Package 'omicsViewer'

May 6, 2026

Title Interactive and explorative visualization of SummarizedExperssionSet or ExpressionSet using omicsViewer

Version 1.17.0

Description omicsViewer visualizes ExpressionSet (or SummarizedExperiment) in an interactive way. The omicsViewer has a separate back- and front-end. In the back-end, users need to prepare an ExpressionSet that contains all the necessary information for the downstream data interpretation. Some extra requirements on the headers of phenotype data or feature data are imposed so that the provided information can be clearly recognized by the front-end, at the same time, keep a minimum modification on the existing ExpressionSet object. The pure dependency on R/Bioconductor guarantees maximum flexibility in the statistical analysis in the back-end. Once the ExpressionSet is prepared, it can be visualized using the front-end, implemented by shiny and plotly. Both features and samples could be selected from (data) tables or graphs (scatter plot/heatmap). Different types of analyses, such as enrichment analysis (using Bioconductor package fgsea or fisher's exact test) and STRING network analysis, will be performed on the fly and the results are visualized simultaneously. When a subset of samples and a phenotype variable is selected, a significance test on means (t-test or ranked based test; when phenotype variable is quantitative) or test of independence (chi-square or fisher's exact test; when phenotype data is categorical) will be performed to test the association between the phenotype of interest with the selected samples. Additionally, other analyses can be easily added as extra shiny modules. Therefore, omicsViewer will greatly facilitate data exploration, many different hypotheses can be explored in a short time without the need for knowledge of R. In addition, the resulting data could be easily shared using a shiny server. Otherwise, a standalone version of omicsViewer together with designated omics data could be easily created by integrating it with portable R, which can be shared with collaborators or submitted as supplementary data together with a manuscript.

Depends R (>= 4.2)

License GPL-2

Imports htmltools, shinydashboard, survminer, survival, fastmatch, reshape2, stringr, beeswarm, grDevices, DT, shiny, shinythemes, shinyWidgets, plotly, networkD3, httr, matrixStats, RColorBrewer, Biobase, fgsea, openxlsx, psych, shinybusy, ggseqlogo, htmlwidgets, graphics, grid, stats, utils, methods, shinyjs, curl, flatxml, ggplot2, S4Vectors, SummarizedExperiment, RSQLite, Matrix, shinycssloaders, ROCR, drc

Suggests BiocStyle, knitr, rmarkdown, unittest

VignetteBuilder knitr

LazyData false

Encoding UTF-8

biocViews Software, Visualization, GeneSetEnrichment, DifferentialExpression, MotifDiscovery, Network, NetworkEnrichment

BugReports https://github.com/mengchen18/omicsViewer

URL https://github.com/mengchen18/omicsViewer

Video https://www.youtube.com/watch?v=0nirB-exquY&list=PLo2m88lJf-RRoLKMY8UEGqCpraKYrX5lk

RoxygenNote 7.3.3

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

git_branch devel

git_last_commit 8461845

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-05

Author Chen Meng [aut, cre]

Maintainer Chen Meng <mengchen18@gmail.com>

Contents

`.e2EC50` *convert e (inflection point) to EC50*

Description

convert e (inflection point) to EC50

Usage

```
.e2EC50(b, d, e, f)
```

Arguments

`b`	Hill's slope. The Hill's slope refers to the steepness of the curve. It could either be positive or negative.
`d`	Hihgest response value.
`e`	Inflection point. The inflection point is defined as the point on the curve where the curvature changes direction or signs. In models where f = 1 (2-4 parameter models), e is EC50.
`f`	Asymmetry factor. When f=1 we have a symmetrical curve around inflection point and so we have a four-parameters logistic equation.

Note

Only has an effect when using LL.5 and LL2.5 model

`.modelFormula` *model fitted by drc*

Description

model fitted by drc

Usage

```
.modelFormula(x, b, c = 0, d = 1, e, f = 1)
```

Arguments

x	numerical vector of doses/time points/concentrations
b	Hill's slope. The Hill's slope refers to the steepness of the curve. It could either be positive or negative.
c	Lowest response value.
d	Hihgest response value.
e	Inflection point. The inflection point is defined as the point on the curve where the curvature changes direction or signs. In models where f = 1 (2-4 parameter models), e is EC50.
f	Asymmetry factor. When f=1 we have a symmetrical curve around inflection point and so we have a four-parameters logistic equation.

Details

func(x) = c + (d - c) / (1 + (x/e)^b)^f

---

app_module *omicsViewer Application Server Logic (Level 0)*

Description

Implements the main server-side logic for the omicsViewer Shiny application. Handles data loading, validation, state management, snapshot functionality, and orchestrates communication between sub-modules. Uses modern Shiny module pattern with `moduleServer`. Primarily intended for developers extending the application.

Usage

```
app_module(
  id,
  .dir,
  filePattern = ".(RDS|db|sqlite|sqlite3)$",
  additionalTabs = NULL,
  ESVObj = reactive(NULL),
  esetLoader = readESVObj,
  exprsGetter = getExprs,
  pDataGetter = getPData,
  fDataGetter = getFData,
  imputeGetter = getExprsImpute,
  defaultAxisGetter = getAx,
  appName = "omicsViewer",
  appVersion = packageVersion("omicsViewer")
)
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in app_ui.
.dir	Reactive expression. Returns the directory path containing data files (ExpressionSet or SummarizedExperiment .RDS files).
filePattern	Character. Regular expression to filter displayed files. Default: ".(RDS|db|sqlite|sqlite3)$" (case-insensitive).
additionalTabs	List or NULL. Custom analysis modules to add to the application. Each element should contain: tabName, moduleName, moduleUi, and moduleServer. Default: NULL (no additional tabs).
ESVObj	Reactive expression. Returns a pre-loaded ExpressionSet or SummarizedExperiment object, bypassing file loading. Default: reactive(NULL).
esetLoader	Function. Loads data objects from disk. Takes file path as input, returns ExpressionSet or SummarizedExperiment. Default: readESVObj.
exprsGetter	Function. Extracts expression matrix from loaded object. Default: getExprs.
pDataGetter	Function. Extracts sample/phenotype metadata. Default: getPData.
fDataGetter	Function. Extracts feature metadata. Default: getFData.
imputeGetter	Function. Extracts imputed expression matrix (if available) for Excel export. Should return NULL if no imputed data. Default: getExprsImpute.
defaultAxisGetter	Function. Determines default plot axes. Takes object and what ("sx", "sy", "fx", "fy") as arguments. Default: getAx.
appName	Character. Application name displayed in UI. Default: "omicsViewer".
appVersion	Character or package_version. Version shown in UI. Default: current package version.

Details

The module coordinates several key functionalities:

- **Data Loading**: Validates file paths, checks file sizes, loads with error handling
- **Data Validation**: Ensures rownames/colnames consistency across expression and metadata
- **State Management**: Tracks selected features/samples across sub-modules
- **Snapshots**: Save and restore analysis states to disk (.ESS files)
- **Data Export**: Generate Excel files with expression data, metadata, and gene sets
- **Module Coordination**: Manages data space (L1_data_space_module) and result space (L1_result_space_module) interactions

Security features include path traversal prevention, file type validation, and size limits (2GB maximum).

Value

NULL (invisibly). The module manages reactive state internally and communicates with child modules. No explicit return value.

See Also

app_ui for the corresponding UI function. L1_data_space_module, L1_result_space_module for sub-modules. omicsViewer for the high-level launcher.

Examples

```
if (interactive()) {
  dir <- system.file("extdata", package = "omicsViewer")
  ui <- fluidPage(app_ui("app"))
  server <- function(input, output, session) {
    app_module("app", .dir = reactive(dir))
  }
  shinyApp(ui = ui, server = server)
}
```

---

app_ui *omicsViewer Application UI (Level 0)*

Description

Generates the user interface for the main omicsViewer application. This function creates a responsive layout with data exploration panels, snapshot functionality, and data export capabilities. Primarily intended for developers extending the application.

Usage

```
app_ui(id, showDropList = TRUE, activeTab = "Feature")
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in app_module.
showDropList	Logical. Whether to display the file selection dropdown menu. Set to FALSE when providing data directly via ESVObj parameter in app_module. Default: TRUE.
activeTab	Character. Initial tab to display when data is loaded. Options: • "Feature" - Feature space scatter plot • "Feature table" - Feature metadata table • "Sample" - Sample space scatter plot • "Sample table" - Sample metadata table • "Cor" - Correlation heatmap • "Heatmap" - Expression heatmap • "Dynamic heatmap" - Interactive heatmap with selection • "Expression" - Expression matrix table • "GSList" - Gene set membership table Default: "Feature".

Value

A `fluidRow` containing the complete UI structure, including:

- File selection dropdown (if `showDropList = TRUE`)
- Data summary display
- Export and snapshot buttons
- Two-column layout with data space (left) and analysis space (right)

See Also

`app_module` for the corresponding server logic. `omicsViewer` for the high-level application launcher.

Examples

```
if (interactive()) {
  dir <- system.file("extdata", package = "omicsViewer")
  ui <- fluidPage(
    app_ui("app", showDropList = TRUE, activeTab = "Feature")
  )
  server <- function(input, output, session) {
    app_module("app", .dir = reactive(dir))
  }
  shinyApp(ui = ui, server = server)
}
```

---

asEsetWithAttr *Convert SummarizedExperiment to ExpressionSet retaining all attributes*

Description

Convert SummarizedExperiment to ExpressionSet retaining all attributes

Usage

```
asEsetWithAttr(x)
```

Arguments

x	an object of class SummarizedExperiment

Value

an object of class ExpressionSet

batch_comparison_module *Batch Comparison Server Function*

Description

Server logic for the batch comparison module. Automatically tests all phenotype variables or all features for differences between selected and unselected samples with appropriate statistical tests and FDR correction.

Usage

```
batch_comparison_module(
  id,
  reactive_expr,
  reactive_phenoData,
  reactive_featureData,
  reactive_i_samples
)
```

Arguments

id	Character. Namespace ID for the Shiny module.
reactive_expr	Reactive expression. Returns numeric matrix with features as rows and samples as columns.
reactive_phenoData	Reactive expression. Returns data.frame of sample metadata.
reactive_featureData	Reactive expression. Returns data.frame of feature metadata.
reactive_i_samples	Reactive expression. Returns integer vector of selected sample indices.

Value

Reactive value containing information about the selected row from the results table.

