Package 'debrowser'

November 12, 2026

Type Package

Title Interactive Differential Expresion Analysis Browser

Version 1.41.2

Date 2026-07-23

Description Bioinformatics platform containing interactive plots and tables for differential gene and region expression studies. Allows visualizing expression data much more deeply in an interactive and faster way. By changing the parameters, users can easily discover different parts of the data that like never have been done before. Manually creating and looking these plots takes time. With DEBrowser users can prepare plots without writing any code. Differential expression, PCA and clustering analysis are made on site and the results are shown in various plots such as scatter, bar, box, volcano, ma plots and Heatmaps.

Depends R (>= 4.2.0),

License GPL-3 + file LICENSE

LazyData true

Imports shiny, jsonlite, shinyjs, shinyBS, shinyWidgets, gplots, DT, ggplot2, annotate, AnnotationDbi, DESeq2, igraph, grDevices, graphics, stats, utils, GenomicRanges, IRanges, S4Vectors, SummarizedExperiment, stringi, reshape2, org.Hs.eg.db, limma, edgeR, clusterProfiler, methods, sva, RCurl, colourpicker, plotly, heatmaply, bslib (>= 0.7.0), htmltools

RoxygenNote 8.0.0

Encoding UTF-8

Suggests testthat (>= 3.2.0), rmarkdown, RSQLite, knitr, digest, shinymanager (>= 1.0.4), shinytest2, chromote, DBI, Harman, pathview, org.Mm.eg.db, apeglm, ashr, commonmark, enrichplot, fgsea, ipaddress (>= 1.0), mockery, msigdbr, openssl, scrypt, UpSetR, DOSE, ellmer (>= 0.1.0), keyring, whisker, withr, xml2

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/UMMS-Biocore/debrowser, https://debrowser.readthedocs.io

BugReports https://github.com/UMMS-Biocore/debrowser/issues/new

biocViews Sequencing, ChIPSeq, RNASeq, DifferentialExpression, GeneExpression, Clustering, ImmunoOncology

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

git_branch devel

git_last_commit f1190cd

git_last_commit_date 2026-07-23

Repository Bioconductor 3.24

Date/Publication 2026-11-12

Author Alper Kucukural [aut, cre], Onur Yukselen [aut], Manuel Garber [aut]

Maintainer Alper Kucukural <alper.kucukural@umassmed.edu>

Contents

accountDropdownServer . . . 7
accountDropdownUI . . . 8
actionButtonDE . . . 8
addDataCols . . . 9
addID . . . 10
aiInterpretServer . . . 10
aiInterpretUI . . . 11
aiSettingsServer . . . 12
aiSettingsUI . . . 12
ai_chat . . . 13
ai_interpret . . . 13
all2all . . . 14
all2allControlsUI . . . 15
applyFilters . . . 15
applyFiltersNew . . . 16
applyFiltersToMergedComparison . . . 16
apply_batch_correction . . . 17
apply_de_filters . . . 18
apply_merged_filters . . . 19
barMainPlotControlsUI . . . 19
batchEffectUI . . . 20
batchMethod . . . 20
BoxMainPlotControlsUI . . . 21
changeClusterOrder . . . 22
checkCountData . . . 22
checkMetaData . . . 23
clusterData . . . 24
clustFunParamsUI . . . 24
compareClust . . . 25
comparisonConcordanceServer . . . 26
comparisonConcordanceUI . . . 27
comparison_labels . . . 28
compute_pill_class . . . 28
concordance_sets . . . 29

concordance_summary . . . 29
condSelectServer . . . 30
condSelectUI . . . 31
cooks_outlier_summary . . . 31
correctCombat . . . 32
correctHarman . . . 33
create_debrowser_user . . . 33
customColorsUI . . . 34
cutOffSelectionServer . . . 34
cutOffSelectionUI . . . 35
cutoff_presets . . . 35
dataLCFUI . . . 36
dataLoadUI . . . 36
debrowserall2all . . . 37
debrowserbarmainplot . . . 37
debrowserbatcheffect . . . 38
debrowserboxmainplot . . . 39
debrowserdataload . . . 39
debrowserdeanalysis . . . 40
debrowserdensityplot . . . 41
debrowserheatmap . . . 41
debrowserhistogram . . . 42
debrowserIQRplot . . . 42
debrowserlowcountfilter . . . 43
debrowsermainplot . . . 43
debrowserpcaplot . . . 44
debrowserqccooks . . . 44
debrowserqcdetectionrate . . . 45
debrowserqcdispersion . . . 46
debrowserqclibrarydepth . . . 46
debrowserqcmtpct . . . 47
debrowserqcsampledist . . . 47
debrowserqcsizefactors . . . 48
default_cutoffs . . . 49
dendControlsUI . . . 49
densityPlotControlsUI . . . 50
deServer . . . 50
detection_rate . . . 51
detect_separator . . . 51
deUI . . . 52
de_assert_count_matrix . . . 53
de_bootswatch_link . . . 53
de_card . . . 54
de_direction_summary . . . 54
de_error . . . 55
de_eyebrow . . . 56
de_headline . . . 56
de_nav_chip . . . 57
de_preset_js . . . 57
de_preset_picker . . . 58
de_progress_label . . . 58
de_stat . . . 59

de_stat_strip . . . 59
de_style_guide . . . 60
de_theme . . . 60
de_workbar . . . 61
distFunParamsUI . . . 61
drawKEGG . . . 62
drawPCAExplained . . . 62
enrichmentGmtServer . . . 63
enrichmentGmtUI . . . 64
enrichmentNesHeatmapServer . . . 64
enrichmentNesHeatmapUI . . . 65
enrichmentServer . . . 65
enrichmentUI . . . 66
exportMenuItems . . . 67
exportMenuServer . . . 67
exportMenuUI . . . 68
fct_upload . . . 68
fileTypes . . . 69
fileUploadBox . . . 69
filter_low_counts . . . 70
filter_params_from_input . . . 70
flag_outliers_2sd . . . 71
fold_to_log2fc . . . 72
generateTestData . . . 72
getAfterLoadMsg . . . 73
getAll2AllPlotUI . . . 73
getBarMainPlot . . . 74
getBarMainPlotUI . . . 74
getBoxMainPlot . . . 75
getBoxMainPlotUI . . . 76
getBSTableUI . . . 76
getColors . . . 77
getColorShapeSelection . . . 78
getCompSelection . . . 78
getCondMsg . . . 79
getCutOffSelection . . . 80
getDataAssesmentText . . . 80
getDataForTables . . . 81
getDataPreparationText . . . 81
getDEAnalysisText . . . 82
getDensityPlot . . . 82
getDensityPlotUI . . . 83
getDEResultsUI . . . 83
getDomains . . . 84
getDown . . . 84
getDownloadSection . . . 85
getEnrichDO . . . 85
getEnrichGO . . . 86
getEnrichKEGG . . . 87
getEntrezIds . . . 87
getEntrezTable . . . 88
getGeneList . . . 89

getGeneSetData . . . 89
getGOLeftMenu . . . 90
getGoPanel . . . 90
getGOPlots . . . 91
getGSEA . . . 92
getHeatmapUI . . . 92
getHelpButton . . . 93
getHideLegendOnOff . . . 93
getHistogramUI . . . 94
getIntroText . . . 94
getIQRPlot . . . 95
getIQRPlotUI . . . 95
getJSLine . . . 96
getKEGGModal . . . 96
getLeftMenu . . . 97
getLegendColors . . . 97
getLegendRadio . . . 98
getLegendSelect . . . 98
getLevelOrder . . . 99
getLogo . . . 99
getMainPanel . . . 100
getMainPlotsLeftMenu . . . 100
getMainPlotUI . . . 101
getMean . . . 101
getMergedComparison . . . 102
getMostVariedList . . . 102
getNormalizedMatrix . . . 103
getOrganism . . . 103
getOrganismBox . . . 104
getOrganismPathway . . . 104
getPCAcontolUpdatesJS . . . 105
getPCAexplained . . . 105
getPCAPlotUI . . . 106
getPCselection . . . 107
getPlotArea . . . 107
getProgramTitle . . . 108
getQAText . . . 108
getQCLeftMenu . . . 109
getQCPanel . . . 109
getSampleDetails . . . 110
getSearchData . . . 110
getSelectedCols . . . 111
getSelectedDatasetInput . . . 111
getSelHeat . . . 112
getShapeColor . . . 113
getStartPlotsMsg . . . 113
getStartupMsg . . . 114
getTableDetails . . . 114
getTableModal . . . 115
getTableStyle . . . 115
getTabUpdateJS . . . 116
getUp . . . 117

getUpDown . . . 117
getVariationData . . . 118
get_most_varied . . . 118
get_table_data . . . 119
gmt_to_pathways . . . 120
heatmapControlsUI . . . 121
heatmapJScode . . . 121
heatmapServer . . . 122
heatmapUI . . . 122
hideObj . . . 123
histogramControlsUI . . . 123
install_cutoff_preset_observers . . . 124
IQRPlotControlsUI . . . 125
kmeansControlsUI . . . 125
lcfMetRadio . . . 126
library_depth_summary . . . 126
list_models . . . 127
log2fc_to_fold . . . 127
mainPlotControlsUI . . . 128
mainScatterNew . . . 128
make_default_metadata . . . 129
match_preset . . . 129
merge_comparisons . . . 130
methods_paragraph . . . 131
msigdb_pathways . . . 131
mt_pct_per_sample . . . 132
nes_heatmap_data . . . 133
niceKmeans . . . 133
normalizationMethods . . . 134
normalize_counts . . . 135
palUI . . . 135
panel.cor . . . 136
panel.hist . . . 136
parse_preset_cookie . . . 137
pcaPlotControlsUI . . . 137
plotData . . . 138
plotMarginsUI . . . 138
plotSizeMarginsUI . . . 139
plotSizeUI . . . 140
plotTypeUI . . . 140
plot_de_direction_bar . . . 141
plot_de_pairwise_heatmap . . . 141
plot_method_scatter . . . 142
plot_method_upset . . . 143
plot_pca . . . 144
prepDataContainer . . . 145
prepGroup . . . 146
prepHeatData . . . 147
prepPCADat . . . 147
progress_message . . . 148
push . . . 148
qcCooksUI . . . 149

qcDetectionRateUI . . . 149
qcDispersionUI . . . 150
qcLibraryDepthUI . . . 150
qcMtPctUI . . . 151
qcSampleDistUI . . . 151
qcSizeFactorsUI . . . 152
qc_keep_cols . . . 152
qc_keep_meta_rows . . . 153
removeCols . . . 153
removeExtraCols . . . 154
require_pkg . . . 154
resend_verification_email . . . 155
reset_debrowser_password . . . 155
round_vals . . . 156
runDE . . . 156
runDESeq2 . . . 157
runEdgeR . . . 158
runHeatmap . . . 159
runHeatmap2 . . . 160
runLimma . . . 160
run_de . . . 161
run_deseq2 . . . 162
run_de_methods . . . 163
run_edger . . . 164
run_gsea . . . 165
run_limma . . . 166
run_pca . . . 167
sample_distance_matrix . . . 168
search_geneset . . . 168
selectGroupInfo . . . 169
select_dataset . . . 170
sepRadio . . . 171
setBatch . . . 171
showObj . . . 172
size_factor_library_summary . . . 172
startDEBrowser . . . 173
startHeatmap . . . 174
textareaInput . . . 174
togglePanels . . . 175
update_progress . . . 175

Index **177**

`accountDropdownServer` *Server: wires the dropdown to current_user, signup modal, signout.*

Description

Server: wires the dropdown to current_user, signup modal, signout.

Usage

```
accountDropdownServer(id)
```

Arguments

id	Shiny module namespace id.

Value

invisible(NULL); called for its observers and outputs side effects (login/logout label, signup modal, sign-out flow).

---

accountDropdownUI *UI: navbar account dropdown.*

Description

UI: navbar account dropdown.

Usage

```
accountDropdownUI(id)
```

Arguments

id	Shiny module namespace id.

Value

A ‘bslib::nav_menu‘ element with a server-side-rendered label and a per-state menu (Sign up / Sign out).

---

actionButtonDE *Buttons including Action Buttons and Event Buttons*

Description

Creates an action button whose value is initially zero, and increments by one each time it is pressed.

Usage

```
actionButtonDE(
  inputId,
  label,
  styleclass = "",
  size = "",
  block = FALSE,
  icon = NULL,
  css.class = "",
  ...
)
```

Arguments

inputId	Specifies the input slot that will be used to access the value.
label	The contents of the button–usually a text label, but you could also use any other HTML, like an image.
styleclass	The Bootstrap styling class of the button–options are primary, info, success, warning, danger, inverse, link or blank
size	The size of the button–options are large, small, mini
block	Whehter the button should fill the block
icon	Display an icon for the button
css.class	Any additional CSS class one wishes to add to the action button
...	Other argument to feed into shiny::actionButton

Value

A ‘shiny::tags$button‘ element wired as a click-counting action button, suitable for embedding in any Shiny UI.

Examples

```
actionButtonDE("goDE", "Go to DE Analysis")
```

---

addDataCols *addDataCols*

Description

add aditional data columns to de results

Usage

```
addDataCols(data = NULL, de_res = NULL, cols = NULL, conds = NULL)
```

Arguments

data	loaded dataset
de_res	de results
cols	columns
conds	inputconds

Value

data

Examples

```
x <- addDataCols()
```

addID *addID*

Description

Adds an id to the data frame being used.

Usage

```
addID(data = NULL)
```

Arguments

data	loaded dataset

Value

data

Examples

```
x <- addID()
```

aiInterpretServer *AI interpretation panel server.*

Description

AI interpretation panel server.

Usage

```
aiInterpretServer(
  id,
  payload_react,
  settings_react,
  payload_shape = "geneset",
  deterministic_methods_react = NULL
)
```

Arguments

id	Module ID (matches [aiInterpretUI()]).
payload_react	Reactive expression returning the payload list per shape: * ‘geneset‘ shape: ‘list(genes, stats, enrichment, context_mode)‘ * ‘de_table‘ shape: ‘list(comparison_label, genes, stats, n_total_de, cutoffs)‘ * ‘concordance‘ shape: ‘list(comparison_labels, concordance_table, per_comparison_top, cutoffs)‘ May return NULL; Ask is disabled in that case.
settings_react	Reactive expression yielding the current AI settings list (from ‘aiSettingsServer‘).

payload_shape — One of '"geneset"', '"de_table"', '"concordance"'. Must match the shape passed to 'aiInterpretUI(id, ..., payload_shape)'. Default '"geneset"'.

deterministic_methods_react — Optional reactive expression yielding a chr(1) Methods paragraph (e.g. from 'methods_paragraph()'). Required for the 'draft_methods' preset; ignored otherwise.

Value

invisible(NULL).

---

aiInterpretUI *AI interpretation panel UI.*

Description

Renders a card with: question dropdown, privacy radio (hidden for 'draft_methods'), Top-N input, "What will be sent?" disclosure, Ask button, and a sanitized-markdown response area. The card is meant to be wrapped in a parent's conditionalPanel so it only renders when settings are valid and analytical results exist.

Usage

```
aiInterpretUI(id, questions = "summarize_geneset", payload_shape = "geneset")
```

Arguments

id — Module ID.

questions — Character vector of preset keys offered in the dropdown. Default '"summarize_geneset"' (E12.A behavior). Supported keys: 'summarize_geneset', 'reconcile_enrichments', 'suggest_followup', 'draft_methods'.

payload_shape — One of '"geneset"', '"de_table"', '"concordance"'. Drives slot building and per-shape privacy defaults. Default '"geneset"' (E12.A behavior).

Value

tagList.

Examples

```
aiInterpretUI("ai_enrichment")
aiInterpretUI("ai_de",
              questions     = c("summarize_geneset", "suggest_followup",
                                "draft_methods"),
              payload_shape = "de_table")
```

aiSettingsServer *Settings server – opens the AI configuration modal on click.*

Description

Settings server – opens the AI configuration modal on click.

Usage

```
aiSettingsServer(id)
```

Arguments

id	Module ID (matches [aiSettingsUI()]).

Value

shiny::reactive yielding the current settings list. Parent modules should consume this to gate the AI panel mount.

aiSettingsUI *Settings nav_menu UI (mounted in the navbar).*

Description

Returns a ‘bslib::nav_menu‘ titled "Settings" with one item: "AI Assistant". Clicking the item opens a modal handled by [aiSettingsServer()].

Usage

```
aiSettingsUI(id)
```

Arguments

id	Module ID.

Value

bslib::nav_menu element.

Examples

```
aiSettingsUI("ai")
```

ai_chat *Build an ellmer chat object for the configured provider/model.*

Description

Build an ellmer chat object for the configured provider/model.

Usage

```
ai_chat(provider, model, api_key = NULL)
```

Arguments

provider	chr(1).
model	chr(1).
api_key	chr(1) or NULL.

Value

An ellmer Chat object. Caller invokes '$chat(prompt)' on it.

ai_interpret *Send a redacted analytical question to a configured LLM provider.*

Description

Pure orchestrator over: payload redaction (per 'privacy_mode'), prompt template rendering (whisker), and dispatch to the provided ellmer chat object. Returns the model's response as 'chr(1)'. Errors are surfaced as classed conditions of class 'ai_error' so callers (the Shiny module) can pattern-match for friendly messages.

Usage

```
ai_interpret(
  question,
  payload,
  privacy_mode,
  provider_chat,
  top_n = 50L,
  template_dir = system.file("templates", package = "debrowser")
)
```

Arguments

question	chr(1). Preset key. v1 supports "summarize_geneset".
payload	list. Pre-built payload. Shape: ‘list(genes = chr, stats = data.frame|NULL, enrichment = list|NULL)‘.
privacy_mode	chr(1). One of "symbols", "stats", "stats_enrichment".
provider_chat	An ellmer chat object (output of ‘ai_chat()‘). The chat object must have a ‘$chat(text)‘ method that returns chr(1) on success or raises an error on failure.
top_n	integer(1). Cap on genes-list length. Default 50.
template_dir	Directory containing ‘ai_*.md‘ templates. Default ‘system.file("templates", package = "debrowser")‘.

Value

character(1). Model’s response text.

`all2all` *all2all*

Description

Prepares all2all scatter plots for given datasets.

Usage

```
all2all(data, cex = 2)
```

Arguments

data	data that have the sample names in the header.
cex	text size

Value

all2all scatter plots

Examples

```
plot <- all2all(mtcars)
```

all2allControlsUI *all2allControlsUI*

Description

Generates the controls in the left menu for an all2all plot

Usage

```
all2allControlsUI(id)
```

Arguments

id	namespace id

Value

returns the controls for left menu

Note

all2allControlsUI

Examples

```
x <- all2allControlsUI("bar")
```

applyFilters *applyFilters*

Description

Applies filters based on user selected parameters to be displayed within the DEBrowser.

Usage

```
applyFilters(filt_data = NULL, cols = NULL, conds = NULL, input = NULL)
```

Arguments

filt_data	loaded dataset
cols	selected samples
conds	seleced conditions
input	input parameters

Value

data

Examples

```
x <- applyFilters()
```

applyFiltersNew *applyFiltersNew*

Description

Apply filters based on foldChange cutoff and padj value. This function adds a "Legend" column with "Up", "Down" or "NS" values for visualization.

Usage

```
applyFiltersNew(data = NULL, input = NULL)
```

Arguments

data	loaded dataset
input	input parameters

Value

data

Examples

```
x <- applyFiltersNew()
```

applyFiltersToMergedComparison
applyFiltersToMergedComparison

Description

Gathers the merged comparison data to be used within the DEBrowser.

Usage

```
applyFiltersToMergedComparison(dc = NULL, nc = NULL, input = NULL)
```

Arguments

dc	all data
nc	the number of comparisons
input	input params

Value

data

Examples

```
x <- applyFiltersToMergedComparison()
```

apply_batch_correction *Apply batch-effect correction.*

Description

Pure function – no Shiny dependency. Accepts batch / treatment column names directly instead of pulling from a reactive ‘input‘ object.

Usage

```
apply_batch_correction(
  counts,
  metadata,
  method = "none",
  batch_col = NULL,
  treatment_col = NULL
)
```

Arguments

counts	Numeric matrix (genes x samples).
metadata	data.frame; first column is sample id.
method	One of "none", "Combat", "CombatSeq", "Harman".
batch_col	Name of the batch column in ‘metadata‘.
treatment_col	Name of the treatment column in ‘metadata‘. May be NULL or "None" – only required for Harman.

Value

Corrected count matrix.

Examples

```
m <- matrix(as.integer(c(100, 200, 150, 80, 250, 130,
                         40,  60,  55, 30,  70,  45)),
            nrow = 2, byrow = TRUE)
colnames(m) <- paste0("S", 1:6)
rownames(m) <- c("Gene1", "Gene2")
# method = "none" returns the matrix unchanged
apply_batch_correction(m, metadata = NULL, method = "none")
```

`apply_de_filters` *Apply DE filters and label Up/Down/NS/MV/GS rows.*

Description

Pure function – no Shiny dependency. Re-normalizes the count columns, computes per-condition x/y log10 means, and labels each row by cutoff.

