Package ‘psichomics’

March 29, 2017

Title Graphical Interface for Alternative Splicing Quantification, Analysis and Visualisation

Version 1.0.8

Encoding UTF-8

Description Package with a Shiny-based graphical interface for the integrated analysis of alternative splicing data from The Cancer Genome Atlas (TCGA). This tool interactively performs survival, principal components and differential splicing analyses with direct incorporation of clinical features (such as tumour stage or survival) associated with TCGA samples.

Depends R (>= 3.3), shiny (>= 1.0.0), shinyBS

License MIT + file LICENSE

LazyData true

RoxygenNote 6.0.1

Imports AnnotationHub, data.table, digest, dplyr, DT (>= 0.2), fastmatch, highcharter (>= 0.5.0), htr, jsonlite, miscTools, plyr, R.utils, shinyjs, stringr, stats, survival, Sushi, tools, utils, XML, methods

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

VignetteBuilder knitr

Collate 'analysis.R' 'analysis_diffSplicing.R'
'analysis_diffSplicing_event.R' 'analysis_diffSplicing_table.R'
'analysis_information.R' 'analysis_pca.R' 'analysis_survival.R'
'analysis_template.R' 'utils.R' 'globalAccess.R' 'app.R'
'data.R' 'formats.R' 'data_firebrowse.R' 'data_gtex.R'
'data_inclusionLevels.R' 'data_local.R' 'events_suppa.R'
'events_vastTools.R' 'events_miso.R' 'events_mats.R' 'events.R'
'formats_firehoseGeneExpression.R'
'formats_firehoseJunctionReads.R'
'formats_firehoseMergeClinical.R' 'formats_gtexClinical.R'
'formats_gtexJunctionReads.R' 'formats_gtexSampleInfo.R'
'groups.R' 'settings.R'

biocViews Sequencing, RNASeq, AlternativeSplicing, DifferentialSplicing, Transcription, GUI, PrincipalComponent, Survival, BiomedicalInformatics, Transcriptomics, Visualization, MultipleComparison
URL https://github.com/nuno-agostinho/psichomics

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

NeedsCompilation no

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

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

R topics documented:

addTCGAd ata .......................................................... 7
analysesServer ....................................................... 7
analysesUI ........................................................... 8
appServer ............................................................ 8
appUI ................................................................. 9
articleUI .............................................................. 9
basicStats ............................................................ 10
browserHistory ..................................................... 10
bsModal2 .............................................................. 11
calculateInclusionLevels ........................................ 11
checkFileFormat ..................................................... 12
checkFirebrowse .................................................... 12
checkIntegrity ....................................................... 13
checkSurvivalInput ................................................ 13
closeProgress ........................................................ 14
createDataTab ....................................................... 14
createDensitySparklines ......................................... 15
createGroup ........................................................ 15
createGroupByAttribute ......................................... 16
createGroupByColumn ............................................. 16
createGroupById ................................................... 17
createGroupFromInput ............................................ 17
createJunctionsTemplate ......................................... 18
dataServer .......................................................... 18
dataUI ................................................................. 19
diffAnalyses ........................................................ 19
diffSplicingEventServer ........................................ 20
diffSplicingEventUI ............................................... 21
diffSplicingServer ................................................ 21
diffSplicingTableServer ......................................... 22
diffSplicingTableUI ............................................... 22
diffSplicingUI ....................................................... 23
disableTab ............................................................ 23
downloadFiles ....................................................... 24
R topics documented:

echoProgress .................................................. 24
enableTab ...................................................... 25
endProcess ..................................................... 25
ensemblToUniprot ............................................ 26
escape .......................................................... 26
export_highcharts ............................................ 27
filterGroups .................................................... 27
firebrowseUI .................................................... 28
fisher ............................................................ 28
fligner ............................................................ 29
getActiveDataset ............................................. 29
getAssemblyVersion .......................................... 30
getAutoNavigation ............................................ 30
getCategories .................................................. 31
getCategory ..................................................... 31
getCategoryData ............................................... 31
getClinicalData ............................................... 32
getClinicalMatchFrom ....................................... 32
getColumnsTime ............................................... 33
getCores ........................................................ 33
gGetData ........................................................ 34
gGetDataRows ................................................... 34
getDifferentialAnalyses ...................................... 35
getDifferentialAnalysesSurvival ......................... 35
getDownloadsFolder ........................................... 36
gETEvent ......................................................... 36
getFirehoseCohorts .......................................... 36
getFirehoseDataTypes ....................................... 37
getFirehoseDateFormat ..................................... 37
getFirehoseDates ............................................. 38
getGlobal ........................................................ 38
getGroupsFrom ............................................... 39
getInclusionLevels .......................................... 39
getInclusionLevelsPCA ...................................... 40
getJunctionQuantification .................................. 40
getMatchingSamples ......................................... 41
getNumerics .................................................... 41
getPatientFromSample ....................................... 42
getPatientId .................................................... 42
getPrecision ..................................................... 43
getPSIperPatient ............................................. 43
getSampleId ..................................................... 44
getSampleInfo .................................................. 45
getSelectedGroups ............................................ 45
getServerFunctions ........................................... 46
getSignificant .................................................. 46
getSpecies ....................................................... 47
getSplicingEventCoordinates .............................. 47
getSplicingEventTypes ...................................... 48
getUiFunctions ................................................ 48
getURLtoDownload ............................................ 49
getValidEvents ............................................... 49
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>globalSelectize</td>
<td>50</td>
</tr>
<tr>
<td>groupByAttribute</td>
<td>51</td>
</tr>
<tr>
<td>groupByExpression</td>
<td>51</td>
</tr>
<tr>
<td>groupByGrep</td>
<td>52</td>
</tr>
<tr>
<td>groupById</td>
<td>52</td>
</tr>
<tr>
<td>groupPerPatient</td>
<td>53</td>
</tr>
<tr>
<td>groupPerSample</td>
<td>53</td>
</tr>
<tr>
<td>groupsServer</td>
<td>54</td>
</tr>
<tr>
<td>groupsServerOnce</td>
<td>54</td>
</tr>
<tr>
<td>groupsUI</td>
<td>55</td>
</tr>
<tr>
<td>gtxDataServer</td>
<td>55</td>
</tr>
<tr>
<td>gtxDataUI</td>
<td>56</td>
</tr>
<tr>
<td>hchart.survfit</td>
<td>56</td>
</tr>
<tr>
<td>hc_scatter</td>
<td>57</td>
</tr>
<tr>
<td>inclusionLevelsInterface</td>
<td>58</td>
</tr>
<tr>
<td>inclusionLevelsServer</td>
<td>58</td>
</tr>
<tr>
<td>inclusionLevelsUI</td>
<td>59</td>
</tr>
<tr>
<td>infoServer</td>
<td>59</td>
</tr>
<tr>
<td>infoUI</td>
<td>60</td>
</tr>
<tr>
<td>insideFile</td>
<td>60</td>
</tr>
<tr>
<td>is.whole</td>
<td>61</td>
</tr>
<tr>
<td>isFirehoseUp</td>
<td>61</td>
</tr>
<tr>
<td>joinEventsPerType</td>
<td>62</td>
</tr>
<tr>
<td>junctionString</td>
<td>62</td>
</tr>
<tr>
<td>kruskal</td>
<td>63</td>
</tr>
<tr>
<td>labelBasedOnCutoff</td>
<td>63</td>
</tr>
<tr>
<td>leven</td>
<td>64</td>
</tr>
<tr>
<td>leveneTest</td>
<td>64</td>
</tr>
<tr>
<td>listAllAnnotations</td>
<td>65</td>
</tr>
<tr>
<td>listSplicingAnnotations</td>
<td>66</td>
</tr>
<tr>
<td>loadAnnotation</td>
<td>66</td>
</tr>
<tr>
<td>loadBy</td>
<td>67</td>
</tr>
<tr>
<td>loadedDataModal</td>
<td>67</td>
</tr>
<tr>
<td>loadFile</td>
<td>68</td>
</tr>
<tr>
<td>loadFileFormats</td>
<td>68</td>
</tr>
<tr>
<td>loadFirehoseData</td>
<td>69</td>
</tr>
<tr>
<td>loadFirehoseFolders</td>
<td>69</td>
</tr>
<tr>
<td>loadGtexData</td>
<td>70</td>
</tr>
<tr>
<td>loadLocalFiles</td>
<td>70</td>
</tr>
<tr>
<td>localDataServer</td>
<td>71</td>
</tr>
<tr>
<td>localDataUI</td>
<td>72</td>
</tr>
<tr>
<td>missingDataModal</td>
<td>72</td>
</tr>
<tr>
<td>modTabPanel</td>
<td>73</td>
</tr>
<tr>
<td>navSelectize</td>
<td>73</td>
</tr>
<tr>
<td>noinfo</td>
<td>74</td>
</tr>
<tr>
<td>operateOnGroups</td>
<td>74</td>
</tr>
<tr>
<td>optimalPSICutoff</td>
<td>75</td>
</tr>
<tr>
<td>optimSurvDiff</td>
<td>76</td>
</tr>
<tr>
<td>optimSurvDiffOptions</td>
<td>77</td>
</tr>
<tr>
<td>parseDateResponse</td>
<td>77</td>
</tr>
<tr>
<td>parseFirehoseMetadata</td>
<td>78</td>
</tr>
<tr>
<td>parseMatsEvent</td>
<td>78</td>
</tr>
</tbody>
</table>
R topics documented:

- parseMatsGeneric ........................................ 79
- parseMisoEvent ........................................... 81
- parseMisoEventID ......................................... 82
- parseMisoGeneric ......................................... 83
- parseMisoID ............................................... 86
- parseSampleGroups ........................................ 86
- parseSplicingEvent ........................................ 87
- parseSuppaAnnotation .................................... 87
- parseSuppaEvent .......................................... 89
- parseSuppaGeneric ........................................ 90
- parseTegaSampleInfo ..................................... 91
- parseUniprotXML .......................................... 92
- parseUrlsFromFirehoseResponse ......................... 92
- parseValidFile ........................................... 93
- parseVastToolsEvent ..................................... 93
- parseVastToolsSE ......................................... 94
- pcaServer ................................................ 95
- pcaUI ..................................................... 96
- performPCA .............................................. 96
- plotDistribution ......................................... 97
- plotMiniSurvivalCurves .................................. 97
- plotPCA .................................................. 98
- plotProtein ............................................... 99
- plotSurvivalCurves ...................................... 99
- plotTranscripts ......................................... 100
- plotVariance ............................................ 101
- prepareAnnotationFromEvents ......................... 101
- prepareFirehoseArchives ................................ 102
- processButton ........................................... 103
- processDatasetNames .................................... 104
- processSurvData ........................................ 104
- processSurvival ........................................ 105
- processSurvTerms ........................................ 105
- psichomics ............................................... 107
- pubmedUI ................................................. 107
- quantifySplicing ........................................ 108
- queryEnsembl ............................................ 108
- queryEnsemblByEvent .................................... 109
- queryEnsemblByGene ...................................... 109
- queryFirehoseData ...................................... 110
- queryPubMed ............................................. 111
- queryUniprot ............................................ 111
- readFile ................................................ 112
- renameDuplicated ....................................... 112
- renameGroups ........................................... 113
- renderDataTableSparklines ............................... 114
- renderGeneticInfo ....................................... 114
- rm.null .................................................. 115
- roundDigits .............................................. 115
- rowVar .................................................... 116
- selectGroupsServer ..................................... 116
- selectGroupsUI .......................................... 117
R topics documented:

setActiveDataset ......................................................... 117
setAssemblyVersion ...................................................... 118
setAutoNavigation ....................................................... 118
setCategory ............................................................... 119
setClinicalMatchFrom .................................................. 119
setCores ................................................................. 120
setData ................................................................. 120
setDifferentialAnalyses .................................................. 121
setDifferentialAnalysesSurvival ...................................... 121
setEvent ................................................................. 122
setFirehoseData ......................................................... 122
setGlobal ................................................................. 123
setGroupsFrom ............................................................ 123
setInclusionLevels ...................................................... 124
setInclusionLevelsPCA .................................................. 124
setLocalData .............................................................. 125
setPatientId ............................................................... 125
setPrecision .............................................................. 126
setSampleId ............................................................... 126
setSampleInfo ............................................................. 127
setSignificant ............................................................ 127
setSpecies ................................................................. 128
settingsServer ........................................................... 128
settingsUI ................................................................. 129
setURLtoDownload ....................................................... 129
showAlert ................................................................. 130
showGroupsTable ........................................................ 130
signifDigits ............................................................... 131
singleDiffAnalyses ....................................................... 131
sortCoordinates .......................................................... 132
spearman ................................................................. 132
startProcess .............................................................. 133
startProgress ............................................................ 133
styleModal ................................................................. 134
survdiff.survTerms ...................................................... 135
 survfit.survTerms ....................................................... 135
tabDataset ............................................................... 136
survivalServer ........................................................... 137
templateServer ........................................................... 138
templateUI ................................................................. 138
testSurvival .............................................................. 139
testSurvivalCutoff ....................................................... 139
textSuggestions ............................................................ 140
timePerPatient ............................................................ 141
trimWhitespace ........................................................... 141
ttest ....................................................................... 142
uniqueBy ................................................................. 142
updateClinicalParams .................................................... 143
updateProgress ........................................................... 143
vennEvents ............................................................... 144
wilcox .................................................................... 144
addTCGAdat* 