See Also

batch_comparison_ui for the corresponding UI function.

Other analysis modules: batch_comparison_ui(), dose_response_module(), dose_response_ui()

batch_comparison_ui *Batch Comparison UI Function*

Description

Creates the user interface for the batch comparison module that automatically compares either all phenotype variables or all features between selected and unselected sample groups.

Usage

```
batch_comparison_ui(id)
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in batch_comparison_module.

Value

A tagList containing:

- Toggle buttons to select comparison mode (phenotype vs features)
- Results table with p-values and FDR correction
- Download functionality for results
- Selected row information panel

See Also

batch_comparison_module for the corresponding server logic.

Other analysis modules: batch_comparison_module(), dose_response_module(), dose_response_ui()

constants *Application Constants*

Description

This file contains named constants used throughout the omicsViewer application. Using named constants instead of "magic numbers" improves code readability, maintainability, and makes it easier to update values in a single location.

`correlationAnalysis` *Correlating a expression matrix with phenotypical variables*

Description

This is a convenience function to perform correlation analysis, the output is in a format ready to be incorporated into object to be visualized by `omicsViewer`.

Usage

```
correlationAnalysis(
  x,
  pheno,
  min.value = MIN_SAMPLES_CORRELATION,
  prefix = "Cor"
)
```

Arguments

`x`	an expression matrix, rows are the features (e.g. proteins), columns are the samples
`pheno`	a `data.frame` storing the numerical phenotypical variable to be correlated with the rows (features) in expression matrix.
`min.value`	the minimum number of samples required in the correlation analysis, if lower than this number, NA will be returned.
`prefix`	prefix of the names. Usually don't need to be changed by the user. When changes are needed, the prefix should be in a format like [analysis name]|[subset] so the "analysis name" and "subset" can be selected in the `omicsViewer`.

Value

Every correlation analysis returns a `data.frame` with five columns: `R` - pearson correlation coefficient `N` - number of values used in the analysis `P` - p-values returned by pearson correlation analysis `logP` - log transformed p-values `range` - the range of values in expression matrix used in the analysis

Examples

```
e1 <- matrix(rnorm(500), 50, 10)
rownames(e1) <- paste0("FT", 1:50)
p1 <- matrix(rnorm(50), 10, 5)
colnames(p1) <- paste0("PH", 1:5)
colnames(e1) <- rownames(p1) <- paste0("S", 1:10)
correlationAnalysis(x = e1, pheno = p1, min.value = 8)
```

csc2list *convert a column compressed sparse matrix to a list*

Description

convert a column compressed sparse matrix to a list

Usage

```
csc2list(x)
```

Arguments

x	a matrix or CsparseMatrix object

Value

a sparse frame in data.frame

documentation_template *Documentation Template for omicsViewer Modules*

Description

This file provides the standard documentation template for all Shiny modules in the omicsViewer package. All module functions should follow this structure to ensure consistency and completeness.

dose_response_module *Dose Response Curve Server Function*

Description

Server logic for the dose response curve visualization module. Fits dose-response curves using the drc package and displays fitted parameters and plots for selected features.

Usage

```
dose_response_module(
  id,
  reactive_expr,
  reactive_phenoData,
  reactive_featureData,
  reactive_i,
  reactive_attr_drc
)
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in dose_response_ui.
reactive_expr	Reactive expression. Returns a numeric matrix with features as rows and samples as columns containing expression/response values.
reactive_phenoData	Reactive expression. Returns a data.frame of sample metadata including dose and curve ID columns.
reactive_featureData	Reactive expression. Returns a data.frame of feature metadata with features as rows.
reactive_i	Reactive expression. Returns a single integer index indicating which feature row to display. Must have length 1.
reactive_attr_drc	Reactive expression. Returns a named character vector with elements "dose_col" and "curveid_col" specifying the column names in phenoData for dose values and curve identifiers.

Details

The module fits 4-parameter or 5-parameter log-logistic curves to the dose-response data and extracts parameters including EC50, hill slope, minimum and maximum responses. Only single feature selection is supported.

Value

NULL (invisibly). The module displays plots and tables but does not return a reactive value.

See Also

dose_response_ui for the corresponding UI function. plotDCMat for the plotting function. drmMat for curve fitting.

Other analysis modules: batch_comparison_module(), batch_comparison_ui(), dose_response_ui()

---

dose_response_ui *Dose Response Curve UI Function*

Description

Creates the user interface for the dose response curve visualization module. Displays dose-response curves with fitted parameters and feature information in a multi-panel layout.

Usage

```
dose_response_ui(id)
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in dose_response_module.

Value

A `fluidRow` containing:

- Plot panel: Dose-response curve visualization
- Parameter table: Fitted curve parameters (hill slope, EC50, etc.)
- Feature table: Selected feature information

See Also

`dose_response_module` for the corresponding server logic. `plotDCMat` for the underlying plotting function.

Other analysis modules: `batch_comparison_module()`, `batch_comparison_ui()`, `dose_response_module()`

`draw_roc_pr` *Drawing ROC and PR curve*

Description

Drawing ROC and PR curve

Usage

```
draw_roc_pr(value, label)
```

Arguments

`value`	a numerical vector indicates the predictions
`label`	true class labels, could be two or more unique values

Examples

```
v <- sort(rnorm(100))
l <- sample(1:2, size = 100, replace = TRUE)
draw_roc_pr(v, l)
l <- rep(c("b", "c", "a", "d"), each = 25)
draw_roc_pr(v, l)
draw_roc_pr(v, sample(l))
```

drmMat *Fitting dose-response models for omics data matrix*

Description

A convenient function to fit dose response models for every row in an omics matrix using drm function in the drc package.

Usage

```
drmMat(
  x,
  fitvar,
  fitvar.name = c("Dose", "Time", "Concentration")[1],
  curveid = NA,
  fct.name = c("LL.4()", "LL.3()", "LL.2()", "LL.5()")[1]
)
```

Arguments

x	a numerical matrix where the rows are features and columns are samples.
fitvar	a numerical variable has the same length as ncol(x) to indicate the dose/time/concentration conditions.
fitvar.name	the name of the fitvar, a length one character. Will be used as the label for x-axis when drawing the dose curve.
curveid	a numeric vector or factor containing the grouping of the columns in x.
fct.name	the function name, e.g. "LL.4()", "LL.3()", "LL.2()" and "LL.5()", which are defined in the drc package.

Value

a list of drc object

error_handling *Error Handling Utilities for API Calls*

Description

Utility functions for robust error handling of external API calls. Provides consistent error messages and graceful degradation when network requests fail.

exprspca *Perform PCA and prepare results for* omicsViewer

Description

This is a convenience function to perform PCA on expression matrix, the output of PCA will be in a format ready to be incorporated into object to be visualized by omicsViewer.

Usage

```
exprspca(x, n = min(8, ncol(x) - 1), prefix = "PCA|All", fillNA = FALSE, ...)
```

Arguments

x	an expression matrix, where rows are features and samples are on columns.
n	number of components to keep
prefix	prefix of the names. Usually don't need to be changed by the user. When changes are needed, the prefix should be in a format like [analysis name]|[subset] so the "analysis name" and "subset" can be selected in the omicsViewer.
fillNA	logical; whether NA should be filled? If FALSE (default), na.omit will be called before PCA. If TRUE, the missing value will be replaced using fillNA.
...	other parameters passed to prcomp

Value

a data.frame storing the PCA results

Examples

```
# reading expression
packdir <- system.file("extdata", package = "omicsViewer")
expr <- read.delim(file.path(packdir, "expressionMatrix.tsv"), stringsAsFactors = FALSE)
# call PCA
pc <- exprspca(expr)
head(pc$samples)
head(pc$features)
```

extendMetaData *Add extra columns to the phenoData/colData or featureData/rowData in ExpressionSet/SummarizedExperiment*

Description

Add extra columns to the phenoData/colData or featureData/rowData in ExpressionSet/SummarizedExperiment

Add extra columns to the phenoData/colData or featureData/rowData in ExpressionSet/SummarizedExperiment

Add extra columns to the phenoData/colData or featureData/rowData in ExpressionSet/SummarizedExperiment

Usage

```
extendMetaData(object, newData, where)

## S4 method for signature 'ExpressionSet,data.frame'
extendMetaData(
  object,
  newData,
  where = c("pData", "fData", "colData", "rowData")[1]
)

## S4 method for signature 'SummarizedExperiment,data.frame'
extendMetaData(
  object,
  newData,
  where = c("pData", "fData", "colData", "rowData")[1]
)

## S4 method for signature 'SummarizedExperiment,DFrame'
extendMetaData(
  object,
  newData,
  where = c("pData", "fData", "colData", "rowData")[1]
)
```

Arguments

object	an object of ExpressionSet-class
newData	a data.frame containing the data to be added
where	where to add the extra columns, should be one of "pData", "fData", "rowData" and "colData".

Value

an object of ExpressionSet-class

Note

The attributes in the pheno data and feature data will be preserved

Examples

```
est <- Biobase::ExpressionSet(assayData=matrix(runif(1000), nrow=100, ncol=10))
Biobase::pData(est)
est <- extendMetaData(est, data.frame(letter = letters[1:10]), where = "pData")
Biobase::pData(est)
```

extractParamDC *Extracting parameters from drc models*

Description

Extracting parameters from drc models

Usage

```
extractParamDC(mod, prefix = "ResponseCurve")
```

Arguments

mod	a drc object
prefix	for column header, the column will be named as prefix|curveid|curveparameter

Note

when LL2.X is used, e is estimated as log(e), this function will return e in linear scale instead.

extractParamDCList *Extracting parameter from a list of drc object*

Description

Extracting parameter from a list of drc object and return a data.frame, which can be incorporated into the object visualized by omicsViewer

Usage

```
extractParamDCList(x, prefix = "ResponseCurve")
```

Arguments

x	a list of drc object
prefix	for column header

Value

a data.frame

fgsea1 — *Wrapper of fgseaMultilevel function to take binary gene set matrix as input*

Description

Wrapper of fgseaMultilevel function to take binary gene set matrix as input

Usage

