Package 'gDRplots'

November 25, 2026

Title 'gDRplots'

Version 1.0.1

Date 2026-11-24

Description gDRplots provides static visualization functions and dedicated helpers for the gDR drug response platform. Designed for high-throughput screening (HTS), pharmacogenomics, and chemical genomics research, it enables comprehensive analysis of single-agent, co-dilution, and drug combination cell viability assays. The package supports quality control (plate layouts, control mapping), data normalization inspection, dose-response curves, distribution boxplots, drug synergy heatmaps with overlaid isobolograms, and combination index plots. Additionally, it integrates with PRISM and DepMap datasets to generate volcano plots and correlation scatter plots for molecular feature associations.

License Artistic-2.0

biocViews Software, Visualization, QualityControl, Normalization, CellBasedAssays, Pharmacogenomics, MicrotitrePlateAssay

Imports ashr, checkmate, data.table, DT, fgsea, ggplot2 (>= 4.0.0), ggrepel, ggtext, gDRutils (>= 1.11.18), patchwork, pheatmap, RColorBrewer, rstudioapi, tools, WGCNA

Suggests BiocStyle, colorspace, gDRtestData (>= 1.6.0), gDRstyle (>= 1.6.0), heatmaply, jsonlite, knitr, matrixStats, mockery, purrr, readxl, rlang, R.utils, scales, S4Vectors, SummarizedExperiment, svglite, qs2, testthat, withr

VignetteBuilder knitr

URL https://github.com/gdrplatform/gDRplots, https://gdrplatform.github.io/gDRplots/

BugReports https://github.com/gdrplatform/gDRplots/issues

Encoding UTF-8

LazyLoad yes

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.1

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

git_branch RELEASE_3_24

git_last_commit a9221ae

git_last_commit_date 2026-11-23

Repository Bioconductor 3.24

Date/Publication 2026-11-24

Author Marc Hafner [aut] (ORCID: <https://orcid.org/0000-0003-1337-7598>),
Luca Gerosa [aut] (ORCID: <https://orcid.org/0000-0001-6805-9410>),
Janina Smola [aut] (ORCID: <https://orcid.org/0009-0007-4347-7748>),
Bartosz Czech [aut] (ORCID: <https://orcid.org/0000-0002-9908-3007>),
Dariusz Scigocki [aut],
Arkadiusz Gladki [aut, cre] (ORCID:
<https://orcid.org/0000-0002-7059-6378>),
William Colgan [cph] (cdsrmodels::lin_associations, adapted in
.lin_associations(); see inst/licenses/cdsrmodels-MIT.txt),
Andrew Boghossian [cph] (cdsrmodels::lin_associations, adapted in
.lin_associations(); see inst/licenses/cdsrmodels-MIT.txt)

Maintainer Arkadiusz Gladki <gladki.arkadiusz@gmail.com>

Contents

gDRplots-package 4
.calc_dose_rank 6
.cgs_diff_by_cotrt 6
.cgs_fgsea 7
.create_PRISM_plot_list 7
.describe_fn_sentence 9
.describe_gr 9
.describe_import 10
.describe_masked 10
.describe_plates_without_barcode 11
.describe_provenance_step 11
.describe_settings 12
.fill_pheatmap_annotation 12
.format_dose_list 13
.get_combo_curves_colors 13
.get_combo_panel_type 14
.get_combo_score_breaks 15
.get_data_type 16
.get_excess_palette 17
.get_gradient_colors 17
.get_info_from_name 18
.get_info_from_names 19
.get_iso_colors 19
.get_n_top_asssoc 20
.get_pheatmap_cluster_param 20
.get_pheatmap_fontsize 22
.get_pheatmap_number_color 22
.get_scale_type 23
.get_smooth_palette 24
.get_tile_size 24
.lin_associations 25
.order_assoc_result 26
.prep_dt_OmicsArmLevelCNA 26
.prep_dt_OmicsSomaticMutationsMatrix 27