Index 145

addTCGAdat*  Creates a UI set with options to add data from TCGA/Firehose

Description

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

Usage

addTCGAdat*(ns)

Arguments

ns  Namespace function

Value

A UI set that can be added to a UI definition

analysesServer  Server logic for the analyses

Description

Server logic for the analyses

Usage

analysesServer(input, output, session)

Arguments

input  Shiny input
output  Shiny output
session  Shiny session

Value

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

**User interface for the data analyses**

**Description**

User interface for the data analyses

**Usage**

```r
analysesUI(id, tab)
```

**Arguments**

- `id` Character: identifier
- `tab` Function to process HTML elements

**Value**

HTML element as character

---

appServer

**Server function**

**Description**

Instructions to build the Shiny app.

**Usage**

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

**Arguments**

- `input` Input object
- `output` Output object
- `session` Session object

**Value**

NULL (this function is used to modify the Shiny session’s state)
**Description**

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

**Usage**

`appUI()`

**Value**

HTML elements

---

**Description**

Return the interface to display an article.

**Usage**

`articleUI(article)`

**Arguments**

- `article` : PubMed article

**Value**

HTML to render an article’s interface
basicStats  

**Basic statistics performed on data**

**Description**

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

**Usage**

```
basicStats(psi, groups)
```

**Arguments**

- **psi**  
  Numeric: quantification of one alternative splicing event
- **groups**  
  Character: group of each PSI index

**Value**

HTML elements

---

browserHistory  

**Enable history navigation**

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

NULL (this function is used to modify the Shiny session’s state)
bsModal2

Modified version of shinyBS::bsModal

Description

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

Usage

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

Arguments

id A unique identifier for the modal window

title The title to appear at the top of the modal

trigger The id of a button or link that will open the modal.

... UI elements to include within the modal

size Character: Modal size ("small", "default" or "large")

footer UI set: List of elements to include in the footer

style Character: message style can be "warning", "error", "info" or NULL

Value

HTML element to create a modified modal

calculateInclusionLevels

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

Description

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

Usage

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

Arguments

eventType Character: type of the alternative event to calculate

junctionQuant Data.frame: junction quantification with samples as columns and junctions as rows

annotation Data.frame: alternative splicing annotation related to event type

minReads Integer: minimum of total reads required to consider the quantification as valid (10 by default)
checkFileFormat

Description
Checks the format of a file

Usage
checkFileFormat(format, head, filename)

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

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

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

checkFirebrowse

Description
Return an user interface depending on the status of the Firebrowse API

Usage
checkFirebrowse(ns)

Arguments
ns Namespace function

Value
HTML elements
**checkIntegrity**

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

**Description**

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

**Usage**

`checkIntegrity(filesToCheck, md5file)`

**Arguments**

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

**Value**

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

**checkSurvivalInput**

*Prepare survival terms in case of valid input*

**Description**

Prepare survival terms in case of valid input

**Usage**

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

**Arguments**

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

**Value**

NULL (this function is used to modify the Shiny session’s state)
closeProgress  
\textit{Close the progress even if there's an error}

\textbf{Description}

Close the progress even if there’s an error

\textbf{Usage}

\begin{verbatim}
closeProgress(message = NULL, global = sharedData)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \textbf{message}  \hspace{1cm} \text{Character: message to show in progress bar}
  \item \textbf{global}  \hspace{1cm} \text{Global Shiny variable where all data is stored}
\end{itemize}

\textbf{Value}

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

createDataTab  
\textit{Render a specific data tab (including data table and related interface)}

\textbf{Description}

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

\textbf{Usage}

\begin{verbatim}
createDataTab(index, data, name, input, output)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \textbf{index}  \hspace{1cm} \text{Integer: index of the data to load}
  \item \textbf{data}  \hspace{1cm} \text{Data frame: data with everything to load}
  \item \textbf{name}  \hspace{1cm} \text{Character: name of the dataset}
  \item \textbf{input}  \hspace{1cm} \text{Shiny session input}
  \item \textbf{output}  \hspace{1cm} \text{Shiny session output}
\end{itemize}

\textbf{Value}

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

Create density sparklines for inclusion levels

Description

Create density sparklines for inclusion levels

Usage

createDensitySparklines(data, events, delim = NULL)

Arguments

data [-v] Character: HTML-formatted data series of interest
events [-v] Character: event identifiers
delim [delim] Character: left and right delimiters in groups that should be removed

Value

HTML element with sparkline data (character)

dcreateGroup

Prepare to create group according to specific details

Description

Prepare to create group according to specific details

Usage

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

Arguments

session [-v] Shiny session
input [-v] Shiny input
output [-v] Shiny output
id [id] Character: identifier of the group selection
type [type] Character: type of group to create

Value

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

Create groups with the indexes from the unique values of a given column from a dataset

Description
Create groups with the indexes from the unique values of a given column from a dataset

Usage
createGroupByAttribute(col, dataset)

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

Value
Named list with the indexes of each unique value from a given column

Examples
```r
df <- data.frame(gender=c("male", "female"),
                 stage=paste("stage", c(1, 3, 1, 4, 2, 3, 2, 2)))
createGroupByAttribute(col="stage", dataset=df)
```

createGroupByColumn

Create groups with the indexes from the unique values of a given column from a dataset

Description
Create groups with the indexes from the unique values of a given column from a dataset

Usage
createGroupByColumn(col, dataset)

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

Value
Named list with the indexes of each unique value from a given column
createGroupById

Create groups from a given string of rows

Description

Create groups from a given string of rows

Usage

createGroupById(session, rows, dataset, identifiers)

Arguments

- **session**: Shiny session
- **rows**: Character: rows separated by a comma
- **dataset**: Matrix or data frame: dataset
- **identifiers**: Character: available identifiers

Value

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

createGroupFromInput

Set new groups according to the user input

Description

Set new groups according to the user input

Usage

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

Arguments

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

Value

Matrix with the group names and respective indexes
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: Number of rows  
- `program`: Character: Program used to get the junctions  
- `event.type`: Character: Event type of the respective events  
- `chromosome`: Character: Chromosome of the junctions  
- `strand`: Character: positive ("+") or negative ("-“) strand of the event  
- `id`: Character: events’ ID

**Value**

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

**Examples**

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

---

**dataServer**

*Server logic of the data module*

**Description**

Server logic of the data module

**Usage**

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

**Arguments**

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

**Value**

Part of the server logic related to this tab
**dataUI**

*User interface of the data module*

**Description**

User interface of the data module

**Usage**

```r
dataUI(id, tab)
```

**Arguments**

- `id` Character: identifier
- `tab` Function to create tab

**Value**

HTML elements

---

**diffAnalyses**

*Perform selected statistical analyses on multiple splicing events*

**Description**

Perform selected statistical analyses on multiple splicing events

**Usage**

```r
diffAnalyses(psi, groups = NULL, analyses = c("wilcoxRankSum", "ttest", "kruskal", "levene", "fligner"), pvalueAdjust = "BH", progress = echoProgress)
```

**Arguments**

- `psi` Data frame or matrix: alternative splicing event quantification
- `groups` Character: group of each sample from the alternative splicing event quantification (if NULL, sample types are used instead, e.g. normal, tumour and metastasis)
- `analyses` Character: analyses to perform (see Details)
- `pvalueAdjust` Character: method used to adjust p-values (see Details)
- `progress` Function to track the progress
Details

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

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

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

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

Value

Table of statistical analyses

Examples

```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"))

# diffSplicingEventServer

diffSplicingEventServer

Description

Server logic for the analyses of a single alternative splicing event

Usage

diffSplicingEventServer(input, output, session)
**diffSplicingEventUI**

**Arguments**

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

**Value**

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

---

**diffSplicingEventUI**  Interface for the analysis of an alternative splicing event

**Description**

Interface for the analysis of an alternative splicing event

**Usage**

diffSplicingEventUI(id)

**Arguments**

- **id**: Character: identifier

**Value**

Character with the HTML interface

---

**diffSplicingServer**  Server logic for the differential splicing analyses

**Description**

Server logic for the differential splicing analyses

**Usage**

diffSplicingServer(input, output, session)

**Arguments**

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

**Value**

NULL (this function is used to modify the Shiny session’s state)
**diffSplicingTableServer**

*Server logic of the exploratory differential analyses*

**Description**

Server logic of the exploratory differential analyses

**Usage**

`diffSplicingTableServer(input, output, session)`

**Arguments**

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

**Value**

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

---

**diffSplicingTableUI**

*Interface for differential analyses on all splicing events*

**Description**

Interface for differential analyses on all splicing events

**Usage**

`diffSplicingTableUI(id)`

**Arguments**

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

**Value**

HTML elements
**diffSplicingUI**  
*User interface for the differential splicing analyses*

**Description**  
User interface for the differential splicing analyses

**Usage**  
diffSplicingUI(id, tab)

**Arguments**

- **id**  
  Character: identifier
- **tab**  
  Function to process HTML elements

**Value**  
HTML element as character

---

**disableTab**  
*Disable a tab from the navbar*

**Description**  
Disable a tab from the navbar

**Usage**  
disableTab(tab)

**Arguments**

- **tab**  
  Character: tab to disable

**Value**  
NULL (this function is used to modify the Shiny session’s state)
downloadFiles | Download files to a given directory

Description
Download files to a given directory

Usage
downloadFiles(url, folder, progress = echoProgress, download = download.file, ...)

Arguments
url | Character: download links
folder | Character: directory to store the downloaded archives
progress | Function to show the progress (default is to print progress to console)
download | Function to use to download files
... | Extra parameters passed to the download function

Value
Invisible TRUE if every file was successfully downloaded

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

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

## End(Not run)

echoProgress | Echo progress to console using cat

Description
Echo progress to console using cat

Usage
echoProgress(..., console = TRUE)

Arguments
... | Strings to print to console
console | Boolean: print to console? TRUE by default
enableTab

**Value**

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

```
enableTab
```

Enable a tab from the navbar

**Description**

Enable a tab from the navbar

**Usage**

```r
enableTab(tab)
```

**Arguments**

- `tab` Character: tab to enable

**Value**

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

```
endProcess
```

Signal the program that a process has ended

**Description**

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

**Usage**