```
fgsea1(gs, stats, gs_desc = NULL, ...)
```

Arguments

gs	either a data.frame or a (sparse) matrix input. If a data.frame object is given, it should have at least three columns named as "featureId", "gsId" and "weight". If a matrix is given, the matrix is binary matrix where rows are features and columns are gene sets. The values in the matrix should be either 1 or 0 representing the presence and absence of a feature in the genesets, repectively.
stats	ranking stats
gs_desc	description of gene sets, it should be a named vector and the names should be the same as colnames(gs)
...	other parameters passed to fgseaMultilevel

Value

a `data.frame` of fgsea results

Examples

```
## not for users
# library(fgsea)
# library(Biobase)
# dat <- readRDS(system.file(package = "omicsViewer", "extdata/demo.RDS"))
# fd <- fData(dat)
# fdgs <- fd[, grep("^GS\|", colnames(fd))]
# res <- fgsea1(fdgs, stats = fd$`t-test|OV_BR|md`, minSize = 5, maxSize = 500)
# res <- fgsea1(
#   fdgs, stats = fd$`t-test|OV_BR|md`,
#   minSize = 5, maxSize = 500, gs_desc = colnames(fdgs))
```

fillNA — *Filling NAs in a matrix using constants calculated from user the defined function*

Description

This function is usually use to impute missing values in expression matrix, where the rows are feature and columns are samples. This function impute the missing values on the row-wise, that is, every row will be imputed using different constant.

Usage

```
fillNA(
  x,
  maxfill = quantile(x, probs = 0.15, na.rm = TRUE),
  fillingFun = function(x) min(x, na.rm = TRUE) - log10(2)
)
```

Arguments

x	a matrix with NA values
maxfill	the maximum filled value, if the value calculated by fillingFun is greater than maxfill, then maxfill will the used to replace NAs.
fillingFun	function to calculate the filling values. It should be a function accept at least one argument "x", which is a row of input expression matrix. The default is function(x) min(x, na.rm = TRUE) - log10(2) corresponds to the "half of lowest detected values" if the expression matrix is log10 transformed. More examples:#' function(x) min(x, na.rm = TRUE) - 1 # half of lowest detected value when expression matrix is in log2 scale function(x) 0 # replace NA by 0

Value

a matrix without NAs

Note

The returned matrix may have -Inf, which may need to be filtered/replaced additionally

Examples

```
m <- matrix(rnorm(200), 20, 10)
m[sample(1:200, size = 20)] <- NA
mf <- fillNA(m)
```

---

filterRow *Filter out rows of expression matrix*

Description

The function is used to filter rows with values of low intensities or do not reproducible presented in replicates.

Usage

```
filterRow(x, max.quantile = NULL, max.value = NULL, var = NULL, min.rep = 2)
```

Arguments

x	an expression matrix
max.quantile	a single numerical value between (0, 1), if the row maximum is smaller than this quantile (calculated from the whole matrix), the row will be removed.
max.value	a single numerical value, if the the maximum value of a rwo is smaller than this value, the row will be removed. Only used if max.quantile is set to "NULL".
var	variables has the same length as the column number in x to indicate which sample is from which group
min.rep	the minimum number of replicate in at least one of the groups, if less than this value, the row will be removed.

Value

a logical vector where the TRUE means row to keep

Examples

```
e1 <- matrix(rnorm(5000, sd = 0.3), 500, 10) + rnorm(500)
f <- filterRow(x = e1, max.quantile = 0.25)
table(f)
```

---

geneshot_module *Geneshot Search Server Function*

Description

Server logic for the Geneshot gene search module. Queries the Geneshot API to find genes associated with user-provided text terms, ranks results by publication metrics, and highlights overlap with selected features.

Usage

```
geneshot_module(
  id,
  pdata,
  fdata,
  expr,
  feature_selected,
  sample_selected,
  object,
  reactive_status = reactive(NULL)
)
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in geneshot_ui.
pdata	Reactive expression. Returns a data.frame of sample metadata (phenotype data). Currently unused but required for module consistency.

`fdata` Reactive expression. Returns a data.frame of feature metadata with features as rows. Used for ID mapping and overlap detection.

`expr` Reactive expression. Returns a numeric matrix with features as rows and samples as columns. Currently unused but required for module consistency.

`feature_selected` Reactive expression. Returns a character vector or integer vector of currently selected feature IDs/indices. Used to highlight overlap with search results.

`sample_selected` Reactive expression. Returns selected sample IDs. Currently unused but required for module consistency.

`object` Reactive expression. Returns the full ExpressionSet or SummarizedExperiment object. Currently unused but required for module consistency.

`reactive_status` Reactive expression. Returns a list containing saved state for session restoration. Optional. Default: `reactive(NULL)`.

Details

The module performs the following steps:

1. Accepts semicolon-separated search terms from user
2. Queries Geneshot API via `getAutoRIF`
3. Filters results to top 20 genes (GENESHOT_TOP_RESULTS)
4. Highlights genes that overlap with currently selected features
5. Displays results in interactive scatter plot and table

Handles API errors gracefully with user notifications.

Value

A reactive expression returning a list with module state for session saving.

See Also

`geneshot_ui` for the corresponding UI function. `getAutoRIF` for the API call function.

Other search modules: `geneshot_ui()`

`geneshot_ui` *Geneshot Search UI Function*

Description

Creates the user interface for the Geneshot gene search module. Allows users to search for genes associated with arbitrary text queries using the Geneshot API and visualize results by publication metrics.

Usage

```
geneshot_ui(id)
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in geneshot_module.

Value

A tagList containing:

- Text input for search terms (semicolon-separated)
- Submit button to trigger search
- Gene ID mapper selector
- Interactive scatter plot of search results
- Results table with download functionality

References

Lachmann et al. (2019) Geneshot: search engine for ranking genes from arbitrary text queries. Nucleic Acids Research 47(W1):W571-W577.

See Also

geneshot_module for the corresponding server logic. getAutoRIF for the underlying API function.

Other search modules: geneshot_module()

---

getAutoRIF *Get genes associated with search terms and AutoRIF annotations*

Description

Get genes associated with search terms and AutoRIF annotations

Usage

```
getAutoRIF(term, rif = c("generif", "autorif")[1], filter = TRUE)
```

Arguments

term	a character vector of terms want to search
rif	either autorif or generif, see "https://maayanlab.cloud/geneshot/"
filter	whether the result should be filtered. The least frequently mentioned genes (most like 1 or 2 times) will be removed.

Value

a data.frame of 4 columns: gene, n, perc, rank.

Note

https://amp.pharm.mssm.edu/geneshot/

References

Alexander Lachmann, Brian M Schilder, Megan L Wojciechowicz, Denis Torre, Maxim V Kuleshov, Alexandra B Keenan, Avi Ma'ayan, Geneshot: search engine for ranking genes from arbitrary text queries, Nucleic Acids Research, Volume 47, Issue W1, 02 July 2019, Pages W571–W577, https://doi.org/10.1093/nar/gkz393

Alexander Lachmann, Brian M Schilder, Megan L Wojciechowicz, Denis Torre, Maxim V Kuleshov, Alexandra B Keenan, Avi Ma'ayan, Geneshot: search engine for ranking genes from arbitrary text queries, Nucleic Acids Research, Volume 47, Issue W1, 02 July 2019, Pages W571–W577, https://doi.org/10.1093/nar/gkz393

Examples

```
a <- getAutoRIF("mtor signaling")
```

---

getMQParams *Parse mqpar.xml file*

Description

Getting the experimental informatione (TMT or label free) from mqpar.xml file.

Usage

```
getMQParams(x)
```

Arguments

x	the path to mqpar.xml file

Value

a list of MQ paramters

---

getStringId *Mapping ids to string ids*

Description

the string ids can be used as background in the string enrichment analysis

Usage

```
getStringId(genes, taxid = 9606, caller = "omicsViewer")
```

Arguments

genes	a character vector of gene ids
taxid	taxonomy ids
caller	your identifier for string-db.org

Note

https://string-db.org/cgi/help.pl?subpage=api

Examples

```
1
# gg = c('P04637', 'P00533', 'P04626', "Q8IYB3", "O75494", "Q9Y696")
# getStringId(gg)
```

---

getUPRefProteomeID *get uniprot reference proteome IDs*

Description

get uniprot reference proteome IDs

get uniprot reference proteome IDs

Usage

```
getUPRefProteomeID(
  domain = c("Eukaryota", "Archaea", "Bacteria", "Viruses")[1]
)

downloadUPRefProteome(
  id,
  domain = c("Eukaryota", "Archaea", "Bacteria", "Viruses")[1],
  destdir = "./"
)
```

Arguments

domain	the domain, one of "Eukaryota", "Archaea", "Bacteria" or "Viruses"
id	the UP id to download
destdir	destination directory

Value

a character vector of UP ids

a character vector of UP ids

Functions

- `getUPRefProteomeID()`: get uniprot reference protein IDs

gsAnnotIdList *Annotation of gene/protein function using multiple IDs.*

Description

Annotation of gene/protein function using multiple IDs.

Usage

```
gsAnnotIdList(
  idList,
  gsIdMap,
  minSize = 5,
  maxSize = 500,
  data.frame = FALSE,
  sparse = TRUE
)
```

Arguments

idList	list of protein IDs, e.g. list(c("ID1", "ID2"), c("ID13"), c("ID4", "ID8", "ID10"))
gsIdMap	a data frame for geneset to id map, it has two columns - id: the ID column - term: annotation terms e.g. gsIdMap <- data.frame(id = c("ID1", "ID2", "ID1", "ID2", "ID8", "ID10"), term = c("T1", "T1", "T2", "T2", "T2", "T2"), stringsAsFactors = FALSE)
minSize	minimum size of gene sets
maxSize	maximum size of gene sets
data.frame	logical; whether to organize the result into data.frame format, see "Value" section.
sparse	logical; whether to return a sparse matrix, only used when data.frame=FALSE

Value

A binary matrix (if data.frame = FALSE), the number of rows is the same with length of idList, the columns are the annotated gene set; or a data.frame (if data.frame = TRUE) with three columns: featureId, gsId, weight.

Examples

```
terms <- data.frame(
  id = c("ID1", "ID2", "ID1", "ID2", "ID8", "ID10"),
 term = c("T1", "T1", "T2", "T2", "T2", "T2"),
  stringsAsFactors = FALSE
)
features <- list(c("ID1", "ID2"), c("ID13"), c("ID4", "ID8", "ID10"))
gsAnnotIdList(idList = features, gsIdMap = terms, minSize = 1, maxSize = 500)

terms <- data.frame(
id = c("ID1", "ID2", "ID1", "ID2", "ID8", "ID10", "ID4", "ID4"),
term = c("T1", "T1", "T2", "T2", "T2", "T2", "T1", "T2"),
stringsAsFactors = FALSE
```