.resolve_group_colors . . . 28
.safe_as_numeric . . . 29
.stop_on_aggregation . . . 29
.table_to_ggplot . . . 30
.theme_bw . . . 30
.trim_labels . . . 31
analyze_cgs . . . 31
analyze_cgs_conditions . . . 33
assert_RGB_format . . . 34
calc_assoc . . . 34
change_NA_into_char . . . 35
compute_distances . . . 36
create_download_link . . . 37
create_log_seq . . . 38
create_PRISM_plot_list_combo . . . 38
create_PRISM_plot_list_sa . . . 40
create_PRISM_summary_list . . . 42
create_zoom_link . . . 42
escape_special_characters . . . 43
estimate_plot_size . . . 44
fill_ann_color_map . . . 44
generate_color_mappings . . . 45
generate_datatable . . . 46
get_ann_color_map . . . 47
get_col_luminance . . . 47
get_hm_title . . . 48
get_iso_colors . . . 49
get_provenance_tables . . . 49
get_qual_colors . . . 50
get_r_file_path . . . 51
heatmap_combo_metrics . . . 51
heatmap_combo_metrics_panel . . . 53
heatmap_combo_with_isoref . . . 56
heatmap_combo_with_isoref_panel . . . 58
heatmap_combo_with_isoref_panel_common . . . 59
heatmap_combo_with_isoref_panel_independent . . . 62
heatmap_control_mapping_qc . . . 63
is_color_dark . . . 64
is_valid_color . . . 65
neutralize_spaces . . . 66
pheatmap_qc . . . 66
pheatmap_with_anno_cd . . . 68
pheatmap_with_anno_combo . . . 70
pheatmap_with_anno_combo_metrics . . . 73
pheatmap_with_anno_sa . . . 75
plot_boxplot_meta . . . 78
plot_boxplot_metric_combo . . . 80
plot_boxplot_metric_combo_by_CLs . . . 81
plot_boxplot_metric_combo_by_drugs . . . 83
plot_boxplot_metric_combo_by_grp . . . 84
plot_boxplot_metric_sa . . . 86
plot_boxplot_metric_sa_by_CLs . . . 88

plot_boxplot_metric_sa_by_drugs . . . 89
plot_boxplot_metric_sa_by_grp . . . 91
plot_boxplot_num . . . 93
plot_boxplot_num_panel . . . 94
plot_cgs_ranking . . . 95
plot_combination_index . . . 96
plot_dose_response_combo . . . 98
plot_dose_response_combo_panel . . . 99
plot_dose_response_sa . . . 100
plot_dose_response_sa_by_CLs . . . 102
plot_dose_response_sa_by_drugs . . . 104
plot_dose_response_sa_qc . . . 105
plot_dose_response_sa_qc_panel . . . 106
plot_fitting_acc . . . 107
plot_plate . . . 108
plot_plate_stack_info . . . 109
plot_scatter_with_corr . . . 110
plot_scatter_with_corr_panel . . . 111
plot_single_plate_stack_info . . . 112
plot_var_distribution_qc . . . 114
plot_var_stat_qc . . . 115
plot_volcano_assoc . . . 116
plot_volcano_assoc_panel . . . 118
prep_assoc_summary . . . 119
prep_double_table_chunk . . . 120
prep_dt_assoc . . . 121
prep_dt_depmap_feat . . . 122
prep_dt_depmap_meta . . . 123
prep_dt_response_dose_sa . . . 124
prep_dt_response_metric_diff . . . 125
prep_dt_response_metric_sa . . . 126
prep_dt_response_scores . . . 127
prep_filename_path . . . 128
prep_hm_limits . . . 130
prep_nested_plot_chunk . . . 130
prep_pheatmap_matrix . . . 132
prep_plot_chunk . . . 133
round_to_unique_string . . . 135
save_plot . . . 136
transform_log_conc . . . 136

Index **138**

`gDRplots-package` *gDRplots: 'gDRplots'*

Description

gDRplots provides static visualization functions and dedicated helpers for the gDR drug response platform. Designed for high-throughput screening (HTS), pharmacogenomics, and chemical genomics research, it enables comprehensive analysis of single-agent, co-dilution, and drug combination cell viability assays. The package supports quality control (plate layouts, control mapping), data normalization inspection, dose-response curves, distribution boxplots, drug synergy heatmaps with overlaid isobolograms, and combination index plots. Additionally, it integrates with PRISM and DepMap datasets to generate volcano plots and correlation scatter plots for molecular feature associations.