```r
dropSignal(id, time = NULL, closeProgressBar = TRUE)
```

**Arguments**

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

**Value**

NULL (this function is used to modify the Shiny session’s state)
ensemblToUniprot  
*Convert a protein’s Ensembl identifier to UniProt identifier*

**Description**

Convert a protein’s Ensembl identifier to UniProt identifier

**Usage**

`ensemblToUniprot(protein)`

**Arguments**

- **protein**  
  Character: Ensembl protein identifier

**Value**

UniProt protein identifier

**Examples**

`ensemblToUniprot("ENSP00000445929")`

---

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

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

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

**Arguments**
- **hc**: A highcharts object
- **fill**: Character: colour fill
- **text**: Character: button text

**Value**
A highcharts object with an export button

---

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

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

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

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

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

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

**User interface of the TCGA/Firebrowse loader**

**Description**

User interface of the TCGA/Firebrowse loader

**Usage**

```r
definefirebrowseUI(id, panel)
```

**Arguments**

- `id` Character: identifier
- `panel` Function to enclose interface

**Value**

HTML of the interface

### fisher

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

**Description**

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

**Usage**

```r
definefisher(psi, groups)
```

**Arguments**

- `psi` Numeric: quantification of one alternative splicing event
- `groups` Character: group of each PSI index

**Value**

HTML elements
**fligner**

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

---

**Description**

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

**Usage**

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

**Arguments**

- `psi` Numeric: quantification of one alternative splicing event
- `groups` Character: group of each PSI index
- `stat` Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)

**Value**

HTML elements

---

**getActiveDataset**

*Get selected dataset*

---

**Description**

Get selected dataset

**Usage**

```
getActiveDataset()
```

**Value**

List of data frames
getAssemblyVersion  
*Get the assembly version of a data category*

**Description**

Get the assembly version of a data category

**Usage**

```r
getAssemblyVersion(category = getCategory())
```

**Arguments**

- **category**  
  Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**

Character value with the assembly version

**Note**

Needs to be called inside a reactive function

---

getAutoNavigation  
*Get if history browsing is automatic*

**Description**

Get if history browsing is automatic

**Usage**

```r
getAutoNavigation()
```

**Value**

Boolean: is navigation of browser history automatic?
**getCategories**

*Get available data categories*

**Description**

Get available data categories

**Usage**

getCategories()

**Value**

Name of all data categories

---

**getCategory**

*Get selected data category*

**Description**

Get selected data category

**Usage**

getCategory()

**Value**

Name of selected data category

---

**getCategoryData**

*Get data of selected data category*

**Description**

Get data of selected data category

**Usage**

getCategoryData()

**Value**

If category is selected, returns the respective data as a data frame; otherwise, returns NULL
getClinicalData  
*Get clinical data of the data category*

**Description**
Get clinical data of the data category

**Usage**
```
getclinicalData()
```

**Value**
Data frame with clinical data

getClinicalMatchFrom  
*Get clinical matches from a given data type*

**Description**
Get clinical matches from a given data type

**Usage**
```
getclinicalMatchFrom(dataset, category = getCategory())
```

**Arguments**
- **dataset**: Character: data set (e.g. "Junction quantification")
- **category**: Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**
Integer with clinical matches to a given dataset

**Note**
Needs to be called inside a reactive function
**getColumnsTime**  
*Retrieve the time for given columns in a clinical dataset*

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

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

**Arguments**
- **clinical**  
  Data frame: clinical data
- **event**  
  Character: name of column containing time of the event of interest
- **timeStart**  
  Character: name of column containing starting time of the interval or follow up time
- **timeStop**  
  Character: name of column containing ending time of the interval
- **followup**  
  Character: name of column containing follow up time

**Value**
Data frame containing the time for the given columns

---

**getCores**  
*Get number of cores to use*

**Description**
Get number of cores to use

**Usage**
```
getCores()
```

**Value**
Numeric value with the number of cores to use
**getData**

*Get global data*

**Description**

Get global data

**Usage**

```r
dataGet()
```

**Value**

Variable containing all data of interest

---

**getDataRows**

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

**Description**

Get rows of a data frame between two row indexes

**Usage**

```r
dataGetRows(i, data, firstRow, lastRow)
```

**Arguments**

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

**Details**

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

**Value**

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

Get the table of differential analyses of a data category

Description
Get the table of differential analyses of a data category

Usage
getDifferentialAnalyses(category = getCategory())

Arguments
category Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

Value
Data frame of differential analyses

Note
Needs to be called inside a reactive function

g får DifferentialAnalysesSurvival

Get the table of differential analyses’ survival data of a data category

Description
Get the table of differential analyses’ survival data of a data category

Usage
g får DifferentialAnalysesSurvival(category = getCategory())

Arguments
category Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

Value
Data frame of differential analyses’ survival data

Note
Needs to be called inside a reactive function
getDownloadsFolder  
*Get the Downloads folder of the user*

**Description**
Get the Downloads folder of the user

**Usage**

```r
getDownloadsFolder()
```

**Value**
Path to Downloads folder

**Examples**

```r
getDownloadsFolder()
```

getEvent  
*Get selected alternative splicing event’s identifier*

**Description**
Get selected alternative splicing event’s identifier

**Usage**

```r
getEvent()
```

**Value**
Alternative splicing event’s identifier as a string

getFirehoseCohorts  
*Query the Firehose API for the cohorts available*

**Description**
Query the Firehose API for the cohorts available

**Usage**

```r
getFirehoseCohorts(cohort = NULL)
```

**Arguments**

- `cohort`  Character: filter by given cohorts (optional)
**getFirehoseDataTypes**

**Value**

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

**Examples**

```r
if (isFirehoseUp()) getFirehoseCohorts()
```

---

**getFirehoseDataTypes**  
*Get data types available from Firehose*

**Description**

Get data types available from Firehose

**Usage**

```r
getFirehoseDataTypes()
```

**Value**

Named character vector

**Examples**

```r
getFirehoseDataTypes()
```

---

**getFirehoseDateFormat**  
*Returns the date format used by the Firehose API*

**Description**

Returns the date format used by the Firehose API

**Usage**

```r
getFirehoseDateFormat()
```

**Value**

Named list with Firehose API’s date formats

**Examples**

```r
format <- psichomics:::getFirehoseDateFormat()

# date format to use in a query to Firehose API
format$query

# date format to parse a date in a response from Firehose API
format$response
```
getFirehoseDates

Query the Firehose API for the datestamps of the data available and parse the response

Usage

getFirehoseDates()

Value

Date with datestamps of the data available

Examples

if (isFirehoseUp()) getFirehoseDates()

getGlobal

Get data from global data

Description

Get data from global data

Usage

global(..., sep = ":")

Arguments

... Arguments to identify a variable
sep Character to separate identifiers

Value

Data from global data
getGroupsFrom  

*Get groups from a given data type*

**Description**

Get groups from a given data type

**Usage**

```r
getGroupsFrom(dataset, category = getCategory(), complete = FALSE, samples = FALSE)
```

**Arguments**

- `dataset` Character: data set (e.g. "Clinical data")
- `category` Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category
- `complete` Boolean: return all the information on groups (TRUE) or just the group names and respective indexes (FALSE)? FALSE by default
- `samples` Boolean: show groups by samples (TRUE) or patients (FALSE)? FALSE by default

**Value**

Matrix with groups of a given dataset

**Note**

Needs to be called inside a reactive function

---

getInclusionLevels  

*Get alternative splicing quantification of the selected data category*

**Description**

Get alternative splicing quantification of the selected data category

**Usage**

```r
getInclusionLevels()
```

**Value**

Data frame with the alternative splicing quantification
### getInclusionLevelsPCA

*Get principal component analysis based on inclusion levels*

**Description**

Get principal component analysis based on inclusion levels

**Usage**

```r
getInclusionLevelsPCA(category = getCategory())
```

**Arguments**

- `category`
  - Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**

`prcomp` object (PCA) of inclusion levels

**Note**

Needs to be called inside a reactive function

### getJunctionQuantification

*Get junction quantification data*

**Description**

Get junction quantification data

**Usage**

```r
getJunctionQuantification(category = getCategory())
```

**Arguments**

- `category`
  - Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**

List of data frames of junction quantification

**Note**

Needs to be called inside a reactive function
getMatchingSamples

Search samples in the clinical dataset and return the ones matching the given index

Usage

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

Arguments

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

Value

Names of the matching samples (if showMatch is TRUE, a integer vector with the patient index and the matching samples as names is shown)

Examples

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

getNumerics

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

Description

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

Usage

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

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Data matrix: table</td>
</tr>
<tr>
<td>by</td>
<td>Character: column names of interest</td>
</tr>
<tr>
<td>toNumeric</td>
<td>Boolean: which columns to convert to numeric (FALSE by default)</td>
</tr>
</tbody>
</table>

**Value**

Processed data matrix

**Examples**

```r
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"])

# Let's check if the same column is now integer
is.numeric(event[, "C1.end"])
```

getPatientFromSample

**Match given sample identifiers and return the respective row in clinical data**

**Description**

Match given sample identifiers and return the respective row in clinical data

**Usage**

```r
getPatientFromSample(sampleId, patientId)
```

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampleId</td>
<td>Character: sample identifiers</td>
</tr>
<tr>
<td>patientId</td>
<td>Character: clinical patient identifiers (if a matrix or data frame is given, its row-names will be retrieved as patient identifiers)</td>
</tr>
</tbody>
</table>

**Value**

Integer vector of the row number in clinical data corresponding to the given IDs (named with the ID)

**Examples**

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

*Get the identifier of patients for a given category*

**Description**

Get the identifier of patients for a given category

**Usage**

```r
getPatientId(category = getCategory())
```

**Arguments**

- `category` Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**

Character vector with identifier of patients

**Note**

Needs to be called inside a reactive function

getPrecision

*Get number of decimal places*

**Description**

Get number of decimal places

**Usage**

```r
getPrecision()
```

**Value**

Numeric value regarding the number of decimal places
**getPSIperPatient**  
Assign alternative splicing quantification to patients based on their samples

**Description**
Match filtered samples with clinical patients to retrieve alternative splicing quantification per clinical patient. Only one sample can be matched with one patient. Normal and control samples are filtered out by default.

**Usage**
```r
getPSIperPatient(psi, match, clinical, pattern = c("Normal", "Control"), filterOut = TRUE)
```

**Arguments**
- `psi`  
  Data frame or matrix: alternative splicing quantification per samples
- `match`  
  Matrix: match between samples and clinical patients
- `clinical`  
  Data frame or matrix: clinical dataset
- `pattern`  
  Character: pattern to use when filtering sample types (normal and control samples are filtered by default)
- `filterOut`  
  Boolean: filter out (TRUE) or filter in (FALSE) samples with the given pattern; by default, filter out

**Value**  
Alternative splicing quantification per clinical patients

---

**getSampleId**  
Get the identifier of samples for a given category

**Description**
Get the identifier of samples for a given category

**Usage**
```r
getSampleId(category = getCategory())
```

**Arguments**
- `category`  
  Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**  
Character vector with identifier of samples
**getSampleInfo**

*Get sample information of the selected data category*

**Description**

Get sample information of the selected data category

**Usage**

getSampleInfo()

**Value**

Data frame with sample information

**getSelectedGroups**

*Get selected groups for a given group selection element*

**Description**

Get selected groups for a given group selection element

**Usage**

getSelectedGroups(input, id, samples = FALSE, dataset = "Clinical data", filter = NULL)

**Arguments**

- **input**: Shiny input
- **id**: Character: identifier of the group selection element
- **samples**: Boolean: show groups by samples (TRUE) or patients (FALSE)? FALSE by default
- **dataset**: Character: data set (e.g. "Clinical data")
- **filter**: Character: only get groups passed

**Value**

List with selected groups (or NULL if no groups were selected)
**getServerFunctions**  
*Matches server functions from a given loader*

**Description**

Matches server functions from a given loader

**Usage**

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

**Arguments**

- `loader`  
  Character: loader to run the functions

- `...`  
  Extra arguments to pass to server functions

- `priority`  
  Character: name of functions to prioritise by the given order; for instance, `c("data", "analyses")` would load "data", then "analyses" then remaining functions

**Value**

Invisible TRUE

---

**getSignificant**  
*Get number of significant digits*

**Description**

Get number of significant digits

**Usage**

```r
getSignificant()
```

**Value**

Numeric value regarding the number of significant digits
**getSpecies**

*Get the species of a data category*

**Description**

Get the species of a data category

**Usage**