```
)
features <- list(F1 = c("ID1", "ID2", "ID4"), F2 = c("ID13"), F3 = c("ID4", "ID8", "ID10"))
gsAnnotIdList(features, gsIdMap = terms, data.frame = TRUE, minSize = 1)
gsAnnotIdList(features, gsIdMap = terms, data.frame = FALSE, minSize = 1)
```

`gslist_module` *Gene Set List Server Function*

Description

Server logic for the gene set membership display module. Retrieves and displays gene set annotations for selected features from the feature data attributes.

Usage

```
gslist_module(id, reactive_featureData, reactive_i)
```

Arguments

`id`	Character. Namespace ID for the Shiny module. Must match the ID used in `gslist_ui`.
`reactive_featureData`	Reactive expression. Returns a data.frame of feature metadata with a "GS" attribute containing gene set membership information. The "GS" attribute should be a data.frame with columns: • featureId: Feature identifiers • gsId: Gene set identifiers • Additional annotation columns (optional)
`reactive_i`	Reactive expression. Returns feature IDs or indices to display. Can be: • Character vector of feature IDs • Integer vector of feature indices • Logical scalar TRUE (show all features) • NULL or NA (show all features)

Details

The module extracts gene set annotations from the "GS" attribute of feature data and filters to show only selected features. Gene sets are displayed with associated feature annotations from columns starting with "Generall".

Value

A reactive expression returning a character vector of feature IDs for the selected table row, or NULL if no row is selected.

See Also

`gslist_ui` for the corresponding UI function.

Other enrichment modules: `gslist_ui()`

gslist_ui *Gene Set List UI Function*

Description

Creates the user interface for the gene set membership display module. Shows which gene sets contain the selected features with downloadable results.

Usage

```
gslist_ui(id)
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in gslist_module.

Value

A tagList containing a data table with download functionality showing gene set memberships for selected features.

See Also

gslist_module for the corresponding server logic.

Other enrichment modules: gslist_module()

hasAttr *Check whether an object has an attribute*

Description

Check whether an object has an attribute

Usage

```
hasAttr(x, attr.name)
```

Arguments

x	the object
attr.name	a character vector containing the name of attributes to be checked

Value

a logical value/vector has the same length as attr.name

hclust2str *Convert hclust object to/from single character*

Description

Convert hclust object to/from single character

Usage

```
hclust2str(x)

str2hclust(x)
```

Arguments

x	a character of length one or an hclust object

Value

a character stores the hclust object

a hclust object

Note

The $call element in hclust will not retained in the conversion. The conversion decrease the precision in $height element.

Examples

```
# not for end users
# m <- matrix(rnorm(50), 25)
# hc <- hclust(dist(m))
# plot(hc)
# te <- hclust2str(hc)
# hc2 <- str2hclust(te)
# plot(hc2)
```

jaccardList *Calculate Jaccard distance from a list*

Description

Calculate Jaccard distance from a list

Usage

```
jaccardList(x)
```

Arguments

x	a list

Value

an dist object

L1_data_space_module *Application level 1 module - data space*

Description

Application level 1 module - data space

Usage

```
L1_data_space_module(
  id,
  expr,
  pdata,
  fdata,
  reactive_x_s = reactive(NULL),
  reactive_y_s = reactive(NULL),
  reactive_x_f = reactive(NULL),
  reactive_y_f = reactive(NULL),
  cormat = reactive(NULL),
  status = reactive(NULL)
)
```

Arguments

id	module id
expr	reactive value; expression matrix
pdata	reactive value; phenotype data
fdata	reactive value; feature data
reactive_x_s	pre-selected x axis for sample space
reactive_y_s	pre-selected y axis for sample space
reactive_x_f	pre-selected x axis for feature space
reactive_y_f	pre-selected y axis for feature space
cormat	reactive value; correlation matrix. if not given, calculated on the fly.
status	intial status

L1_data_space_ui *Application level 1 ui - data space*

Description

Application level 1 ui - data space

Usage

```
L1_data_space_ui(id, activeTab = "Feature")
```

Arguments

id	id
activeTab	one of "Feature", "Feature table", "Sample", "Sample table", "Heatmap"

L1_result_space_module *Utility L1 result space ui*

Description

Utility L1 result space ui

Usage

```
L1_result_space_module(
  id,
  reactive_expr,
  reactive_phenoData,
  reactive_featureData,
  reactive_i = reactive(NULL),
  reactive_highlight = reactive(NULL),
  additionalTabs = NULL,
  object = NULL,
  status = reactive(NULL)
)
```

Arguments

id	module id
reactive_expr	expression matrix
reactive_phenoData	phentype data
reactive_featureData	feature data
reactive_i	row ID/name of rows selected

reactive_highlight	col ID/name of columns selected
additionalTabs	additional tabs added to "Analyst" panel
object	originally loaded object, mostly an ExpressionSet or SummarizedExperiment object
status	intial status

L1_result_space_ui *Utility L1 result space ui*

Description

Utility L1 result space ui

Usage

```
L1_result_space_ui(id)
```

Arguments

id	id

---

list2csc *convert a list to column compressed sparse matrix*

Description

convert a list to column compressed sparse matrix

Usage

```
list2csc(l, dimnames)
```

Arguments

l	a data.frame with at least two columns - featureId, gsId; optionally a "weight" column.
dimnames	a list of dimnames, should contain at least one element for the row names.

Value

a sparse matrix, CsparseMatrix, column compressed

meta_scatter_module *Utility - scatter plot for meta shiny module*

Description

Utility - scatter plot for meta shiny module

Usage

```
meta_scatter_module(
  id,
  reactive_meta = reactive(NULL),
  reactive_expr = reactive(NULL),
  combine = c("pheno", "feature"),
  source = "plotlyscattersource",
  reactive_x = reactive(NULL),
  reactive_y = reactive(NULL),
  reactive_status = reactive(NULL)
)
```

Arguments

id	Character. Namespace ID for the Shiny module.
reactive_meta	reactive meta data, phenotype data or feature data
reactive_expr	reactive expression data
combine	how to combine the expression and meta data, pheno or feature?
source	source id for plotly object
reactive_x	reactive value for pre-selected x-aixs
reactive_y	reactive value for pre-selected y-aixs
reactive_status	the status of scatter plot, e.g. x-, y-axis, color variable, shape variable, etc.

meta_scatter_ui *Meta Scatter Plot UI Function*

Description

Creates the user interface for the metadata scatter plot visualization module. Provides interactive scatter plots with advanced selection tools including corner selection for volcano plots.

Usage

```
meta_scatter_ui(id)
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in meta_scatter_module.

Value

A tagList containing:

- Figure attribute selector (color, shape, size controls)
- Clear selection button
- X-axis and Y-axis variable selectors
- Interactive plotly scatter plot with lasso/box selection

See Also

meta_scatter_module for the corresponding server logic. plotly_scatter_module for the scatter plot implementation.

---

multi.t.test *Function to perform multiple t-tests on an expression matrix*

Description

This is a convenience function to perform multiple student's t-test. The output is in a format ready to be incorporated into object to be visualized by omicsViewer. This function use t.test.

Usage

```
multi.t.test(x, pheno, compare = NULL, fillNA = FALSE, ...)
```

Arguments

x	an expression matrix, usually log10 transformed.
pheno	phenotype data of x, the number of rows in pheno must equal the number of columns of x. Please refer to examples for more details.
compare	NULL or a matrix with three columns to define the comparisons to do. When a matrix is given, the first column should be one of the column headers in pheno; then the second and third columns should be two values presented (more than once) in the columns of pheno selected by the values in the first column. The samples mapped to the two values are compared. If paired comparisons to be done, the orders of samples should be mapped
fillNA	logical; whether NA should be filled? If FALSE (default), t test will be performed whenever possible. If not possible, then NA will be returned. If TRUE, the missing value will be replaced using fillNA.
...	other parameters passed to t.test

Value

a data.frame stores the t-test results with the follow columns: mean|[selected header in pheno]|[group 1 in test] - The mean value of group 1 n value|[selected header in pheno]|[group 1 in test] - The number of value used in the test for group 1 quantile|[selected header in pheno]|[group 1 in test] - The quantile of means values in group 1 mean|[selected header in pheno]|[group 2 in test] - The mean value of group 2 n value|[selected header in pheno]|[group 2 in test] - The number of value used in the test for group 2 quantile|[selected header in pheno]|[group

2 in test] - The quantile of means values in group 2 ttest|[group 1 in test]_vs_[group 2 in test]|pvalue - The p-value return by t.test ttest|[group 1 in test]_vs_[group 2 in test]|log.pvalue - The -log10 transformed p-value ttest|[group 1 in test]_vs_[group 2 in test]|fdr - The BH method corrected p-values, e.g. FDR ttest|[group 1 in test]_vs_[group 2 in test]|log.fdr - The -log10 transformed FDR ttest|[group 1 in test]_vs_[group 2 in test]|mean.diff - The difference between the means of the two groups, e.g. fold change

Examples

```
# reading expression
packdir <- system.file("extdata", package = "omicsViewer")
expr <- read.delim(file.path(packdir, "expressionMatrix.tsv"), stringsAsFactors = FALSE)
# reading phenotype data
pd <- read.delim(file.path(packdir, "sampleGeneral.tsv"), stringsAsFactors = FALSE)

## Single t-test
head(pd)
# define comparisons
tests <- c("Origin", "RE", "ME")
tres <- multi.t.test(x = expr, pheno = pd, compare = tests)

## multiple t-test
head(pd)
# define comparisons
tests <- rbind(
c("Origin", "RE", "ME"),
c("Origin", "RE", "LE"),
c('TP53.Status', "MT", "WT")
)
tres <- multi.t.test(x = expr, pheno = pd, compare = tests)
```

---

nColors *Generating k distinct colors*

Description

Mainly used in the shiny app to generate reproducible k distinct colors.

Usage

```
nColors(k, stop = FALSE)
```

Arguments

k	a number between 1 to 60 tells how many distinct colors to use
stop	logical; whether the function should return an error message if k is not in the range of 2 to 60. Default FALSE, the function will return NULL.

Value

a vector of hex code for k colors or NULL

Examples