Usage

```
apply_de_filters(filt_data, cols, conds, params = list())
```

Arguments

`filt_data`	data.frame with columns including ‘foldChange‘, ‘padj‘, plus all ‘cols‘.
`cols`	Character vector of sample column names.
`conds`	Character vector of per-column condition labels (length == length(cols)). Values look like "Cond1"/"Cond2"/...
`params`	Named list with components: - ‘padj_cutoff‘ (numeric or numeric-string) - ‘fold_cutoff‘ (numeric or numeric-string) - ‘dataset‘ ("up"/"down"/"up+down"/"alldetected"/"selected"/"most-varied"/"comparisons"/"searched") - ‘compselect‘ (integer; default 1) - ‘norm_method‘ (forwarded to [normalize_counts()]) - ‘geneset_area‘ (string of search terms; "" = none) - ‘method_tab‘ (UI tab id; geneset overlay only applies on "panel1") - ‘top_n‘, ‘min_count‘ (only for ‘dataset == "most-varied"‘)

Value

data.frame with added ‘x‘, ‘y‘, ‘Legend‘, ‘Size‘ columns; or NULL.

Examples

```
cols <- c("S1", "S2", "S3", "S4")
conds <- c("Cond1", "Cond1", "Cond2", "Cond2")
fd <- data.frame(
  S1 = c(100L, 5L), S2 = c(120L, 8L),
  S3 = c(10L, 200L), S4 = c(12L, 220L),
  foldChange = c(10, 0.05), padj = c(0.001, 0.001),
  row.names = c("GeneA", "GeneB")
)
params <- list(padj_cutoff = 0.05, fold_cutoff = 2, dataset = "up+down",
               compselect = 1, norm_method = "none")
apply_de_filters(fd, cols, conds, params)
```

apply_merged_filters *Apply Up/Down cutoffs across a merged-comparisons table.*

Description

Pure version of ‘applyFiltersToMergedComparison()‘.

Usage

```
apply_merged_filters(dc, nc, params = list())
```

Arguments

dc	List of per-comparison containers (each has ‘init_data‘, ‘cols‘, ‘cond_names‘).
nc	Number of comparisons.
params	Named list with ‘norm_method‘.

Value

Merged data.frame with a ‘Legend‘ column (‘"Sig"‘ / ‘"NS"‘).

Examples

```
init1 <- data.frame(
  S1 = c(100L, 40L), S2 = c(200L, 60L),
  S3 = c(10L, 150L), S4 = c(12L, 160L),
  foldChange = c(10, 0.25), padj = c(0.01, 0.01),
  row.names = c("GeneA", "GeneB")
)
dc <- list(list(
  init_data = init1,
  cols = c("S1", "S2", "S3", "S4"),
  cond_names = c("Treat", "Ctrl")
))
params <- list(norm_method = "none", padj_cutoff = 0.05, fold_cutoff = 2)
apply_merged_filters(dc, nc = 1, params = params)
```

barMainPlotControlsUI *barMainPlotControlsUI*

Description

Generates the controls in the left menu for a bar main plot

Usage

```
barMainPlotControlsUI(id)
```

Arguments

id	namespace id

Value

returns the controls for left menu

Note

barMainPlotControlsUI

Examples

```
x <- barMainPlotControlsUI("bar")
```

---

batchEffectUI *batchEffectUI Creates a panel to coorect batch effect*

Description

batchEffectUI Creates a panel to coorect batch effect

Usage

```
batchEffectUI(id)
```

Arguments

id	namespace id

Value

panel

Examples

```
x <- batchEffectUI("batcheffect")
```

---

batchMethod *batchMethod*

Description

select batch effect method

Usage

```
batchMethod(id)
```

Arguments

id	namespace id

Value

radio control

Note

batchMethod

Examples

```
x <- batchMethod("batch")
```

BoxMainPlotControlsUI *BoxMainPlotControlsUI*

Description

Generates the controls in the left menu for a Box main plot

Usage

```
BoxMainPlotControlsUI(id)
```

Arguments

id	namespace id

Value

returns the controls for left menu

Note

BoxMainPlotControlsUI

Examples

```
x <- BoxMainPlotControlsUI("box")
```

changeClusterOrder *changeClusterOrder*

Description

change order of K-means clusters

Usage

```
changeClusterOrder(order = NULL, cld = NULL)
```

Arguments

order	order
cld	data

Value

heatmap plot area

Note

changeClusterOrder

Examples

```
x <- changeClusterOrder()
```

checkCountData *checkCountData*

Description

Returns if there is a problem in the count data.

Usage

```
checkCountData(input = NULL, sep = NULL)
```

Arguments

input	inputs
sep	optional override for the field separator; defaults to ‘input$countdataSep‘ when NULL

Value

error if there is a problem about the loaded data

Note

checkCountData

Examples

```
x <- checkCountData()
```

---

checkMetaData *checkMetaData*

Description

Returns if there is a problem in the count data.

Usage

```
checkMetaData(input = NULL, counttable = NULL, sep = NULL)
```

Arguments

input	input
counttable	counttable
sep	optional override for the field separator; defaults to 'input$metadataSep' when NULL

Value

error if there is a problem about the loaded data

Note

checkMetaData

Examples

```
x <- checkMetaData()
```

clusterData *clusterData*

Description

Gathers the Cluster analysis data to be used within the GO Term plots.

Usage

```
clusterData(dat = NULL)
```

Arguments

dat	the data to cluster

Value

clustered data

Note

clusterData

Examples

```
mycluster <- clusterData()
```

clustFunParamsUI *clustFunParamsUI*

Description

get cluster function parameter control

Usage

```
clustFunParamsUI()
```

Value

cluster params

Note

clustFunParamsUI

Examples

```
x <- clustFunParamsUI()
```

compareClust *compareClust*

Description

Compares the clustered data to be displayed within the GO Term plots.

Usage

```
compareClust(
  dat = NULL,
  ont = "CC",
  org = "org.Hs.eg.db",
  fun = "enrichGO",
  title = "Ontology Distribution Comparison",
  pvalueCutoff = 0.01
)
```

Arguments

dat	data to compare clusters
ont	the ontology to use
org	the organism used
fun	fun
title	title of the comparison
pvalueCutoff	pvalueCutoff

Value

compared cluster

Note

compareClust

Examples

```
x <- compareClust()
```

`comparisonConcordanceServer` *Server for the Comparison Concordance tab.*

Description

Server for the Comparison Concordance tab.

Usage

```
comparisonConcordanceServer(
  id,
  de_results_react,
  comparisons_react = NULL,
  ai_settings_react = NULL,
  fgsea_pathways_react = NULL,
  deterministic_methods_react = NULL
)
```

Arguments

`id`	Module ID.
`de_results_react`	Reactive yielding the named list of per-comparison DE result data.frames (typically 'de_results_list' from 'R/server.R'). Each element must contain 'ID', 'log2FoldChange', and 'padj' columns; entries are keyed by 'comparison_labels()' so colliding labels stay unique.
`comparisons_react`	Reactive yielding the full comparison list (typically 'dc()' from 'R/server.R'); each entry should have 'cond_names' so the pairwise DEG heatmap can label group axes. When NULL or missing cond_names, the heatmap card shows an empty-state.
`ai_settings_react`	Optional reactive yielding AI settings list (from 'aiSettingsServer'). When NULL, no AI panel is mounted.
`fgsea_pathways_react`	Optional reactive yielding the active fgsea pathway set ('names()' over the gene-set lookup). Drives the in-card pathway picker for 'reconcile_enrichments'.
`deterministic_methods_react`	Optional reactive yielding a chr(1) Methods paragraph (e.g. from 'methods_paragraph()'). Consumed by the 'draft_methods' preset.

Value

invisible(NULL).

Examples

```
de <- list(
  "Treat vs Ctrl" = data.frame(ID = paste0("G", 1:5),
    log2FoldChange = c(2, -1, 0, 3, -2),
    padj = c(0.01, 0.04, 0.5, 0.02, 0.03)),
  "Drug vs Vehicle" = data.frame(ID = paste0("G", 1:5),
    log2FoldChange = c(1.5, -0.8, 0.2, 2.1, -1.5),
    padj = c(0.02, 0.03, 0.6, 0.01, 0.04))
)
shiny::shinyApp(
  ui = comparisonConcordanceUI("cc"),
  server = function(input, output, session) {
    comparisonConcordanceServer("cc", shiny::reactive(de))
  }
)
```

---

comparisonConcordanceUI

UI for the Comparison Concordance top-level tab.

Description

UI for the Comparison Concordance top-level tab.

Usage

```
comparisonConcordanceUI(id)
```

Arguments

id	Module ID.

Value

A ‘shiny::tagList‘ with a control card (padj / |log2FC|), an UpSet card, a pairwise scatter card with x/y comparison selectizes, and a concordance summary table card.

Examples

```
comparisonConcordanceUI("demo")
```

comparison_labels *Build unique display labels for a list of comparisons.*

Description

Each comparison contributes a label "<treatment> vs <control>" derived from its ‘cond_names‘ field. When two comparisons collide on the same label, suffixes " (1)", " (2)", ... are appended in input order so ‘names()‘ of the final list stays unique. Comparisons missing ‘cond_names‘ fall back to "comparison_<index>".

Usage

```
comparison_labels(comparisons)
```

Arguments

comparisons	List of per-comparison entries (typically ‘dc()‘ from server.R or ‘comparisons_spec()‘); each entry should have a ‘cond_names‘ character vector of length >= 2 (treatment first).

Value

Character vector of length ‘length(comparisons)‘. Empty character vector when input is empty.

Examples

```
comparisons <- list(
  list(cond_names = c("Treat", "Ctrl")),
  list(cond_names = c("Drug", "Vehicle"))
)
comparison_labels(comparisons)
```

compute_pill_class *Compute the CSS class name for a pill given its progress state.*

Description

Used by the JS handler in ‘getTabUpdateJS()‘ (R/funcs.R) and as a direct render path for static initial state.

Usage

```
compute_pill_class(state)
```

Arguments

state	character, progress state.

Value

character, CSS class name or empty string.

Examples

```
compute_pill_class("done")
```

`concordance_sets` *Build per-method significant-gene sets.*

Description

Build per-method significant-gene sets.

Usage

```
concordance_sets(de_list, padj_cutoff = 0.05, lfc_cutoff = 0)
```

Arguments

`de_list`	Output of [run_de_methods()].
`padj_cutoff`	Maximum padj for "significant" (default 0.05).
`lfc_cutoff`	Minimum |log2FoldChange| for "significant" (default 0; pass 0.585 for |fold|>=1.5, 1 for |fold|>=2).

Value

Named list of character vectors of significant gene IDs. Names match ‘names(de_list)‘.

Examples

```
de_list <- list(
  EdgeR = data.frame(ID = paste0("G", 1:5),
    log2FoldChange = c(2, -1, 0, 3, -2),
    padj = c(0.01, 0.04, 0.5, 0.02, 0.03)),
  Limma = data.frame(ID = paste0("G", 1:5),
    log2FoldChange = c(1.8, -0.9, 0.1, 2.5, -1.8),
    padj = c(0.02, 0.03, 0.6, 0.01, 0.04))
)
concordance_sets(de_list, padj_cutoff = 0.05)
```

`concordance_summary` *Pairwise concordance summary across DE methods.*

Description

For each unordered pair of methods, computes set sizes, intersection size, Jaccard index, and Spearman correlation of log2FoldChange over the set of genes present in both per-method tables (excluding NAs).

Usage

```
concordance_summary(de_list, padj_cutoff = 0.05, lfc_cutoff = 0)
```

Arguments

de_list	Output of [run_de_methods()].
padj_cutoff	Maximum padj for "significant" (default 0.05).
lfc_cutoff	Minimum |log2FoldChange| for "significant" (default 0; pass 0.585 for |fold|>=1.5, 1 for |fold|>=2).

Value

data.frame with one row per pair, columns: ‘method1‘, ‘method2‘, ‘n_method1‘, ‘n_method2‘, ‘n_overlap‘, ‘jaccard‘, ‘spearman_lfc‘, ‘n_genes_compared‘. Empty data.frame (zero rows) when fewer than 2 methods are present.

Examples

```
de_list <- list(
  EdgeR = data.frame(ID = paste0("G", 1:5),
    log2FoldChange = c(2, -1, 0, 3, -2),
    padj = c(0.01, 0.04, 0.5, 0.02, 0.03)),
  Limma = data.frame(ID = paste0("G", 1:5),
    log2FoldChange = c(1.8, -0.9, 0.1, 2.5, -1.8),
    padj = c(0.02, 0.03, 0.6, 0.01, 0.04))
)
concordance_summary(de_list, padj_cutoff = 0.05)
```

---

condSelectServer *Comparison-Selection wizard server.*

Description

Comparison-Selection wizard server.

Usage

```
condSelectServer(id, data = NULL, metadata = NULL, initial_spec = NULL)
```

Arguments

id	module namespace id (must match the id passed to ‘condSelectUI‘).
data	count matrix.
metadata	sample-metadata data.frame; first column is the sample id.
initial_spec	Optional saved comparisons_spec to populate the wizard with on mount. When the module is created during an active bookmark restore, the module’s own ‘onRestore‘ hook handles this; when the module is created LATER (e.g. by the D2.5 DE auto-replay state machine in deServer, after the active-restore window has closed), pass the captured spec here so the per-comparison reactiveValues and UI cards reflect the restored state.

Value

list with ‘n_comparisons‘, ‘start_de‘, ‘is_ready‘, ‘comparisons_spec‘.

Examples

```
counts <- matrix(as.integer(c(100, 200, 10, 12, 40, 60)),
                 nrow = 2,
                 dimnames = list(c("G1", "G2"), paste0("S", 1:3)))
meta <- data.frame(Sample = paste0("S", 1:3),
                   Condition = c("Ctrl", "Ctrl", "Treat"))
shiny::shinyApp(
  ui = condSelectUI("cs"),
  server = function(input, output, session) {
    condSelectServer("cs", data = counts, metadata = meta)
  }
)
```

---

condSelectUI *Comparison-Selection wizard UI.*

Description

Comparison-Selection wizard UI.

Usage

```
condSelectUI(id)
```

Arguments

id	module namespace id.

Value

a ‘de_card‘ containing the wizard’s static skeleton; comparison panels are rendered dynamically by ‘condSelectServer‘ via ‘uiOutput("comparison_panels")‘.

Examples

```
condSelectUI("demo")
```

---

cooks_outlier_summary *Per-sample Cook’s distance outlier counts.*

Description

Counts how many genes in each sample have a Cook’s distance above a threshold; samples with disproportionately many high-Cook genes drove DE calls more than their share and may warrant a downstream re-check. Default threshold is the canonical DESeq2 vignette cut, ‘4 / (n_samples - n_params)‘, where ‘n_params‘ is the column count of ‘model.matrix(design(dds), colData(dds))‘. The chosen threshold is attached as ‘attr(out, "threshold")‘.

Usage

```
cooks_outlier_summary(dds, threshold = NULL)
```

Arguments

dds	A fitted 'DESeqDataSet'. Must carry the 'cooks' assay (i.e. 'DESeq()' was run); otherwise raises 'qc_input_error'.
threshold	Optional numeric Cook's-distance cut. NULL (default) uses the DESeq2 vignette cut.

Value

data.frame with columns 'sample', 'n_high_cooks', 'total_genes', 'high_cooks_pct'. The active threshold is attached as 'attr(out, "threshold")'.

Examples

```
cooks_outlier_summary(dds)
```

---

correctCombat *Correct Batch Effect using Combat in sva package*

Description

Batch effect correction

Usage

```
correctCombat(input = NULL, idata = NULL, metadata = NULL, method = NULL)
```

Arguments

input	input values
idata	data
metadata	metadata
method	method: either Combat or CombatSeq

Value

data

Examples

```
x <- correctCombat()
```

correctHarman *Correct Batch Effect using Harman*

Description

Batch effect correction

Usage

```
correctHarman(input = NULL, idata = NULL, metadata = NULL)
```

Arguments

input	input values
idata	data
metadata	metadata

Value

data

Examples

```
x <- correctHarman()
```

create_debrowser_user *Create a debrowser user from the R console.*

Description

Opens `users.sqlite`, writes a new `kind = "shinymanager"` row, and closes the connection. Use this to bootstrap the first user when running `startDEBrowser(hosted = TRUE)` for the first time, since the in-app signup modal lives behind the login wall and is unreachable until you have an account.

Usage

```
create_debrowser_user(user_id, password, email = NA_character_)
```

Arguments

user_id	Username for login.
password	Plaintext password (will be hashed via scrypt).
email	Optional email. Default `NA_character_`.

Value

The created user_id, invisibly.

Examples

```
create_debrowser_user("alice", "hunter2!", "alice@example.com")
startDEBrowser(hosted = TRUE)  # then log in as alice / hunter2!
```

`customColorsUI` *customColorsUI*

Description

get Custom Color controls

Usage

```
customColorsUI(id)
```

Arguments

`id`	namespace ID

Value

color range

Note

`getColRng`

Examples

```
x <- customColorsUI("heatmap")
```

`cutOffSelectionServer` *cutOffSelectionServer*

Description

Server-side companion for cutOffSelectionUI. Wires the preset radio-group buttons to the namespaced numeric inputs via install_cutoff_preset_observers.

Usage

```
cutOffSelectionServer(id)
```

Arguments

`id`	Module id (matches cutOffSelectionUI(id)).

Value

Invisible NULL.

Examples

```
if (FALSE) cutOffSelectionServer("DEResults1")
```

---

cutOffSelectionUI *cutOffSelectionUI*

Description

Gathers the cut off selection for DE analysis

Usage

```
cutOffSelectionUI(id)
```

Arguments

id	namespace id

Value

returns the left menu according to the selected tab;

Note

cutOffSelectionUI

Examples

```
x <- cutOffSelectionUI("cutoff")
```

---

cutoff_presets *Cutoff preset table.*

Description

Each row defines a one-click preset for the Strict / Standard button bar. log2fc is shared across presets by design – only padj differs.

Usage

```
cutoff_presets()
```

Value

data.frame with columns: name, label, padj, log2fc.

Examples

```
cutoff_presets()
```

dataLCFUI *dataLCFUI Creates a panel to filter low count genes and regions*

Description

dataLCFUI Creates a panel to filter low count genes and regions

Usage

```
dataLCFUI(id)
```

Arguments

id	namespace id

Value

panel

Examples

```
x <- dataLCFUI("lcf")
```

dataLoadUI *dataLoadUI*

Description

Creates a panel to upload the data

Usage

```
dataLoadUI(id)
```

Arguments

id	namespace id

Value

panel

Examples

```
x <- dataLoadUI("load")
```

debrowserall2all *debrowserall2all*

Description

Module for a bar plot that can be used in data prep, main plots low count removal modules or any desired module

Usage

```
debrowserall2all(id, data = NULL, cex = 2)
```

Arguments

id	namespace id
data	a matrix that includes expression values
cex	the size of the dots

Value

all2all plot

Examples

```
x <- debrowserall2all("all2all")
```

debrowserbarmainplot *debrowserbarmainplot*

Description

Module for a bar plot that can be used in data prep, main plots low count removal modules or any desired module

Usage

```
debrowserbarmainplot(
  id,
  data = NULL,
  cols = NULL,
  conds = NULL,
  cond_names = NULL,
  key = NULL
)
```

Arguments

id	namespace id
data	a matrix that includes expression values
cols	columns
conds	conditions
cond_names	condition names
key	the gene or region name

Value

density plot

Examples

```
x <- debrowserbarmainplot("bar")
```

---

debrowserbatcheffect *debrowserbatcheffect*

Description

Module to correct batch effect

Usage

```
debrowserbatcheffect(id, ldata = NULL)
```

Arguments

id	namespace id
ldata	loaded data

Value

main plot

panel

Examples

```
x <- debrowserbatcheffect("batch")
```

debrowserboxmainplot *debrowserboxmainplot*

Description

Module for a box plot that can be used in DEanalysis main part and used heatmaps

Usage

```
debrowserboxmainplot(
  id,
  data = NULL,
  cols = NULL,
  conds = NULL,
  cond_names = NULL,
  key = NULL
)
```

Arguments

id	namespace id
data	a matrix that includes expression values
cols	columns
conds	conditions
cond_names	condition names
key	the gene or region name

Value

density plot

Examples

```
x <- debrowserboxmainplot("box")
```

debrowserdataload *debrowserdataload*

Description

Module to load count data and metadata

Usage

```
debrowserdataload(id, nextpagebutton = NULL)
```

Arguments

id	namespace id
nextpagebutton	the name of the next page button after loading the data

Value

main plot

panel

Examples

```
x <- debrowserdataload("load")
```

---

debrowserdeanalysis *debrowserdeanalysis*

Description

Module to perform and visualize DE results.