```r
getSpecies(category = getCategory())
```

**Arguments**

- `category`
  - Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**

Character value with the species

**Note**

Needs to be called inside a reactive function

---

**getSplicingEventCoordinates**

*Returns the coordinates of interest for a given event type*

**Description**

Returns the coordinates of interest for a given event type

**Usage**

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

**Arguments**

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

**Value**

Coordinates of interest according to the alternative splicing event type
getSplicingEventTypes  *Splicing event types available*

**Description**
Splicing event types available

**Usage**

```r
getSplicingEventTypes()
```

**Value**
Named character vector with splicing event types

**Examples**

```r
getSplicingEventTypes()
```

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

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

**Usage**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ns</code></td>
<td>Shiny function to create namespaced IDs</td>
</tr>
<tr>
<td><code>loader</code></td>
<td>Character: loader to run the functions</td>
</tr>
<tr>
<td><code>...</code></td>
<td>Extra arguments to pass to the user interface (UI) functions</td>
</tr>
<tr>
<td><code>priority</code></td>
<td>Character: name of functions to prioritise by the given order; for instance, <code>c(&quot;data&quot;, &quot;analyses&quot;)</code> would load &quot;data&quot;, then &quot;analyses&quot; then remaining functions</td>
</tr>
</tbody>
</table>

**Value**
List of functions related to the given loader
getURLtoDownload  
*Get the URL links to download*

**Description**

Get the URL links to download

**Usage**

getURLtoDownload()

**Value**

Character vector with URLs to download

**Note**

Needs to be called inside a reactive function

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

**Description**

Filters the events with valid elements according to the given validator

**Usage**

getValidEvents(event, validator, areMultipleExonsValid = FALSE)

**Arguments**

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

**Details**

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

**Value**

Data.frame with valid events
Examples

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

---

globalSelectize  
Create a selectize input available from any page

Description

Create a selectize input available from any page

Usage

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

Arguments

```r
id  
Character: input identifier

placeholder  
Character: input placeholder
```

Value

HTML element for a global selectize input
**groupByAttribute**  
*User interface to group by attribute*

**Description**  
User interface to group by attribute

**Usage**  
`groupByAttribute(ns, dataset, id, example)`

**Arguments**
- **ns**: Namespace function
- **dataset**: Data frame: dataset of interest
- **id**: Character: identifier
- **example**: Character: text to show as an example

**Value**
- HTML elements

**groupByExpression**  
*User interface to group by subset expression*

**Description**  
User interface to group by subset expression

**Usage**  
`groupByExpression(ns, id)`

**Arguments**
- **ns**: Namespace function
- **id**: Character: identifier

**Value**
- HTML elements
**groupById**

User interface to group by row

### Description

User interface to group by row

### Usage

```r
groupById(ns, id, choices)
```

### Arguments

- **ns**: Namespace function
- **id**: Character: identifier
- **choices**: Character: identifier suggestions

### Value

HTML elements

---

**groupByGrep**

User interface to group by grep expression

### Description

User interface to group by grep expression

### Usage

```r
groupByGrep(ns, dataset, id)
```

### Arguments

- **ns**: Namespace function
- **dataset**: Data frame: dataset of interest
- **id**: Character: identifier

### Value

HTML elements
groupPerPatient  

Assign one group to each patient

**Description**

Assign one group to each patient

**Usage**

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

**Arguments**

- `groups` List of integers: clinical groups
- `patients` Integer: total number of clinical patients (remaining patients will be filled with missing values)
- `includeOuterGroup` Boolean: join the patients that have no groups?
- `outerGroupName` Character: name to give to outer group

**Value**

Character vector where each element corresponds to the group of a clinical patient

**Examples**

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

---

groupPerSample  

Assign one group to each sample

**Description**

Assign one group to each sample

**Usage**

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

**Arguments**

- `groups` List of characters: list of samples
- `samples` Character: all available samples
- `includeOuterGroup` Boolean: join the patients that have no groups?
- `outerGroupName` Character: name to give to outer group
Value

Character vector where each element corresponds to the group of a sample

Examples

groups <- list(letters[1:3], letters[10:12], letters[5:8])
names(groups) <- paste("Stage", 1:3)
samples <- letters
groupPerSample(groups, samples)

Description

Server function for data grouping

Usage

groupsServer(input, output, session, datasetName)

Arguments

- input: Shiny input
- output: Shiny output
- session: Shiny session
- datasetName: Character: name of dataset

Value

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

Description

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

Usage

groupsServerOnce(input, output, session)

Arguments

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

Value

NULL (this function is used to modify the Shiny session’s state)
**groupsUI**

*Creates UI elements for the grouping feature*

**Description**

Creates UI elements for the grouping feature

**Usage**

`groupsUI(id)`

**Arguments**

- `id` Character: identifier

**Value**

HTML elements

---

**gtexDataServer**

*Server logic to load GTEx data*

**Description**

Server logic to load GTEx data

**Usage**

`gtexDataServer(input, output, session)`

**Arguments**

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

**Value**

NULL (this function is used to modify the Shiny session’s state)
hchart survfit

---

**gtxDataUI**

*Interface to load GTEx data*

**Description**

Interface to load GTEx data

**Usage**

```r
gtxDataUI(id, panel)
```

**Arguments**

- `id` Character: namespace identifier
- `panel` Function to deal with the interface

**Value**

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

---

**hchart survfit**

*Plot survival curves using Highcharts*

**Description**

Plot survival curves using Highcharts

**Usage**

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

**Arguments**

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

Arguments

hc Highchart object
x Numeric: X axis
y Numeric: Y axis
z Numeric: Z axis to set the bubble size (optional)
label Character: data label for each point (optional)
showInLegend Boolean: show the data in the legend box? FALSE by default
... Extra attributes of the data series to plot

Value

Highchart object containing information for a scatter plot
inclusionLevelsInterface

Interface to quantify alternative splicing

Description

Interface to quantify alternative splicing

Usage

inclusionLevelsInterface(ns)

Arguments

ns Namespace function

Value

HTML elements

inclusionLevelsServer Server logic of the alternative splicing event quantification module

Description

Server logic of the alternative splicing event quantification module

Usage

inclusionLevelsServer(input, output, session)

Arguments

input Shiny input
output Shiny output
session Shiny session

Value

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

**Description**

Interface of the alternative splicing event quantification module

**Usage**

inclusionLevelsUI(id, panel)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Character: identifier</td>
</tr>
<tr>
<td>panel</td>
<td>Function to process HTML elements</td>
</tr>
</tbody>
</table>

**Value**

HTML elements

---

**infoServer**

**Server logic**

**Description**

Server logic

**Usage**

infoServer(input, output, session)

**Arguments**

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

**Value**

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

Information’s user interface

Description

Information’s user interface

Usage

infoUI(id)

Arguments

id  Character: identifier

Value

HTML elements

insideFile  

Get psichomics file inside a given directory

Description

Get psichomics file inside a given directory

Usage

insideFile(...)  

Arguments

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

Value

Loaded file
is.whole

Check if a number is whole

Description

Check if a number is whole

Usage

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

Arguments

x
  Object to be tested

tol
  Numeric: tolerance used for comparison

Value

TRUE if number is whole; otherwise, FALSE

isFirehoseUp

Check if the Firehose API is running

Description

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

Usage

isFirehoseUp()

Value

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

Examples

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

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

**Usage**  
`joinEventsPerType(events, types)`

**Arguments**
- `events`: Data frame or matrix: alternative splicing events
- `types`: Character: alternative splicing types

**Value**
List of events joined by alternative splicing event type

---

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

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

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

**Arguments**
- `chr`: Character: chromosome
- `strand`: Character: strand
- `junc5`: Integer: 5’ end junction
- `junc3`: Integer: 3’ end junction
- `showStrand`: Boolean: include strand?

**Value**
Formatted character string
kruskal

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

**Description**

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

**Usage**

```r
kruskal(psi, groups, stat = NULL)
```

**Arguments**

- `psi`: Numeric: quantification of one alternative splicing event
- `groups`: Character: group of each PSI index
- `stat`: Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)

**Value**

HTML elements

---

labelBasedOnCutoff

Label groups based on a given cut-off

**Description**

Label groups based on a given cut-off

**Usage**

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

**Arguments**

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

**Value**

Labeled groups
Examples

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

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

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

---

**Description**

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

**Usage**

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

**Arguments**

- `psi`: Numeric: quantification of one alternative splicing event
- `groups`: Character: group of each PSI index
- `stat`: Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)

**Value**

HTML elements

---

**Description**

Levene's test

**Usage**

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

**Arguments**

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

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

```r
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)
```

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 the alternative splicing annotation files available*

**Description**

List the alternative splicing annotation files available

**Usage**

listSplicingAnnotations()

**Value**

Named character vector with splicing annotation files available

**Examples**

listSplicingAnnotations()

loadAnnotation

*Load alternative splicing annotation from AnnotationHub*

**Description**

Load alternative splicing annotation from AnnotationHub

**Usage**

loadAnnotation(annotation)

**Arguments**

- **annotation** Character: annotation to load

**Value**

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

**Examples**

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

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

Usage
loadBy(loader, FUN)

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

Value
Boolean vector

loadedDataModal

Description
Create a modal warning the user of already loaded data

Usage
loadedDataModal(session, modalId, replaceButtonId, keepButtonId)

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

Value
HTML elements for a warning modal reminding data is loaded
**loadFile**

*Loads a file according to its format*

---

**Description**

Loads a file according to its format

**Usage**

`loadFile(format, file)`

**Arguments**

- `format`  
  Environment: format of the file
- `file`  
  Character: file to load

**Details**

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

**Value**

Data frame with the loaded file

---

**loadFileFormats**

*Loads file formats*

---

**Description**

Loads file formats

**Usage**

`loadFileFormats()`

**Value**

Loaded file formats available
loadFirehoseData

Downloads and processes data from the Firehose API and loads it into R

Description

Downloads and processes data from the Firehose API and loads it into R

Usage

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

Arguments

folder Character: directory to store the downloaded archives (by default, it saves in the user’s "Downloads" folder)
data Character: data to load
exclude Character: files and folders to exclude from downloading and from loading into R (by default, it excludes ".aux.", ".mage-tab." and "MANIFEST.TXT" files)
... Extra parameters to be passed to queryFirehoseData
progress Function to show the progress (default is to print progress to console)
download Boolean: download missing files through the function download.file (TRUE by default)

Value

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

Examples

## Not run:
loadFirehoseData(cohort = "ACC", data_type = "Clinical")
## End(Not run)

loadFirehoseFolders

Load Firehose folders

Description

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

Usage

loadFirehoseFolders(folder, exclude = "", progress = echoProgress)
### loadLocalFiles

**Description**

Load local files

**Usage**

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

**Arguments**

- **folder**: Character: folder(s) in which to look for Firehose files
- **exclude**: Character: files to exclude from the loading
- **progress**: Function to show the progress (default is to print progress to console)

**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).

### loadGtexData

**Description**

Load GTEx data given input

**Usage**

```r
loadGtexData(input, replace = TRUE)
```

**Arguments**

- **input**: Shiny input
- **replace**: Boolean: replace loaded data? TRUE by default

**Value**

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

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

Value

List of data frames from valid files

Examples

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

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

## End(Not run)
localDataUI  
*Interface to load local data*

**Description**

Interface to load local data

**Usage**

`localDataUI(id, panel)`

**Arguments**

- **id**
  - Character: namespace identifier
- **panel**
  - Function to deal with the interface

**Value**

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

missingDataModal  
*Missing information modal template*

**Description**

Missing information modal template

**Usage**

`missingDataModal(session, dataType, buttonId)`

`loadRequiredData(modal = NULL)`

`missingDataGuide(dataType)`

**Arguments**

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

**Value**

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

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

**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 pass to tab
- `icon` Character: name of the icon
- `menu` Boolean: create a dropdown menu-like tab? FALSE by default

**Value**

HTML interface for a tab panel

**Note**

Icon is hidden at small viewports

---

**navSelectize**

*Create a special selectize input in the navigation bar*

**Description**

Create a special selectize input in the navigation bar

**Usage**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Character: input identifier</td>
</tr>
<tr>
<td>label</td>
<td>Character: input label</td>
</tr>
<tr>
<td>placeholder</td>
<td>Character: input placeholder</td>
</tr>
</tbody>
</table>

**Value**

HTML element to be included in a navigation bar

---

**noinfo**

*Interface when no information could be retrieved*

**Description**

Interface when no information could be retrieved

**Usage**