Value

package help page

Note

To learn more about functions start with 'help(package = "gDRplots")'

Author(s)

Maintainer: Arkadiusz Gladki <gladki.arkadiusz@gmail.com> (ORCID)

Authors:

- Arkadiusz Gladki <gladki.arkadiusz@gmail.com> (ORCID)
- Marc Hafner (ORCID)
- Luca Gerosa (ORCID)
- Janina Smola (ORCID)
- Bartosz Czech (ORCID)
- Dariusz Scigocki

Other contributors:

- William Colgan (cdsrmodels::lin_associations, adapted in .lin_associations(); see inst/licenses/cdsrmodels-MIT.txt) [copyright holder]
- Andrew Boghossian (cdsrmodels::lin_associations, adapted in .lin_associations(); see inst/licenses/cdsrmodels-MIT.txt) [copyright holder]

See Also

Useful links:

- https://github.com/gdrplatform/gDRplots
- https://gdrplatform.github.io/gDRplots/
- Report bugs at https://github.com/gdrplatform/gDRplots/issues

.calc_dose_rank *Calculate Dose Ranks*

Description

Helper function to rank unique positive concentration values. Zero values or NAs are assigned a rank of "-".

Usage

```
.calc_dose_rank(vals)
```

Arguments

vals	numeric vector of concentration values.

Value

A character vector of ranks.

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

.cgs_diff_by_cotrt *Reshape co-treatment differences to one row per drug and co-treatment concentration*

Description

prep_dt_response_metric_diff returns one difference column per metric and co-treatment concentration (e.g. RV_gDR_x_mean_cotrt_diff_0.03_drug_1). A cell line screened at one concentration has NA in the columns of the others, so the columns are stacked into one column per metric, with the concentration in column cotrt. Rows without any value at a concentration are dropped.

Usage

```
.cgs_diff_by_cotrt(metrics_diff)
```

Arguments

metrics_diff	data.table returned by prep_dt_response_metric_diff.

Value

data.table with the identifier columns, cotrt and one column per metric.

.cgs_fgsea *Run GSEA on one metric of one cell line (or cell line comparison)*

Description

Drugs without a finite value of the metric are left out, since `fgsea` accepts only finite statistics. When no drug is left, an empty result is returned.

Usage

```
.cgs_fgsea(dt, metric, moa_list, name_col, moa_col, unit_name)
```

Arguments

`dt`	data.table with the metric column, drug names and MOAs.
`metric`	string, the metric column.
`moa_list`	named list of drug names per MOA.
`name_col, moa_col`	strings, the drug name and MOA columns.
`unit_name`	string, the cell line (or comparison) used in messages.

Value

data.table with the `fgsea` result and a `median` column; `leadingEdge` holds the leading-edge drugs as a comma-separated string.

.create_PRISM_plot_list *Create a list of PRISM plots for a selected type of experiment*

Description

This internal function was created to avoid code duplication and maintain consistency. For user, it is contained in a single-purpose function, depending on the experiment type.

Usage

```
.create_PRISM_plot_list(
  experiment_type,
  dt_metrics_sa,
  dt_metrics_combo = NULL,
  dt_average = NULL,
  dt_scores = NULL,
  drug1_name_vec = NULL,
  drug2_name_vec = NULL,
  normalization_type_vec = "RV",
  metric = c("xc50", "x_mean", "x_max"),
  metric_scores = c("hsa_score", "bliss_score"),
```

```
  fit_source = "gDR",
  meta_data_path,
  feat_data_path,
  feature_sets,
  metadata_columns = NULL,
  clear_taxonomy_info = TRUE,
  with_decoding = FALSE
)
```

Arguments

experiment_type	string with the type of experiment one of: "sa" for the single-agent experiment "combo" for the combination experiment
dt_metrics_sa	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay "Metrics") and single-agent SummarizedExperiment
dt_metrics_combo	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay "Metrics") and combo SummarizedExperiment
dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assa "Averaged") and single-agent SummarizedExperiment
dt_scores	data.table representing data from the scores assay, outputted by gDRutils::convert_se_assay_ "scores") and combo SummarizedExperiment
drug1_name_vec	character vector with drug names to be plotted (identifiers DrugName)
drug2_name_vec	character vector with co-drug names to be plotted (identifiers DrugName_2)
normalization_type_vec	character vector with normalization types to be selected one of: "GR" ("GR-value") or "RV" ("RelativeViability") or both
metric	character vector with names of metric; chosen from: "xc50" ("GR50" or "IC50" - respectively depending on normalization_type), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean")
metric_scores	character vector with names of combo metric; chosen from: "hsa_score"("Bliss Excess GR" or "Bliss Excess RV" - respectively depending on normalization_type), "bliss_score" ("Bliss Score GR" or "Bliss Score RV")
fit_source	string source name for metrics
meta_data_path	string path to metadata file describing all cancer models/cell lines which are referenced by a dataset contained within the DepMap portal. It is usually a file named Model.csv or Model.csv.gz.
feat_data_path	string path to the directory containing the molecular feature set file to load from DepMap.
feature_sets	character vector containing the names of the molecular feature sets to load from DepMap. These names should also correspond to the file names containing the feature data (without the extension, which is assumed to be csv or csv.gz)
metadata_columns	character vector with the metadata columns to load for DepMap cell lines
clear_taxonomy_info	logical flag whether to remove taxonomy information for gene names in table with the molecular feature sets from DepMap.
with_decoding	logical whether the feature OmicsArmLevelCNA, OmicsSomaticMutationsMatrixHotspot and OmicsSomaticMutationsMatrixDamaging should be encoded into a 0-1 scheme