Usage

```
debrowserdeanalysis(
  id,
  data = NULL,
  metadata = NULL,
  columns = NULL,
  conds = NULL,
  params = NULL
)
```

Arguments

id	namespace id
data	a matrix that includes expression values
metadata	metadata
columns	columns
conds	conditions
params	de parameters

Value

DE panel

Examples

```
x <- debrowserdeanalysis("de")
```

debrowserdensityplot *debrowserdensityplot*

Description

Module for a density plot that can be used in data prep and low count removal modules

Usage

```
debrowserdensityplot(id, data = NULL)
```

Arguments

id	namespace id
data	a matrix that includes expression values

Value

density plot

Examples

```
x <- debrowserdensityplot("density")
```

debrowserheatmap *debrowserheatmap*

Description

Heatmap module to create interactive heatmaps and get selected list from a heatmap

Usage

```
debrowserheatmap(id, expdata = NULL)
```

Arguments

id	namespace id
expdata	a matrix that includes expression values

Value

heatmapply plot

Examples

```
x <- debrowserheatmap("heatmap")
```

debrowserhistogram *debrowserhistogram*

Description

Module for a histogram that can be used in data prep and low count removal modules

Usage

```
debrowserhistogram(id, data = NULL)
```

Arguments

id	namespace id
data	a matrix that includes expression values

Value

histogram

Examples

```
x <- debrowserhistogram("histogram")
```

debrowserIQRplot *debrowserIQRplot*

Description

Module for an IQR plot that can be used in data prep and low count removal modules

Usage

```
debrowserIQRplot(id, data = NULL)
```

Arguments

id	namespace id
data	a matrix that includes expression values

Value

IQR

Examples

```
x <- debrowserIQRplot("iqr")
```

debrowserlowcountfilter *debrowserlowcountfilter*

Description

Module to filter low count genes/regions

Usage

```
debrowserlowcountfilter(id, ldata = NULL)
```

Arguments

id	namespace id
ldata	loaded data

Value

main plot

panel

Examples

```
x <- debrowserlowcountfilter("lcf")
```

debrowsermainplot *debrowsermainplot*

Description

Module for a scatter, volcano and ma plots that are going to be used as a mainplot in debrowser

Usage

```
debrowsermainplot(id, data = NULL, cond_names = NULL)
```

Arguments

id	namespace id
data	a matrix that includes expression values
cond_names	condition names

Value

main plot

panel

Examples

```
x <- debrowsermainplot("main")
```

`debrowserpcaplot` *debrowserpcaplot*

Description

Module for a pca plot with its loadings as a mainplot in debrowser

Usage

```
debrowserpcaplot(id, pcadata = NULL, metadata = NULL)
```

Arguments

`id`	namespace id
`pcadata`	a matrix that includes expression values
`metadata`	metadata to color the plots

Value

main plot

panel

Examples

```
x <- debrowserpcaplot("pca")
```

`debrowserqccooks` *debrowserqccooks*

Description

Server factory for the Cook's-distance outlier-count QC card. Computes `cooks_outlier_summary` and renders a vertical bar of 'n_high_cooks' per sample. Subtitle reports the active threshold.

Usage

```
debrowserqccooks(id, dds = NULL, selected_samples = NULL)
```

Arguments

id	character, namespace id matching 'qcCooksUI(id)'
dds	A fitted 'DESeqDataSet', or NULL.
selected_samples	Optional character vector of sample names to show. NULL (default) shows every sample in the dds; otherwise only bars for samples whose name is in 'selected_samples' are rendered (mirrors the QC sidebar's column-selector behaviour).

Value

invisible(NULL); wires 'output$body', 'output$plot', and 'output$dl'.

Examples

```
debrowserqccooks("cooks", dds)
```

---

debrowserqcdetectionrate
debrowserqcdetectionrate

Description

Server factory for the feature-detection-rate QC card. Computes `detection_rate` and renders a horizontal 'plotly' bar of 'detection_pct' per sample.

Usage

```
debrowserqcdetectionrate(id, counts = NULL)
```

Arguments

id	character, namespace id matching 'qcDetectionRateUI(id)'
counts	numeric matrix or data.frame (rows = features, cols = samples); the module no-ops if NULL

Value

invisible(NULL); the module wires 'output$plot' and 'output$dl'

Examples

```
debrowserqcdetectionrate("detectionRate", counts)
```

debrowserqcdispersion *debrowserqcdispersion*

Description

Server factory for the dispersion-estimates QC card. Wraps `DESeq2::plotDispEsts(dds)` in ‘renderPlot‘ (base-R graphics, not plotly). Renders an empty-state alert when ‘dds‘ is NULL (no DE run yet, or method was not DESeq2).

Usage

```
debrowserqcdispersion(id, dds = NULL)
```

Arguments

`id`	character, namespace id matching ‘qcDispersionUI(id)‘
`dds`	A fitted ‘DESeqDataSet‘, or NULL.

Value

invisible(NULL); wires ‘output$body‘ and ‘output$plot‘.

Examples

```
debrowserqcdispersion("dispersion", dds)
```

debrowserqclibrarydepth *debrowserqclibrarydepth*

Description

Server factory for the library-depth QC card. Computes per-sample depth via `library_depth_summary` and renders a horizontal ‘plotly‘ bar colored by the optional ‘group_col‘. Outliers (>2 SD from the mean depth) are drawn with a red marker outline.

Usage

```
debrowserqclibrarydepth(id, counts = NULL, meta = NULL, group_col = NULL)
```

Arguments

`id`	character, namespace id matching ‘qcLibraryDepthUI(id)‘
`counts`	numeric matrix or data.frame (rows = features, cols = samples); the module no-ops if NULL
`meta`	optional metadata data.frame with a ‘samples‘ column
`group_col`	optional name of a column in ‘meta‘ used to color bars

Value

invisible(NULL); the module wires ‘output$plot‘ and ‘output$dl‘

Examples

```
debrowserqclibrarydepth("libraryDepth", counts, meta, "samples")
```

---

debrowserqcmtpct *debrowserqcmtpct*

Description

Server factory for the mitochondrial-percentage QC card. Computes `mt_pct_per_sample` and either renders a ‘plotly‘ bar with a horizontal threshold line at ‘threshold_pct‘, or an empty-state info alert when no MT genes are detected.

Usage

```
debrowserqcmtpct(id, counts = NULL, threshold_pct = 5)
```

Arguments

`id`	character, namespace id matching ‘qcMtPctUI(id)‘
`counts`	numeric matrix or data.frame (rows = features, cols = samples); the module no-ops if NULL
`threshold_pct`	numeric, the horizontal warning threshold drawn over the bar plot (default 5)

Value

invisible(NULL); the module wires ‘output$body‘, ‘output$plot‘, and ‘output$dl‘

Examples

```
debrowserqcmtpct("mtPct", counts, threshold_pct = 5)
```

---

debrowserqcsampledist *debrowserqcsampledist*

Description

Server factory for the sample-distance heatmap. Computes `sample_distance_matrix` and renders it via `heatmaply::heatmaply`.

Usage

```
debrowserqcsampledist(id, counts = NULL)
```

Arguments

id	character, namespace id matching 'qcSampleDistUI(id)'
counts	numeric matrix or data.frame (rows = features, cols = samples); the module no-ops if NULL

Value

invisible(NULL); the module wires 'output$plot' and 'output$dl'

Examples

```
debrowserqcsampledist("sampleDist", counts)
```

---

debrowserqcsizefactors

debrowserqcsizefactors

Description

Server factory for the size-factors-vs-library-size QC card. Computes size_factor_library_summary and renders two scaled bars per sample plus the Spearman correlation in the subtitle.

Usage

```
debrowserqcsizefactors(id, dds = NULL, selected_samples = NULL)
```

Arguments

id	character, namespace id matching 'qcSizeFactorsUI(id)'
dds	A fitted 'DESeqDataSet', or NULL.
selected_samples	Optional character vector of sample names to show. NULL (default) shows every sample in the dds; otherwise only bars for samples whose name is in 'selected_samples' are rendered (mirrors the QC sidebar's column-selector behaviour).

Value

invisible(NULL); wires 'output$body', 'output$plot', and 'output$dl'.

Examples

```
debrowserqcsizefactors("sizeFactors", dds)
```

default_cutoffs *Default DE significance cutoffs.*

Description

Single source of truth for the values that populate the DE cutoff inputs and the generateTestData() fallback in R/mainScatter.R.

Usage

```
default_cutoffs()
```

Value

Named list with components: padj, log2fc, gopvalue.

Examples

```
default_cutoffs()
```

dendControlsUI *dendControlsUI*

Description

get distance metric parameters

Usage

```
dendControlsUI(id, dendtype = "Row")
```

Arguments

id	module ID
dendtype	Row or Col

Value

controls

Note

dendControlsUI

Examples

```
x <- dendControlsUI("heatmap")
```

densityPlotControlsUI *densityPlotControlsUI*

Description

Generates the controls in the left menu for a densityPlot

Usage

```
densityPlotControlsUI(id)
```

Arguments

id	namespace id

Value

returns the left menu

Note

densityPlotControlsUI

Examples

```
x <- densityPlotControlsUI("density")
```

deServer *deServer*

Description

Sets up shinyServer to be able to run DEBrowser interactively.

Usage

```
deServer(input, output, session)
```

Arguments

input	input params from UI
output	output params to UI
session	session variable

Value

the panel for main plots;

Note

deServer

Examples

```
deServer
```

`detection_rate` *Per-sample gene detection rate.*

Description

Counts the number of features with values strictly greater than ‘threshold‘ per sample and reports detection percentage.

Usage

```
detection_rate(counts, threshold = 0)
```

Arguments

`counts`	Numeric matrix or data.frame (rows = features, cols = samples).
`threshold`	Detection threshold (default 0). A feature is "detected" in a sample iff its value is strictly greater than ‘threshold‘.

Value

data.frame with columns ‘sample‘, ‘n_detected‘, ‘total_features‘, ‘detection_pct‘ (one row per sample).

Examples

```
m <- matrix(c(0, 0, 5, 5, 0, 5), nrow = 3,
            dimnames = list(NULL, c("a", "b")))
detection_rate(m)
```

`detect_separator` *Auto-detect the field separator in a count-data file.*

Description

Tries tab, comma, semicolon. Returns the highest-scoring delimiter with score >= ‘min_score‘, with the tie-break order tab > comma > semicolon. Returns NA when no candidate clears the threshold.

Usage

```
detect_separator(path, sample_lines = 50L, min_score = 3L)
```

Arguments

`path`	character, path to the file (may end in ‘.gz‘).
`sample_lines`	integer, number of lines to read for scoring.
`min_score`	integer, minimum numeric-column count required to accept a candidate delimiter.

Details

Decompresses '.gz' files and strips a leading UTF-8 BOM before scoring. Excel files are not handled here – callers should branch on extension before calling this.

Counts files default to 'min_score = 3' (3+ numeric columns is a strong signal). Metadata files have only 1-2 numeric columns typically; pass 'min_score = 1' to keep the helper useful for them.

Value

one of tab, comma, semicolon, or 'NA_character_'.

Examples

```
tmp <- tempfile(fileext = ".txt")
writeLines(c("Gene\tS1\tS2\tS3",
             "GeneA\t100\t200\t150",
             "GeneB\t40\t60\t55"), tmp)
detect_separator(tmp)
unlink(tmp)
```

---

deUI *deUI*

Description

Creates a shinyUI to be able to run DEBrowser interactively. B1 shell: bslib::page_navbar with 5 nav panels (Data Prep / Main Plots / QC Plots / GO Term / Tables), Slate + OK-blue theme, light/dark toggle.

Usage

```
deUI(req = NULL)
```

Arguments

req	Shiny request object (auto-supplied by Shiny when 'deUI' is used as the 'ui' argument to 'shinyApp()').

Details

Accepts a Shiny 'request' argument so that the theme can be swapped at runtime via the '?preset=NAME' query parameter (e.g. '?preset=zephyr'). Unknown or missing 'preset' keeps the default theme.

Value

the page tagList for DEBrowser

Note

deUI

Examples

```
shiny::shinyApp(ui = deUI, server = deServer)
```

`de_assert_count_matrix` *Validate that x is a non-empty numeric count matrix.*

Description

Raises ‘de_error()‘ with one of: ‘"null_input"‘, ‘"empty_matrix"‘, ‘"non_numeric"‘. Otherwise returns ‘x‘ invisibly.

Usage

```
de_assert_count_matrix(x)
```

Arguments

`x`	Object to validate.

Value

‘x‘ (invisibly) if valid.

Examples

```
m <- matrix(c(100L, 200L, 150L, 40L, 60L, 55L), nrow = 2)
colnames(m) <- paste0("S", 1:3)
rownames(m) <- c("GeneA", "GeneB")
de_assert_count_matrix(m)
```

`de_bootswatch_link` *Build a bootswatch CDN <link> tag for a given preset*

Description

Returns NULL when ‘preset‘ is NULL, "bootstrap", or "shiny" (no CDN counterpart). Otherwise returns a ‘<link rel="stylesheet">‘ pointing at ‘https://bootswatch.com/5/<preset>/bootstrap.min.css‘. Used by ‘deUI()‘.

Usage

```
de_bootswatch_link(preset)
```

Arguments

`preset`	bslib preset name or NULL.

Value

an ‘htmltools::tag‘ or NULL.

de_card *de_card*

Description

DEBrowser card wrapper used at every former ‘shinydashboard::box()‘ site. Optionally renders a small primary download button on the right of the card header.

Usage

```
de_card(title, ..., download_id = NULL, full_screen = FALSE, class = NULL)
```

Arguments

title	character – card title shown in the header
...	card body content
download_id	character or NULL – if non-NULL, render a ‘downloadButton‘ with this id in the header
full_screen	logical – passed through to ‘bslib::card()‘
class	character or NULL – extra CSS class(es) forwarded to ‘bslib::card()‘ (e.g. "de-subcard", "de-comparison")

Value

a ‘bslib::card‘ tagList

Examples

```
x <- de_card("Heatmap", shiny::plotOutput("heat"))
```

de_direction_summary *Per-entry up / down / total significant gene counts.*

Description

Counts significant genes per entry of a named DE-results list at the given padj / |log2FoldChange| cutoffs. Up = significant + log2FC > 0; Down = significant + log2FC < 0. NA padj / NA log2FoldChange are treated as not-significant.

Usage

```
de_direction_summary(de_list, padj_cutoff = 0.05, lfc_cutoff = 0)
```

Arguments

de_list	Named list of DE result data.frames; each must contain ‘padj‘ and ‘log2FoldChange‘ columns.
padj_cutoff	Maximum padj for "significant" (default 0.05).
lfc_cutoff	Minimum |log2FoldChange| for "significant" (default 0).

Value

data.frame with columns ‘comparison‘, ‘n_up‘, ‘n_down‘, ‘n_sig‘ (= ‘n_up + n_down‘). One row per entry of ‘de_list‘, in input order.

Examples

```
de_list <- list(
  comp1 = data.frame(
    log2FoldChange = c(2, -1, 0, 3, -2),
    padj = c(0.01, 0.04, 0.5, 0.02, 0.03)
  ),
  comp2 = data.frame(
    log2FoldChange = c(1.5, -0.8, 0.2, 2.1, -1.5),
    padj = c(0.02, 0.03, 0.6, 0.01, 0.04)
  )
)
de_direction_summary(de_list, padj_cutoff = 0.05)
```

---

de_error *Raise a structured DEBrowser error.*

Description

All errors raised by the ‘fct_*‘ pure functions go through this helper so callers (tests, scripts, Shiny modules) can dispatch on the error class rather than parsing message strings.

Usage

```
de_error(message, class = character(), ...)
```

Arguments

message	Human-readable error message.
class	Optional character vector of additional classes to prepend to the condition's class list. Always inherits from ‘"debrowser_error"‘.
...	Extra named fields stored on the condition for caller use.

Value

Nothing – always raises.

Examples

```
tryCatch(de_error("bad input"), error = function(e) e$message)
```

de_eyebrow *de_eyebrow*

Description

B3 helper. Renders a small numbered chip + eyebrow label above a panel headline. The CSS for ‘.de-eyebrow‘ and ‘.de-eyebrow-chip‘ lives in inst/extdata/www/debrowser.css and is dormant until the redesign layer is toggled on via ‘data-debrowser-redesign="1"‘ on ‘<html>‘.

Usage

```
de_eyebrow(num, label)
```

Arguments

num	character / numeric – the chip number (1..N)
label	character – the eyebrow text

Value

an ‘htmltools::tag‘ (‘ ‘).

Examples

```
x <- de_eyebrow(1, "Upload & configure")
```

de_headline *de_headline*

Description

B3 helper. Page-level headline rendered just under the eyebrow.

Usage

```
de_headline(text)
```

Arguments

text	character – the headline text

Value

an ‘htmltools::tag‘ (‘<h2 class="de-headline">‘).

Examples

```
x <- de_headline("Bring your counts & metadata in.")
```

de_nav_chip *de_nav_chip*

Description

B3.2 helper. Renders an HTML title for ‘bslib::nav_panel(title = ...)‘ that consists of a small numbered chip + the panel name. Matches the mockup header pattern (numbered tabs like in aidrift.geniohub.com). Optionally accepts a ‘de_progress_label()‘-style progress-pill marker so the Data Prep tab keeps its workflow checkmark behavior.

Usage

```
de_nav_chip(num, label, progress_pill = NULL)
```

Arguments

num	integer/character – the chip number (1..N)
label	character – the tab label
progress_pill	optional progress pill key (e.g. ‘"data_prep"‘)

Value

an ‘htmltools::HTML‘ blob suitable for ‘nav_panel(title = ...)‘

Examples

```
x <- de_nav_chip(1, "Data Prep")
```

de_preset_js *Tiny JS that wires the preset picker to a cookie*

Description

On change of ‘#de_preset_picker‘: writes a ‘debrowser_preset‘ cookie (1-year expiry, path /) – or deletes it when the user picks "Default" – strips any ‘?preset=‘ from the current URL, and reloads. Uses standard DOM APIs and jQuery’s delegated event binding (already in Shiny).

Usage

```
de_preset_js()
```

Value

an ‘htmltools::tag‘ (‘<script>‘).

de_preset_picker *Render the navbar preset picker*

Description

Compact native '<select>' styled with standard Bootstrap classes ('form-select form-select-sm') – no custom classes. Sits next to the dark-mode toggle in 'deUI()'. The 'change' event is wired in JS to write a 'debrowser_preset' cookie and reload (see 'de_preset_js()').

Usage

```
de_preset_picker(current = NULL)
```

Arguments

current	Currently active preset name, or NULL for default.

Value

an 'htmltools::tag' ('<select>').

de_progress_label *Render a label suitable for nav_panel(title=) or actionLink(label=) that includes a slot the JS handler can decorate with a progress icon.*

Description

Render a label suitable for nav_panel(title=) or actionLink(label=) that includes a slot the JS handler can decorate with a progress icon.

Usage

```
de_progress_label(name, key)
```

Arguments

name	character, the visible label (e.g. "Upload", "Data Prep").
key	character, the progress key the JS handler listens for.

Value

a 'tags$span' HTML wrapper.

Examples

```
de_progress_label("Upload", "upload")
```

de_stat *de_stat*

Description

B3 helper. Single stat inside a ‘de_stat_strip()‘.

Usage

```
de_stat(value, label, color = "var(--de-cyan)")
```

Arguments

value	character – the value (e.g. "6")
label	character – the trailing label (e.g. "samples")
color	CSS color – the dot color (default cyan)

Value

an ‘htmltools::tag‘ (‘‘ with a dot + value + label).

Examples

```
x <- de_stat("6", "samples", color = "var(--de-cyan)")
```

de_stat_strip *de_stat_strip*

Description

B3 helper. Renders a compact pill-shaped strip of stats (counts of samples / genes / conditions / current method) just under the headline. Each item is built with ‘de_stat()‘.

Usage

```
de_stat_strip(...)
```

Arguments

...	‘de_stat()‘ items

Value

an ‘htmltools::tag‘ (‘ ‘).

Examples

```
x <- de_stat_strip(de_stat("6", "samples"), de_stat("30,739", "genes"))
```

de_style_guide *Open the DEBrowser design-system style guide*

Description

Opens the standalone component gallery shipped at ‘inst/extdata/www/style-guide.html‘ in the default browser. Every component is rendered from ‘debrowser.css‘ in both themes, so it doubles as the client handoff artifact and the visual-regression baseline for the CSS consolidation (see ‘docs/design/06-handoff-plan.md‘ and ‘DESIGN.md‘).

Usage

```
de_style_guide(theme = c("light", "dark"))
```

Arguments

theme	one of "light" or "dark" – sets the initial ‘data-bs-theme‘ via the page’s ‘?theme=‘ query parameter.

Value

(invisibly) the path that was opened.

Examples

```
## Not run:
  de_style_guide("dark")

## End(Not run)
```

de_theme *de_theme*

Description

DEBrowser bslib theme. Default behavior: hand-rolled Slate + OK-blue palette with Inter typography on Bootstrap 5. When ‘preset‘ is supplied (e.g. ‘"zephyr"‘, ‘"lumen"‘, ‘"cosmo"‘), returns a bare Bootstrap 5 theme with only Inter font applied – preset-specific colors come from the bootswatch CDN stylesheet that ‘deUI()‘ injects separately, so each preset URL is unique and tabs don’t collide on the same compiled bootstrap.min.css. Used as the ‘theme‘ argument to ‘bslib::page_navbar()‘ in [deUI()]. The URL-param playground (‘?preset=NAME‘) wires this up.

Usage

```
de_theme(preset = NULL)
```

Arguments

preset	Optional bslib preset name. One of ‘.de_theme_presets‘. ‘NULL‘ (default) returns the standard custom theme.

Value

a ‘bs_theme‘ object

Examples

```
x <- de_theme()
y <- de_theme(preset = "zephyr")
```

---

de_workbar *de_workbar*

Description

B3 helper. Small breadcrumb / toolbar row that sits above a tab's headline. Accepts a current-tab label and optional trailing actions.

Usage

```
de_workbar(crumb, ...)
```

Arguments

crumb	character – current tab label, shown bold
...	trailing tags (e.g. ‘actionButton‘s)

Value

an ‘htmltools::tag‘ (‘ ‘).

Examples

```
x <- de_workbar("Data Prep")
```

---

distFunParamsUI *distFunParamsUI*

Description

get distance metric parameters

Usage

```
distFunParamsUI()
```

Value

funParams

Note

distFunParamsUI

Examples

```
x <- distFunParamsUI()
```

`drawKEGG` *drawKEGG*

Description

draw KEGG patwhay with expression values

Usage

```
drawKEGG(input = NULL, dat = NULL, pid = NULL)
```

Arguments

`input`	input
`dat`	expression matrix
`pid`	pathway id

Value

enriched DO

Note

`drawKEGG`

Examples

```
x <- drawKEGG()
```

`drawPCAExplained` *Creates a more detailed plot using the PCA results from the selected dataset.*

Description

Creates a more detailed plot using the PCA results from the selected dataset.