```r
noinfo(output, title = paste("No information available for the gene", 
"associated with this event."),
description = "Select another alternative splicing event.")
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>Shiny output</td>
</tr>
<tr>
<td>title</td>
<td>Character: title of the message to show to the user</td>
</tr>
<tr>
<td>description</td>
<td>Character: description of the message to show to the user</td>
</tr>
</tbody>
</table>

**Value**

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

---

**operateOnGroups**

*Set operations on groups*

**Description**

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

**Usage**

```r
operateOnGroups(input, session, FUN, buttonId, symbol = " ", datasetName, 
sharedData = sharedData)
```
optimalPSIcutoff

Arguments

input  Shiny input
session Shiny session
FUN  Function: operation to set
buttonId Character: ID of the button to trigger operation
symbol Character: operation symbol
datasetName Character: name of dataset
sharedData Shiny app’s global variable

Value

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

optimalPSIcutoff Calculate optimal alternative splicing quantification cut-off to separate survival curves

Description

Calculate optimal alternative splicing quantification cut-off to separate survival curves

Usage

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

Arguments

clinical  Data frame: clinical data
psi  Numeric: PSI values to test against the cut-off
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
followup  Character: name of column containing follow up time
session  Shiny session (only used for the visual interface)
filter  Boolean or numeric: elements to use (all by default)
survTime  survTime object: times to follow up, time start, time stop and event (optional)

Details

timeStop is only considered if censoring is either interval or interval2
Value

Optimal alternative splicing quantification cut-off

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 <- optimalPSIcutoff(clinical, psi, "right", event, timeStart)
```

---

### optimSurvDiff

*Optimal survival difference given an inclusion level cut-off for a specific alternative splicing event*

Description

Optimal survival difference given an inclusion level cut-off for a specific alternative splicing event

Usage

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

Arguments

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

Value

NULL (this function is used to modify the Shiny session’s state) Calculate optimal survival cut-off for the inclusion levels of a given alternative splicing event
optimSurvDiffOptions

**Interface for calculating optimal cut-off and p-value for survival curves differences**

**Description**

Interface for calculating optimal cut-off and p-value for survival curves differences

**Usage**

```r
optimSurvDiffOptions(ns)
```

**Arguments**

- `ns` Namespace function

**Value**

HTML elements to calculate optimal survival difference

---

parseDateResponse

**Parse the date from a response**

**Description**

Parse the date from a response

**Usage**

```r
parseDateResponse(string)
```

**Arguments**

- `string` Character: dates

**Value**

Parsed date
parseFirehoseMetadata  
*Query the Firehose API for metadata and parse the response*

**Description**
Query the Firehose API for metadata and parse the response

**Usage**
```r
parseFirehoseMetadata(type, ...)  
```

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

**Value**
List with parsed JSON response

**Examples**
```r
psichomics:::parseFirehoseMetadata("Dates")
psichomics:::parseFirehoseMetadata("Centers")
psichomics:::parseFirehoseMetadata("HeartBeat")

# Get the abbreviation and description of all cohorts available
psichomics:::parseFirehoseMetadata("Cohorts")
# Get the abbreviation and description of the selected cohorts
psichomics:::parseFirehoseMetadata("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)
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

List containing the event attributes and junctions

Examples

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

Description

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

Usage

```r
parseMatsGeneric(junctions, strand, coords, plus_pos, minus_pos)
parseMatsSE(junctions, strand)
parseMatsMXE(junctions, strand)
parseMatsRI(junctions, strand)
parseMatsA3SS(junctions, strand)
parseMatsA5SS(junctions, strand)
parseMatsAFE(junctions, strand)
parseMatsALE(junctions, strand)
```
Arguments

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

Details

The following event types are ready to be parsed:

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

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

Value

Data frame with parsed junctions

See Also

`parseMatsEvent`

Examples

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

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

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

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

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

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

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

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

---

**parseMisoEvent**

*Parse an alternative splicing event from MISO*

**Description**

Parse an alternative splicing event from MISO

**Usage**

`parseMisoEvent(event)`

**Arguments**

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

**Details**

More information about MISO available at [http://miso.readthedocs.org](http://miso.readthedocs.org)

**Value**

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

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

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

Description

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

Usage

parseMisoEventID(eventID, annotation, IDcolumn)

Arguments

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

Details

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

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

Value

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

Note

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

```r
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)
```

Description

Parse junctions of an event from MISO according to event type

Usage

```r
parseMisoGeneric(event, validator, eventType, coord, plusIndex, minusIndex)
parseMisoSE(event)
parseMisoMXE(event)
parseMisoRI(event, strand)
parseMisoA5SS(event)
parseMisoA3SS(event, plusIndex, minusIndex)
parseMisoTandemUTR(event, minusIndex)
parseMisoAFE(event)
parseMisoALE(event)
```

Arguments

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

The following event types are available to be parsed:

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

Value

List of parsed junctions

See Also

`parseMisoEvent`

Examples

```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 mRNA 16854 18061 . - .
chr1 SE exon 16854 17955 . - .
chr1 SE exon 17915 18061 . - .")
psichomics:::parseMisoSE(event)

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

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

chr1 RI exon 17233 17364 . - .
chr1 RI exon 17601 17742 . - .

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

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

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

# alternative first exon (AFE) event
event <- read.table(text = "
  chr12 AFE gene 57916659 57920171 . + .
  chr12 AFE mRNA 57919131 57928171 . + .
  chr12 AFE exon 57919131 57928171 . + .
  chr12 AFE mRNA 57916659 57918199 . + .
  chr12 AFE exon 57916659 57917875 . + .
  chr12 AFE exon 57918863 57918199 . + .")

# alternative last exon (ALE) event
event <- read.table(text = "
  chr6 ALE gene 30620579 30822593 . + .
  chr6 ALE mRNA 30822190 30822593 . + .
  chr6 ALE exon 30822190 30822593 . + .
  chr6 ALE mRNA 30620579 30620982 . + .
  chr6 ALE exon 30620579 30620982 . + .")
parseMisoId  
*Parse MISO’s alternative splicing event identifier*

**Description**

Parse MISO’s alternative splicing event identifier

**Usage**

`parseMisoId(id)`

**Arguments**

- `id`  Character: MISO alternative splicing event identifier

**Value**

Character with the parsed ID

**Examples**

```r
id <- paste0("ID=ENSMUSG00000026150.chr1:82723803:82723911:+@chr1:82724642:82724813:;", "+@chr1:82725791:82725911:+.B;Parent=ENSMUSG00000026150.chr1:82723803:;", "+82723911:+@chr1:82724642:82724813:+@chr1:82725791:82725911:+")

psichomics:::parseMisoId(id)
```

---

parseSampleGroups  
*Return the type of a given sample*

**Description**

Return the type of a given sample

**Usage**

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

**Arguments**

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

**Value**

Types of the TCGA samples

**Examples**

```r
parseSampleGroups(c("TCGA-01A-Tumour", "TCGA-10B-Normal"))
```
parseSplicingEvent  Parse an alternative splicing event based on a given identifier

Description

Parse an alternative splicing event based on a given identifier

Usage

parseSplicingEvent(event)

Arguments

event  Character: event identifier

Value

Parsed event

Examples

events <- c("SE_1--123_456_789_1024_TST",
"MX_3+t_473_578_686_736_834_937_HEY/YOU")
parseSplicingEvent(events)

parseSuppaAnnotation  Get events from alternative splicing annotation

Description

Get events from alternative splicing annotation

Usage

parseSuppaAnnotation(folder, types = c("SE", "AF", "AL", "MX", "A5", "A3", "RI"), genome = "hg19")
parseVastToolsAnnotation(folder, types = c("ALT3", "ALT5", "COMBI", "IR", "MERGE3m", "MIC", "EXSK", "MULTI"), genome = "Hsa", complexEvents = FALSE)
parseMisoAnnotation(folder, types = c("SE", "AFE", "ALE", "MXE", "A5SS", "A3SS", "RI", "TandemUTR"), genome = "hg19")
parseMatsAnnotation(folder, types = c("SE", "AFE", "ALE", "MXE", "A5SS", "A3SS", "RI"), genome = "fromGTF", novelEvents = TRUE)
Arguments

folder  Character: path to folder

types  Character: type of events to retrieve (depends on the program of origin; see details)

genome  Character: genome of interest (for instance, "hg19"; depends on the program of origin)

complexEvents  Boolean: should complex events in A3SS and A5SS be parsed? FALSE by default

novelEvents  Boolean: parse events detected due to novel splice sites (TRUE by default)

Details

Type of parseable events:

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

Value

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

Examples

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

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

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

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

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

Parses splicing events of a specific event type from SUPPA

Description

Parses splicing events of a specific event type from SUPPA

Usage

parseSuppaEvent(event)

Arguments

event Character vector: Splicing event attributes and junction positions

Details

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

The following event types are available to be parsed:

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

Value

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

Note

It only allows to parse one event type at once.

Examples

event <- "ENSG0000000419;A3:20:49557492-49557642:49557470-49557642:-"
psiChomics::parseSuppaEvent(event)
parseSuppaGeneric  Parse junctions of an event from SUPPA

**Description**

Parse junctions of an event from SUPPA

**Usage**

parseSuppaGeneric(junctions, strand, coords, plus_pos, minus_pos)
parseSuppaSE(junctions, strand)
parseSuppaRI(junctions, strand)
parseSuppaALE(junctions, strand)
parseSuppaAFE(junctions, strand)
parseSuppaMXE(junctions, strand)
parseSuppaA3SS(junctions, strand)
parseSuppaA5SS(junctions, strand)

**Arguments**

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

**Details**

The following event types are available to be parsed:

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

**Value**

Data frame of parsed junctions
See Also

parseSuppaEvent

Examples

# Parse generic event (in this case, an exon skipping event)
junctions <- read.table(text = "169768099 169770024 169770112 169771762")
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 24795476 24795797")
psichomics:::parseSuppaALE(junctions, "+")
junctions <- read.table(text = "169763871 169764046 169767998 169765124 169767998")
psichomics:::parseSuppaAFE(junctions, "+")
junctions <- read.table(text = "202060671 202068453 202068489 202073793 202060671 202072798 202072906 202073793")
psichomics:::parseSuppaMXE(junctions, "+")
junctions <- read.table(text = "169772450 169773216 169772450 169773253")
psichomics:::parseSuppaA3SS(junctions, "+")
junctions <- read.table(text = "50193276 50197008 50192997 50197008")
psichomics:::parseSuppaA5SS(junctions, "+")

parseTcgaSampleInfo  Parse and prepare sample information from TCGA samples

Description

Parse and prepare sample information from TCGA samples

Usage

parseTcgaSampleInfo(samples, category = getCategory())

Arguments

samples Character: sample identifiers
category Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category
parseUrlsFromFirehoseResponse

Description
Retrieve URLs from a response to a Firehose data query

Usage
parseUrlsFromFirehoseResponse(res)

Arguments
res Response from http::GET to a Firehose data query

Value
Named character with URLs

Examples
res <- psychomics::queryFirehoseData(cohort = "ACC")
url <- psychomics::parseUrlsFromFirehoseResponse(res)

Value
Data frame containing metadata associated with each TCGA sample

parseUniprotXML  Parse XML from Uniprot's RESTful service

Description
Parse XML from Uniprot’s RESTful service

Usage
parseUniprotXML(xml)

Arguments
xml response from Uniprot

Value
List containing protein length and data frame of protein features
**parseValidFile**

*Parse file given a list of file formats*

**Description**

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

**Usage**