Value

A named list with elements:

- `ls_plot` nested list of plots for selected type of experiment
- `ls_assoc_data` nested list of table with association data

Author(s)

Janina Smoła <janina.smola@external.roche.com>

.describe_fn_sentence *Describe a recorded function in a sentence*

Description

Uses, in this order, the description set with `gDRutils::set_fit_fn_info`, the one-line source of a function written inline, or the package, name and version of the function.

Usage

```
.describe_fn_sentence(lead, fn)
```

Arguments

`lead`	string; the start of the sentence.
`fn`	list; a function record from `provenance$steps`, or `NULL`.

Value

A string, or `NULL` when `fn` is `NULL`.

.describe_gr *Describe how GR values were computed in normalization*

Description

Describe how GR values were computed in normalization

Usage

```
.describe_gr(gr)
```

Arguments

`gr`	list with `n_pairs`, a named vector counting the drug x cell line pairs per GR method (`day0`, `ref_div_time`, `none`), and optionally `no_gr_cell_lines`; or `NULL`.

Value

A string, or `NULL` when `gr` is `NULL`.

.describe_import *Describe the import step recorded by gDRimport*

Description

Describe the import step recorded by gDRimport

Usage

```
.describe_import(settings)
```

Arguments

settings	list; the record that gDRimport attaches to the loaded data (see gDRimport::get_import_provenanc gDRimport (version), instrument or, for PRISM data, source$format, files (a table with a role column) and corrections, including plates_without_barcode.

Value

A character vector of sentences, or NULL when settings is NULL.

.describe_masked *Describe the masked wells removed when the SE was created*

Description

Describe the masked wells removed when the SE was created

Usage

```
.describe_masked(masked)
```

Arguments

masked	list with n_wells and, when known, n_pairs (drug x cell line pairs affected) and n_flag_missing (masked wells without a masked flag), or NULL.

Value

A string, or NULL when masked is NULL.

.describe_plates_without_barcode

Describe the plates read without a barcode

Description

Describe the plates read without a barcode

Usage

```
.describe_plates_without_barcode(plates)
```

Arguments

plates	table recorded by gDRimport in corrections$plates_without_barcode: one row per plate, with file, plate and used, the barcode the plate was imported under (NA when it was skipped); or NULL.

Value

A character vector with a sentence for the plates imported under another barcode and one for the skipped plates, or NULL when plates is empty.

.describe_provenance_step

Describe one recorded processing step in a sentence

Description

Steps recorded by gDRimport and gDRcore ("import", "create", "normalize", "average", "growth_rates") have their own description; any other step lists its settings.

Usage

```
.describe_provenance_step(step, entry)
```

Arguments

step	string; name of the step.
entry	list; the step entry from provenance$steps, with settings and, for some steps, functions.

Value

A string.

.describe_settings *List the settings of a step without a dedicated description*

Description

List the settings of a step without a dedicated description

Usage

```
.describe_settings(settings)
```

Arguments

settings	list of settings, or NULL.

Value

A string, cut at 300 characters.

.fill_pheatmap_annotation *Prep annotation data.table acc to metric matrix for pheatmap::pheatmat*

Description

Prep annotation data.table acc to metric matrix for pheatmap::pheatmat

Usage

```
.fill_pheatmap_annotation(
  dt_anno,
  mat_with_metric,
  anno_var = gDRutils::get_env_identifiers("cellline_name")
)
```

Arguments

dt_anno	data.table that specifies the annotations shown on left side of the heatmap or shown above the heatmap - depending on the anno_var. Each row defines the features for a specific row. The rows in the data and in the annotation are matched using corresponding names from the required anno_var column.
mat_with_metric	numeric matrix with metric values; must have named rows and columns
anno_var	string with variable describing annotation dimension: one of: CellLineName for rows or DrugName for column.