```
nColors(5)
nColors(1, stop = FALSE)
```

`normalize.nQuantiles` *Normalization using n quantiles*

Description

Normalization using n quantiles

Usage

```
normalize.nQuantiles(x, probs = 0.5, shareFeature = FALSE, ref = 1)
```

Arguments

`x`	an expression matrix, usually log transformed
`probs`	the quantiles to be aligned across samples. If `probs` is a length 1 numerical vector, the quantiles will aligned. As a special case, `probs = 0.5` equals the median centering. If `probs`' length is > 1, a shift and scaling factor of samples will be calculating by fitting linear models using quantiles of samples, the median and variance of samples will be corrected using the intersect and slope of the fitted model.
`shareFeature`	logocal; if TRUE, the normalization will be based on the shared features between samples
`ref`	the columns name or index to specify the reference sample, only used when `shareFeature = TRUE`

Value

a normalized matrix

Examples

```
e1 <- matrix(rnorm(5000), 500, 10)
e1[, 6:10] <- 0.3 *e1[, 6:10] + 3
boxplot(e1)
# median centering, no variance correction
e2 <- normalize.nQuantiles(x = e1, probs = 0.5)
boxplot(e2)
# median centering + variance stablization
e3 <- normalize.nQuantiles(x = e1, probs = seq(0.25, 0.75, by = 0.1))
boxplot(e3)
```

normalize.totsum *Normalize total sum*

Description

Normalize total sum

Usage

```
normalize.totsum(x)
```

Arguments

x	a log10 transformed expression matrix

Value

a normalized matrix

Examples

```
e1 <- matrix(rnorm(5000), 500, 10)
e1[, 6:10] <- e1[, 6:10]+3
boxplot(e1)
e2 <- normalize.totsum(x = e1)
boxplot(e2)
```

normalizeColWise *Column-wise normalization of expression matrix*

Description

A wrapper function of all column-wise normalization methods

Usage

```
normalizeColWise(
  x,
  method = c("Median centering", "Median centering (shared ID)", "Total sum",
    "median centering + variance stablization")[1]
)
```

Arguments

x	an expression matrix where rows are features and columns are samples, usually log transformed.
method	normalization method to use "Median centering" - median centering, see normalize.nQuantiles "Median centering (shared ID)" - median centering using shared features, see normalize.nQuantiles "Total sum" - total sum normalization "median centering + variance stablization" - 10 quantile normalization using 0.25, 0.3, ..., 0.75, see normalize.nQuantiles

Value

a normalized matrix

Examples

```
e1 <- matrix(rnorm(5000), 100, 50)+10
boxplot(e1)
e2 <- normalizeColWise(x = e1, method = "Median centering")
boxplot(e2)
```

`normalizeData` *Normalized expression matrix*

Description

A wrapper function of all normalization methods, including row-wise or column-wise normalization.

Usage

```
normalizeData(
  x,
 colWise = c("None", "Median centering", "Median centering (shared ID)", "Total sum",
    "median centering + variance stablization")[1],
  rowWise = c("None", "Reference", "Batch mean", "Batch reference")[1],
  ref = NULL,
  batch = NULL
)
```

Arguments

`x`	an expression matrix where rows are features and columns are samples, usually log transformed.
`colWise`	column-wise normalization method to use, see `normalizeColWise`
`rowWise`	row-wise normalization method to used `Reference` - using `removeVarQC` method `Batch mean` - using `rowshift` method without reference samples `Batch reference` - using `rowshift` method with reference samples
`ref`	index of reference samples
`batch`	batch factor

Value

a normalized matrix

Examples

```
e1 <- matrix(rnorm(5000), 100, 50)+10
boxplot(e1)
e2 <- normalizeData(x = e1, ref = seq(5, 45, by = 10), rowWise = "Reference")
boxplot(e2)
```

`omicsViewer` *Launch the omicsViewer Shiny Application*

Description

Starts an interactive Shiny application for exploring omics data, including visualization of expression matrices, feature and sample metadata, statistical analyses, and functional enrichment results. The viewer supports both `ExpressionSet` and `SummarizedExperiment` objects.

Usage

```
omicsViewer(
  dir,
  additionalTabs = NULL,
  filePattern = ".(RDS|DB|SQLITE|SQLITE3)$",
  ESVObj = NULL,
  esetLoader = readESVObj,
  exprsGetter = getExprs,
  pDataGetter = getPData,
  fDataGetter = getFData,
  defaultAxisGetter = getAx,
  appName = "omicsViewer",
  appVersion = packageVersion("omicsViewer")
)
```

Arguments

`dir`	Character. Path to directory containing the `ExpressionSet` or `SummarizedExperiment` object saved as .RDS file. Provide only the directory path, not the full file path. The viewer will list all compatible files in this directory.
`additionalTabs`	List. Optional custom Shiny modules to add as tabs in the "Analyst" panel. Each element should be a list with: `tabName` (character), `moduleName` (character), `moduleUi` (UI function), and `moduleServer` (server function).
`filePattern`	Character. Regular expression pattern to filter files displayed in the directory. Default: `".(RDS|DB|SQLITE|SQLITE3)$"` (case-insensitive).
`ESVObj`	ExpressionSet or SummarizedExperiment. Optional pre-loaded object to view directly without file selection. If provided, the file dropdown will show "ESVObj.RDS".
`esetLoader`	Function. Custom loader for reading saved objects. Default: `readESVObj`. Should accept a file path and return an ExpressionSet or SummarizedExperiment object.
`exprsGetter`	Function. Extracts expression matrix from loaded object. Default: `getExprs`. Should return a numeric matrix.
`pDataGetter`	Function. Extracts phenotype/sample metadata. Default: `getPData`. Should return a data.frame with rownames matching sample names.
`fDataGetter`	Function. Extracts feature metadata. Default: `getFData`. Should return a data.frame with rownames matching feature names.

`defaultAxisGetter`
: Function. Determines default axes for plots. Takes two arguments: `x` (the loaded object) and `what` (one of "sx", "sy", "fx", "fy" for sample/feature space x/y axes). Should return column name from metadata.

`appName`
: Character. Application title displayed in the UI. Default: "omicsViewer".

`appVersion`
: Character or package_version. Version number displayed in UI. Default: current package version.

Value

NULL (invisibly). Launches the Shiny application. The app runs until stopped by the user.

See Also

`prepOmicsViewer` for preparing data objects for visualization. `app_module` for the main application module (developers only).

Examples

```
if (interactive()) {
  # Basic usage with example data
  omicsViewer(system.file("extdata", package = "omicsViewer"))

  # With pre-loaded object
  packdir <- system.file("extdata", package = "omicsViewer")
  eset <- readRDS(file.path(packdir, "exampleEset.RDS"))
  omicsViewer(packdir, ESVObj = eset)
}
```

---

parseDatTerm *Extract function annotation from uniprot .dat file*

Description

Extract function annotation from uniprot .dat file

Usage

```
parseDatTerm(file, outputDir = NULL, ...)
```

Arguments

`file`
: the .dat or .dat.gz file

`outputDir`
: dir of output file

`...`
: other parameters passed to readLines

Value

a data.frame parse from .dat file

plotDC *Draw dose-response curves*

Description

Draw dose-response curves

Usage

```
plotDC(mod, ylab = "Abundance", lty = 2, pch = 19, cex = 1, logx = FALSE)
```

Arguments

mod	an drc object
ylab	ylab in plot function
lty	lty in plot function
pch	pch in plot function
cex	cex in plot function
logx	whether the x-axis should be in log scale

plotDCMat *Draw dose response curve given parameters in the omicsViewer object*

Description

Draw dose response curve given the feature Data/rowData, phenotype data/colData and expression matrix. The function is usually used in shinyApp.

Usage

```
plotDCMat(
  expr,
  pd,
  fd,
  featid,
  dose.var,
  curve.var = NULL,
  only.par = FALSE,
  ...
)
```

Arguments

expr	expression matrix
pd	phenotype data or colData
fd	feature data or rowData
featid	feature id to be visualized
dose.var	the column header indicating the dose/time/concentration
curve.var	the column header indicating the curve ids
only.par	logical value. If true, no plot generated, the function only returns the parameters of models.
...	other parameters passed to plot function, except col, pch, xlab, ylab

---

plotly_boxplot_module *Shiny module for boxplot using plotly - Module*

Description

Shiny module for boxplot using plotly - Module

Usage

```
plotly_boxplot_module(
  id,
  reactive_param_plotly_boxplot,
  reactive_checkpoint = reactive(TRUE)
)
```

Arguments

id	module id
reactive_param_plotly_boxplot	reactive value; argument passed to plotly_boxplot
reactive_checkpoint	reactive_value; check this value before render any plot/executing any calculation

Value

do not return any values

Examples

```
if (interactive()) {

  library(shiny)

  ui <- fluidPage(
    plotly_boxplot_ui("testplotly")
  )

  server <- function(input, output, session) {
```

```
  x <- cbind(matrix(rnorm(10000, mean = 3), 1000, 10), matrix(rnorm(20000), 1000, 20))
  x[sample(1:length(x), size = 0.3*length(x))] <- NA
  rownames(x) <- paste("R", 1:nrow(x), sep = "")
  colnames(x) <- paste("C", 1:ncol(x), sep = "")
  plotly_boxplot_module("testplotly",
             reactive_param_plotly_boxplot = reactive(list(
               x = x# , i  = c(4, 20, 80)# , highlight = c(1, 4, 5, 20), extvar = 1:30
             ))
  )
 }

 shinyApp(ui, server)
}
```

`plotly_boxplot_ui` *Shiny module for boxplot using plotly - UI*

Description

Function should only be used for the developers

Usage

```
plotly_boxplot_ui(id)
```

Arguments

id	id

Value

a tagList of UI components

a tagList of UI components

Examples

```
if (interactive()) {

 library(shiny)

 ui <- fluidPage(
  plotly_boxplot_ui("testplotly")
 )

 server <- function(input, output, session) {

  x <- cbind(matrix(rnorm(10000, mean = 3), 1000, 10), matrix(rnorm(20000), 1000, 20))
  x[sample(1:length(x), size = 0.3*length(x))] <- NA
  rownames(x) <- paste("R", 1:nrow(x), sep = "")
  colnames(x) <- paste("C", 1:ncol(x), sep = "")
  plotly_boxplot_module("testplotly",
             reactive_param_plotly_boxplot = reactive(list(
               x = x# , i  = c(4, 20, 80)# , highlight = c(1, 4, 5, 20), extvar = 1:30
```