```r
describeValidFile(file, formats)
```

**Arguments**

- `file` Character: file to parse
- `formats` List of file formats to check

**Details**

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

**Value**

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

---

**parseVastToolsEvent**

*Parses an alternative splicing event from VAST-TOOLS*

**Description**

Parses an alternative splicing event from VAST-TOOLS

**Usage**

```r
describeVastToolsEvent(event)
```

**Arguments**

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

**Details**

Junctions are parsed from

**Value**

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

Only supports to parse one event at a time.

Examples

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

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

Description

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

Usage

parseVastToolsSE(junctions)
parseVastToolsRI(junctions, strand)
parseVastToolsA3SS(junctions)
parseVastToolsA5SS(junctions)

Arguments

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

Details

The following event types are available to be parsed:

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

Value

List of parsed junctions

See Also

parseVastToolsEvent
Examples

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

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

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

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

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

---

**pcaServer**

*Server logic for the principal component analysis*

**Description**

Server logic for the principal component analysis

**Usage**

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

**Arguments**

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

**Value**

NULL (this function is used to modify the Shiny session’s state)
**pcaUI**  
*User interface of the principal component analysis*

**Description**  
User interface of the principal component analysis

**Usage**  
pcaUI(id)

**Arguments**
- **id**  
  Character: identifier

**Value**  
HTML element

**performPCA**  
*Perform principal component analysis after processing missing values from data frame*

**Description**  
Perform principal component analysis after processing missing values from data frame

**Usage**  
performPCA(data, center = TRUE, scale. = FALSE, naTolerance = 0)

**Arguments**
- **data**  
  Data frame: data
- **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.
- **naTolerance**  
  Integer: percentage of NA tolerance

**Value**  
PCA result in a prcomp object

**Examples**
performPCA(USArrests)
plotDistribution  

Plot distribution through a density plot

Description

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

Usage

plotDistribution(psi, groups, rug = TRUE, vLine = TRUE, ..., title = NULL)

Arguments

psi  
Numeric: quantification of one alternative splicing event

groups  
Character: group of each PSI index

rug  
Boolean: include rug plot to better visualise data distribution (TRUE by default)

vLine  
Boolean: include vertical plot lines to indicate the mean and median of each group even when those groups are omitted

...  
Extra parameters passed to density to create the kernel density estimates

title  
Character: plot title

Value

Highcharter object with density plot

Examples

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

plotMiniSurvivalCurves

Perform and plot survival curves

Description

Perform and plot survival curves

Usage

plotMiniSurvivalCurves(i, input, survParams, clinical, match, psi, censoring, event, timeStart, timeStop)
Arguments

i Numeric: index of the survival curves plot of interest
input Shiny input
survParams List of parameters to plot survival curves
clinical Data frame: clinical data
match Integer: samples matched with clinical patients
psi Data frame or matrix: alternative splicing quantification
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

Value

A "highchart" object to plot

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)

Arguments

pca prcomp object
pcX Character: name of the xAxis of interest from the PCA
pcY Character: name of the yAxis of interest from the PCA
groups Matrix: groups to plot indicating the index of interest of the samples (use clinical or sample groups)
individuals Boolean: plot PCA individuals (TRUE by default)
loadings Boolean: plot PCA loadings/rotations (FALSE by default)

Value

Scatterplot as an Highcharter object
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)
```

---

plotProtein

*Plot protein features*

Description

Plot protein features

Usage

```r
plotProtein(protein)
```

Arguments

- **protein** Character: UniProt protein identifier

Value

highchart object

Examples

```r
## Not run:
plotProtein("P38398")
## End(Not run)
```

---

plotSurvivalCurves

*Plot survival curves*

Description

Plot survival curves

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>surv</td>
<td>Survival object</td>
</tr>
<tr>
<td>mark</td>
<td>Boolean: mark times? TRUE by default</td>
</tr>
<tr>
<td>interval</td>
<td>Boolean: show interval ranges? FALSE by default</td>
</tr>
<tr>
<td>pvalue</td>
<td>Numeric: p-value of the survival curves</td>
</tr>
<tr>
<td>title</td>
<td>Character: plot title</td>
</tr>
<tr>
<td>scale</td>
<td>Character: time scale; default is “days”</td>
</tr>
</tbody>
</table>

Value

Plot of survival curves

Examples

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

plotTranscripts

Plot transcripts

Description

Plot transcripts

Usage

`plotTranscripts(info, eventPosition)`

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>info</td>
<td>Information retrieved from ENSEMBL</td>
</tr>
<tr>
<td>eventPosition</td>
<td>Numeric: coordinates of the alternative splicing event</td>
</tr>
</tbody>
</table>

Value

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

Examples

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

Create the explained variance plot

Description
Create the explained variance plot

Usage
plotVariance(pca)

Arguments
pca
PCA values

Value
Plot variance as an Highcharter object

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

prepareAnnotationFromEvents

Prepare annotation from alternative splicing events

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

Usage
prepareAnnotationFromEvents(...)

Arguments
... Data frame(s) of alternative splicing events to include in the annotation
Events from two or more data frames are cross-referenced based on each event’s chromosome, strand and specific coordinates relevant for each event type:

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

Value

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

Note

When cross-referencing events, gene information is discarded.

Examples

```r
# 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)
```

prepareFirehoseArchives

Prepares Firehose archives in a given directory

Description

Checks Firehose 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
prepareFirehoseArchives(archive, md5, folder, outdir)

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

Value
Invisible TRUE if successful

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

processButton
Style button used to initiate a process

Description
Style button used to initiate a process

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

Arguments
id Character: button identifier
label Character: label
... Extra parameters to pass to actionButton
class Character: class

Value
HTML for a button
### processDatasetNames

**Process dataset names**

**Description**

Process dataset names

**Usage**

```r
processDatasetNames(data)
```

**Arguments**

- `data`: List of lists of data frames

**Details**

Avoid duplicated names and append the technology used for junction quantification

**Value**

Processed list of lists of data frames

---

### processSurvData

**Process survival data to calculate survival curves**

**Description**

Process survival data to calculate survival curves

**Usage**

```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
- `followup`: Character: name of column containing follow up time
- `group`: Character: group of each individual
- `clinical`: Data frame: clinical data
- `survTime`: survTime object: Times to follow up, time start, time stop and event (optional)
Details

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

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

Value

Data frame with terms needed to calculate survival curves

---

**processSurvival**

*Check if survival analyses successfully completed or returned errors*

**Description**

Check if survival analyses successfully completed or returned errors

**Usage**

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

**Arguments**

- `session` Shiny session
- `...` Arguments to pass to function `processSurvTerms`

**Value**

List with survival analysis results

---

**processSurvTerms**

*Process survival curves terms to calculate survival curves*

**Description**

Process survival curves terms to calculate survival curves

**Usage**

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

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

Details

timeStop is only considered if censoring is either interval or interval2

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

Value

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

Examples

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

**Start graphical interface of PSICHOMICS**

**Description**

Start graphical interface of PSICHOMICS

**Usage**

```r
call = psychomics(..., reset = FALSE)
```

**Arguments**

- `...`: Parameters to pass to the function `runApp`
- `reset`: Boolean: reset Shiny session? FALSE by default; requires the package `devtools` to reset data

**Value**

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

**Examples**

```r
## Not run:
psychomics()
## 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**

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

**Arguments**

- `gene`: Character: gene
- `...`: Arguments to pass to `queryPubMed` function

**Value**

HTML interface of relevant PubMed articles
**quantifySplicing**  
*Quantify alternative splicing events*

**Description**  
Quantify alternative splicing events

**Usage**

```r
quantifySplicing(annotation, junctionQuant, eventType = c("SE", "MXE", "ALE", "AFE", "A3SS", "A5SS"), minReads = 10, progress = echoProgress)
```

**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: minimum of read counts to consider a junction read in calculations
- **progress**: Function to track the progress

**Value**

Data frame with the quantification of the alternative splicing events

**Examples**

```r
# 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"))
```

---

**queryEnsembl**  
*Query the Ensembl REST API*

**Description**  
Query the Ensembl REST API

**Usage**

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

**Arguments**

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

Value

Parsed response or NULL if there’s no response

Examples

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

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

queryEnsemblByEvent  Query information from Ensembl by a given alternative splicing event

Description

Query information from Ensembl by a given alternative splicing event

Usage

queryEnsemblByEvent(event, ...)

Arguments

  event  Character: alternative splicing event identifier
  ...    Arguments to pass to queryEnsemblByGene

Value

Information from Ensembl

Examples

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

queryEnsemblByGene  Query information from Ensembl by a given gene

Description

Query information from Ensembl by a given gene

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene</td>
<td>Character: gene identifier</td>
</tr>
<tr>
<td>species</td>
<td>Character: species (can be NULL when handling an ENSEMBL identifier)</td>
</tr>
<tr>
<td>assembly</td>
<td>Character: assembly version (can be NULL when handling an ENSEMBL identifier)</td>
</tr>
</tbody>
</table>

Value

Information from Ensembl

Examples

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

queryFirehoseData

Query the Firehose API for TCGA data

Description

Query the Firehose API for TCGA data

Usage

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

Arguments

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

Value

Response from the Firehose API (it needs to be parsed)
Examples

cohort <- psichomics:::getFirehoseCohorts()[1]
psichomics:::queryFirehoseData(cohort = cohort, data_type = "mRNASeq")

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

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

queryPubMed

Query the PubMed REST API

Description

Query the PubMed REST API

Usage

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

Arguments

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

Value

Parsed response

Examples

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

queryUniprot

Query the Uniprot REST API

Description

Query the Uniprot REST API

Usage

queryUniprot(protein, format = "xml")
renameDuplicated

Arguments

protein  Character: protein to query
format   Character: format of the response

Value

Parsed response

Examples

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

readFile

Load local file

Description

Load local file

Usage

readFile(file)

Arguments

file  Character: path to the file

Value

Loaded file

Examples

junctionQuant <- readFile("ex_junctionQuant.RDS")

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)
renameGroups

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 doesn’t return the comparator values.

Examples

```r
c(\text{psichomics:::renameDuplicated}(check = \text{c("blue", "red")}, comp = \text{c(\text{"green"}, \text{"blue"}})))
```

---

renameGroups  Rename duplicated names from a new group

Description

Rename duplicated names from a new group

Usage

```r
c(renameGroups(new, old))
```

Arguments

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

Value

Character with no duplicated group names

Note

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

*Render a data table with Sparkline HTML elements*

**Description**

Render a data table with Sparkline HTML elements

**Usage**

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

**Arguments**

- `...`: Arguments to pass to `renderDataTable`
- `options`: List of options to pass to `renderDataTable`

**Details**

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

**Value**

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

---

renderGeneticInfo

*Render genetic information*

**Description**

Render genetic information

**Usage**

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

**Arguments**

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

**Value**

HTML elements to render gene, protein and transcript annotation
**rm.null**

*Filter NULL elements from vector or list*

**Description**

Filter NULL elements from vector or list

**Usage**

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

**Arguments**

- `v` Vector or list

**Value**

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

---

**roundDigits**

*Round by the given number of digits*

**Description**

Round by the given number of digits

**Usage**

```r
roundDigits(n)
```

**Arguments**

- `n` Numeric: number to round

**Value**

Formatted number with a given numeric precision
**rowVar**  
*Sample variance by row*

---

**Description**

Calculate the sample variance of each row in the given matrix

**Usage**

```
rowVar(x, na.rm = FALSE)
```

**Arguments**

- `x`  
  Matrix
- `na.rm`  
  Boolean: should the NAs be ignored? FALSE by default

**Value**

Variance for each row

---

**selectGroupsServer**  
*Group selection logic*

---

**Description**

Group selection logic

**Usage**

```
selectGroupsServer(session, id)
```

**Arguments**

- `session`  
  Shiny session
- `id`  
  Character: identifier of the group selection