Usage

```
drawPCAExplained(explainedData = NULL)
```

Arguments

`explainedData`	selected data

Value

explained plot

Examples

```
x <- drawPCAExplained()
```

`enrichmentGmtServer` *Server for the gene-set source picker.*

Description

Returns a reactive that yields a named list of gene-symbol vectors - the same shape `gmt_to_pathways` produces - or NULL until the user has loaded a source.

Usage

```
enrichmentGmtServer(id)
```

Arguments

`id`	Module ID.

Details

Both branches populate an internal reactiveVal so the load step is observable independently of any downstream consumer (the previous eventReactive design was lazy and didn't fire until Submit). A small status output reports load success / failure / set count next to the picker.

Value

Reactive expression yielding the parsed pathways list.

Examples

```
shiny::shinyApp(
  ui = enrichmentGmtUI("gmt"),
  server = function(input, output, session) {
    enrichmentGmtServer("gmt")
  }
)
```

enrichmentGmtUI *UI for the gene-set source picker (Enrichment tab sidebar).*

Description

Two sources: manual '.gmt' upload (auto-loads on file pick) and MSigDB (loads on explicit "Load gene sets" click). A status line below the picker reports how many gene sets are currently loaded so the user has immediate feedback before pressing Submit.

Usage

```
enrichmentGmtUI(id)
```

Arguments

id	Module ID.

Value

Shiny tagList for inclusion in a 'bslib::accordion_panel'.

Examples

```
enrichmentGmtUI("demo")
```

enrichmentNesHeatmapServer *Server for the NES heatmap card.*

Description

Server for the NES heatmap card.

Usage

```
enrichmentNesHeatmapServer(id, results_by_comparison)
```

Arguments

id	Module ID.
results_by_comparison	Reactive yielding a named list of run_gsea outputs (one per comparison).

Value

invisible(NULL).

Examples

```
shiny::shinyApp(
  ui = enrichmentNesHeatmapUI("nes"),
  server = function(input, output, session) {
    enrichmentNesHeatmapServer("nes", shiny::reactive(list()))
  }
)
```

`enrichmentNesHeatmapUI` *UI for the NES heatmap card.*

Description

Renders only when [enrichmentServer] passes more than one comparison's GSEA results.

Usage

```
enrichmentNesHeatmapUI(id)
```

Arguments

`id`	Module ID.

Value

A 'bslib::card'.

Examples

```
enrichmentNesHeatmapUI("demo")
```

`enrichmentServer` *Server for the Enrichment tab.*

Description

Server for the Enrichment tab.

Usage

```
enrichmentServer(id, de_results)
```

Arguments

`id`	Module ID.
`de_results`	Reactive yielding either: - a single data.frame with columns 'ID' (or 'gene'), 'log2FoldChange', 'padj' (one comparison), or - a named list of such data.frames (multi-comparison; enables the NES heatmap).

Value

invisible(NULL).

Examples

```
de <- data.frame(ID = paste0("G", 1:5),
                 log2FoldChange = c(2, -1, 0, 3, -2),
                 padj = c(0.01, 0.04, 0.5, 0.02, 0.03))
shiny::shinyApp(
  ui = enrichmentUI("e1"),
  server = function(input, output, session) {
    enrichmentServer("e1", shiny::reactive(de))
  }
)
```

---

enrichmentUI *UI for the Enrichment tab.*

Description

Lives next to the existing GO Term tab. E2 will fold GO Term into here as the ORA mode; E11 adds the "Compare methods" mode.

Usage

```
enrichmentUI(id)
```

Arguments

id	Module ID.

Value

A ‘bslib::layout_sidebar‘ tagList.

Examples

```
enrichmentUI("demo")
```

`exportMenuItems` *Export menu items (without nav_menu wrapper) for embedding in the account dropdown. Returns a list of '<li>'s, each carrying either a shiny-download-link (downloads) or a 'data-debrowser-input' action link (modal opens). IDs are namespaced under 'id' so the existing 'exportMenuServer(id, ...)' observers/handlers wire up unchanged.*

Description

Export menu items (without nav_menu wrapper) for embedding in the account dropdown. Returns a list of '<li>'s, each carrying either a shiny-download-link (downloads) or a 'data-debrowser-input' action link (modal opens). IDs are namespaced under 'id' so the existing 'exportMenuServer(id, ...)' observers/handlers wire up unchanged.

Usage

```
exportMenuItems(id)
```

Arguments

`id`	Module ID (matches the one passed to 'exportMenuServer()').

Value

list of 'shiny::tags$li' elements.

`exportMenuServer` *Export menu server – wires download handlers from a state reactive.*

Description

Export menu server – wires download handlers from a state reactive.

Usage

```
exportMenuServer(id, state_react)
```

Arguments

`id`	Module ID (matches [exportMenuUI()]).
`state_react`	reactive expression returning the plain-list state snapshot consumed by [build_session_blocks()]. May return NULL when no DE has run yet; both download handlers no-op (showNotification) in that case.

Value

Invisibly NULL.

Examples

```
shiny::shinyApp(
  ui = bslib::page_navbar(exportMenuUI("exp")),
  server = function(input, output, session) {
    exportMenuServer("exp", shiny::reactive(NULL))
  }
)
```

`exportMenuUI` *Export menu UI – navbar dropdown.*

Description

Mounted in the page_navbar after ‘nav_spacer()‘. Six items: - "R script" downloads a runnable .R reproducibility script - "Rmd source" (E3.B) downloads the raw .Rmd body, no render - "HTML" renders an .Rmd to HTML (gated on rmarkdown) - "View HTML in tab" (E3.B) renders + opens in a new browser tab - "Jupyter notebook" (E3.B) downloads the same content as .ipynb with R-kernel code cells - "Copy methods text" opens a modal with a manuscript-ready paragraph (E9) plus a "Download as .txt" button

Usage

```
exportMenuUI(id)
```

Arguments

`id`	Module ID.

Details

Items are disabled at the server level when DE has not yet been run; the UI emits the disabled-attribute via output bindings.

Value

bslib::nav_menu element.

Examples

```
exportMenuUI("demo")
```

`fct_upload` *Pure helpers for the Quick Start wizard (B2b).*

Description

These are testthat-tested in isolation. The file has no ‘library()‘ calls and no Shiny references – keep it that way.

fileTypes *fileTypes*

Description

Returns fileTypes that are going to be used in creating fileUpload UI

Usage

```
fileTypes()
```

Value

file types

Note

fileTypes

Examples

```
x <- fileTypes()
```

fileUploadBox *fileUploadBox*

Description

File upload module

Usage

```
fileUploadBox(id = NULL, inputId = NULL, label = NULL, helper = NULL)
```

Arguments

id	namespace id
inputId	input file ID
label	label
helper	optional help-text string shown beneath the card title

Value

radio control

Note

fileUploadBox

Examples

```
x <- fileUploadBox("meta", "metadata", "Metadata")
```

`filter_low_counts` *Filter low-count rows from a count matrix.*

Description

Filter low-count rows from a count matrix.

Usage

```
filter_low_counts(counts, method = "max", cutoff = 10, min_samples = NULL)
```

Arguments

`counts`	Numeric matrix (genes x samples).
`method`	One of "max", "mean", "cpm".
`cutoff`	Numeric threshold; meaning depends on ‘method‘.
`min_samples`	For method="cpm": the row is kept if CPM > cutoff in at least ‘min_samples‘ samples. Defaults to ‘ncol(counts) - 1‘.

Value

Filtered count matrix (rows preserved by row order).

Examples

```
m <- matrix(c(1, 50, 100, 5, 80, 120,
              1,  2,   3, 1,  2,   3),
            nrow = 2, byrow = TRUE)
colnames(m) <- paste0("S", 1:6)
rownames(m) <- c("GeneA", "GeneB")
filter_low_counts(m, method = "max", cutoff = 10)
```

`filter_params_from_input`
Translate a Shiny ‘input‘ reactive into a structured filter-params list.

Description

All A3b pure functions accept a named list of filter parameters. This helper bridges Shiny modules to those functions in a single place so the field-name mapping is documented and centralised.

Usage

```
filter_params_from_input(input)
```

Arguments

input	A Shiny input reactive (or a plain list with the same fields).

Value

Named list with components: padj_cutoff, fold_cutoff, dataset, compselect, norm_method, geneset_area, method_tab, min_count, top_n, selected_plot.

Examples

```
fake_input <- list(
  padj = 0.05, log2fc_cutoff = 1, dataset = "up+down",
  compselect = 1, norm_method = "TMM", genesetarea = "",
  methodtabs = "panel1", mincount = 10, topn = 500,
  selectedplot = NULL
)
filter_params_from_input(fake_input)
```

---

flag_outliers_2sd *Flag values further than 2 standard deviations from the mean.*

Description

Pure helper used by library_depth_summary. NA values map to FALSE in the output. Returns all FALSE if ‘length(x) < 2‘ or if the standard deviation is zero.

Usage

```
flag_outliers_2sd(x)
```

Arguments

x	Numeric vector.

Value

Logical vector the same length as ‘x‘. TRUE iff ‘abs(x - mean(x, na.rm = TRUE)) > 2 * sd(x, na.rm = TRUE)‘.

Examples

```
flag_outliers_2sd(c(1, 1, 1, 1, 1, 10))
flag_outliers_2sd(c(1, 1, 1, 1))          # zero variance -> all FALSE
flag_outliers_2sd(c(NA, 1, 1, 1, 10))     # NA position -> FALSE
```

fold_to_log2fc *Convert fold-change cutoff to |log2FC|.*

Description

Convert fold-change cutoff to |log2FC|.

Usage

```
fold_to_log2fc(x)
```

Arguments

x	Numeric fold-change cutoff.

Value

|log2FC| cutoff (log2(x)).

Examples

```
fold_to_log2fc(2)   # 1
fold_to_log2fc(4)   # 2
```

generateTestData *generateTestData*

Description

This generates a test data that is suitable to main plots in debrowser

Usage

```
generateTestData(dat = NULL)
```

Arguments

dat	DESeq results will be generated for loaded data

Value

testData

Examples

```
x <- generateTestData()
```

getAfterLoadMsg *getAfterLoadMsg*

Description

Generates and displays the message to be shown after loading data within the DEBrowser.

Usage

```
getAfterLoadMsg()
```

Value

return After Load Msg

Note

getAfterLoadMsg

Examples

```
x <- getAfterLoadMsg()
```

getAll2AllPlotUI *getAll2AllPlotUI*

Description

all2all plots UI.

Usage

```
getAll2AllPlotUI(id)
```

Arguments

id	namespace id

Value

A ‘shiny::uiOutput‘ placeholder under namespace ‘id‘ that the all2all module’s server fills with the all-pairs scatter matrix plot.

Note

getAll2AllPlotUI

Examples

```
x <- getAll2AllPlotUI("bar")
```

getBarMainPlot *getBarMainPlot*

Description

Makes Density plots

Usage

```
getBarMainPlot(
  data = NULL,
  cols = NULL,
  conds = NULL,
  cond_names = NULL,
  key = NULL,
  title = "",
  input = NULL
)
```

Arguments

data	count or normalized data
cols	cols
conds	conds
cond_names	condition names
key	key
title	title
input	input

Value

A ‘plotly‘ bar plot for the selected gene/region across samples.

Examples

```
getBarMainPlot()
```

getBarMainPlotUI *getBarMainPlotUI*

Description

main bar plots UI.

Usage

```
getBarMainPlotUI(id)
```

Arguments

id	namespace id

Value

the panel for Density plots;

Note

getBarMainPlotUI

Examples

```
x <- getBarMainPlotUI("bar")
```

---

getBoxMainPlot *getBoxMainPlot*

Description

Makes Density plots

Usage

```
getBoxMainPlot(
  data = NULL,
  cols = NULL,
  conds = NULL,
  cond_names = NULL,
  key = NULL,
  title = "",
  input = NULL
)
```

Arguments

data	count or normalized data
cols	cols
conds	conds
cond_names	condition names
key	key
title	title
input	input

Value

A ‘plotly‘ box plot for the selected gene/region across samples.

Examples

```
getBoxMainPlot()
```

`getBoxMainPlotUI` *getBoxMainPlotUI*

Description

main Box plots UI.

Usage

```
getBoxMainPlotUI(id)
```

Arguments

id	namespace id

Value

the panel for Density plots;

Note

`getBoxMainPlotUI`

Examples

```
x <- getBoxMainPlotUI("box")
```

`getBSTableUI` *getBSTableUI prepares a Modal to put a table*

Description

getBSTableUI prepares a Modal to put a table

Usage

```
getBSTableUI(
  name = NULL,
  label = NULL,
  trigger = NULL,
  size = "large",
  modal = NULL
)
```

Arguments

name	name
label	label
trigger	trigger button for the modal
size	size of the modal
modal	modal yes/no

Value

the modal

Examples

```
x <- getBSTableUI()
```

---

getColors *getColors*

Description

get colors for the domains

Usage

```
getColors(domains = NULL)
```

Arguments

domains	domains to be colored

Value

colors

Examples

```
x <- getColors()
```

getColorShapeSelection *getColorShapeSelection*

Description

Generates the fill and shape selection boxes for PCA plots. metadata file has to be loaded in this case

Usage

```
getColorShapeSelection(metadata = NULL, input = NULL, session = NULL)
```

Arguments

metadata	metadata table
input	input
session	session

Value

Color and shape selection boxes

Examples

```
x <- getColorShapeSelection()
```

getCompSelection *getCompSelection*

Description

Gathers the user selected comparison set to be used within the DEBrowser.

Usage

```
getCompSelection(name = NULL, count = NULL)
```

Arguments

name	the name of the selectInput
count	comparison count

Value

A 'shiny::selectInput' (or NULL when only one comparison exists) listing the available comparison indices.

Note

getCompSelection

Examples

```
x <- getCompSelection(name = "comp", count = 2)
```

---

getCondMsg *getCondMsg*

Description

Generates and displays the current conditions and their samples within the DEBrowser.

Usage

```
getCondMsg(dc = NULL, input = NULL, cols = NULL, conds = NULL)
```

Arguments

dc	columns
input	selected comparison
cols	columns
conds	selected conditions

Value

return conditions

Note

getCondMsg

Examples

```
x <- getCondMsg()
```

getCutOffSelection *getCutOffSelection*

Description

Gathers the cut off selection for DE analysis

Usage

```
getCutOffSelection(nc = 1)
```

Arguments

nc	total number of comparisons

Value

returns the left menu according to the selected tab;

Note

`getCutOffSelection`

Examples

```
x <- getCutOffSelection()
```

getDataAssesmentText *getDataAssesmentText DataAssesment text*

Description

getDataAssesmentText DataAssesment text

Usage

```
getDataAssesmentText()
```

Value

help text for data assesment

Examples

```
x <- getDataAssesmentText()
```

`getDataForTables` *getDataForTables get data to fill up tables tab*

Description

getDataForTables get data to fill up tables tab

Usage

```
getDataForTables(
  input = NULL,
  init_data = NULL,
  filt_data = NULL,
  selected = NULL,
  getMostVaried = NULL,
  mergedComp = NULL,
  explainedData = NULL
)
```

Arguments

`input`	input parameters
`init_data`	initial dataset
`filt_data`	filt_data
`selected`	selected genes
`getMostVaried`	most varied genes
`mergedComp`	merged comparison set
`explainedData`	pca gene set

Value

data

Examples

```
x <- getDataForTables()
```

`getDataPreparationText` *getDataPreparationText DataPreparation text*

Description

getDataPreparationText DataPreparation text

Usage

```
getDataPreparationText()
```

Value

help text for data preparation

Examples

```
x <- getDataPreparationText()
```

---

getDEAnalysisText *getDEAnalysisText DEAnalysis text*

Description

getDEAnalysisText DEAnalysis text

Usage

```
getDEAnalysisText()
```

Value

help text for DE Analysis

Examples

```
x <- getDEAnalysisText()
```

---

getDensityPlot *getDensityPlot*

Description

Makes Density plots

Usage

```
getDensityPlot(data = NULL, input = NULL, title = "")
```

Arguments

data	count or normalized data
input	input
title	title

Value

A ‘plotly‘ density plot showing the per-sample distributions of the input count matrix (one density curve per sample).

Examples

```
getDensityPlot()
```

`getDensityPlotUI` *getDensityPlotUI*

Description

Density plot UI.

Usage

```
getDensityPlotUI(id)
```

Arguments

`id`	namespace id

Value

the panel for Density plots;

Note

`getDensityPlotUI`

Examples

```
x <- getDensityPlotUI("density")
```

`getDEResultsUI` *getDEResultsUI Creates a panel to visualize DE results*

Description

getDEResultsUI Creates a panel to visualize DE results

Usage

```
getDEResultsUI(id)
```

Arguments

`id`	namespace id

Value

panel

Examples

```
x <- getDEResultsUI("batcheffect")
```

`getDomains` *getDomains*

Description

Get domains for the main plots.

Usage

```
getDomains(filt_data = NULL)
```

Arguments

`filt_data`	data to get the domains

Value

domains

Examples

```
x <- getDomains()
```

`getDown` *getDown get down regulated data*

Description

getDown get down regulated data

Usage

```
getDown(filt_data = NULL)
```

Arguments

`filt_data`	filt_data

Value

data

Examples

```
x <- getDown()
```

getDownloadSection *getDownloadSection*

Description

download section button and dataset selection box in the menu for user to download selected data.

Usage

```
getDownloadSection(choices = NULL)
```

Arguments

choices	main vs. QC section

Value

the panel for download section in the menu;

Note

getDownloadSection

Examples

```
x <- getDownloadSection()
```

getEnrichDO *getEnrichDO*

Description

Gathers the Enriched DO Term analysis data to be used within the GO Term plots.

Usage

```
getEnrichDO(genelist = NULL, pvalueCutoff = 0.01)
```

Arguments

genelist	gene list
pvalueCutoff	the p value cutoff

Value

enriched DO

Note

getEnrichDO

Examples

```
x <- getEnrichDO()
```

---

getEnrichGO *getEnrichGO*

Description

Gathers the Enriched GO Term analysis data to be used within the GO Term plots.

Usage

```
getEnrichGO(
  genelist = NULL,
  pvalueCutoff = 0.01,
  org = "org.Hs.eg.db",
  ont = "CC"
)
```

Arguments

genelist	gene list
pvalueCutoff	p value cutoff
org	the organism used
ont	the ontology used

Value

Enriched GO

Note

getEnrichGO

Examples

```
x <- getEnrichGO()
```

getEnrichKEGG *getEnrichKEGG*

Description

Gathers the Enriched KEGG analysis data to be used within the GO Term plots.

Usage

```
getEnrichKEGG(genelist = NULL, pvalueCutoff = 0.01, org = "org.Hs.eg.db")
```

Arguments

genelist	gene list
pvalueCutoff	the p value cutoff
org	the organism used

Value

Enriched KEGG

Note

getEnrichKEGG

Examples

```
x <- getEnrichKEGG()
```

getEntrezIds *getEntrezIds*

Description

Gathers the gene list to use for GOTerm analysis.

Usage

```
getEntrezIds(genes = NULL, org = "org.Hs.eg.db")
```

Arguments

genes	gene list with fold changes
org	orgranism for gene symbol entrez ID conversion

Value

ENTREZ ID list

Note

GOTerm

getEntrezIds symobol to ENTREZ ID conversion

Examples

```
x <- getEntrezIds()
```

---

getEntrezTable *getEntrezTable*

Description

Gathers the entrezIds of the genes in given list and their data

Usage

```
getEntrezTable(genes = NULL, dat = NULL, org = "org.Hs.eg.db")
```

Arguments

genes	gene list
dat	data matrix
org	orgranism for gene symbol entrez ID conversion

Value

table with the entrez IDs in the rownames

Note

GOTerm

getEntrezTable symobol to ENTREZ ID conversion

Examples

```
x <- getEntrezTable()
```

getGeneList *getGeneList*

Description

Gathers the gene list to use for GOTerm analysis.

Usage

```
getGeneList(
  genes = NULL,
  org = "org.Hs.eg.db",
  fromType = "SYMBOL",
  toType = c("ENTREZID")
)
```

Arguments

genes	gene list
org	orgranism for gene symbol entrez ID conversion
fromType	from Type
toType	to Type

Value

ENTREZ ID list

Note

GOTerm

getGeneList symobol to ENTREZ ID conversion

Examples

```
x <- getGeneList(c("OCLN", "ABCC2"))
```

getGeneSetData *getGeneSetData*

Description

Gathers the specified gene set list to be used within the DEBrowser.

Usage

```
getGeneSetData(data = NULL, geneset = NULL)
```

Arguments

data	loaded dataset
geneset	given gene set

Value

data

Examples

```
x <- getGeneSetData()
```

---

getGOLeftMenu *getGOLeftMenu*

Description

Generates the GO Left menu to be displayed within the DEBrowser.

Usage

```
getGOLeftMenu()
```

Value

returns the left menu according to the selected tab;

Note

getGOLeftMenu

Examples

```
x <- getGOLeftMenu()
```

---

getGoPanel *getGoPanel*

Description

Creates go term analysis panel within the shiny display.

Usage

```
getGoPanel()
```

Value

the panel for go term analysis;

Note

getGoPanel

Examples

```
x <- getGoPanel()
```

---

getGOPlots *getGOPlots*

Description

Go term analysis panel. Generates appropriate GO plot based on user selection.