Value

data.table with annotation updated to mat_with_metric

Author(s)

Janina Smoła <janina.smola@external.roche.com>

.format_dose_list *Format Dose List for Legend*

Description

Helper function to create a formatted string of dose ranks and values for the plot legend, breaking lines based on 'n_per_line'.

Usage

```
.format_dose_list(dose_vec, n_per_line)
```

Arguments

dose_vec	numeric vector of unique positive doses.
n_per_line	integer number of items to display per line.

Value

A single character string with HTML line breaks.

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

.get_combo_curves_colors *Get color palette for the dose response curves for combination data*

Description

Get color palette for the dose response curves for combination data

Usage

```
.get_combo_curves_colors(ls_conc_2)
```

Arguments

ls_conc_2	factor vector with values for Concentration_2

Value

gDR palette for Concentration_2 given in ls_conc_2

Examples

```
ls_conc <- factor(c("0.001", "0.01", "1"))
gDRplots:::.get_combo_curves_colors(ls_conc)
```

`.get_combo_panel_type` *Check type of concentrations set for combination of drug per cell line*

Description

This function checks if the concentration vectors per cell line have common part or not. It is required to decide which function to use for plotting the heatmaps panel (the set of heatmaps with combo metrics plot for combination of selected drug with selected codrug, each heatmap per one cell line): `heatmap_combo_with_isoref_panel_common` that plot heatmaps with shared axes or `heatmap_combo_with_isoref_panel_independent` that plot heatmaps independently.

Usage

```
.get_combo_panel_type(ls_vec_conc)
```

Arguments

`ls_vec_conc`	a list with vectors with concentration per cell line

Details

Possible combinations (0 is not taken into account as it is always present):

- `common`
  - all vectors have common part and start and end conc are the same
    0, 0.03, 0.1, 0.3, 1
  - all vectors have common part and start and end conc are the same, but there are some gap
    0, 0.003, 0.01, 0.03, 0.1, 0.3, 1
    0, 0.003, 0.01, 0.03, ___, 0.3, 1
  - one vector has all concentrations and the other vectors have gaps, also at the start or end conc
    0, 0.003, 0.01, 0.03, 0.1, 0.3, 1
    0, 0.003, 0.01, 0.03, ___, 0.3, 1
    0, 0.003, 0.01, 0.03, 0.1, 0.3, _
- `independent`
  - all vectors have common part and start conc are the same, but end conc is different (no gap)
    0, 0.003, 0.01, 0.03, 0.1, 0.3, __
    0, 0.003, 0.01, 0.03, 0.1, 0.3, 1

- no common part between vectors
  0, 0.003, 0.01, 0.03, __, __, __
  0, ____, ____, ____, 0.1, 0.3, 1

- all vectors have common part but start and end conc are different (shifted range)
  0, 0.003, 0.01, 0.03, 0.1, 0.3, 1, __, __
  0, ____, ____, 0.03, 0.1, 0.3, 1, 3, 10

Value

a string decribing type of concentration list - one of:

- `common` vectors in the input list have common part; heatmaps in the panel should be created jointly with function `heatmap_combo_with_isoref_panel_common`
- `independent` vectors in the input list do not have common part; heatmaps in the panel should be created independently with function `heatmap_combo_with_isoref_panel_independent`

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
ls_conc <- list(c(0, 0.003, 0.01, 0.03), c(0, 0.003, 0.01, 0.03, 0.1))
gDRplots:::.get_combo_panel_type(ls_conc)
```

`.get_combo_score_breaks` *Color breaks for a combo score heatmap: a fixed scale for Bliss and HSA, otherwise symmetric around 0 and spanning the finite values of the matrix*

Description

Color breaks for a combo score heatmap: a fixed scale for Bliss and HSA, otherwise symmetric around 0 and spanning the finite values of the matrix

Usage

```
.get_combo_score_breaks(mat, metric, no_breaks)
```

`.get_data_type` *Check data type*

Description

Check data type

Usage

```
.get_data_type(dt_, desc_col = NULL)
```

Arguments

`dt_`	`data.table` with dependent variables data in the wide format, where rows are samples, columns are feature levels
`desc_col`	a character vector with column names describing the data and which do not contain data itself

Value

a string describing type of data - "numeric" or "categorical"

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
tab_cat <- data.table::data.table(
  ID = sprintf("ID_%s", seq_len(5)),
  brown = c(0, 1, 1, 0, 0),
  blue = c(1, 0, NA, 0, 1),
  green = c(0, 0, 0, 1, 0)
)
gDRplots:::.get_data_type(dt_ = tab_cat, desc_col = "ID")

tab_feat <- data.table::data.table(
  ID = sprintf("ID_%s", seq_len(5)),
  grp = LETTERS[seq_len(5)],
  low = c(0, 1, 1, NA, 0),
  med = c(1, 1, NA, 0, 1),
  high = c(0, 1, 0, 1, 0)
)
gDRplots:::.get_data_type(dt_ = tab_feat, desc_col = c("ID", "grp"))
```