**Value**

Server logic for group selection
selectGroupsUI

Group selection interface

Description
Group selection interface

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

Arguments
- id: Character: identifier of the group selection
- label: Character: selectize label
- placeholder: Character: selectize placeholder
- noGroupsLabel: Character: label to show when no groups may be selected (if NULL, the option to show no groups will not be shown)
- groupsLabel: Character: label to show to the option of using groups when no groups may be selected

Value
Interface for group selection

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

See Also
selectGroupsServer getSelectedGroups

setActiveDataset

Set active dataset

Description
Set active dataset

Usage
setActiveDataset(dataset)

Arguments
- dataset: Character: dataset
**setAutoNavigation**

**Value**

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

**Note**

Needs to be called inside a reactive function

---

**setAssemblyVersion**  
*Set the assembly version of a data category*

**Description**

Set the assembly version of a data category

**Usage**

```
setAssemblyVersion(value, category = getCategory())
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Character: assembly version</td>
</tr>
<tr>
<td>category</td>
<td>Character: data category (e.g. &quot;Carcinoma 2016&quot;); by default, it uses the selected data category</td>
</tr>
</tbody>
</table>

**Value**

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

**Note**

Needs to be called inside a reactive function

---

**setAutoNavigation**  
*Set if history browsing is automatic*

**Description**

Set if history browsing is automatic

**Usage**

```
setAutoNavigation(param)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>param</td>
<td>Boolean: is navigation of browser history automatic?</td>
</tr>
</tbody>
</table>

**Value**

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

Needs to be called inside a reactive function

---

**setCategory**  
*Set data category*

**Description**

Set data category

**Usage**

```
setCategory(category)
```

**Arguments**

- **category**  
  Character: data category

**Value**

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

**Note**

Needs to be called inside a reactive function

---

**setClinicalMatchFrom**  
*Set clinical matches from a given data type*

**Description**

Set clinical matches from a given data type

**Usage**

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

**Arguments**

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

**Value**

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

**Note**

Needs to be called inside a reactive function
**setCores**  
*Set number of cores*

**Description**  
Set number of cores

**Usage**  
```
setCores(cores)
```

**Arguments**

- `cores`  
  Character: number of cores

**Value**  
NULL (this function is used to modify the Shiny session’s state)

**Note**  
Needs to be called inside a reactive function

---

**setData**  
*Set data of the global data*

**Description**  
Set data of the global data

**Usage**  
```
setData(data)
```

**Arguments**

- `data`  
  Data frame or matrix to set as data

**Value**  
NULL (this function is used to modify the Shiny session’s state)

**Note**  
Needs to be called inside a reactive function
setDifferentialAnalyses

Set the table of differential analyses of a data category

Description
Set the table of differential analyses of a data category

Usage
setDifferentialAnalyses(table, category = getCategory())

Arguments
- table: Character; differential analyses table
- category: Character; data category (e.g., "Carcinoma 2016"); by default, it uses the selected data category

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

Note
Needs to be called inside a reactive function

setDifferentialAnalysesSurvival

Set the table of differential analyses’ survival data of a data category

Description
Set the table of differential analyses’ survival data of a data category

Usage
setDifferentialAnalysesSurvival(table, category = getCategory())

Arguments
- table: Character; differential analyses’ survival data
- category: Character; data category (e.g., "Carcinoma 2016"); by default, it uses the selected data category

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

Note
Needs to be called inside a reactive function
**setEvent**  
*Set event*

**Description**  
Set event

**Usage**  

```r
setEvent(event)
```

**Arguments**  

- **event**  
  Character: event

**Value**  

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

**Note**  

Needs to be called inside a reactive function

---

**setFirehoseData**  
*Set data from Firehose*

**Description**  
Set data from Firehose

**Usage**  

```r
setFirehoseData(input, output, session, replace = TRUE)
```

**Arguments**  

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

**Value**  

NULL (this function is used to modify the Shiny session’s state)
**setGlobal**  
*Set element as globally accessible*

**Description**  
Set element as globally accessible

**Usage**  
```r
setGlobal(..., value, sep = "_")
```

**Arguments**

- `...`: Arguments to identify a variable
- `value`: Any value to attribute to an element
- `sep`: Character to separate identifier

**Details**  
Set element inside the global variable

**Value**  
NULL (this function is used to modify the Shiny session’s state)

**Note**  
Needs to be called inside a reactive function

---

**setGroupsFrom**  
*Set groups from a given data type*

**Description**  
Set groups from a given data type

**Usage**  
```r
setGroupsFrom(dataset, groups, category = getCategory())
```

**Arguments**

- `dataset`: Character: data set (e.g. "Clinical data")
- `groups`: Matrix: groups of dataset
- `category`: Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**  
NULL (this function is used to modify the Shiny session’s state)
**setInclusionLevels**  
*Set inclusion levels for a given data category*

**Description**  
Set inclusion levels for a given data category

**Usage**  
```r
setInclusionLevels(value, category = getCategory())
```

**Arguments**
- `value`: Data frame or matrix: inclusion levels
- `category`: Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**
NULL (this function is used to modify the Shiny session’s state)

**Note**
Needs to be called inside a reactive function

---

**setInclusionLevelsPCA**  
*Get principal component analysis based on inclusion levels*

**Description**  
Get principal component analysis based on inclusion levels

**Usage**  
```r
setInclusionLevelsPCA(pca, category = getCategory())
```

**Arguments**
- `pca`: `prcomp` object (PCA) of inclusion levels
- `category`: Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**
NULL (this function is used to modify the Shiny session’s state)

**Note**
Needs to be called inside a reactive function
setLocalData

Load local files

**Description**

Load local files

**Usage**

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

**Arguments**

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

**Value**

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

---

setPatientId

Set the identifier of patients for a data category

**Description**

Set the identifier of patients for a data category

**Usage**

```r
setPatientId(value, category = getCategory())
```

**Arguments**

- **value**: Character: identifier of patients
- **category**: Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**

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

**Note**

Needs to be called inside a reactive function
setPrecision  

Description
Set number of decimal places

Usage
setPrecision(precision)

Arguments
precision Numeric: number of decimal places

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

Note
Needs to be called inside a reactive function

setSampleId  

Description
Set the identifier of samples for a data category

Usage
setSampleId(value, category = getCategory())

Arguments
value Character: identifier of samples
category Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

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

Note
Needs to be called inside a reactive function
**setSampleInfo**

*Set sample information for a given data category*

**Description**

Set sample information for a given data category

**Usage**

```r
setSampleInfo(value, category = getCategory())
```

**Arguments**

- `value`: Data frame or matrix: sample information
- `category`: Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

**Value**

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

**Note**

Needs to be called inside a reactive function

---

**setSignificant**

*Set number of significant digits*

**Description**

Set number of significant digits

**Usage**

```r
setSignificant(significant)
```

**Arguments**

- `significant`: Character: number of significant digits

**Value**

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

**Note**

Needs to be called inside a reactive function
setSpecies

Set the species of a data category

Description
Set the species of a data category

Usage
setSpecies(value, category = getCategory())

Arguments
- value: Character: species
- category: Character: data category (e.g. "Carcinoma 2016"); by default, it uses the selected data category

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

Note
Needs to be called inside a reactive function

settingsServer

Server logic of the settings

Description
Server logic of the settings

Usage
settingsServer(input, output, session)

Arguments
- input: Shiny input
- output: Shiny output
- session: Shiny session

Value
NULL (this function is used to modify the Shiny session’s state)
settingsUI

**User interface of the settings**

**Description**
User interface of the settings

**Usage**
settingsUI(id, tab)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Character: identifier</td>
</tr>
<tr>
<td>tab</td>
<td>Function to create tabs</td>
</tr>
</tbody>
</table>

**Value**
HTML elements

---

setURLtoDownload

**Set URL links to download**

**Description**
Set URL links to download

**Usage**
setURLtoDownload(url)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>Character: URL links to download</td>
</tr>
</tbody>
</table>

**Value**
NULL (this function is used to modify the Shiny session’s state)

**Note**
Needs to be called inside a reactive function
showAlert  

**Show an alert**

**Description**

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

**Usage**

```r
showAlert(session, ..., title = NULL, style = NULL, dismissable = TRUE,
           alertId = "alert")
errorAlert(session, ..., title = NULL, dismissable = TRUE,
           alertId = "alert")
warningAlert(session, ..., title = NULL, dismissable = TRUE,
              alertId = "alert")
```

**Arguments**

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

**Value**

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

**See Also**

showModal

---

showGroupsTable  

**Present groups table**

**Description**

Present groups table

**Usage**

```r
showGroupsTable(datasetName)
```
**signifDigits**

**Arguments**
- `datasetName`  Character: name of dataset

**Value**
- Matrix with groups ordered (or NULL if no groups exist)

---

**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("wilcoxRankSum", "ttest", "kruskal", "levene", "fligner"))
```

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

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

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

Value

A row from a data frame with the results

```
sortCoordinates  Sort coordinates for some event types
```

Description

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

Usage

```
sortCoordinates(events)
```

Arguments

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

Value

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

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

Description

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

Usage

```
spearman(psi, groups)
```

Arguments

- `psi` Numeric: quantification of one alternative splicing event
- `groups` Character: group of each PSI index
**startProcess**

**Value**

HTML elements

---

**startProcess**  
*Signal the program that a process is starting*

**Description**

Style button to show processing is in progress

**Usage**

`startProcess(id)`

**Arguments**

- `id`  
  Character: button identifier

**Value**

Start time of the process

---

**startProgress**  
*Create a progress object*

**Description**

Create a progress object

**Usage**

`startProgress(message, divisions, global = sharedData)`

**Arguments**

- `message`  
  Character: progress message
- `divisions`  
  Integer: number of divisions in the progress bar
- `global`  
  Shiny’s global variable

**Value**

`NULL` (this function is used to modify the Shiny session’s state)
styleModal

Style and show a modal

Description
You can also use errorModal and warningModal to use template modals already stylised to show errors and warnings respectively.

Usage

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

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

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

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

Arguments

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

Value

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

See Also

- showAlert
**survdiff.survTerms**

Test difference between two or more survival curves using processed survival terms

**Description**

Test difference between two or more survival curves using processed survival terms

**Usage**

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

**Arguments**

- `survTerms`: survTerms object: processed survival terms
- `...`: Extra arguments passed to `survdiff`

**Value**

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

**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"
etvent <- "days_to_death"
formulaStr <- "patient.stage_event.pathologic_stage + patient.gender"
survTerms <- processSurvTerms(clinical, censoring="right", event, timeStart, formulaStr=formulaStr)
survdiff.survTerms(survTerms)
```

**survfit.survTerms**

Compute estimate of a survival curve using processed survival terms

**Description**

Compute estimate of a survival curve using processed survival terms

**Usage**

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

Arguments

survTerms  survTerms object: processed survival terms
...  Extra arguments passed to survfit

Value

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

Examples

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

survivalServer  Server logic of survival analysis

Description

Server logic of survival analysis

Usage

survivalServer(input, output, session)

Arguments

input  Shiny input
output  Shiny output
session  Shiny session

Value

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

User interface of survival analysis

Description
User interface of survival analysis

Usage
survivalUI(id)

Arguments

id Character: namespace identifier

Value
Character with HTML

tabDataset

Creates a tabPanel template for a datatable with a title and description

Description
Creates a tabPanel template for a datatable with a title and description

Usage
tabDataset(ns, title, tableId, columns, visCols, data, description = NULL)

Arguments

ns Namespace function
title Character: tab title
tableId Character: id of the datatable
columns Character: column names of the datatable
visCols Boolean: visible columns
data Data frame: dataset of interest
description Character: description of the table (optional)

Value
The HTML code for a tabPanel template
templateServer  

Server logic of template

Description
Server logic of template

Usage

templateServer(input, output, session)

Arguments

input  Shiny input
output  Shiny output
session  Shiny session

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

templateUI  

User interface of template

Description
User interface of template

Usage

templateUI(id)