Usage

```
getGOPlots(dataset = NULL, GSEARes = NULL, input = NULL)
```

Arguments

dataset	the dataset used
GSEARes	GSEA results
input	input params

Value

the panel for go plots;

Note

getGOPlots

Examples

```
x <- getGOPlots()
```

getGSEA *getGSEA*

Description

Gathers the Enriched KEGG analysis data to be used within the GO Term plots.

Usage

```
getGSEA(
  dataset = NULL,
  pvalueCutoff = 0.01,
  org = "org.Hs.eg.db",
  sortfield = "log2FoldChange"
)
```

Arguments

dataset	dataset
pvalueCutoff	the p value cutoff
org	the organism used
sortfield	sort field for GSEA

Value

GSEA

Note

getGSEA

Examples

```
x <- getGSEA()
```

getHeatmapUI *getHeatmapUI*

Description

Generates the left menu to be used for heatmap plots

Usage

```
getHeatmapUI(id)
```

Arguments

id	module ID

Value

heatmap plot area

Note

getHeatmapUI

Examples

```
x <- getHeatmapUI("heatmap")
```

---

getHelpButton *getHelpButton prepares a helpbutton for to go to a specific site in the documentation*

Description

getHelpButton prepares a helpbutton for to go to a specific site in the documentation

Usage

```
getHelpButton(name = NULL, link = NULL)
```

Arguments

name	name that are going to come after info
link	link of the help

Value

the info button

Examples

```
x <- getHelpButton()
```

---

getHideLegendOnOff *getHideLegendOnOff*

Description

hide legend

Usage

```
getHideLegendOnOff(id = "pca")
```

Arguments

id	namespace id

Value

A ‘shiny::radioButtons‘ toggling whether the PCA plot legend is rendered or hidden.

Examples

```
x <- getHideLegendOnOff("pca")
```

---

getHistogramUI *getHistogramUI*

Description

Histogram plots UI.

Usage

```
getHistogramUI(id)
```

Arguments

id	namespace id

Value

the panel for PCA plots;

Note

getHistogramUI

Examples

```
x <- getHistogramUI("histogram")
```

---

getIntroText *getIntroText Intro text*

Description

getIntroText Intro text

Usage

```
getIntroText()
```

Value

the JS for tab updates

Examples

```
x <- getIntroText()
```

getIQRPlot *getIQRPlot*

Description

Makes IQR boxplot plot

Usage

```
getIQRPlot(data = NULL, input = NULL, title = "")
```

Arguments

data	count or normalized data
input	input
title	title

Value

A ‘plotly‘ IQR boxplot showing the per-sample inter-quartile spread of the input data.

Examples

```
getIQRPlot()
```

getIQRPlotUI *getIQRPlotUI*

Description

IQR plot UI.

Usage

```
getIQRPlotUI(id)
```

Arguments

id	namespace id

Value

the panel for IQR plots;

Note

getIQRPlotUI

Examples

```
x <- getIQRPlotUI("IQR")
```

getJSLine *getJSLine*

Description

heatmap JS code for selection functionality

Usage

```
getJSLine()
```

Value

JS Code

Examples

```
x <- getJSLine()
```

getKEGGModal *getKEGGModal prepares a modal for KEGG plots*

Description

getKEGGModal prepares a modal for KEGG plots

Usage

```
getKEGGModal()
```

Value

the info button

Examples

```
x <- getKEGGModal()
```

getLeftMenu *getLeftMenu*

Description

Generates the left menu for for plots within the DEBrowser.

Usage

```
getLeftMenu(input = NULL)
```

Arguments

input	input values

Value

returns the left menu according to the selected tab;

Note

getLeftMenu

Examples

```
x <- getLeftMenu()
```

getLegendColors *getLegendColors*

Description

Generates colors according to the data

Usage

```
getLegendColors(Legend = c("up", "down", "NS"))
```

Arguments

Legend	unique Legends

Value

mainPlotControls

Note

getLegendColors

Examples

```
x <- getLegendColors(c("up", "down", "GS", "NS"))
```

getLegendRadio *getLegendRadio*

Description

Radio buttons for the types in the legend

Usage

```
getLegendRadio(id)
```

Arguments

id	namespace id

Value

radio control

Note

getLegendRadio

Examples

```
x <- getLegendRadio("deprog")
```

getLegendSelect *getLegendSelect*

Description

select legend

Usage

```
getLegendSelect(id = "pca")
```

Arguments

id	namespace id

Value

A ‘shiny::selectInput‘ letting the user pick whether the PCA plot’s legend tracks the color or shape mapping.

Note

getLegendSelect

Examples

```
x <- getLegendSelect("pca")
```

`getLevelOrder` *getLevelOrder*

Description

Generates the order of the overlapping points

Usage

```
getLevelOrder(Level = c("up", "down", "NS"))
```

Arguments

`Level`	factor levels shown in the legend

Value

order

Note

`getLevelOrder`

Examples

```
x <- getLevelOrder(c("up", "down", "GS", "NS"))
```

`getLogo` *getLogo*

Description

Generates and displays the logo to be shown within DEBrowser.

Usage

```
getLogo()
```

Value

return logo

Note

`getLogo`

Examples

```
x <- getLogo()
```

getMainPanel *getMainPanel*

Description

main panel for volcano, scatter and maplot. Barplot and box plots are in this page as well.

Usage

```
getMainPanel()
```

Value

the panel for main plots;

Note

getMainPanel

Examples

```
x <- getMainPanel()
```

getMainPlotsLeftMenu *getMainPlotsLeftMenu*

Description

Generates the Main PLots Left menu to be displayed within the DEBrowser.

Usage

```
getMainPlotsLeftMenu()
```

Value

returns the left menu according to the selected tab;

Note

getMainPlotsLeftMenu

Examples

```
x <- getMainPlotsLeftMenu()
```

getMainPlotUI *getMainPlotUI*

Description

main plot for volcano, scatter and maplot.

Usage

```
getMainPlotUI(id)
```

Arguments

id	namespace id

Value

the panel for main plots;

Note

getMainPlotUI

Examples

```
x <- getMainPlotUI("main")
```

getMean *getMean*

Description

Gathers the mean for selected condition.

Usage

```
getMean(data = NULL, selcols = NULL)
```

Arguments

data	dataset
selcols	input cols

Value

data

Examples

```
x <- getMean()
```

getMergedComparison *getMergedComparison*

Description

Gathers the merged comparison data to be used within the DEBrowser.

Usage

```
getMergedComparison(dc = NULL, nc = NULL, input = NULL)
```

Arguments

dc	data container
nc	the number of comparisons
input	input params

Value

data

Examples

```
x <- getMergedComparison()
```

getMostVariedList *getMostVariedList*

Description

Calculates the most varied genes to be used for specific plots within the DEBrowser.

Usage

```
getMostVariedList(datavar = NULL, cols = NULL, input = NULL)
```

Arguments

datavar	loaded dataset
cols	selected columns
input	input

Value

data

Examples

```
x <- getMostVariedList()
```

getNormalizedMatrix *getNormalizedMatrix*

Description

Normalizes the matrix passed to be used within various methods within DEBrowser. Requires edgeR package

Usage

```
getNormalizedMatrix(M = NULL, method = "TMM")
```

Arguments

M	numeric matrix
method	normalization method for edgeR. default is TMM

Value

normalized matrix

Note

getNormalizedMatrix

Examples

```
x <- getNormalizedMatrix(mtcars)
```

getOrganism *getOrganism*

Description

getOrganism

Usage

```
getOrganism(org)
```

Arguments

org	organism

Value

organism name for keg

Note

getOrganism

Examples

```
x <- getOrganism()
```

---

getOrganismBox *getOrganismBox*

Description

Get the organism Box.

Usage

```
getOrganismBox()
```

Value

selectInput

Note

getOrganismBox

getOrganismBox makes the organism box

Examples

```
x <- getOrganismBox()
```

---

getOrganismPathway *getOrganismPathway*

Description

getOrganismPathway

Usage

```
getOrganismPathway(org)
```

Arguments

org	organism

Value

organism name for pathway

Note

getOrganismPathway

Examples

```
x <- getOrganismPathway()
```

---

getPCAcontolUpdatesJS *getPCAcontolUpdatesJS in the prep menu we have two PCA plots to show how batch effect correction worked. One set of PCA input controls updates two PCA plots with this JS.*

Description

getPCAcontolUpdatesJS in the prep menu we have two PCA plots to show how batch effect correction worked. One set of PCA input controls updates two PCA plots with this JS.

Usage

```
getPCAcontolUpdatesJS()
```

Value

the JS for tab updates

Examples

```
x <- getTabUpdateJS()
```

---

getPCAexplained *getPCAexplained*

Description

Creates a more detailed plot using the PCA results from the selected dataset.

Usage

```
getPCAexplained(datasetInput = NULL, pca_data = NULL, input = NULL)
```

Arguments

datasetInput	selected data
pca_data	from user
input	input params

Value

explained plot

Examples

```
load(system.file("extdata", "demo", "demodata.Rda", package = "debrowser"))
input <- c()
input$qcplot <- "pca"
input$col_list <- colnames(demodata[, seq_len(6L)])
dat <- getNormalizedMatrix(demodata[, seq_len(6L)])
pca_data <- run_pca(dat)
x <- getPCAexplained(dat, pca_data, input)
```

---

getPCAPlotUI *getPCAPlotUI*

Description

PCA plots UI.

Usage

```
getPCAPlotUI(id)
```

Arguments

id	namespace id

Value

the panel for PCA plots;

Note

getPCAPlotUI

Examples

```
x <- getPCAPlotUI("pca")
```

getPCselection *getPCselection*

Description

Generates the PC selection number to be used within DEBrowser.

Usage

```
getPCselection(id, num = 1, xy = "x")
```

Arguments

id	namespace id
num	PC selection number
xy	x or y coordinate

Value

PC selection for PCA analysis

Note

getPCselection

Examples

```
x <- getPCselection("pca")
```

getPlotArea *getPlotArea*

Description

returns plot area either for heatmaply or heatmap.2

Usage

```
getPlotArea(input = NULL, session = NULL)
```

Arguments

input	input variables
session	session

Value

heatmapply/heatmap.2 plot area

Examples

```
x <- getPlotArea()
```

`getProgramTitle` *getProgramTitle*

Description

Generates the title of the program to be displayed within DEBrowser. If it is called in a program, the program title will be hidden

Usage

```
getProgramTitle(session = NULL)
```

Arguments

`session`	session var

Value

program title

Note

`getProgramTitle`

Examples

```
title <- getProgramTitle()
```

`getQAText` *getQAText Some questions and answers*

Description

getQAText Some questions and answers

Usage

```
getQAText()
```

Value

help text for QA

Examples

```
x <- getQAText()
```

getQCLeftMenu *getQCLeftMenu*

Description

Generates the left menu to be used for QC plots within the DEBrowser.

Usage

```
getQCLeftMenu(input = NULL)
```

Arguments

input	input values

Value

QC left menu

Note

getQCLeftMenu

Examples

```
x <- getQCLeftMenu()
```

getQCPanel *getQCPanel*

Description

Gathers the conditional panel for QC plots

Usage

```
getQCPanel(input = NULL)
```

Arguments

input	user input

Value

the panel for QC plots

Note

getQCSection

Examples

```
x <- getQCPanel()
```

`getSampleDetails` *getSampleDetails*

Description

get sample details

Usage

```
getSampleDetails(output = NULL, summary = NULL, details = NULL, data = NULL)
```

Arguments

`output`	output
`summary`	summary output name
`details`	details ouput name
`data`	data

Value

panel

Examples

```
x <- getSampleDetails()
```

`getSearchData` *getSearchData*

Description

search the geneset in the tables and return it

Usage

```
getSearchData(dat = NULL, input = NULL)
```

Arguments

`dat`	table data
`input`	input params

Value

data

Examples

```
x <- getSearchData()
```

`getSelectedCols` *getSelectedCols*

Description

gets selected columns

Usage

```
getSelectedCols(data = NULL, datasetInput = NULL, input = NULL)
```

Arguments

`data`	all loaded data
`datasetInput`	selected dataset
`input`	user input params

Value

A subset of ‘data‘ (rows of ‘datasetInput‘, columns selected in ‘input$col_list‘), or NULL when neither input is provided.

Examples

```
getSelectedCols()
```

`getSelectedDatasetInput` *getSelectedDatasetInput*

Description

Gathers the user selected dataset output to be displayed.

Usage

```
getSelectedDatasetInput(
  rdata = NULL,
  getSelected = NULL,
  getMostVaried = NULL,
  mergedComparison = NULL,
  input = NULL
)
```

Arguments

rdata	filtered dataset
getSelected	selected data
getMostVaried	most varied data
mergedComparison	merged comparison data
input	input parameters

Value

data

Examples

```
x <- getSelectedDatasetInput()
```

---

getSelHeat *getSelHeat*

Description

heatmap selection functionality

Usage

```
getSelHeat(expdata = NULL, input = NULL)
```

Arguments

expdata	selected genes
input	input params

Value

plot

Examples

```
x <- getSelHeat()
```

getShapeColor *getShapeColor*

Description

Generates the fill and shape selection boxes for PCA plots. metadata file has to be loaded in this case

Usage

```
getShapeColor(input = NULL)
```

Arguments

input	input values

Value

Color and shape from selection boxes or defaults

Examples

```
x <- getShapeColor()
```

getStartPlotsMsg *getStartPlotsMsg*

Description

Generates and displays the starting messgae to be shown once the user has first seen the main plots page within DEBrowser.

Usage

```
getStartPlotsMsg()
```

Value

return start plot msg

Note

getStartPlotsMsg

Examples

```
x <- getStartPlotsMsg()
```

getStartupMsg *getStartupMsg*

Description

Generates and displays the starting message within DEBrowser.

Usage

```
getStartupMsg()
```

Value

return startup msg

Note

getStartupMsg

Examples

```
x <- getStartupMsg()
```

getTableDetails *getTableDetails*

Description

get table details To be able to put a table into two lines are necessary; into the server part; getTableDetails(output, session, "dataname", data, modal=TRUE) into the ui part; uiOutput(ns("dataname"))

Usage

```
getTableDetails(
  output = NULL,
  session = NULL,
  tablename = NULL,
  data = NULL,
  modal = NULL
)
```

Arguments

output	output
session	session
tablename	table name
data	matrix data
modal	if it is true, the matrix is going to be in a modal

Value

panel

Examples

```
x <- getTableDetails()
```

---

getTableModal *getTableModal prepares table modal for KEGG*

Description

getTableModal prepares table modal for KEGG

Usage

```
getTableModal()
```

Value

the info button

Examples

```
x <- getTableModal()
```

---

getTableStyle *getTableStyle*

Description

User defined selection that selects the style of table to display within the DEBrowser.

Usage

```
getTableStyle(
  dat = NULL,
  input = NULL,
  padj = c("padj"),
  foldChange = c("foldChange"),
  DEsection = TRUE
)
```

Arguments

dat	dataset
input	input params
padj	the name of the padj value column in the dataset
foldChange	the name of the foldChange column in the dataset
DEsection	if it is in DESection or not

Value

A 'DT::datatable' HTML widget with row colouring driven by the supplied padj / log2FoldChange thresholds.

Note

getTableStyle

Examples

```
x <- getTableStyle()
```

---

getTabUpdateJS *getTabUpdateJS*

Description

Returns a '<script>' tag that installs a Shiny custom-message handler for type "debrowser-progress". The handler decorates wizard progress icons ('.de-progress-icon[data-progress-key=...]') and parent pills ('a[data-progress-pill=...]') with done/locked/skipped CSS classes. Producer side: 'update_progress(session, key, state)' in R/de_progress.R.

Usage

```
getTabUpdateJS()
```

Details

Name retained for export-compatibility; historically this tag drove wizard tab updates, but progressive reveal moved to server-side observers in B1 and progress decoration is the current responsibility.

Value

a '<script>' tag with the custom-message handler installed.

Examples

```
x <- getTabUpdateJS()
```

getUp *getUp get up regulated data*

Description

getUp get up regulated data

Usage

```
getUp(filt_data = NULL)
```

Arguments

filt_data	filt_data

Value

data

Examples

```
x <- getUp()
```

getUpDown *getUpDown get up+down regulated data*

Description

getUpDown get up+down regulated data

Usage

```
getUpDown(filt_data = NULL)
```

Arguments

filt_data	filt_data

Value

data

Examples

```
x <- getUpDown()
```

getVariationData *getVariationData*

Description

Adds an id to the data frame being used.

Usage

```
getVariationData(inputdata = NULL, cols = NULL, conds = NULL, key = NULL)
```

Arguments

inputdata	dataset
cols	columns
conds	conditions
key	gene or region name

Value

plotdata

Examples

```
x <- getVariationData()
```

get_most_varied *Compute the most-varied genes by coefficient of variation.*

Description

Compute the most-varied genes by coefficient of variation.

Usage

```
get_most_varied(datavar, cols, params = list())
```

Arguments

datavar	data.frame with sample columns.
cols	Character vector of sample column names.
params	Named list with ‘top_n‘ (int) and ‘min_count‘ (int).

Value

data.frame of the top-N most-varied rows.

Examples

```
cols <- paste0("S", 1:6)
df <- as.data.frame(matrix(
  c(100, 200, 150, 80, 250, 130,
    40,  60,  55, 30,  70,  45,
    10,  10,  10, 10,  10,  10),
  nrow = 3, byrow = TRUE,
  dimnames = list(paste0("Gene", 1:3), cols)
))
get_most_varied(df, cols, list(top_n = 2, min_count = 0))
```

`get_table_data` *Build the (data, padj_colname, fold_colname) tuple for the Tables tab.*

Description

Pure version of ‘getDataForTables()‘.

Usage

```
get_table_data(
  init_data = NULL,
  filt_data = NULL,
  selected = NULL,
  get_most_varied_data = NULL,
  merged_comp = NULL,
  explained_data = NULL,
  params = list()
)
```

Arguments

`init_data`	Initial DE result.
`filt_data`	Filtered DE result; defaults to ‘init_data‘ if NULL.
`selected`	Genes the user lasso-selected (used when ‘dataset == "selected"‘).
`get_most_varied_data`	Most-varied subset (used when ‘dataset == "most-varied"‘).
`merged_comp`	Merged comparisons table.
`explained_data`	Unused; preserved for legacy signature parity.
`params`	Named list with ‘dataset‘, ‘geneset_area‘.

Value

list(data, padj_colname, fold_colname).

Examples

```
init_data <- data.frame(
  foldChange = c(5, 0.1, 1.0), padj = c(0.01, 0.01, 0.5),
  Legend = c("Up", "Down", "NS"),
  row.names = c("GeneA", "GeneB", "GeneC")
)
result <- get_table_data(
  init_data = init_data,
  params = list(dataset = "alldetected", geneset_area = "")
)
result[[1]]  # the data.frame
```

`gmt_to_pathways` *Read a GMT file into a named list of gene-symbol vectors.*

Description

Thin wrapper around `fgsea::gmtPathways()` that adds an explicit file-existence check (raising classed `de_error`) and a `require_pkg("fgsea")` gate so callers without 'fgsea' installed see a friendly install message rather than a stack trace.

Usage

```
gmt_to_pathways(path)
```

Arguments

`path`	Path to a '.gmt' file (tab-separated, columns: name, url, gene1, gene2, ...).

Value

Named list, names = pathway names, elements = character vectors of gene symbols (or whatever ID the GMT uses).

Examples

```
p <- system.file("extdata", "test-gmt", "hallmark-mini.gmt",
                 package = "debrowser")
gmt_to_pathways(p)
```

heatmapControlsUI *heatmapControlsUI*

Description

Generates the left menu to be used for heatmap plots

Usage

```
heatmapControlsUI(id)
```

Arguments

id	module ID

Value

HeatmapControls

Note

heatmapControlsUI

Examples

```
x <- heatmapControlsUI("heatmap")
```

heatmapJScode *heatmapJScode*

Description

heatmap JS code for selection functionality

Usage

```
heatmapJScode()
```

Value

JS Code

Examples

```
x <- heatmapJScode()
```

heatmapServer *heatmapServer*

Description

Sets up shinyServer to be able to run heatmapServer interactively.

Usage

```
heatmapServer(input, output, session)
```

Arguments

input	input params from UI
output	output params to UI
session	session variable

Value

the panel for main plots;

Note

heatmapServer

Examples

```
heatmapServer
```

heatmapUI *heatmapUI*

Description

Creates a shinyUI to be able to run DEBrowser interactively.

Usage

```
heatmapUI(input, output, session)
```

Arguments

input	input variables
output	output objects
session	session

Value

the panel for heatmapUI;

Note

heatmapUI

Examples

```
shiny::shinyApp(ui = heatmapUI, server = function(input, output) {})
```

---

hideObj *hideObj*

Description

Hides a shiny object.

Usage

```
hideObj(btns = NULL)
```

Arguments

btns	hide group of objects with shinyjs

Value

invisible(NULL); called for the side effect of calling ‘shinyjs::hide()‘ on each supplied id.

Examples

```
x <- hideObj()
```

---

histogramControlsUI *histogramControlsUI*

Description

Generates the controls in the left menu for a histogram

Usage

```
histogramControlsUI(id)
```

Arguments

id	namespace id

Value

returns the left menu

Note

histogramControlsUI

Examples

```
x <- histogramControlsUI("histogram")
```

---

install_cutoff_preset_observers

Install Shiny observers that synchronise a preset-button input with the (padj, log2fc_cutoff) numeric inputs.