```
                ))
    )
  }

  shinyApp(ui, server)
}
```

`plotly_scatter_module` *Shiny module for scatter plot using plotly - Module*

Description

Function should only be used for the developers

Usage

```
plotly_scatter_module(
  id,
  reactive_param_plotly_scatter,
  reactive_regLine = reactive(FALSE),
  reactive_checkpoint = reactive(TRUE),
  htest_var1 = reactive(NULL),
  htest_var2 = reactive(NULL)
)
```

Arguments

`id`	module id
`reactive_param_plotly_scatter`	reactive parammeters for plotly_scatter
`reactive_regLine`	logical show or hide the regression line
`reactive_checkpoint`	checkpoint
`htest_var1`	when the plot is a beeswarmplot, two groups could be selected for two group comparison, this argument gives the default value. Mainly used for restoring the saved session.
`htest_var2`	see above

Value

a list containing the information about the selected data points

an reactive object containing the information of selected, brushed points.

Examples

```
if (interactive()) {
  library(shiny)

  # two random variables
  x <- rnorm(30)
  y <- x + rnorm(30, sd = 0.5)

  # variables mapped to color, shape and size
  cc <- sample(letters[1:4], replace = TRUE, size = 30)
  shape <- sample(c("S1", "S2", "S3"), replace = TRUE, size = 30)
  sz <- sample(c(10, 20, 30, replace = TRUE, size = 30))

  ui <- fluidPage(
    plotly_scatter_ui("test_scatter")
  )

  server <- function(input, output, session) {
    v <- plotly_scatter_module("test_scatter",
                    # reactive_checkpoint = reactive(FALSE),
                    reactive_param_plotly_scatter = reactive(list(
                      x = x, y = y,
                      color = cc,
                      shape = shape,
                      size = sz,
                      tooltips = paste("A", 1:30)
                    )))
    observe(print(v()))
  }
  shinyApp(ui, server)



  # example beeswarm horizontal
  x <- rnorm(30)
  y <- sample(c("x", "y", "z"), size = 30, replace = TRUE)
  shinyApp(ui, server)

  # example beeswarm vertical
  x <- sample(c("x", "y", "z"), size = 30, replace = TRUE)
  y <- rnorm(30)
  shinyApp(ui, server)

  # return values
  x <- c(5, 6, 3, 4, 1, 2)
  y <- c(5, 6, 3, 4, 1, 2)
  ui <- fluidPage(
    plotly_scatter_ui("test_scatter")
  )
  server <- function(input, output, session) {
    v <- plotly_scatter_module("test_scatter",
                    reactive_param_plotly_scatter = reactive(list(
                      x = x, y = y, tooltips = paste("A", 1:6), highlight = 2:4
                    )))

    observe(print(v()))
```

```
  }
  shinyApp(ui, server)
}
```

`plotly_scatter_ui` *Shiny module for scatter plot using plotly - UI*

Description

Function should only be used for the developers

Usage

```
plotly_scatter_ui(id, height = "400px")
```

Arguments

`id`	id
`height`	figure height

Value

a tagList of UI components

Examples

```
if (interactive()) {
  library(shiny)

  # two random variables
  x <- rnorm(30)
  y <- x + rnorm(30, sd = 0.5)

  # variables mapped to color, shape and size
  cc <- sample(letters[1:4], replace = TRUE, size = 30)
  shape <- sample(c("S1", "S2", "S3"), replace = TRUE, size = 30)
  sz <- sample(c(10, 20, 30, replace = TRUE, size = 30))

  ui <- fluidPage(
    plotly_scatter_ui("test_scatter")
  )

  server <- function(input, output, session) {
    v <- callModule(plotly_scatter_module, id = "test_scatter",
                    # reactive_checkpoint = reactive(FALSE),
                    reactive_param_plotly_scatter = reactive(list(
                      x = x, y = y,
                      color = cc,
                      shape = shape,
                      size = sz,
                      tooltips = paste("A", 1:30)
                    )))
    observe(print(v()))
  }
```

```
  shinyApp(ui, server)



  # example beeswarm horizontal
  x <- rnorm(30)
  y <- sample(c("x", "y", "z"), size = 30, replace = TRUE)
  shinyApp(ui, server)

  # example beeswarm vertical
  x <- sample(c("x", "y", "z"), size = 30, replace = TRUE)
  y <- rnorm(30)
  shinyApp(ui, server)

  # return values
  x <- c(5, 6, 3, 4, 1, 2)
  y <- c(5, 6, 3, 4, 1, 2)
  ui <- fluidPage(
    plotly_scatter_ui("test_scatter")
  )
  server <- function(input, output, session) {
    v <- callModule(plotly_scatter_module, id = "test_scatter",
                    reactive_param_plotly_scatter = reactive(list(
                      x = x, y = y, tooltips = paste("A", 1:6), highlight = 2:4
                    )))

    observe(print(v()))
  }
  shinyApp(ui, server)
}
```

`plot_roc_pr_module` *Shiny module for ROC/PR plot - Module*

Description

Shiny module for ROC/PR plot - Module

Usage

```
plot_roc_pr_module(id, reactive_param, reactive_checkpoint = reactive(TRUE))
```

Arguments

`id`	module id
`reactive_param`	reactive value; argument pass to draw_roc_pr
`reactive_checkpoint`	reactive_value; check this value before render any plot/executing any calculation

Value

do not return any values

Examples

```
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    sliderInput("ngrp", label = "Number of groups", min = 2, max = 5, value = 2),
    plot_roc_pr_ui("testplot")
  )

  server <- function(input, output, session) {
    ng <- reactive(
      sample(letters[1:input$ngrp], size = 100, replace = TRUE)
    )
    plot_roc_pr_module("testplot",
      reactive_param = reactive(list(
        x = ng(),
        y = rnorm(100)
      ))
    )
  }
 shinyApp(ui, server)
}
```

`prepOmicsViewer` *Prepare Omics Data for Visualization with omicsViewer*

Description

A comprehensive data preparation function that processes expression matrices and associated metadata for interactive visualization with `omicsViewer`. Automatically performs dimensionality reduction (PCA), statistical testing (t-tests), and integrates gene set annotations, STRING database IDs, and survival data.

Usage

```
prepOmicsViewer(
  expr,
  pData,
  fData,
  PCA = TRUE,
  ncomp = min(8, ncol(expr)),
  pca.fillNA = TRUE,
  t.test = NULL,
  ttest.fillNA = FALSE,
  ...,
  gs = NULL,
  stringDB = NULL,
  surv = NULL,
  SummarizedExperiment = TRUE
)
```

Arguments

expr	Numeric matrix. Expression data with features in rows and samples in columns. Should be log-transformed (e.g., log2 or log10). Row and column names must be unique. Missing values (NA) are permitted if pca.fillNA or ttest.fillNA are TRUE.
pData	Data.frame. Sample/phenotype metadata with one row per sample. Row names must match column names of expr. Should contain grouping variables for statistical tests.
fData	Data.frame. Feature metadata with one row per feature. Row names must match row names of expr. Can include gene symbols, descriptions, database IDs, etc.
PCA	Logical. Whether to perform Principal Component Analysis. Default: TRUE. Results are added to both sample and feature metadata.
ncomp	Integer. Number of principal components to compute. Default: minimum of 8 or the number of samples. Ignored if PCA = FALSE.
pca.fillNA	Logical. If TRUE, missing values in expr are imputed before PCA by replacing with minimum value * 0.9. Default: TRUE. Two PCAs are performed: one with imputation and one without (if possible).
t.test	Matrix or NULL. Definition of t-tests to perform. Should be an n×3 matrix where each row specifies: [column_name, group1, group2]. The column should exist in pData. Example: rbind(c("Treatment", "Drug", "Control"), c("Genotype", "WT", "KO")). Results are added as columns to fData. NULL = no t-tests.
ttest.fillNA	Logical. Whether to impute missing values before t-tests. Default: FALSE (features with NAs are excluded from testing).
...	Additional arguments passed to t.test, such as paired = TRUE for paired t-tests or var.equal = TRUE for equal variance assumption.
gs	Gene set annotations in one of two formats: • Data.frame with columns: featureId (indices), gsId (gene set IDs), weight (optional weights). See gsAnnotIdList. • Matrix or sparse matrix (dgCMatrix) with features in rows and gene sets in columns. Values indicate membership (0/1 or weights). NULL = no gene set annotations. Enables ORA and GSEA analyses in viewer.
stringDB	Character vector of length nrow(expr). Protein/gene identifiers compatible with STRING database queries (e.g., Ensembl protein IDs, gene names). NULL = STRING network analysis disabled.
surv	Survival data in one of three formats: • Vector of length ncol(expr): single survival time with censoring indicated by "+" suffix (e.g., "120+", "45"). • Matrix/data.frame: multiple survival endpoints with samples in rows. Column names will be prefixed with "Surv|all|". Values must be numeric with optional "+" suffix. NULL = no survival analysis.
SummarizedExperiment	Logical. If TRUE, returns a SummarizedExperiment object; if FALSE, returns an ExpressionSet. Default: TRUE.

Details

The function performs the following processing steps:

1. Validates dimensions and ensures unique row/column names
2. Standardizes column names by prefixing with data type (e.g., "General|All|")
3. Performs PCA on expression data (with and without imputation)
4. Conducts statistical tests (t-tests) between specified groups
5. Computes feature rankings across samples
6. Integrates gene set, STRING, and survival annotations
7. Sets sensible default axes for visualization

All metadata columns are prefixed with standardized headers following the pattern "Category|Subcategory|Variable" to organize variables in the viewer interface.

Value

A `SummarizedExperiment` or `ExpressionSet` object ready for visualization with `omicsViewer`.
The object includes:

- Expression matrix (and optionally imputed matrix)
- Enhanced metadata with PCA results, t-test statistics, rankings
- Gene set annotations (as attributes)
- Default axis selections (as attributes: "sx", "sy", "fx", "fy")

an object of `ExpressionSet` or `SummarizedExperiment` that can be visualized using `omicsViewer`

See Also

`omicsViewer` for launching the viewer. `multi.t.test` for details on t-test implementation. `gsAnnotIdList` for gene set annotation formatting.

Examples