Arguments

id  Character: namespace identifier

Value
HTML elements for the interface of the template
testSurvival

Test the survival difference between survival groups

Description
Test the survival difference between survival groups

Usage

\texttt{testSurvival(survTerms, \ldots)}

Arguments

- \texttt{survTerms} \hspace{1cm} survTerms object: processed survival terms
- \texttt{\ldots} \hspace{1cm} Extra arguments passed to \texttt{survdiff}

Value
p-value of the survival difference or NA

Note
Instead of raising errors, an NA is returned

Examples

\begin{verbatim}
require("survival")
data <- aml
timeStart <- "event"
event <- "event"
followup <- "time"
data$event <- NA
data$event[aml$status == 1] <- aml$time[aml$status == 1]
censoring <- "right"
formulaStr <- "x"
survTerms <- processSurvTerms(data, censoring=censoring, event=event,
timeStart=timeStart, followup=followup,
formulaStr=formulaStr)
testSurvival(survTerms)
\end{verbatim}

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

\texttt{testSurvivalCutoff(cutoff, data, filter = TRUE, clinical, \ldots,
session = NULL)}
textSuggestions

**Arguments**

- **cutoff** Numeric: Cut-off of interest
- **data** Numeric: elements of interest to test against the cut-off
- **filter** Boolean or numeric: elements to use (all by default)
- **clinical** Data frame: clinical data
- ... Arguments to pass to `processSurvTerms`
- **session** Shiny session

**Value**

- p-value of the survival difference

---

textSuggestions Create script for autocompletion of text input

**Description**

Uses the JavaScript library `jquery.textcomplete`

**Usage**

```r
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**

```r
words <- c("tumor_stage", "age", "gender")
psychomics:::textSuggestions("textareaid", words)
```
timePerPatient

*Get all columns matching a given string and return a single vector with the max time for each patient if available*

**Description**

Get all columns matching a given string and return a single vector with the max time for each patient if available

**Usage**

```r
timePerPatient(col, clinical)
```

**Arguments**

- `col` Character: column of interest
- `clinical` Data.frame: clinical data

**Value**

Numeric vector with days recorded for columns of interest

---

trimWhitespace

*Trims whitespace from a word*

**Description**

Trims whitespace from a word

**Usage**

```r
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 "))
```
ttest

Perform unpaired t-test analysis and return interface to show the results

**Usage**

```r
ttest(psi, groups, stat = NULL)
```

**Arguments**

- `psi`: Numeric: quantification of one alternative splicing event
- `groups`: Character: group of each PSI index
- `stat`: Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)

**Value**

HTML elements

uniqueBy

Check unique rows of a data frame based on a set of its columns

**Description**

Check unique rows of a data frame based on a set of its columns

**Usage**

```r
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)

Arguments
session  Shiny session

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

updateProgress  Update a progress object

Description
Update a progress object

Usage
updateProgress(message = "Hang in there", value = NULL, max = NULL,
detail = NULL, divisions = NULL, global = sharedData, console = TRUE)

Arguments
message  Character: progress message
value    Integer: current progress value
max      Integer: maximum progress value
detail   Character: detailed message
divisions Integer: number of divisions in the progress bar
global   Shiny’s global variable
console  Boolean: print message to console? (TRUE by default)

Details
If divisions isn’t NULL, a progress bar is started with the given divisions. If value is NULL, the progress bar will be incremented by one; otherwise, the progress bar will be incremented by the integer given in value.

Value
NULL (this function is used to modify the Shiny session’s state)
vennEvents

**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

**Description**

Perform Wilcoxon analysis and return interface to show the results

**Usage**

wilcox(psi, groups, stat = NULL)

**Arguments**

- psi: Numeric: quantification of one alternative splicing event
- groups: Character: group of each PSI index
- stat: Data frame or matrix: values of the analyses to be performed (if NULL, the analyses will be performed)

**Value**

HTML elements
Index

addTCGAd ata, 7
analysesServer, 7
analysesUI, 8
appServer, 8
appUI, 9
articleUI, 9

basicStats, 10
browserHistory, 10
bsModal2, 11

calculateInclusionLevels, 11
checkFileFormat, 12
checkFirebrowse, 12
checkIntegrity, 13
checkSurvivalInput, 13
closeProgress, 14
createDataTab, 14
createDensitySparklines, 15
createGroup, 15
createGroupByAttribute, 16
createGroupByColumn, 16
createGroupById, 17
createGroupFromInput, 17
createJunctionsTemplate, 18
dataServer, 18
dataUI, 19
diffAnalyses, 19
diffSplicingEventServer, 20
diffSplicingEventUI, 21
diffSplicingServer, 21
diffSplicingTableServer, 22
diffSplicingTableUI, 22
diffSplicingUI, 23
disableTab, 23
downloadFiles, 24

echoProgress, 24
enableTab, 25
dropProcess, 25
ensemblToUniprot, 26
errorAlert (showAlert), 130
errorModal (styleModal), 134

escape, 26
export_highcharts, 27

filterGroups, 27
firebrowseUI, 28
fisher, 28
fligner, 29

getActiveDataset, 29
getAssemblyVersion, 30
getAutoNavigation, 30
getCategories, 31
getCategory, 31
getCategoryData, 31
getClinicalData, 32
getClinicalMatchFrom, 32
getColumnsTime, 33, 105, 106
getCores, 33
data, 34
dataRows, 34
generateDifferentialAnalyses, 35
generateDifferentialAnalysesSurvival, 35
generateDifferentialAnalysesSurvival, 35
generateDownloadsFolder, 36
generateEvent, 36
generateFirehoseCohorts, 36
generateFirehoseDataTypes, 37
generateFirehoseDateFormat, 37
generateFirehoseDates, 38
generateGlobal, 38
generateGroupsFrom, 39
generateInclusionLevels, 39
generateInclusionLevelsPCA, 40
generateJunctionQuantification, 40
generateMatchingSamples, 41
generateNumerics, 41
generatePatientFromSample, 42
generatePatientId, 43
generatePrecision, 43
generatePSIperPatient, 44
generateSampleId, 44
generateSampleInfo, 45
generateSelectedGroups, 45
generateServerFunctions, 46
generateSignificant, 46
INDEX

getSpecies, 47
getSplicingEventCoordinates, 47
getSplicingEventTypes, 48
getUiFunctions, 48
getURLtoDownload, 49
getValidEvents, 49
globalSelectize, 50
groupByAttribute, 51
groupByExpression, 51
groupByGrep, 52
groupById, 52
groupPerPatient, 53
groupPerSample, 53
groupsServer, 54
groupsServerOnce, 54
groupsUI, 55
gtexDataServer, 55
gtexDataUI, 56
hc_scatter, 57
hchart.survfit, 56
inclusionLevelsInterface, 58
inclusionLevelsServer, 58
inclusionLevelsUI, 59
infoModal(styleModal), 134
infoServer, 59
infoUI, 60
insideFile, 60
is. whole, 61
isFirehoseUp, 61
joinEventsPerType, 62
junctionString, 62
kruskal, 63
labelBasedOnCutoff, 63
levene, 64
leveneTest, 64
listAllAnnotations, 65
listSplicingAnnotations, 66
loadAnnotation, 66
loadBy, 67
loadDataModal, 67
loadFile, 68
loadFileFormats, 68
loadFirehoseData, 69
loadFirehoseFolders, 69
loadGtexData, 70
loadLocalFiles, 70
loadRequiredData(missingDataModal), 72
localDataServer, 71
localDataUI, 72
missingDataGuide(missingDataModal), 72
missingDataModal, 72
modTabPanel, 73
navSelectize, 73
noinfo, 74
operateOnGroups, 74
optimalPSIcutoff, 75
optimSurvDiff, 76
optimSurvDiffOptions, 77
parseDateResponse, 77
parseFirehoseMetadata, 78
parseMatsA3SS(parseMatsGeneric), 79
parseMatsA5SS(parseMatsGeneric), 79
parseMatsSAFE(parseMatsGeneric), 79
parseMatsALE(parseMatsGeneric), 79
parseMatsAnnotation
(parseSuppaAnnotation), 87
parseMatsEvent, 78, 80
parseMatsGeneric, 79
parseMatsMXE(parseMatsGeneric), 79
parseMatsRI(parseMatsGeneric), 79
parseMatsSE(parseMatsGeneric), 79
parseMisoA3SS(parseMisoGeneric), 83
parseMisoA5SS(parseMisoGeneric), 83
parseMisoSAFE(parseMisoGeneric), 83
parseMisoALE(parseMisoGeneric), 83
parseMisoAnnotation
(parseSuppaAnnotation), 87
parseMisoEvent, 81, 84
parseMisoEventID, 82
parseMisoGeneric, 83
parseMisoId, 86
parseMisoMXE(parseMisoGeneric), 83
parseMisoRI(parseMisoGeneric), 83
parseMisoSE(parseMisoGeneric), 83
parseMisoTandemUTR(parseMisoGeneric), 83
parseSampleGroups, 86
parseSplicingEvent, 87
parseSuppaA3SS(parseSuppaGeneric), 90
parseSuppaA5SS(parseSuppaGeneric), 90
parseSuppaSAFE(parseSuppaGeneric), 90
parseSuppaALE(parseSuppaGeneric), 90
parseSuppaAnnotation, 87
parseSuppaEvent, 89, 91
parseSuppaGeneric, 90
parseSuppaMXE(parseSuppaGeneric), 90
parseSuppaRI(parseSuppaGeneric), 90
INDEX

parseSuppaSE (parseSuppaGeneric), 90
parseTcgaSampleInfo, 91
parseUniprotXML, 92
parseUrlsFromFirehoseResponse, 92
parseValidFile, 93
parseVastToolsA3SS (parseVastToolsSE), 94
parseVastToolsA5SS (parseVastToolsSE), 94
parseVastToolsAnnotation (parseSuppaAnnotation), 87
parseVastToolsEvent, 93, 94
parseVastToolsRI (parseVastToolsSE), 94
pcaServer, 95
pcaUI, 96
performPCA, 96
plotDistribution, 97
plotMiniSurvivalCurves, 97
plotPCA, 98
plotProtein, 99
plotSurvivalCurves, 99
plotTranscripts, 100
plotVariance, 101
prepareAnnotationFromEvents, 101
prepareFirehoseArchives, 102
processButton, 103
processDatasetNames, 104
processSurvData, 104
processSurvival, 105
processSurvTerms, 105
psichomics, 107
pubmedUI, 107
quantifySplicing, 108
queryEnsembl, 108
queryEnsemblByEvent, 109
queryEnsemblByGene, 109
queryFirehoseData, 69, 110
queryPubMed, 111
queryUniprot, 111
readFile, 112
renameDuplicated, 112
renameGroups, 113
renderDataTable, 114
renderDataTableSparklines, 114
renderGeneticInfo, 114
rm.null, 115
roundDigits, 115
rowVar, 116
scale, 96
selectGroupsServer, 116
selectGroupsUI, 117
setActiveDataset, 117
setAssemblyVersion, 118
setAutoNavigation, 118
setCategory, 119
setClinicalMatchFrom, 119
setCores, 120
setData, 120
setFirehoseData, 122
setGlobal, 123
setGroupsFrom, 123
setInclusionLevels, 124
setInclusionLevelsPCA, 124
setLocalData, 125
setPatientId, 125
setPrecision, 126
setSampleId, 126
setSampleInfo, 127
setSignificant, 127
setSpecies, 128
settingsServer, 128
settingsUI, 129
setURLtoDownload, 129
showAlert, 130
showGroupsTable, 130
signifDigits, 131
singleDiffAnalyses, 131
sortCoordinates, 132
spearman, 132
startProcess, 133
startProgress, 133
styleModal, 134
survdiff.survTerms, 135
survfit.survTerms, 135
survivalServer, 136
survivalUI, 137
tabDataset, 137
templateServer, 138
templateUI, 138
testSurvival, 139
testSurvivalCutoff, 139
textSuggestions, 140
timePerPatient, 141
trimWhitespace, 141
ttest, 142
uniqueBy, 142
updateClinicalParams, 143
updateProgress, 143

vennEvents, 144

warningAlert (showAlert), 130
warningModal (styleModal), 134
wilcox, 144