.get_excess_palette *Get color palette for the excess values*

Description

Get color palette for the excess values

Usage

```
.get_excess_palette(no_breaks)
```

Arguments

no_breaks	numeric number of breaks on scale

Value

gDR palette for excess values with given no_breaks

Examples

```
gDRplots:::.get_excess_palette(20)
```

.get_gradient_colors *Generate Gradient Colors for Dose Legends*

Description

Internal helper to create a named vector of colors. Each drug is assigned a base color, and individual concentrations are assigned a gradient intensity of that color.

Usage

```
.get_gradient_colors(
  col_names,
  col_concs,
  col_labels,
  base_map,
  untreated_tag = gDRutils::get_env_identifiers("untreated_tag")
)
```

Arguments

col_names	character vector of drug names corresponding to the data rows.
col_concs	numeric or character vector of concentrations.
col_labels	character vector of labels to be used as names for the returned colors (e.g., "Drug 0.5").
base_map	character named vector mapping drug names to their base (darkest) color.
untreated_tag	character vector of identifiers for untreated/vehicle controls. Defaults to gDRutils::get_env_iden

Value

A named `character` vector of hex color codes.

`.get_info_from_name` *Help function*

Description

This function retrieves information about the feature name and drug name from a file name, which has the schema <chunk_name>__<feature_name>_<drug_name>_<gDR_metric_name> Assumption: - <feature_name> is assumed not to contain any underscores (usually written in camel case) - <gDR_metric_name> starts with normalization type that is `RV` or `GR`

Usage

```
.get_info_from_name(file_name, normalization_type = "RV")
```

Arguments

`file_name`	A string with file name
`normalization_type`	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")

Value

A named list with elements:

- `drug_grid` a string with drug name
- `feat_meta` a string with omic name (feature or meta)

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
f_n <- "name_chunk__FEAT_DRUG_ABC_RV_gDR_log10_xc50.xlsx"
gDRplots:::.get_info_from_name(f_n)
```

.get_info_from_names *Extract drug grid and feature metadata from file names*

Description

Extract drug grid and feature metadata from file names

Usage

```
.get_info_from_names(file_names, normalization_type = "RV")
```

Arguments

file_names	character vector of file names following the pattern prefix__feat_drug_NORM_...
normalization_type	string normalization type ("GR" or "RV")

Value

named list with drug_grid and feat_meta

Examples

```
gDRplots:::.get_info_from_names("plt__metricA_DrugX_RV_extra", "RV")
```

.get_iso_colors *Get color palette for the isobologram levels*

Description

Get color palette for the isobologram levels

Usage

```
.get_iso_colors(iso_levels)
```

Arguments

iso_levels	character vector with isobologram levels

Value

gDR palette for isoline given in iso_levels

Examples

```
ls_iso_lvl <- c("0.25", "0.5", "0.75")
gDRplots:::.get_iso_colors(ls_iso_lvl)
```

.get_n_top_asssoc *Get n-top linear associations*

Description

Currently, n-top linear associations are selected based on values of `q_value` (column sorted increasing) and values of `rho` (column sorted decreasing)

Usage

```
.get_n_top_asssoc(dt_assoc, n_top = 4)
```

Arguments

`dt_assoc`	`data.table` with the calculated linear association between DepMap and metrics outputted by `calc_assoc`
`n_top`	number of requested top linear associations

Value

a vector with name of the n-top linear associations; when `n_top` will be higher than number of available features - only available will be returned.

Examples

```
Y <- matrix(seq(0.5, 2, length.out = 50), nrow = 50,
            dimnames = list(sprintf("row_%s", 1:50), "met_1"))
X <- matrix(
  withr::with_seed(42, sample(c(NA, seq(0.35, 23.5, 1.25)),
                              size = 50*20, replace = TRUE)),
  nrow = 50,
  dimnames = list(sprintf("row_%s", 1:50), sprintf("feat_%s", 1:20)))
tab_assoc <- calc_assoc(X, Y)
gDRplots:::.get_n_top_asssoc(tab_assoc)
```

.get_pheatmap_cluster_param *Compute value of param cluster_rows or cluster_cols in pheatmap::pheatmap*

Description

The `cluster_rows` and `cluster_cols` parameters pheatmap::pheatmap may take values: - boolean determining if rows/columns should be clustered - `hclust` object, this function allows to calculate proper value taking into account matrix, additional condition and selected function used to compute the distance in `hclust` object

Usage