```
packdir <- system.file("extdata", package = "omicsViewer")
# reading expression
expr <- read.delim(file.path(packdir, "expressionMatrix.tsv"), stringsAsFactors = FALSE)
colnames(expr) <- make.names(colnames(expr))
rownames(expr) <- make.names(rownames(expr))
# reading feature data
fd <- read.delim(file.path(packdir, "featureGeneral.tsv"), stringsAsFactors = FALSE)
# reading phenotype data
pd <- read.delim(file.path(packdir, "sampleGeneral.tsv"), stringsAsFactors = FALSE)

#  reading other datasets
drugData <- read.delim(file.path(packdir, "sampleDrug.tsv"))
# survival data
# this data is from cell line, the survival data are fake data to
# show how to use the survival data in #' omicsViewer
surv <- read.delim(file.path(packdir, "sampleSurv.tsv"))
# gene set information
genesets <- read_gmt(file.path(packdir, "geneset.gmt"), data.frame = TRUE)
gsannot <- gsAnnotIdList(idList = rownames(fd), gsIdMap = genesets, data.frame = TRUE)
```

```
# Define t-test to be done, a matrix nx3
# every row define a t-test, the format
# [column header] [group 1 in the test] [group 2 in the test]
tests <- rbind(
 c("Origin", "RE", "ME"),
 c("Origin", "RE", "LE"),
 c('TP53.Status', "MT", "WT")
 )
# prepare column for stringDB query
strid <- sapply(strsplit(fd$Protein.ID, ";|-"), "[", 1)
###
d <- prepOmicsViewer(
  expr = expr, pData = pd, fData = fd,
  PCA = TRUE, pca.fillNA = TRUE,
  t.test = tests, ttest.fillNA = FALSE,
  gs = gsannot, stringDB = strid, surv = surv)
# feature space - default x axis
attr(d, "fx") <- "ttest|RE_vs_ME|mean.diff"
# feature space - default y axis
attr(d, "fy") <- "ttest|RE_vs_ME|log.fdr"
# sample space - default x axis
attr(d, "sx") <- "PCA|All|PC1("
# sample space - default y axis
attr(d, "sy") <- "PCA|All|PC2("
# Save object and view
# saveRDS(d, file = "dtest.RDS")
##  to open the viewer
# omicsViewer("./")
```

`read.proteinGroups` *Reading proteinGroup table of MaxQuant output*

Description

A convenience function to read the proteinGroups table of MaxQuant output. The function organize the result into different tables, e.g. iBAQ.

Usage

```
read.proteinGroups(x, quant = c("LF", "TMT")[1])
```

Arguments

`x`	the proteinGroup.txt file returned by MaxQuant search
`quant`	the quantification method, LF or TMT

Value

a list of tables extracted from proteinGroups.txt file

read.proteinGroups.lf *Read protein groups output of maxquant output and split it to columns*

Description

Read protein groups output of maxquant output and split it to columns

Usage

```
read.proteinGroups.lf(file)
```

Arguments

file	Maxquant proteinGroup.txt file path

Value

a list of tables extracted from proteinGroups.txt file

readESVObj *Read the object of SummarizedExperiment or ExpressetSet to be visualized using omicsViewer*

Description

This function accept a path to a sqlite database or RDS object. If an RDS file to be read, The function is similar to readRDS. It reads the object to R working environment and perform extra two things. 1. If the loaded data an class of SummarizedExperiment, it will be converted to ExpressionSet; 2. If the gene set annotatio is in matrix format, the gene set annotation is converted to data.frame format.

Usage

```
readESVObj(x)
```

Arguments

x	the path of an object of SummarizedExperiment or ExpressionSet, passed to readRDS

Value

an object of class ExpressionSet or SummarizedExperiment to be visualzied.

Examples

```
file <- system.file("extdata/demo.RDS", package = "omicsViewer")
obj <- readESVObj(file)
```

read_gmt *Reading gene set .gmt file*

Description

Frequently the .gmt files are downloaed from MSigDB database

Usage

```
read_gmt(x, id = NA, data.frame = FALSE)
```

Arguments

x	the name/path of the gmt file to be read
id	the id used in gene sets, if is not NA, it should be either "SYMBOL" or "ENTREZ". Usually only used when reading the .gmt file downloaded from MSigDB.
data.frame	logical; whether to organize the data in data.frame format. Default is FALSE, a list will be returned.

Value

a list or data frame of gene set. When data.frame = TRUE, the returned object is a data.frame with two columns: id and term.

Examples

```
file <- system.file("extdata", package = "omicsViewer")
file <- file.path(file, "geneset.gmt")
gs <- read_gmt(file)
```

removeVarQC *Removing variance of reference samples*

Description

This normalization removes the variance in reference samples. The method do not need to specific the batch assignment but cannot work with data contains less than five common reference samples. A typical use of this normalization is to correct some drifting effect in mass spec based label free proteomics or untargeted metabolomics experiment. Usually, this is a very strong normalization should only be used with good reasons.

Usage

```
removeVarQC(x, ref, positive = TRUE, ...)
```

Arguments

x	an expression matrix
ref	the index of reference samples
positive	logical; force only positive values in the resulted matrix
...	if given, normalize.nQuantiles will be called first, the arguments here will be passed to normalize.nQuantiles

Value

a normalized matrix

Examples

```
e1 <- matrix(rnorm(5000), 100, 50)+10
e2 <- removeVarQC(x = e1, ref = seq(5, 45, by = 10))
boxplot(e2)
```

---

rowshift — *Row-wise normalization of expression matrix with or without reference sample*

Description

Row-wise normalization of expression matrix with or without reference sample

Usage

```
rowshift(x, batch, ref = NULL, useMean = FALSE)
```

Arguments

x	an expression matrix where rows are features, e.g. genes, proteins and columns are samples. The values in the matrix are usually log transformed.
batch	a factor or vector has the same length as ncol(x) to indicate the batch assignment of samples.
ref	a logical vector has the same length as ncol(x) to indicated which columns are the common references among batches. If it is NULL (by default), the mean of all channels will be used as batch reference. When NA present in the reference channels, the mean values will be used in correction.
useMean	logical; whether to use means of batches, usually set to TRUE when no reference available

Value

a matrix (hopefully without/with less batch effect)

Examples

```
e1 <- matrix(rnorm(5000), 500, 10)
e1[, 6:10] <- e1[, 6:10] + 3
boxplot(e1)
f <- rep(c("a", "b"), each = 5)
e2 <- rowshift(x = e1, batch = f)
boxplot(e2)
```

---

safe_GET *Safe HTTP GET wrapper with error handling*

Description

Wraps httr::GET with comprehensive error handling including: - Network connectivity errors - HTTP status code errors (4xx, 5xx) - Timeout handling - SSL/TLS errors

Usage

```
safe_GET(url, query = NULL, timeout = 30, api_name = "API", ...)
```

Arguments

url	The URL to fetch
query	Query parameters (optional)
timeout	Timeout in seconds (default: 30)
api_name	Name of the API for error messages (default: "API")
...	Additional parameters passed to httr::GET

Value

A list with: - success: logical, TRUE if request succeeded - data: response content if successful, NULL otherwise - error: error message if failed, NULL otherwise - status_code: HTTP status code if available

---

safe_POST *Safe HTTP POST wrapper with error handling*

Description

Wraps httr::POST with comprehensive error handling including: - Network connectivity errors - HTTP status code errors (4xx, 5xx) - Timeout handling - SSL/TLS errors

Usage

```
safe_POST(
  url,
  body = NULL,
  encode = "json",
  timeout = 30,
  api_name = "API",
  ...
)
```

Arguments

url	The URL to post to
body	Request body
encode	Encoding method (default: "json")
timeout	Timeout in seconds (default: 30)
api_name	Name of the API for error messages (default: "API")
...	Additional parameters passed to httr::POST

Value

A list with: - success: logical, TRUE if request succeeded - data: response content if successful, NULL otherwise - error: error message if failed, NULL otherwise - status_code: HTTP status code if available

---

saveOmicsViewerDb *Save the xcmsViewer result object as sqlite database*

Description

Save the xcmsViewer result object as sqlite database

Usage

```
saveOmicsViewerDb(obj, db.file, overwrite = TRUE)

## S4 method for signature 'SummarizedExperiment,character'
saveOmicsViewerDb(obj, db.file, overwrite = TRUE)

## S4 method for signature 'ExpressionSet,character'
saveOmicsViewerDb(obj, db.file, overwrite = TRUE)
```

Arguments

obj	an object of class ExpressionSet or SummarizedExperiment
db.file	a character indicate file name of the database file
overwrite	logical. whether the database should be overwritten if exist already.

Value

the directory where the database saved

Examples

```
f <- system.file("extdata", "demo.RDS", package = "omicsViewer")
es <- readRDS(f)
# The following line will write a database file on your disk
# saveOmicsViewerDb(es, db.file = "./omicsViewerData.db")
```

stringD3Net *Drawing network given network and gene set result*

Description

Drawing network given network and gene set result

Usage

```
stringD3Net(ntwk, gsa, i, label = FALSE)
```

Arguments

ntwk	network result, often returned by "stringNetwork" function.
gsa	gene set result, often returned by "stringGSA" function
i	row index of gsa, which should be highlighted in the network
label	whether the point should be labelled in the network

stringGSA *Performing gene set analysis using string-db*

Description

Performing gene set analysis using string-db

Usage

```
stringGSA(
  genes,
  taxid = 9606,
  background = NULL,
  backgroundStringId = FALSE,
  caller = "omicsViewer"
)
```

Arguments

genes	a character vector of gene ids
taxid	taxonomy ids (species NCBI identifier)
background	the background genes
backgroundStringId	logical value, whether the identifier given in background is already stringid, only used when background != NULL. You can use getStringId to convet your id to string id.
caller	your identifier for string-db.org

Note

https://string-db.org/cgi/help.pl?subpage=api

Examples

```
1
# gg = c('P04637', 'P00533', 'P04626', "Q8IYB3", "O75494", "Q9Y696")
# u <- stringGSA(gg)
```

`stringNetwork` *Retrieve string network*

Description

get string network

Usage

```
stringNetwork(genes, taxid = 9606, caller = "omicsViewer")
```

Arguments

`genes`	the gene ids
`taxid`	taxonomy ids
`caller`	your identifier for string-db.org

`string_module` *STRING Network Analysis Server Function*

Description

Server logic for the STRING protein-protein interaction network analysis module. Queries the STRING database API to retrieve protein interactions and perform functional enrichment analysis on selected features.

Usage

```
string_module(
  id,
  reactive_ids,
  reactive_status = reactive(NULL),
  active = reactive(FALSE)
)
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in string_ui.
reactive_ids	Reactive expression. Returns a character vector of protein/gene identifiers to query. Maximum of 300 features allowed (enforced by STRING API limits).
reactive_status	Reactive expression. Returns a list containing saved state for session restoration (tax ID, label settings). Optional. Default: reactive(NULL).
active	Reactive expression. Returns a logical indicating whether the module should auto-run on initialization. Used for session restoration. Default: reactive(FALSE).

