
subjectInfoFileInput (geneExprFileInput), 54
subjectMultiMatchWarning, 199
subsetGeneExpressionFromMatchingGenes, 200
successAlert (showAlert), 192
survdiffTerms, 17, 56, 94, 115, 154, 155, 170, 200, 203, 209
survfit.survTerms, 17, 56, 85, 94, 115, 154, 155, 170, 201, 202, 209
survival::survdiff, 200, 202, 209
survivalServer (appServer), 12
survivalUI (appUI), 14
t.sticky, 203
tabDataset, 204
table2html, 205
tableRow, 206
templateServer (appServer), 12
templateUI (appUI), 14
testGroupIndependence, 32, 62, 72, 76, 84, 116, 143, 206
testSingleIndependence, 208
testSurvival, 17, 56, 94, 115, 154, 155, 170, 201, 203, 209
testSurvivalCutoff, 210
textSuggestions, 211
toJSarray, 212
traceInList, 212
transformData, 212
transformOptions, 213
transformValues, 213
trimWhitespace, 214
ttest (wilcox), 217
uniqueBy, 215
updateClinicalParams, 215
updateFileBrowserInput, 49, 216
updateProgress (startProgress), 197
validateCssUnit(), 165
vennEvents, 216
INDEX

voom, 111, 112

warning, 140
warningAlert (showAlert), 192
warningDialog (inlineDialog), 89
warningModal (styleModal), 198
wilcox, 217