```
.get_pheatmap_cluster_param(
  mat_to_cluster,
  distfun = stats::dist,
  additional_condition = TRUE,
  na_rows_last = FALSE
)
```

Arguments

mat_to_cluster	numeric matrix to be clustered; cluster dimension must be named
distfun	function used to compute the distance (dissimilarity) between rows; defaults to stats::dist using euclidean euclidean.
additional_condition	additional logical flag whether rows/columns should be clustered
na_rows_last	logical flag; if TRUE, rows with only NAs are left out when computing distances and join the tree at a height above all other rows

Details

To compute the correct value when clustering columns - use the transposed source matrix as mat_to_cluster

Value

logical flag determining if rows should be clustered or hclust object.

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

compute_distances

Examples

```
mat <- matrix(1:24, nrow = 4)
rownames(mat) <- sprintf("row_%s", 1:4)
colnames(mat) <- sprintf("col_%s", 1:6)
gDRplots:::.get_pheatmap_cluster_param(mat)
gDRplots:::.get_pheatmap_cluster_param(t(mat))
gDRplots:::.get_pheatmap_cluster_param(t(mat), distfun = gDRplots::compute_distances)

mat[2,2] <- NA
mat[2,1] <- Inf
gDRplots:::.get_pheatmap_cluster_param(mat)
gDRplots:::.get_pheatmap_cluster_param(mat, distfun = gDRplots::compute_distances)
gDRplots:::.get_pheatmap_cluster_param(t(mat), distfun = gDRplots::compute_distances)
add_cond <- NCOL(mat) > 10
gDRplots:::.get_pheatmap_cluster_param(mat, distfun = gDRplots::compute_distances,
                                       additional_condition = add_cond)
```

.get_pheatmap_fontsize *Get fontsize for rownames or colnames in pheatmap::pheatmap*

Description

Get fontsize for rownames or colnames in pheatmap::pheatmap

Usage

```
.get_pheatmap_fontsize(
  matrix,
  dimension = c("row", "col"),
  threshold_count = 40L
)
```

Arguments

matrix	numeric matrix with metric values.
dimension	character value, either "row" or "col", indicating whether to compute fontsize for rows or columns.
threshold_count	integer value of the number of rows/columns for which the font size remains standard.

Value

numeric value of font size.

Author(s)

Janina Smoła <janina.smola@external.roche.com>

.get_pheatmap_number_color *Compute color for number font in pheatmap::pheatmap based on given color palette and breaks*

Description

Compute color for number font in pheatmap::pheatmap based on given color palette and breaks

Usage

```
.get_pheatmap_number_color(
  mat_with_metric,
  colors_vec,
  breaks,
  dark_color_font = "white",
  light_color_font = "black"
)
```

Arguments

mat_with_metric	numeric matrix with metric values; must have named rows and columns
colors_vec	character vector of colors (valid name or hex) used in heatmap; must to be one item shorter than no_breaks
breaks	numeric vector of breaks on scale used for mapping values to colors
dark_color_font	string with valid color name of font for field with dark background
light_color_font	string with valid color name of font for field without dark background

Value

named matrix with number color

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mat <- matrix(-14:30, ncol = 5,
              dimnames = list(letters[1:9], LETTERS[1:5]))
no_breaks <- 15
breaks <- seq(from = min(mat), to = max(mat), length.out = no_breaks + 1)
ls_colors <- c("limegreen", "darkblue", "orange")
hm_colors <- grDevices::colorRampPalette(ls_colors)(no_breaks)

number_color <- gDRplots:::.get_pheatmap_number_color(mat, hm_colors, breaks)

pheatmap::pheatmap(mat,
                   breaks = breaks,
                   color = hm_colors,
                   display_numbers = TRUE,
                   number_color = number_color,
                   cluster_rows = FALSE,
                   cluster_cols = FALSE)
```

.get_scale_type *Get the scale type of a native combo metric*

Description

Get the scale type of a native combo metric

Usage

```
.get_scale_type(metric)
```

Arguments

metric	string, one of: "smooth", "hsa_excess" or "bliss_excess"

Value

"smooth" for "smooth", "excess" otherwise

.get_smooth_palette *Get color palette for the smooth values*

Description

Get color palette for the smooth values

Usage

```
.get_smooth_palette(no_breaks)
```

Arguments

no_breaks	numeric number of breaks on scale

Value

gDR palette for smooth values with given no_breaks

Examples

```
gDRplots:::.get_smooth_palette(25)
```

.get_tile_size *Calculate the size of tiles based on pos_x/pos_y values*

Description

Since ggplot2::geom_tile uses the center of the tile and its size (x, y, width, height), x and y are given as pos_x and pos_y, it is required to calculate the width and height.