Details

API Interaction The module makes two separate API calls to STRING database:

1. Network query: Retrieves protein-protein interactions
2. Enrichment query: Performs gene set enrichment analysis

Both queries are rate-limited and may fail with network errors. The module handles errors gracefully with user notifications.

Feature Limits - Maximum 300 genes (STRING_MAX_GENES constant) - Maximum 999 network edges displayed (STRING_MAX_NETWORK_EDGES constant)

Value

A reactive expression returning a list with:

- tax: Taxonomy ID used
- showLabel: Logical, whether labels are shown in network

See Also

string_ui for the corresponding UI function. stringNetwork for network retrieval. stringGSA for enrichment analysis. stringD3Net for network visualization.

Other network modules: string_ui()

---

string_ui *STRING Network Analysis UI Function*

Description

Creates the user interface for the STRING protein-protein interaction network analysis module. Displays network visualization, enrichment results, and taxonomy configuration.

Usage

```
string_ui(id)
```

Arguments

id	Character. Namespace ID for the Shiny module. Must match the ID used in string_module.

Value

A tagList containing:

- Taxonomy code input
- Feature count display with max limit warning
- Action button to trigger analysis
- Enrichment results table with download
- Interactive network visualization with label toggle

See Also

string_module for the corresponding server logic. stringNetwork for network API calls. stringGSA for enrichment analysis.

Other network modules: string_module()

---

triselector_module *The three-step selector - the module function*

Description

The selector is used to select columns of phenotype and feature data. Function should only be used for the developers.

Usage

```
triselector_module(
  id,
  reactive_x,
  reactive_selector1 = reactive(NULL),
  reactive_selector2 = reactive(NULL),
  reactive_selector3 = reactive(NULL),
  label = "Group Label:"
)
```

Arguments

id	module id
reactive_x	an nx3 matrix
reactive_selector1	default value for selector 1
reactive_selector2	default value for selector 2
reactive_selector3	default value for selector 3
label	of the triselector

Value

an reactive object containing the selected values

Examples

```
if (interactive()) {
  library(shiny)
  library(Biobase)

  file <- system.file("extdata/demo.RDS", package = "omicsViewer")
  dat <- readRDS(file)
  fData <- fData(dat)
  triset <- stringr::str_split_fixed(colnames(fData), '\\|', n= 3)

  ui <- fluidPage(
    triselector_ui("tres"),
    triselector_ui("tres2")
  )
  server <- function(input, output, session) {
    v1 <- triselector_module("tres", reactive_x = reactive(triset),
                    reactive_selector1 = reactive("ttest"),
                    reactive_selector2 = reactive("RE_vs_ME"),
                    reactive_selector3 = reactive("mean.diff")
    )
    v2 <- triselector_module("tres2", reactive_x = reactive(triset),
                    reactive_selector1 = reactive("ttest"),
                    reactive_selector2 = reactive("RE_vs_ME"),
                    reactive_selector3 = reactive("log.fdr"))
    observe({
      print("//////////////////////////")
      print(v1())
    })
  }

  shinyApp(ui, server)
}
```

---

triselector_ui *The three-step selector - the ui function*

Description

Function should only be used for the developers

Usage

```
triselector_ui(id, right_margin = "20")
```

Arguments

id	id
right_margin	margin on the right side, in px. For example, "20" translates to "20px".

Value

a tagList of UI components

Examples

```
if (interactive()) {
  library(shiny)
  library(Biobase)

  file <- system.file("extdata/demo.RDS", package = "omicsViewer")
  dat <- readRDS(file)
  fData <- fData(dat)
  triset <- stringr::str_split_fixed(colnames(fData), '\\|', n= 3)

  ui <- fluidPage(
    triselector_ui("tres"),
    triselector_ui("tres2")
  )
  server <- function(input, output, session) {
    v1 <- triselector_module("tres", reactive_x = reactive(triset),
                    reactive_selector1 = reactive("ttest"),
                    reactive_selector2 = reactive("RE_vs_ME"),
                    reactive_selector3 = reactive("mean.diff")
    )
    v2 <- triselector_module("tres2", reactive_x = reactive(triset),
                    reactive_selector1 = reactive("ttest"),
                    reactive_selector2 = reactive("RE_vs_ME"),
                    reactive_selector3 = reactive("log.fdr"))
    observe({
      print("//////////////////////////")
      print(v1())
    })
  }

  shinyApp(ui, server)
}
```

`trisetter` *Create a nx3 matrix that can be use for triselector given a meta and expression table*

Description

only used inside reactive

Usage

```
trisetter(meta, expr = NULL, combine)
```

Arguments

`meta`	a meta data, usually either phenotype data or feature data
`expr`	expression matrix, optional.

combine	how the meta and expression to be combined. Should be either "pheno" or "feature' or "none".

Value

a nx3 matrix

a data.frame with 3 columns

---

validate_api_response *Validate API response data*

Description

Validates that API response data is in expected format and not empty.

Usage

```
validate_api_response(data, expected_type = "data.frame", api_name = "API")
```

Arguments

data	The response data to validate
expected_type	Expected type ("data.frame", "list", "character", etc.)
api_name	Name of the API for error messages

Value

A list with: - valid: logical, TRUE if data is valid - error: error message if invalid, NULL otherwise

---

validate_character_vector
Validate character vector input

Description

Validate character vector input

Usage

```
validate_character_vector(
  vec,
  name = "vector",
  required = TRUE,
  min_length = 1,
  max_length = Inf,
  allow_empty = FALSE
)
```

Arguments

vec	Reactive expression or vector to validate
name	Character. Name of the vector for error messages
required	Logical. Whether NULL values are allowed. Default: TRUE
min_length	Integer. Minimum length required. Default: 1
max_length	Integer. Maximum length allowed. Default: Inf
allow_empty	Logical. Whether empty strings are allowed. Default: FALSE

Value

List with $valid (logical) and $message (character) fields

---

validate_dataframe *Input Validation Utilities*

Description

Provides comprehensive validation functions for Shiny module inputs to ensure data integrity and provide user-friendly error messages.

Usage

```
validate_dataframe(
  data,
  name = "data",
  required = TRUE,
  min_rows = 1,
  min_cols = 1
)
```

Arguments

data	Reactive expression or data.frame to validate
name	Character. Name of the data for error messages
required	Logical. Whether NULL values are allowed. Default: TRUE
min_rows	Integer. Minimum number of rows required. Default: 1
min_cols	Integer. Minimum number of columns required. Default: 1

Value

List with $valid (logical) and $message (character) fields

Examples

```
## Not run:
validation <- validate_dataframe(
  pData(eset),
  name = "phenotype data",
  min_rows = 3,
  min_cols = 2
)
if (!validation$valid) {
  showNotification(validation$message, type = "error")
  return(NULL)
}

## End(Not run)
```

`validate_matching_dimensions` *Validate matching dimensions*

Description

Validate matching dimensions

Usage

```
validate_matching_dimensions(
  obj1,
  obj2,
  dimension = c("rows", "cols"),
  name1 = "object1",
  name2 = "object2"
)
```

Arguments

`obj1`	First object (matrix or data.frame)
`obj2`	Second object (matrix or data.frame)
`dimension`	Character. Either "rows" or "cols" to check
`name1`	Character. Name of first object for error messages
`name2`	Character. Name of second object for error messages

Value

List with $valid (logical) and $message (character) fields

validate_numeric_matrix *Validate numeric matrix input*

Description

Validate numeric matrix input

Usage

```
validate_numeric_matrix(
  mat,
  name = "matrix",
  required = TRUE,
  min_rows = 1,
  min_cols = 1,
  allow_na = TRUE
)
```

Arguments

mat	Reactive expression or matrix to validate
name	Character. Name of the matrix for error messages
required	Logical. Whether NULL values are allowed. Default: TRUE
min_rows	Integer. Minimum number of rows required. Default: 1
min_cols	Integer. Minimum number of columns required. Default: 1
allow_na	Logical. Whether NA values are allowed. Default: TRUE

Value

List with $valid (logical) and $message (character) fields

---

validate_numeric_range *Validate numeric range*

Description

Validate numeric range

Usage

```
validate_numeric_range(
  value,
  name = "value",
  min = -Inf,
  max = Inf,
  allow_na = FALSE
)
```

Arguments

value	Numeric value to validate
name	Character. Name for error messages
min	Numeric. Minimum allowed value (inclusive). Default: -Inf
max	Numeric. Maximum allowed value (inclusive). Default: Inf
allow_na	Logical. Whether NA is allowed. Default: FALSE

Value

List with $valid (logical) and $message (character) fields

---

validate_triselector_input

Validate triselector input matrix

Description

Validate triselector input matrix

Usage

```
validate_triselector_input(
  triset,
  name = "triselector data",
  allow_null = FALSE
)
```

Arguments

triset	Matrix or reactive expression. Should be nx3 matrix from str_split_fixed
name	Character. Name for error messages
allow_null	Logical. Whether NULL is allowed (e.g., during app startup). Default: FALSE

Value

List with $valid (logical) and $message (character) fields

validMQFolder *MQ folder validator Validate whether a folder is a MQ output folder*

Description

MQ folder validator Validate whether a folder is a MQ output folder

Usage

```
validMQFolder(dir)
```

Arguments

dir	the directory to check

Details

from the root level, these files exist: mqpar.xml [[combined/]txt/]proteinGroups.txt

Value

a list containing the info about MQ folder check

varSelector *variable selector*

Description

variable selector

Usage

```
varSelector(x, expr, meta, alternative = NULL)
```

Arguments

x	variable return by triselector, a list of length three named as "analysis", "subset" and "variable'
expr	the expression matrix
meta	a meta matrix
alternative	alternative value to be returned when nothing to select

Value

the selected values in input argument x

Index