Description

Two observers are installed on the supplied session: (1) a click on the 'cutoff_preset' radio group fills the two numeric inputs with that preset's values; (2) a manual edit on either numeric input clears the preset highlight if the resulting (padj, log2fc) pair no longer matches any preset (or selects the matching preset if it happens to match one exactly).

Usage

```
install_cutoff_preset_observers(input, session)
```

Arguments

input, session	A Shiny input/session pair (either the top-level session for the global widget, or a moduleServer session for the namespaced widget).

Details

The observers guard against echoing each other via an 'isolate()' + '!identical()' check on the current radio-group selection. Both use 'ignoreInit = TRUE' to avoid the page-load click cascade.

Value

Invisible NULL; observers are installed as a side effect.

Examples

```
# Inside a Shiny server function:
#   install_cutoff_preset_observers(input, session)
# The call has side effects (observers) and requires a live
# reactive context, so the body is wrapped to skip at example
# time but still satisfies BiocCheck's runnable-example rule.
if (interactive()) {
  message("install_cutoff_preset_observers requires a Shiny session")
}
```

IQRPlotControlsUI *IQRPlotControlsUI*

Description

Generates the controls in the left menu for an IQR plot#'

Usage

```
IQRPlotControlsUI(id)
```

Arguments

id	namespace id

Value

returns the left menu

Note

IQRPlotControlsUI

Examples

```
x <- IQRPlotControlsUI("IQR")
```

kmeansControlsUI *kmeansControlsUI*

Description

get kmeans controls

Usage

```
kmeansControlsUI(id)
```

Arguments

id	module ID

Value

controls

Note

kmeansControlsUI

Examples

```
x <- kmeansControlsUI("heatmap")
```

lcfMetRadio *lcfMetRadio*

Description

Radio buttons for low count removal methods

Usage

```
lcfMetRadio(id)
```

Arguments

id	namespace id

Value

radio control

Note

lcfMetRadio

Examples

```
x <- lcfMetRadio("lcf")
```

library_depth_summary *Per-sample library depth summary.*

Description

Computes column sums (sequencing depth) for a count matrix, optionally annotates each sample with a group label from a metadata data.frame, and flags outliers using a 2-sigma rule on the depth values.

Usage

```
library_depth_summary(counts, meta = NULL, group_col = NULL)
```

Arguments

counts	Numeric matrix or data.frame (rows = features, cols = samples).
meta	Optional data.frame with at least a ‘samples‘ column matching ‘colnames(counts)‘ plus the column named in ‘group_col‘. If NULL, ‘group‘ is ‘NA_character_‘ for every sample.
group_col	Name of the column in ‘meta‘ to use as the group label. Ignored if ‘meta‘ is NULL.

Value

data.frame with columns ‘sample‘, ‘depth‘, ‘group‘, ‘is_outlier_2sd‘ (one row per sample, ordered as ‘colnames(counts)‘).

Examples

```
m <- matrix(rpois(60, lambda = 10), nrow = 10,
            dimnames = list(NULL, paste0("s", 1:6)))
library_depth_summary(m)
```

---

list_models *List available models for a configured LLM provider.*

Description

Calls the provider's models endpoint via ellmer. Returns the user's available model ids on success, or a hardcoded fallback list on failure. Always returns a non-empty character vector.

Usage

```
list_models(provider, api_key = NULL)
```

Arguments

provider	chr(1). "anthropic" | "openai" | "ollama".
api_key	chr(1) or NULL. Required for anthropic/openai; NULL for ollama.

Value

character vector of model ids.

---

log2fc_to_fold *Convert |log2FC| cutoff to fold-change cutoff.*

Description

Convert |log2FC| cutoff to fold-change cutoff.

Usage

```
log2fc_to_fold(x)
```

Arguments

x	Numeric |log2FC| cutoff.

Value

Fold-change cutoff (2^x).

Examples

```
log2fc_to_fold(1)   # 2
log2fc_to_fold(2)   # 4
```

`mainPlotControlsUI` *mainPlotControlsUI*

Description

Generates the left menu to be used for main plots

Usage

```
mainPlotControlsUI(id)
```

Arguments

`id`	module ID

Value

mainPlotControls

Note

`mainPlotControlsUI`

Examples

```
x <- mainPlotControlsUI("main")
```

`mainScatterNew` *mainScatterNew*

Description

Creates the main scatter, volcano or MA plot to be displayed within the main panel.

Usage

```
mainScatterNew(input = NULL, data = NULL, cond_names = NULL, source = NULL)
```

Arguments

`input`	input params
`data`	dataframe that has log2FoldChange and log10padj values
`cond_names`	condition names
`source`	for event triggering to select genes

Value

scatter, volcano or MA plot

Examples

```
x <- mainScatterNew()
```

---

make_default_metadata *Build a single-condition single-batch metadata data frame.*

Description

Used as a fallback when the user uploads counts without metadata. Downstream condSelect's existing "need >=2 conditions" validation will surface the requirement when the user proceeds – no new validation is added here.

Usage

```
make_default_metadata(counts)
```

Arguments

counts	data frame whose column names are the sample IDs.

Value

data frame with columns 'Sample', 'Condition', 'Batch'.

Examples

```
counts <- data.frame(S1 = c(100, 40), S2 = c(200, 60), S3 = c(150, 55))
rownames(counts) <- c("GeneA", "GeneB")
make_default_metadata(counts)
```

---

match_preset *Identify which preset a (padj, log2fc) pair matches.*

Description

Uses a small numeric tolerance because numericInput round-trips floats via JSON and exact equality occasionally fails.

Usage

```
match_preset(padj, log2fc, tol = 1e-09)
```

Arguments

padj, log2fc	Numeric scalars from the cutoff inputs.
tol	Match tolerance.

Value

Character scalar ("strict" | "standard") or NA_character_ if either input is invalid or no preset matches.

Examples

```
match_preset(0.01, 1)   # "strict"
match_preset(0.05, 1)   # "standard"
match_preset(0.10, 1)   # NA_character_
```

---

merge_comparisons *Merge per-comparison DE results into one wide table.*

Description

Pure version of 'getMergedComparison()'. The legacy version reads 'input$norm_method'; this takes the same value off 'params'.

Usage

```
merge_comparisons(dc, nc, params = list())
```

Arguments

dc	List of per-comparison containers (each has 'init_data', 'cols', 'cond_names').
nc	Number of comparisons.
params	Named list with 'norm_method'.

Value

Merged data.frame (samples + per-comparison foldChange/padj cols).

Examples

```
init1 <- data.frame(
  S1 = c(100L, 40L), S2 = c(200L, 60L),
  S3 = c(10L, 150L), S4 = c(12L, 160L),
  foldChange = c(10, 0.25), padj = c(0.01, 0.01),
  row.names = c("GeneA", "GeneB")
)
dc <- list(list(
  init_data = init1,
  cols = c("S1", "S2", "S3", "S4"),
  cond_names = c("Treat", "Ctrl")
))
merge_comparisons(dc, nc = 1, params = list(norm_method = "none"))
```

methods_paragraph *Manuscript-ready methods paragraph.*

Description

Joins the non-NA per-step sentences from [methods_sentences()] into a single paragraph (~150-250 words for typical sessions). Embeds inline citations from '.method_refs' and version stamps via [.pkg_version_or_unknown()]. Used by 'emit_r_script()' (header comment), 'emit_rmd()' (Methods section), and the Export menu's "Copy methods text" modal.

Usage

```
methods_paragraph(blocks)
```

Arguments

blocks	Output of [build_session_blocks()].

Value

character(1). A single paragraph; no embedded newlines.

msigdb_pathways *Fetch MSigDB gene sets as a named list of gene-symbol vectors.*

Description

Wraps `msigdbr::msigdbr()` and reshapes the long-format result into the same named-list-of-character-vectors shape that `gmt_to_pathways` returns, so the rest of the Enrichment tab is source-agnostic. `msigdbr` is in Suggests; the `require_pkg("msigdbr")` gate produces a friendly install prompt for users without it.

Usage

```
msigdb_pathways(
  species = "Homo sapiens",
  collection = "H",
  subcollection = NULL
)
```

Arguments

species	Character; an MSigDB-supported species name (see `msigdbr::msigdbr_species()`). Default `"Homo sapiens"`.
collection	Character; the top-level MSigDB collection code (e.g. `"H"` for Hallmark, `"C2"` for curated, `"C5"` for ontology). See `msigdbr::msigdbr_collections()`.
subcollection	Optional character; the subcollection code (e.g. `"CP:KEGG"`, `"GO:BP"`). NULL returns all subcollections under the given top-level collection.

Details

Pathway names use the canonical `gs_name` (e.g. `HALLMARK_HYPOXIA`). Genes use the human-readable `gene_symbol` column.

Value

Named list – names are pathway names, elements are character vectors of gene symbols. Empty list with a classed `empty_input` error if msigdbr returns no rows for the given species/collection/subcollection combination.

Examples

```
p <- msigdb_pathways("Homo sapiens", "H")
length(p)               # 50 (Hallmark)
head(p[["HALLMARK_HYPOXIA"]])
```

`mt_pct_per_sample` *Per-sample mitochondrial-transcript percentage.*

Description

Identifies mitochondrial-genome rows by matching ‘rownames(counts)‘ against a curated list of MT gene symbols across the common naming conventions: human HGNC (‘MT-ND1‘, ‘MT-CO1‘), mouse MGI (‘mt-Nd1‘, ‘mt-Co1‘), the dash-stripped Ensembl-style variants (‘MTND1‘, ‘mtNd1‘), and the prefix-stripped suffixes (‘ND1‘, ‘Nd1‘). Avoids false positives like ‘Mtor‘, ‘Mthfr‘, ‘Atp6v0a1‘ by anchoring to the exact suffix set.

Usage

```
mt_pct_per_sample(counts, pattern = NULL)
```

Arguments

`counts`	Numeric matrix or data.frame (rows = features, cols = samples).
`pattern`	Optional regex to override the default matcher. If NULL (the default), the curated MT symbol list is used.

Details

Returns each sample’s mitochondrial count, total count, and MT percentage.

Value

data.frame with columns ‘sample‘, ‘mt_count‘, ‘total‘, ‘mt_pct‘. Zero rows if no rownames match.

Examples

```
m <- matrix(c(1, 2, 0, 4, 5, 0), nrow = 3,
            dimnames = list(c("MT-ND1", "ACTB", "GAPDH"), c("a", "b")))
mt_pct_per_sample(m)
# mouse MGI (with or without the dash) also matches:
m2 <- matrix(c(3, 0, 7, 0), nrow = 2,
             dimnames = list(c("mt-Nd1", "Actb"), c("a", "b")))
mt_pct_per_sample(m2)
```

---

nes_heatmap_data *Reshape per-comparison GSEA results for the NES heatmap.*

Description

Used by enrichmentNesHeatmapServer to render a tile plot of NES across pathways (rows) by comparisons (columns).

Usage

```
nes_heatmap_data(results_by_comparison, sig_only = FALSE, sig_threshold = 0.05)
```

Arguments

results_by_comparison	Named list of run_gsea outputs. Names become column labels in the heatmap.
sig_only	Logical; if TRUE, retain only pathways significant in at least one comparison.
sig_threshold	padj cutoff used when ‘sig_only = TRUE‘.

Value

data.frame with columns ‘pathway‘, ‘comparison‘, ‘NES‘, ‘padj‘.

Examples

```
nes_heatmap_data(list(c1 = run_gsea(de1, paths),
                      c2 = run_gsea(de2, paths)))
```

---

niceKmeans *niceKmeans*

Description

Generates hierarchially clustered K-means clusters

Usage

```
niceKmeans(df = NULL, input = NULL, iter.max = 1000, nstart = 100)
```

Arguments

df	data
input	user inputs
iter.max	max iteration for kmeans clustering
nstart	n for kmeans clustering

Value

heatmap plot area

Note

niceKmeans

Examples

```
x <- niceKmeans()
```

---

normalizationMethods *normalizationMethods*

Description

Select box to select normalization method prior to batch effect correction

Usage

```
normalizationMethods(id)
```

Arguments

id	namespace id

Value

radio control

Note

normalizationMethods

Examples

```
x <- normalizationMethods("batch")
```

normalize_counts *Normalize a count matrix.*

Description

Pure function: takes a numeric matrix, returns a normalized numeric matrix. Wraps ‘edgeR::calcNormFactors‘ + ‘edgeR::equalizeLibSizes‘ for TMM/RLE/upper-quartile, DESeq2’s median-of-ratios for ‘"MRN"‘, identity for ‘"none"‘.

Usage

```
normalize_counts(counts, method = "TMM")
```

Arguments

counts	Numeric matrix or data.frame (genes x samples).
method	One of "TMM", "RLE", "upperquartile", "MRN", "none".

Value

Normalized numeric matrix with the same shape and dimnames.

Examples

```
m <- matrix(as.integer(c(100, 200, 150, 80, 250, 130,
                         40,  60,  55, 30,  70,  45)),
            nrow = 2, byrow = TRUE)
colnames(m) <- paste0("S", 1:6)
rownames(m) <- c("Gene1", "Gene2")
normalize_counts(m, method = "TMM")
```

palUI *palUI*

Description

get pallete

Usage

```
palUI(id)
```

Arguments

id	namespace ID

Value

pals

Note

palUI

Examples

```
x <- palUI("heatmap")
```

---

panel.cor *panel.cor*

Description

Prepares the correlations for the all2all plot.

Usage

```
panel.cor(x, y, prefix = "rho=", cex.cor = 2, ...)
```

Arguments

x	numeric vector x
y	numeric vector y
prefix	prefix for the text
cex.cor	correlation font size
...	additional parameters

Value

all2all correlation plots

Examples

```
panel.cor(c(1, 2, 3), c(4, 5, 6))
```

---

panel.hist *panel.hist*

Description

Prepares the historgram for the all2all plot.

Usage

```
panel.hist(x, ...)
```

Arguments

x	a vector of values for which the histogram is desired
...	any additional params

Value

all2all histogram plots

Examples

```
panel.hist(1)
```

---

parse_preset_cookie *Parse the debrowser_preset value out of an HTTP Cookie header*

Description

Returns NULL when the cookie is missing or empty. Used by ‘deUI()‘ so the preset choice from the navbar picker persists across page reloads.

Usage

```
parse_preset_cookie(cookie_header)
```

Arguments

cookie_header	value of ‘req$HTTP_COOKIE‘ (string or NULL).

Value

a single string (preset name) or NULL.

---

pcaPlotControlsUI *pcaPlotControlsUI*

Description

Generates the PCA PLots Left menu to be displayed within the DEBrowser.

Usage

```
pcaPlotControlsUI(id = "pca")
```

Arguments

id	namespace id

Value

returns the left menu according to the selected tab;

Note

pcaPlotControlsUI

Examples

```
x <- pcaPlotControlsUI("pca")
```

`plotData` *plotData*

Description

prepare plot data for mainplots

Usage

```
plotData(pdata = NULL, input = NULL)
```

Arguments

`pdata`	data
`input`	input

Value

prepdata

Note

`plotData`

Examples

```
x <- plotData()
```

`plotMarginsUI` *plotMarginsUI*

Description

Margins module for plotly plots

Usage

```
plotMarginsUI(id, t = 20, b = 100, l = 100, r = 20)
```

Arguments

`id`	id
`t`	top margin
`b`	bottom margin
`l`	left margin
`r`	right margin

Value

size and margins controls

Note

plotMarginsUI

Examples

```
x <- plotMarginsUI("heatmap")
```

---

plotSizeMarginsUI *plotSizeMarginsUI*

Description

Size and margins module for plotly plots

Usage

```
plotSizeMarginsUI(id, w = 800, h = 640, t = 20, b = 100, l = 100, r = 20)
```

Arguments

id	id
w	width
h	height
t	top margin
b	bottom margin
l	left margin
r	right margin

Value

size and margins controls

Note

plotSizeMarginsUI

Examples

```
x <- plotSizeMarginsUI("heatmap")
```

plotSizeUI *plotSizeUI*

Description

Size module for plotly plots

Usage

```
plotSizeUI(id, w = 800, h = 600)
```

Arguments

id	id
w	width
h	height

Value

size and margins controls

Note

plotSizeUI

Examples

```
x <- plotSizeUI("heatmap")
```

plotTypeUI *plotTypeUI*

Description

Plot download type

Usage

```
plotTypeUI(id)
```

Arguments

id	id

Value

size and margins controls

Note

plotTypeUI

Examples

```
x <- plotTypeUI("heatmap")
```

---

plot_de_direction_bar *Horizontal bar plot of up / down DEG counts per comparison.*

Description

Up bars (red) extend right; down bars (blue) extend left from the zero line. Comparisons are ordered top-to-bottom by total DEG count.

Usage

```
plot_de_direction_bar(summary, subtitle = NULL)
```

Arguments

summary	data.frame as returned by [de_direction_summary()]; must contain 'comparison', 'n_up', 'n_down', 'n_sig'.
subtitle	Optional subtitle (typically a cutoff label like "Threshold: padj <= 0.05").

Value

ggplot object.

Examples

```
summary <- data.frame(
  comparison = c("Treat vs Ctrl", "Drug vs Vehicle"),
  n_up   = c(120L, 45L),
  n_down = c(80L,  30L),
  n_sig  = c(200L, 75L),
  stringsAsFactors = FALSE
)
plot_de_direction_bar(summary, subtitle = "padj <= 0.05")
```

---

plot_de_pairwise_heatmap *Symmetric pairwise heatmap of significant DEG counts between groups.*

Description

Builds a 'groups x groups' symmetric matrix (diagonal NA) where each off-diagonal cell shows the number of significant DEGs in the comparison between those two groups. Group identities come from 'cond_names' of 'comparisons'; sig counts come from 'summary', matched on 'comparison_labels(comparisons)'. When the same group pair appears in multiple comparisons, the last seen wins.

Usage

```
plot_de_pairwise_heatmap(summary, comparisons, subtitle = NULL)
```

Arguments

summary	data.frame as returned by [de_direction_summary()].
comparisons	List of comparison spec/dc-style entries with ‘cond_names‘ (treatment, control) used as group labels. Length and order should mirror the ‘de_list‘ that produced ‘summary‘.
subtitle	Optional subtitle (typically a cutoff label).

Value

ggplot object.

Examples

```
summary_df <- data.frame(
  comparison = c("Treat vs Ctrl", "Drug vs Vehicle"),
  n_up   = c(120L, 45L),
  n_down = c(80L,  30L),
  n_sig  = c(200L, 75L),
  stringsAsFactors = FALSE
)
comparisons <- list(
  list(cond_names = c("Treat", "Ctrl")),
  list(cond_names = c("Drug", "Vehicle"))
)
plot_de_pairwise_heatmap(summary_df, comparisons)
```

---

plot_method_scatter *Pairwise log2FC scatter between two named entries of a DE list.*

Description

Joins the two per-entry tables on gene ID and renders a scatter of log2FoldChange (id1 on x, id2 on y). Spearman rho appears in the subtitle. The y=x reference line is drawn dashed.

Usage

```
plot_method_scatter(de_list, id1, id2)
```

Arguments

de_list	Named list of DE result data.frames (each with at minimum ‘ID‘ and ‘log2FoldChange‘ columns).
id1, id2	Names present in ‘names(de_list)‘ to put on x and y.

Details

Generic across "DE list dimensions": entries can be DE methods on the same comparison, OR comparisons on the same method, OR any other axis as long as each named entry contains a ‘data.frame(ID, log2FoldChange, ...)‘.

Value

ggplot object.

Examples

```
de_list <- list(
  EdgeR = data.frame(ID = paste0("G", 1:5),
    log2FoldChange = c(2, -1, 0, 3, -2),
    padj = c(0.01, 0.04, 0.5, 0.02, 0.03)),
  Limma = data.frame(ID = paste0("G", 1:5),
    log2FoldChange = c(1.8, -0.9, 0.1, 2.5, -1.8),
    padj = c(0.02, 0.03, 0.6, 0.01, 0.04))
)
plot_method_scatter(de_list, id1 = "EdgeR", id2 = "Limma")
```

---

plot_method_upset *UpSet plot of DE-gene overlap across methods.*

Description

Gated by ‘require_pkg("UpSetR")‘. Returns the ‘UpSetR::upset()‘ object – caller (renderPlot) is responsible for printing.

Usage

```
plot_method_upset(de_list, padj_cutoff = 0.05, lfc_cutoff = 0)
```

Arguments

de_list	Output of [run_de_methods()].
padj_cutoff	Maximum padj for "significant" (default 0.05).
lfc_cutoff	Minimum |log2FoldChange| for "significant" (default 0; pass 0.585 for |fold|>=1.5, 1 for |fold|>=2).

Value

Result of ‘UpSetR::upset()‘ (a list with class ‘"upset"‘). When fewer than 2 methods produce non-empty sets, returns NULL (caller should render an empty-state instead).

Examples

```
if (requireNamespace("UpSetR", quietly = TRUE)) {
  de_list <- list(
    EdgeR = data.frame(ID = paste0("G", 1:5),
      log2FoldChange = c(2, -1, 0, 3, -2),
      padj = c(0.01, 0.04, 0.5, 0.02, 0.03)),
    Limma = data.frame(ID = paste0("G", 1:5),
      log2FoldChange = c(1.8, -0.9, 0.1, 2.5, -1.8),
      padj = c(0.02, 0.03, 0.6, 0.01, 0.04))
  )
  plot_method_upset(de_list)
}
```

plot_pca *plot_pca*

Description

Plots the PCA results for the selected dataset.

Usage

```
plot_pca(
  dat = NULL,
  pcx = 1,
  pcy = 2,
  metadata = NULL,
  color = NULL,
  shape = NULL,
  size = NULL,
  textonoff = "On",
  legendSelect = "samples",
  input = NULL
)
```

Arguments

dat	data
pcx	x axis label
pcy	y axis label
metadata	additional data
color	color for plot
shape	shape for plot
size	size of the plot
textonoff	text on off
legendSelect	select legend
input	input param

Value

pca list

Examples

```
load(system.file("extdata", "demo", "demodata.Rda",
  package = "debrowser"
))
metadata <- cbind(
  colnames(demodata[, seq_len(6L)]),
  colnames(demodata[, seq_len(6L)]),
  c(rep("Cond1", 3), rep("Cond2", 3))
)
```

```
colnames(metadata) <- c("samples", "color", "shape")

a <- plot_pca(
  getNormalizedMatrix(
    demodata[rowSums(demodata[, seq_len(6L)]) > 10, seq_len(6L)]
  ),
  metadata = metadata, color = "samples",
  size = 5, shape = "shape"
)
```

`prepDataContainer` *Run DE per comparison and return the downstream 'dclist' payload.*

Description

Run DE per comparison and return the downstream 'dclist' payload.

Usage

```
prepDataContainer(data, metadata, comparisons_spec)
```

Arguments

`data`	count matrix (rows = features, cols = samples).
`metadata`	sample-metadata data.frame; first column is the sample id.
`comparisons_spec`	list of per-comparison spec lists with components 'treatment_samples', 'control_samples', 'treatment_label', 'control_label', 'de_method', 'method_params', 'covariates', 'meta_column' (NA_character_ when manual mode).

Value

list of length 'length(comparisons_spec)' with components 'conds', 'cols', 'cond_names', 'init_data', 'demethod_params', 'dds' per comparison. 'dds' is the fitted 'DESeqDataSet' for DESeq2 runs and NULL for edgeR/limma. Returns NULL if no comparison produced usable results.

Examples

```
set.seed(42)
counts <- matrix(
  as.integer(abs(rnorm(60, mean = 100, sd = 30))),
  nrow = 10, ncol = 6,
  dimnames = list(paste0("G", 1:10), paste0("S", 1:6))
)
meta <- data.frame(
  Sample    = paste0("S", 1:6),
  Condition = c(rep("Ctrl", 3), rep("Treat", 3)),
  stringsAsFactors = FALSE
)
spec <- list(list(
  treatment_samples = paste0("S", 4:6),
  control_samples   = paste0("S", 1:3),
```

```
  treatment_label   = "Treat",
  control_label     = "Ctrl",
  de_method         = "EdgeR",
  method_params     = list(
    edgeR_normfact = "TMM",
    dispersion     = "0",
    edgeR_testType = "exactTest"
  ),
  covariates        = character(0),
  meta_column       = NA_character_
))
## prepDataContainer must run inside a Shiny session
## (delegates to a moduleServer-based DE analysis module).
prepDataContainer(counts, meta, spec)
```

---

prepGroup *prepGroup*

Description

prepare group table

Usage

```
prepGroup(conds = NULL, cols = NULL, metadata = NULL, covariates = NULL)
```

Arguments

conds	inputconds
cols	columns
metadata	metadata
covariates	covariates

Value

data

Examples

```
x <- prepGroup()
```

prepHeatData *prepHeatData*

Description

scales the data

Usage

```
prepHeatData(expdata = NULL, input = NULL)
```

Arguments

expdata	a matrixthat includes expression values
input	input variables

Value

heatdata

Examples

```
x <- prepHeatData()
```

prepPCADat *prepPCADat*

Description

prepares pca data with metadata. If metadata doesn't exists it puts all the sampels into a signlge group; "Conds".

Usage

```
prepPCADat(pca_data = NULL, metadata = NULL, input = NULL, pcx = 1, pcy = 2)
```

Arguments

pca_data	pca run results
metadata	additional meta data
input	input
pcx	x axis label
pcy	y axis label

Value

Color and shape from selection boxes or defaults

Examples

```
x <- prepPCADat()
```

progress_message *Build a named list payload for the "debrowser-progress" custom message.*

Description

Build a named list payload for the "debrowser-progress" custom message.

Usage

```
progress_message(key, state)
```

Arguments

key	character, progress key (e.g. "upload", "filter", "data_prep").
state	character, one of "pending", "done", "locked", "skipped", or "" (empty clears the icon).

Value

named list with elements ‘key‘ and ‘state‘.

Examples

```
progress_message("upload", "done")
```

push *push*

Description

Push an object to the list.

Usage

```
push(l, ...)
```

Arguments

l	that are going to push to the list
...	list object

Value

combined list

Examples

```
mylist <- list()
newlist <- push(1, mylist)
```

qcCooksUI *qcCooksUI*

Description

UI factory for the Cook's-distance outlier-count QC card. The body is a 'uiOutput' so the server can switch between the bar plot and an empty-state alert when no fitted 'DESeqDataSet' is available.

Usage

```
qcCooksUI(id)
```

Arguments

id	character, namespace id

Value

a 'bslib::card' tagList

Examples

```
qcCooksUI("cooks")
```

qcDetectionRateUI *qcDetectionRateUI*

Description

UI factory for the feature-detection-rate QC card. Renders a 'de_card' containing a 'plotly' bar of detection percentage per sample plus a one-line caption.

Usage

```
qcDetectionRateUI(id)
```

Arguments

id	character, namespace id

Value

a 'bslib::card' tagList

Examples

```
qcDetectionRateUI("detectionRate")
```

qcDispersionUI *qcDispersionUI*

Description

UI factory for the DESeq2 dispersion-estimates QC card. The body is a ‘uiOutput‘ so the server can switch between the base-R dispersion plot and an empty-state alert when no fitted ‘DESeqDataSet‘ is available.

Usage

```
qcDispersionUI(id)
```

Arguments

id	character, namespace id

Value

a ‘bslib::card‘ tagList

Examples

```
qcDispersionUI("dispersion")
```

qcLibraryDepthUI *qcLibraryDepthUI*

Description

UI factory for the library-depth QC card. Renders a ‘de_card‘ containing a ‘plotly‘ bar of total counts per sample (with a CSV download button) and a one-line caption explaining the 2-SD outlier flag.

Usage

```
qcLibraryDepthUI(id)
```

Arguments

id	character, namespace id

Value

a ‘bslib::card‘ tagList

Examples

```
qcLibraryDepthUI("libraryDepth")
```

qcMtPctUI *qcMtPctUI*

Description

UI factory for the mitochondrial-percentage QC card. The body is a ‘uiOutput‘ so the server can switch between a plot and an empty-state alert when no MT genes are detected.

Usage

```
qcMtPctUI(id)
```

Arguments

id	character, namespace id

Value

a ‘bslib::card‘ tagList

Examples

```
qcMtPctUI("mtPct")
```

qcSampleDistUI *qcSampleDistUI*

Description

UI factory for the sample-distance QC card. Renders a ‘de_card‘ containing a ‘heatmaply‘ heatmap and a caption explaining the distance metric.

Usage

```
qcSampleDistUI(id)
```

Arguments

id	character, namespace id

Value

a ‘bslib::card‘ tagList

Examples

```
qcSampleDistUI("sampleDist")
```

qcSizeFactorsUI *qcSizeFactorsUI*

Description

UI factory for the size-factors-vs-library-size QC card. The body is a ‘uiOutput‘ so the server can render the comparison plot or an empty-state alert when no fitted ‘DESeqDataSet‘ is available.

Usage

```
qcSizeFactorsUI(id)
```

Arguments

id	character, namespace id

Value

a ‘bslib::card‘ tagList

Examples

```
qcSizeFactorsUI("sizeFactors")
```

qc_keep_cols *Subset a count matrix to a user-selected column list.*

Description

Mirrors the QC sidebar's column-selector contract: NULL ‘selected‘ (the pre-render state) returns ‘counts‘ unchanged; otherwise keeps only the columns whose name is in ‘selected‘. Empty intersection returns a zero-column matrix (matches `getSelectedCols` on an empty selection).

Usage

```
qc_keep_cols(counts, selected = NULL)
```

Arguments

counts	Numeric matrix or data.frame, or NULL.
selected	Character vector of sample names to keep, or NULL.

Value

The subset matrix (or ‘counts‘ unchanged if ‘selected‘ is NULL); NULL passes through.

Examples

```
m <- matrix(1:6, nrow = 2,
            dimnames = list(NULL, c("a", "b", "c")))
qc_keep_cols(m, c("a", "c"))
qc_keep_cols(m, NULL)  # unchanged
```

qc_keep_meta_rows *Subset a sample-metadata data.frame by a user-selected column list.*

Description

Companion to qc_keep_cols: keeps the rows of ‘meta‘ whose ‘samples‘ column matches ‘selected‘. NULL ‘selected‘ returns ‘meta‘ unchanged. NULL ‘meta‘ passes through. If ‘meta‘ has no ‘samples‘ column, returns it unchanged.

Usage

```
qc_keep_meta_rows(meta, selected = NULL)
```

Arguments

meta	data.frame with a ‘samples‘ column, or NULL.
selected	Character vector of sample names to keep, or NULL.

Value

The subset data.frame, or ‘meta‘ unchanged when there is nothing to do; NULL passes through.

Examples

```
m <- data.frame(samples = c("a", "b", "c"), x = 1:3)
qc_keep_meta_rows(m, c("a", "c"))
```

removeCols *removeCols*

Description

remove unnecessary columns

Usage

```
removeCols(cols = NULL, dat = NULL)
```

Arguments

cols	columns that are going to be removed from data frame
dat	data

Value

data

Examples

```
x <- removeCols()
```

removeExtraCols *removeExtraCols*

Description

remove extra columns for QC plots

Usage

```
removeExtraCols(dat = NULL)
```

Arguments

dat	selected data

Value

'dat' with QC-extraneous columns (padj/foldChange/Legend/etc.) stripped, leaving only sample columns suitable for plotting.

Examples

```
removeExtraCols()
```

require_pkg *Require a Suggested package, with a friendly error if missing.*

Description

Used to gate features that depend on packages declared in 'Suggests:' rather than 'Imports:'. Raises a 'de_error()' with class '"missing_suggested_pkg"' that names the package and the feature that needs it, plus the install command.

Usage

```
require_pkg(pkg, feature = pkg)
```

Arguments

pkg	Package name (single string).
feature	Short human description of the feature that requires it (e.g. '"Harman batch correction"').

Value

'TRUE' invisibly if available; otherwise raises.

Examples

```
# Returns TRUE invisibly when the package is present:
require_pkg("stats", feature = "basic statistics")
```

resend_verification_email

Re-send the verification email for an existing unverified user.

Description

Generates a fresh token + 24h expiry and pushes a new send. Safe to call repeatedly – each call invalidates any previous link.

Usage

```
resend_verification_email(user_id, base_url = NULL)
```

Arguments

user_id	Username.
base_url	Optional app base URL (used to build the verify link).

Value

TRUE on success; errors if user is unknown or already verified.

reset_debrowser_password

Reset a debrowser user's password from the R console.

Description

Updates the hashed_pw column for an existing user. Useful when the password is forgotten and there is no admin UI yet.

Usage

```
reset_debrowser_password(user_id, new_password)
```

Arguments

user_id	Username.
new_password	New plaintext password (hashed before storage).

Value

TRUE on success; errors if the user does not exist.

Examples

```
reset_debrowser_password("alice", "newhunter2!")
```

round_vals *round_vals*

Description

Plot PCA results.

Usage

```
round_vals(l)
```

Arguments

l	the value

Value

round value

Examples

```
x <- round_vals(5.1323223)
```

runDE *runDE*

Description

Run DE algorithms on the selected parameters. Output is to be used for the interactive display.

Usage

```
runDE(
  data = NULL,
  metadata = NULL,
  columns = NULL,
  conds = NULL,
  params = NULL,
  return_dds = FALSE
)
```

Arguments

data	A matrix that includes all the expression raw counts, rownames has to be the gene, isoform or region names/IDs
metadata	metadata of the matrix of expression raw counts
columns	is a vector that includes the columns that are going to be analyzed. These columns has to match with the given data.
conds	experimental conditions. The order has to match with the column order

params	all params for the DE methods
return_dds	Logical. When TRUE the function returns a list ‘list(res, dds)‘ for DESeq2 (so QC cards can introspect the fit) and ‘list(res, dds = NULL)‘ for edgeR / limma. Default FALSE returns the per-method data.frame as before.

Value

de results

Examples

```
x <- runDE()
```

---

runDESeq2 *runDESeq2*

Description

Run DESeq2 algorithm on the selected conditions. Output is to be used for the interactive display.

Usage

```
runDESeq2(
  data = NULL,
  metadata = NULL,
  columns = NULL,
  conds = NULL,
  params = NULL,
  return_dds = FALSE
)
```

Arguments

data	A matrix that includes all the expression raw counts, rownames has to be the gene, isoform or region names/IDs
metadata	metadata of the matrix of expression raw counts
columns	is a vector that includes the columns that are going to be analyzed. These columns has to match with the given data.
conds	experimental conditions. The order has to match with the column order
params	fitType: either "parametric", "local", or "mean" for the type of fitting of dispersions to the mean intensity. See estimateDispersions for description. betaPrior: whether or not to put a zero-mean normal prior on the non-intercept coefficients See nbinomWaldTest for description of the calculation of the beta prior. By default, the beta prior is used only for the Wald test, but can also be specified for the likelihood ratio test. testType: either "Wald" or "LRT", which will then use either Wald significance tests (defined by nbinomWaldTest), or the likelihood ratio test on the difference in deviance between a full and reduced model formula (defined by nbinomLRT) shrinkage: Adds shrunken log2 fold changes (LFC) and SE to a results table from DESeq run without LFC shrinkage. For

	consistency with results, the column name lfcSE is used here although what is returned is a posterior SD. Three shrinkage estimators for LFC are available via type (see the vignette for more details on the estimators). The apeglm publication demonstrates that 'apeglm' and 'ashr' outperform the original 'normal' shrinkage estimator.
return_dds	Logical. When TRUE the function returns 'list(res, dds)' so QC cards can introspect the fitted DESeqDataSet. Default FALSE preserves the legacy data.frame return shape.

Value

deseq2 results

Examples

```
x <- runDESeq2()
```

---

runEdgeR *runEdgeR*

Description

Run EdgeR algorithm on the selected conditions. Output is to be used for the interactive display.

Usage

```
runEdgeR(
  data = NULL,
  metadata = NULL,
  columns = NULL,
  conds = NULL,
  params = NULL
)
```

Arguments

data	A matrix that includes all the expression raw counts, rownames has to be the gene, isoform or region names/IDs
metadata	metadata of the matrix of expression raw counts
columns	is a vector that includes the columns that are going to be analyzed. These columns has to match with the given data.
conds	experimental conditions. The order has to match with the column order
params	normfact: Calculate normalization factors to scale the raw library sizes. Values can be "TMM","RLE","upperquartile","none". dispersion: either a numeric vector of dispersions or a character string indicating that dispersions should be taken from the data object. If a numeric vector, then can be either of length one or of length equal to the number of genes. Allowable character values are "common", "trended", "tagwise" or "auto". Default behavior ("auto" is to use most complex dispersions found in data object. testType: exactTest or glmLRT.

exactTest: Computes p-values for differential abundance for each gene between two digital libraries, conditioning on the total count for each gene. The counts in each group as a proportion of the whole are assumed to follow a binomial distribution. glmLRT: Fit a negative binomial generalized log-linear model to the read counts for each gene. Conduct genewise statistical tests for a given coefficient or coefficient contrast.

Value

edgeR results

Examples

```
x <- runEdgeR()
```

`runHeatmap` *runHeatmap*

Description

Creates a heatmap based on the user selected parameters within shiny

Usage

```
runHeatmap(input = NULL, session = NULL, expdata = NULL)
```

Arguments

`input`	input variables
`session`	session
`expdata`	a matrix that includes expression values

Value

heatmapply plot

Examples

```
x <- runHeatmap()
```

runHeatmap2 *runHeatmap2*

Description

Creates a heatmap based on the user selected parameters within shiny

Usage

```
runHeatmap2(input = NULL, session = NULL, expdata = NULL)
```

Arguments

input	input variables
session	session
expdata	a matrix that includes expression values

Value

heatmap.2

Examples

```
x <- runHeatmap2()
```

runLimma *runLimma*

Description

Run Limma algorithm on the selected conditions. Output is to be used for the interactive display.

Usage

```
runLimma(
  data = NULL,
  metadata = NULL,
  columns = NULL,
  conds = NULL,
  params = NULL
)
```

Arguments

data	A matrix that includes all the expression raw counts, rownames has to be the gene, isoform or region names/IDs
metadata	metadata of the matrix of expression raw counts
columns	is a vector that includes the columns that are going to be analyzed. These columns has to match with the given data.
conds	experimental conditions. The order has to match with the column order
params	normfact: Calculate normalization factors to scale the raw library sizes. Values can be "TMM","RLE","upperquartile","none". fitType, fitting method; "ls" for least squares or "robust" for robust regression normBet: Normalizes expression intensities so that the intensities or log-ratios have similar distributions across a set of arrays.

Value

Limma results

Examples

```
x <- runLimma()
```

---

run_de *Dispatch a DE run by method name.*

Description

Dispatch a DE run by method name.

Usage

```
run_de(
  method,
  counts,
  metadata = NULL,
  columns = NULL,
  conds = NULL,
  params = list(),
  return_dds = FALSE
)
```

Arguments

method	One of "DESeq2", "EdgeR", "Limma".
counts	A numeric count matrix or data.frame (genes x samples).
metadata	Sample metadata; first column is sample id.
columns	Character vector of sample column names to use.
conds	Factor of conditions, length == length(columns).

params	Named list with components: covariates (character "|"-joined or "NoCovariate"), fit_type ("parametric"/"local"/"mean"), beta_prior (logical), test_type ("Wald"/"LRT"), shrinkage ("None"/"apeglm"/"ashr"/"normal").
return_dds	Logical. Forwarded to [run_deseq2()] for the DESeq2 branch; ignored for edgeR/limma. When TRUE for DESeq2 the function returns 'list(res, dds)'; for non-DESeq2 methods the standard per-method result object is wrapped to 'list(res = <obj>, dds = NULL)' so downstream code can pattern-match a single shape.

Value

Method-specific result object.

Examples

```
set.seed(42)
counts <- matrix(
  as.integer(abs(rnorm(60, mean = 100, sd = 30))),
  nrow = 10, ncol = 6,
  dimnames = list(paste0("G", 1:10), paste0("S", 1:6))
)
conds <- c("Cond1", "Cond1", "Cond1", "Cond2", "Cond2", "Cond2")
run_de("EdgeR", counts, columns = colnames(counts), conds = conds)
```

---

run_deseq2 *Run DESeq2 on a count matrix.*

Description

Pure function: takes plain R objects, returns a 'DESeqResults' object. No Shiny dependency. Errors are raised via [de_error()] with classes: - 'too_few_columns' - 'null_input'

Usage

```
run_deseq2(
  counts,
  metadata = NULL,
  columns = NULL,
  conds = NULL,
  params = list(),
  return_dds = FALSE
)
```

Arguments

counts	A numeric count matrix or data.frame (genes x samples).
metadata	Sample metadata; first column is sample id.
columns	Character vector of sample column names to use.
conds	Factor of conditions, length == length(columns).
params	Named list with components: covariates (character "|"-joined or "NoCovariate"), fit_type ("parametric"/"local"/"mean"), beta_prior (logical), test_type ("Wald"/"LRT"), shrinkage ("None"/"apeglm"/"ashr"/"normal").

return_dds	Logical. If TRUE, return a list with components ‘res‘ (DESeqResults) and ‘dds‘ (the fitted DESeqDataSet), so downstream QC cards (Dispersion / SizeFactors / Cook’s) can introspect the fit. Default FALSE preserves the legacy data.frame-shaped contract.

Value

DESeqResults if ‘return_dds = FALSE‘; otherwise ‘list(res = DESeqResults, dds = DESeqDataSet)‘.

Examples

```
set.seed(42)
counts <- matrix(
  as.integer(abs(rnorm(60, mean = 100, sd = 30))),
  nrow = 10, ncol = 6,
  dimnames = list(paste0("G", 1:10), paste0("S", 1:6))
)
conds <- c("Cond1", "Cond1", "Cond1", "Cond2", "Cond2", "Cond2")
run_deseq2(counts, columns = colnames(counts), conds = conds,
           params = list(test_type = "Wald", shrinkage = "None"))
```

---

run_de_methods *Run multiple DE methods on the same comparison.*

Description

Wraps [run_de()] for each requested method and returns a named list of per-method result data.frames keyed by ‘methods‘. The returned frames are normalized to a common shape: ‘data.frame(ID, log2FoldChange, padj, pvalue, stat)‘ so downstream helpers can join by ‘ID‘ without per-method special-casing.

Usage

```
run_de_methods(
  counts,
  metadata = NULL,
  columns = NULL,
  conds = NULL,
  methods = c("DESeq2", "EdgeR", "Limma"),
  params_per_method = list()
)
```

Arguments

counts	Numeric count matrix or data.frame (genes x samples).
metadata	Sample metadata data.frame.
columns	Character vector of sample column names to use.
conds	Factor of conditions, length == length(columns).
methods	Character vector of method names; subset of ‘c("DESeq2", "EdgeR", "Limma")‘. Default all three.

`params_per_method`
: Optional named list keyed by method name whose values are the per-method ‘params‘ list passed to [run_de()]. Missing methods get an empty list (defaults take over inside the per-method runner).

Details

DESeq2’s native return is a ‘DESeqResults‘ object – coerced via ‘as.data.frame()‘. edgeR / limma already return data.frames.

Value

Named list of per-method data.frames with columns ID, log2FoldChange, padj, pvalue, stat. ‘length()‘ equals ‘length(methods)‘. Methods that error out are dropped from the returned list and a warning is signalled (not an error) so partial results remain usable.

Examples

```
set.seed(42)
counts <- matrix(
  as.integer(abs(rnorm(60, mean = 100, sd = 30))),
  nrow = 10, ncol = 6,
  dimnames = list(paste0("G", 1:10), paste0("S", 1:6))
)
conds <- c("Cond1", "Cond1", "Cond1", "Cond2", "Cond2", "Cond2")
run_de_methods(counts, columns = colnames(counts), conds = conds,
               methods = c("EdgeR", "Limma"))
```

---

run_edger *Run edgeR on a count matrix.*

Description

Run edgeR on a count matrix.

Usage

```
run_edger(
  counts,
  metadata = NULL,
  columns = NULL,
  conds = NULL,
  params = list()
)
```

Arguments

`counts`	A numeric count matrix or data.frame (genes x samples).
`metadata`	Sample metadata; first column is sample id.
`columns`	Character vector of sample column names to use.
`conds`	Factor of conditions, length == length(columns).
`params`	Named list with components: covariates, norm_fact ("TMM"/"RLE"/"upperquartile"/"none"), dispersion (numeric or character "common"/"trended"/"tagwise"/"auto"), test_type ("exactTest"/"glmLRT").

Value

data.frame with columns log2FoldChange, pvalue, padj, stat.

Examples

```
set.seed(42)
counts <- matrix(
  as.integer(abs(rnorm(60, mean = 100, sd = 30))),
  nrow = 10, ncol = 6,
  dimnames = list(paste0("G", 1:10), paste0("S", 1:6))
)
conds <- c("Cond1", "Cond1", "Cond1", "Cond2", "Cond2", "Cond2")
run_edger(counts, columns = colnames(counts), conds = conds)
```

---

run_gsea *Run pre-ranked GSEA on a DE result table.*

Description

Pure function: takes a DE table (one row per gene with an effect-size column) and a named list of pathways, returns a tidy data.frame of enrichment scores ordered by descending |NES|. No Shiny calls.

Usage

```
run_gsea(
  de_table,
  pathways,
  min_size = 15L,
  max_size = 500L,
  n_perm = 1000L,
  seed = 1L,
  stat_col = "log2FoldChange",
  id_col = "gene"
)
```

Arguments

de_table	data.frame with at minimum ‘id_col‘ (gene ID) and ‘stat_col‘ (numeric effect size, e.g. log2 fold change).
pathways	Named list, output of gmt_to_pathways or built from ‘msigdbr‘ (Phase E2).
min_size, max_size	Pathway-size filter passed to ‘fgsea::fgsea‘.
n_perm	Permutations for ‘fgsea::fgsea‘ ‘nPermSimple‘ arg.
seed	RNG seed for reproducibility.
stat_col	Column in ‘de_table‘ providing the ranking statistic.
id_col	Column in ‘de_table‘ providing the gene ID matching ‘pathways‘.

Details

Symbols in ‘de_table[[id_col]]‘ are matched against the gene IDs in ‘pathways‘ directly – caller is responsible for ID-space consistency (typically both SYMBOL or both ENTREZID).

Value

data.frame with columns ‘pathway‘, ‘size‘, ‘NES‘, ‘padj‘, ‘pval‘, ‘leading_edge‘ (last is a list-column of character vectors), ordered by descending ‘|NES|‘.

Examples

```
p <- gmt_to_pathways(system.file("extdata", "test-gmt",
                                 "hallmark-mini.gmt",
                                 package = "debrowser"))
de <- data.frame(gene = c("PGK1", "PDK1"),
                 log2FoldChange = c(3, 2.5))
run_gsea(de, p)
```

---

run_limma *Run limma-voom on a count matrix.*

Description

Run limma-voom on a count matrix.

Usage

```
run_limma(
  counts,
  metadata = NULL,
  columns = NULL,
  conds = NULL,
  params = list()
)
```

Arguments

counts	A numeric count matrix or data.frame (genes x samples).
metadata	Sample metadata; first column is sample id.
columns	Character vector of sample column names to use.
conds	Factor of conditions, length == length(columns).
params	Named list: covariates, norm_fact, fit_type ("ls"/"robust"), norm_bet ("none"/"scale"/"quantile"/...).

Value

data.frame with columns log2FoldChange, pvalue, padj, stat.

Examples

```
set.seed(42)
counts <- matrix(
  as.integer(abs(rnorm(60, mean = 100, sd = 30))),
  nrow = 10, ncol = 6,
  dimnames = list(paste0("G", 1:10), paste0("S", 1:6))
)
conds <- c("Cond1", "Cond1", "Cond1", "Cond2", "Cond2", "Cond2")
run_limma(counts, columns = colnames(counts), conds = conds)
```

`run_pca` *run_pca*

Description

Runs PCA on the selected dataset.

Usage

```
run_pca(x = NULL, retx = TRUE, center = TRUE, scale = TRUE)
```

Arguments

`x`	dataframe with experiment data
`retx`	specifies if the data should be returned
`center`	center the PCA (Boolean)
`scale`	scale the PCA (Boolean)

Value

pca list

Examples

```
load(system.file("extdata", "demo", "demodata.Rda",
  package = "debrowser"
))
pca_data <- run_pca(getNormalizedMatrix(
  demodata[rowSums(demodata[, seq_len(6L)]) > 10, seq_len(6L)]
))
```

sample_distance_matrix

Sample-to-sample distance matrix from variance-stabilized counts.

Description

Applies DESeq2's variance-stabilizing transformation (vst when 'nrow(counts) >= 30', otherwise varianceStabilizingTransformation) and computes euclidean distances between samples on the transformed scale. Falls back to 'log2(counts + 1)' if vst fails (e.g., all-zero rows).

Usage

```
sample_distance_matrix(counts)
```

Arguments

counts	Numeric matrix or data.frame (rows = features, cols = samples).

Value

Symmetric numeric matrix of size 'ncol(counts) x ncol(counts)', diagonal = 0, dimnames = 'colnames(counts)' on both axes.

Examples

```
m <- matrix(rpois(600, lambda = 10), nrow = 100,
            dimnames = list(NULL, paste0("s", 1:6)))
d <- sample_distance_matrix(m)
isSymmetric(d)
```

search_geneset

Search a data.frame's 'ID' column for a gene-set list.

Description

Search a data.frame's 'ID' column for a gene-set list.

Usage

```
search_geneset(dat, params = list())
```

Arguments

dat	data.frame with an 'ID' column (or first column treated as ID).
params	Named list with 'geneset_area' (string of search terms).

Value

Filtered data.frame; or 'dat' unchanged if 'geneset_area' is empty.

Examples

```
dat <- data.frame(
  ID = c("BRCA1", "TP53", "MYC"),
  padj = c(0.01, 0.02, 0.5),
  stringsAsFactors = FALSE
)
# Empty search returns all rows:
search_geneset(dat, params = list(geneset_area = ""))
```

`selectGroupInfo` *selectGroupInfo*

Description

Group info column selection. This can be used in batch effect or coloring the groups in the plots.

Usage

```
selectGroupInfo(
  metadata = NULL,
  input = NULL,
  selectname = "groupselect",
  label = "Group info"
)
```

Arguments

`metadata`	metadata
`input`	input values
`selectname`	name of the select box
`label`	label of the select box

Value

A ‘shiny::selectInput‘ listing metadata-column choices (with "None" prepended), or NULL when no metadata is supplied.

Note

`selectGroupInfo`

Examples

```
x <- selectGroupInfo()
```

`select_dataset` *Pick a subset of ‘rdata‘ based on the ‘dataset‘ filter param.*

Description

Pick a subset of ‘rdata‘ based on the ‘dataset‘ filter param.

Usage

```
select_dataset(
  rdata,
  get_selected = NULL,
  get_most_varied_data = NULL,
  merged_comparison = NULL,
  params = list()
)
```

Arguments

`rdata`	Filtered data.frame (typically the output of [apply_de_filters()]).
`get_selected`	Optional; the user’s lasso/click selection.
`get_most_varied_data`	Optional; the most-varied subset to use when ‘dataset == "most-varied"‘.
`merged_comparison`	Optional; merged comparisons table.
`params`	Named list with ‘dataset‘ and (optionally) ‘selected_plot‘, ‘geneset_area‘.

Value

Subset data.frame.

Examples

```
rdata <- data.frame(
  foldChange = c(5, 0.1, 1.0),
  padj = c(0.01, 0.01, 0.5),
  Legend = c("Up", "Down", "NS"),
  row.names = c("GeneA", "GeneB", "GeneC")
)
select_dataset(rdata, params = list(dataset = "alldetected"))
```

`sepRadio` *sepRadio*

Description

Radio button for separators

Usage

```
sepRadio(id, name)
```

Arguments

`id`	module id
`name`	name

Value

radio control

Note

`sepRadio`

Examples

```
x <- sepRadio("meta", "metadata")
```

`setBatch` *setBatch to skip batch effect correction batch variable set with the filter results*

Description

setBatch to skip batch effect correction batch variable set with the filter results

Usage

```
setBatch(fd = NULL)
```

Arguments

`fd`	filtered data

Value

fd data

Examples

```
x <- setBatch()
```

showObj *showObj*

Description

Displays a shiny object.

Usage

```
showObj(btns = NULL)
```

Arguments

btns	show group of objects with shinyjs

Value

invisible(NULL); called for the side effect of calling ‘shinyjs::show()‘ on each supplied id.

Examples

```
x <- showObj()
```

---

size_factor_library_summary
Per-sample size-factor vs. library-size summary.

Description

Pulls the size factors that DESeq2 estimated for the comparison and the raw library sizes from the same ‘DESeqDataSet‘, returned in a tidy data.frame plus ‘sf_scaled‘/‘lib_scaled‘ columns rescaled to [0,1] for a side-by-side bar comparison. Spearman’s rho between size factor and library size is attached as ‘attr(out, "spearman_rho")‘; values close to 1 confirm DESeq2 picked up depth differences without unexpected per-sample composition shifts.

Usage

```
size_factor_library_summary(dds)
```

Arguments

dds	A fitted ‘DESeqDataSet‘.

Value

data.frame with columns ‘sample‘, ‘size_factor‘, ‘library_size‘, ‘sf_scaled‘, ‘lib_scaled‘. Attribute ‘spearman_rho‘ holds the rank correlation between size factor and library size (NA for n < 2).

Examples

```
size_factor_library_summary(dds)
```

startDEBrowser *startDEBrowser*

Description

Starts the DEBrowser to be able to run interactively.

Usage

```
startDEBrowser(hosted = FALSE, trusted_proxies = character(0), port = 3838)
```

Arguments

hosted	Logical. When TRUE, enables hosted-mode behaviors: auth-provider chain, per-user AI settings (D2.6), bookmark ownership enforcement (D2.3+). Default FALSE preserves the single-user desktop launch experience. May also be set via the 'DEBROWSER_HOSTED' env var.
trusted_proxies	Character vector of proxy IPs and CIDR networks (e.g. 'c("127.0.0.1", "10.0.0.0/8")') whose 'X-Forwarded-User' header is trusted as the authenticated user. Only used when 'hosted = TRUE'. Empty by default.
port	Integer TCP port to bind. Default '3838' matches the shiny-server convention so bookmark URLs stay stable across restarts (essential for '?_state_id_=...' links the user copies from the share modal). Pass 'NULL' to let Shiny pick a random free port (legacy behavior).

Value

the app

Note

startDEBrowser

Examples

```
startDEBrowser()
startDEBrowser(hosted = TRUE,
               trusted_proxies = c("127.0.0.1", "10.0.0.0/8"))
```

startHeatmap *startHeatmap*

Description

Starts the DEBrowser heatmap

Usage

```
startHeatmap()
```

Value

the app

Note

startHeatmap

Examples

```
startHeatmap()
```

textareaInput *textareaInput*

Description

Generates a text area input to be used for gene selection within the DEBrowser.

Usage

```
textareaInput(id, label, value, rows = 20, cols = 35, class = "form-control")
```

Arguments

id	id of the control
label	label of the control
value	initial value
rows	the # of rows
cols	the # of cols
class	css class

Value

A ‘shiny::tags$div‘ containing a ‘<label>‘ and a ‘<textarea>‘ Shiny-bound input.

Examples

```
x <- textareaInput("genesetarea", "Gene Set",
  "Fgf21",
  rows = 5, cols = 35
)
```

`togglePanels` *togglePanels*

Description

User defined toggle to display which panels are to be shown within DEBrowser.

Usage

```
togglePanels(num = NULL, nums = NULL, session = NULL)
```

Arguments

`num`	selected panel
`nums`	all panels
`session`	session info

Value

invisible(NULL); called for the navbar nav_show / nav_hide / nav_select side effects on the method-tabs nav.

Note

`togglePanels`

Examples

```
x <- togglePanels()
```

`update_progress` *Push a progress update to the browser for one key.*

Description

Sends a Shiny custom message of type "debrowser-progress" with a payload built by ‘progress_message(key, state)‘. The handler installed by ‘getTabUpdateJS()‘ (R/funcs.R) reads the message and toggles CSS classes on the matching icon span and parent pill.

Usage

```
update_progress(session, key, state)
```

Arguments

session	Shiny session object.
key	character, progress key.
state	character, progress state.

Value

Invisibly NULL.

Examples

```
if (interactive()) {
  update_progress(session, "upload", "done")
}
```

Index

∗ **internal**
de_bootswatch_link, 53
de_preset_js, 57
de_preset_picker, 58
fct_upload, 68
parse_preset_cookie, 137

accountDropdownServer, 7
accountDropdownUI, 8
actionButtonDE, 8
addDataCols, 9
addID, 10
ai_chat, 13
ai_interpret, 13
aiInterpretServer, 10
aiInterpretUI, 11
aiSettingsServer, 12
aiSettingsUI, 12
all2all, 14
all2allControlsUI, 15
apply_batch_correction, 17
apply_de_filters, 18
apply_merged_filters, 19
applyFilters, 15
applyFiltersNew, 16
applyFiltersToMergedComparison, 16

barMainPlotControlsUI, 19
batchEffectUI, 20
batchMethod, 20
BoxMainPlotControlsUI, 21

changeClusterOrder, 22
checkCountData, 22
checkMetaData, 23
clusterData, 24
clustFunParamsUI, 24
compareClust, 25
comparison_labels, 28
comparisonConcordanceServer, 26
comparisonConcordanceUI, 27
compute_pill_class, 28
concordance_sets, 29
concordance_summary, 29
condSelectServer, 30
condSelectUI, 31
cooks_outlier_summary, 31, *44*
correctCombat, 32
correctHarman, 33
create_debrowser_user, 33
customColorsUI, 34
cutoff_presets, 35
cutOffSelectionServer, 34
cutOffSelectionUI, 35

dataLCFUI, 36
dataLoadUI, 36
de_assert_count_matrix, 53
de_bootswatch_link, 53
de_card, 54
de_direction_summary, 54
de_error, 55
de_eyebrow, 56
de_headline, 56
de_nav_chip, 57
de_preset_js, 57
de_preset_picker, 58
de_progress_label, 58
de_stat, 59
de_stat_strip, 59
de_style_guide, 60
de_theme, 60
de_workbar, 61
debrowserall2all, 37
debrowserbarmainplot, 37
debrowserbatcheffect, 38
debrowserboxmainplot, 39
debrowserdataload, 39
debrowserdeanalysis, 40
debrowserdensityplot, 41
debrowserheatmap, 41
debrowserhistogram, 42
debrowserIQRplot, 42
debrowserlowcountfilter, 43
debrowsermainplot, 43
debrowserpcaplot, 44
debrowserqccooks, 44
debrowserqcdetectionrate, 45

debrowserqcdispersion, 46
debrowserqclibrarydepth, 46
debrowserqcmtpct, 47
debrowserqcsampledist, 47
debrowserqcsizefactors, 48
default_cutoffs, 49
dendControlsUI, 49
densityPlotControlsUI, 50
deServer, 50
detect_separator, 51
detection_rate, *45*, 51
deUI, 52
distFunParamsUI, 61
drawKEGG, 62
drawPCAExplained, 62

enrichmentGmtServer, 63
enrichmentGmtUI, 64
enrichmentNesHeatmapServer, 64, *133*
enrichmentNesHeatmapUI, 65
enrichmentServer, 65
enrichmentUI, 66
exportMenuItems, 67
exportMenuServer, 67
exportMenuUI, 68

fct_upload, 68
fileTypes, 69
fileUploadBox, 69
filter_low_counts, 70
filter_params_from_input, 70
flag_outliers_2sd, 71
fold_to_log2fc, 72

generateTestData, 72
get_most_varied, 118
get_table_data, 119
getAfterLoadMsg, 73
getAll2AllPlotUI, 73
getBarMainPlot, 74
getBarMainPlotUI, 74
getBoxMainPlot, 75
getBoxMainPlotUI, 76
getBSTableUI, 76
getColors, 77
getColorShapeSelection, 78
getCompSelection, 78
getCondMsg, 79
getCutOffSelection, 80
getDataAssesmentText, 80
getDataForTables, 81
getDataPreparationText, 81
getDEAnalysisText, 82
getDensityPlot, 82
getDensityPlotUI, 83
getDEResultsUI, 83
getDomains, 84
getDown, 84
getDownloadSection, 85
getEnrichDO, 85
getEnrichGO, 86
getEnrichKEGG, 87
getEntrezIds, 87
getEntrezTable, 88
getGeneList, 89
getGeneSetData, 89
getGOLeftMenu, 90
getGoPanel, 90
getGOPlots, 91
getGSEA, 92
getHeatmapUI, 92
getHelpButton, 93
getHideLegendOnOff, 93
getHistogramUI, 94
getIntroText, 94
getIQRPlot, 95
getIQRPlotUI, 95
getJSLine, 96
getKEGGModal, 96
getLeftMenu, 97
getLegendColors, 97
getLegendRadio, 98
getLegendSelect, 98
getLevelOrder, 99
getLogo, 99
getMainPanel, 100
getMainPlotsLeftMenu, 100
getMainPlotUI, 101
getMean, 101
getMergedComparison, 102
getMostVariedList, 102
getNormalizedMatrix, 103
getOrganism, 103
getOrganismBox, 104
getOrganismPathway, 104
getPCAcontolUpdatesJS, 105
getPCAexplained, 105
getPCAPlotUI, 106
getPCselection, 107
getPlotArea, 107
getProgramTitle, 108
getQAText, 108
getQCLeftMenu, 109
getQCPanel, 109
getSampleDetails, 110

getSearchData, 110
getSelectedCols, 111
getSelectedDatasetInput, 111
getSelHeat, 112
getShapeColor, 113
getStartPlotsMsg, 113
getStartupMsg, 114
getTableDetails, 114
getTableModal, 115
getTableStyle, 115
getTabUpdateJS, 116
getUp, 117
getUpDown, 117
getVariationData, 118
gmt_to_pathways, *63*, 120, *131*, *165*

heatmapControlsUI, 121
heatmapJScode, 121
heatmapServer, 122
heatmapUI, 122
hideObj, 123
histogramControlsUI, 123

install_cutoff_preset_observers, 124
IQRPlotControlsUI, 125

kmeansControlsUI, 125

lcfMetRadio, 126
library_depth_summary, *46*, *71*, 126
list_models, 127
log2fc_to_fold, 127

mainPlotControlsUI, 128
mainScatterNew, 128
make_default_metadata, 129
match_preset, 129
merge_comparisons, 130
methods_paragraph, 131
msigdb_pathways, 131
mt_pct_per_sample, *47*, 132

nes_heatmap_data, 133
niceKmeans, 133
normalizationMethods, 134
normalize_counts, 135

palUI, 135
panel.cor, 136
panel.hist, 136
parse_preset_cookie, 137
pcaPlotControlsUI, 137
plot_de_direction_bar, 141
plot_de_pairwise_heatmap, 141
plot_method_scatter, 142
plot_method_upset, 143
plot_pca, 144
plotData, 138
plotMarginsUI, 138
plotSizeMarginsUI, 139
plotSizeUI, 140
plotTypeUI, 140
prepDataContainer, 145
prepGroup, 146
prepHeatData, 147
prepPCADat, 147
progress_message, 148
push, 148

qc_keep_cols, 152, *153*
qc_keep_meta_rows, 153
qcCooksUI, 149
qcDetectionRateUI, 149
qcDispersionUI, 150
qcLibraryDepthUI, 150
qcMtPctUI, 151
qcSampleDistUI, 151
qcSizeFactorsUI, 152

removeCols, 153
removeExtraCols, 154
require_pkg, 154
resend_verification_email, 155
reset_debrowser_password, 155
round_vals, 156
run_de, 161
run_de_methods, 163
run_deseq2, 162
run_edger, 164
run_gsea, *64*, *133*, 165
run_limma, 166
run_pca, 167
runDE, 156
runDESeq2, 157
runEdgeR, 158
runHeatmap, 159
runHeatmap2, 160
runLimma, 160

sample_distance_matrix, *47*, 168
search_geneset, 168
select_dataset, 170
selectGroupInfo, 169
sepRadio, 171
setBatch, 171
showObj, 172
size_factor_library_summary, *48*, 172

startDEBrowser, 173
startHeatmap, 174

textareaInput, 174
togglePanels, 175

update_progress, 175

varianceStabilizingTransformation, *168*
vst, *168*