Usage

```
.get_tile_size(pos_vec)
```

Arguments

pos_vec	numeric vector with pos_x or pos_y values

Value

size of tile in ggplot2::geom_tile

`.lin_associations` *Compute pairwise linear associations between columns of X and Y*

Description

Adapted from `lin_associations` in the cdsrmodels package (MIT License; authors William Colgan and Andrew Boghossian) to remove a GitHub-only package dependency. The license text and copyright notice are in `inst/licenses/cdsrmodels-MIT.txt` (`system.file("licenses", "cdsrmodels-MIT.txt", package = "gDRplots")`). Uses `WGCNA::cor` for fast pairwise correlation on large matrices and `ashr::ash` for empirical-Bayes shrinkage of effect sizes.

Usage

```
.lin_associations(
  X,
  Y,
  n.min = 4L,
  shrinkage = TRUE,
  alpha = 0,
  MHC_direction = NULL
)
```

Arguments

`X`	`matrix` of independent variables (rows = samples, cols = features).
`Y`	`matrix` or `vector` of response variables (rows = samples).
`n.min`	integer; minimum number of degrees of freedom (finite paired observations minus 2) required to compute a p-value (default 4, i.e. at least 6 paired observations).
`shrinkage`	logical; apply `ashr` shrinkage (default `TRUE`).
`alpha`	numeric; `ashr` alpha parameter (default 0).
`MHC_direction`	character; `"x"` or `"y"` — direction of multiple-hypothesis correction (Benjamini-Hochberg). With `"y"`, p-values are adjusted across the features of `X` separately for each column of `Y`; with `"x"`, across the columns of `Y` separately for each feature of `X`. Defaults to `"x"` when `length(Y) >= length(X)`, otherwise `"y"` (same default as `cdsrmodels::lin_associations`).

Value

Named list with elements `N`, `rho`, `beta`, `beta.se`, `p.val`, `q.val`, and `res.table` (a `data.frame` from `ashr`).

.order_assoc_result *Order and rename columns in associations results*

Description

Order and rename columns in associations results

Usage

```
.order_assoc_result(res_dt)
```

Arguments

res_dt	data.table of associations results

Value

data.table of associations results with reorder and rename columns

.prep_dt_OmicsArmLevelCNA *Encode OmicsArmLevelCNA as not mutated and mutated*

Description

OmicsArmLevelCNA is arm-level copy number alteration inferred using absolute copy number data from PureCN, method from the Ben-David et al. (2021) paper (https://www.nature.com/articles/s41586-020-03114-6). Chromosome arms: *1* indicates arm-level gain, *-1* indicates arm-level loss, and *0* indicates copy-neutral.

Usage

```
.prep_dt_OmicsArmLevelCNA(dt_depmap)
```

Arguments

dt_depmap	data.table with dependent variables data load from DepMap. (rows are samples, columns are features or meta); outputted by one of prep_dt_depmap_feat for OmicsArmLevelCNA

Details

This function transform each chromosome column (e.g., 3p) into two new binary columns: 3p_loss and 3p_gain. 3p_loss is *1* for values of *-1* in the original column and *0* otherwise. 3p_gain is *1* for values of *1* in the original column and *0* otherwise. The original chromosome column is then removed.

Value

data.table with OmicsArmLevelCNA decoded as mutated - not mutated

Author(s)

Janina Smoła <janina.smola@external.roche.com>

`.prep_dt_OmicsSomaticMutationsMatrix` *Binarize somatic mutations in OmicsSomaticMutationsMatrixHotspot and OmicsSomaticMutationsMatrixDamaging*

Description

OmicsSomaticMutationsMatrixHotspot is genotyped matrix determining for each cell line whether each gene has at least one hot spot mutation. A variant is considered a hot spot if it's present in one of the following: Hess et al. (2019) paper, OncoKB hotspot, COSMIC mutation significance tier 1.

Usage

```
.prep_dt_OmicsSomaticMutationsMatrix(dt_depmap)
```

Arguments

`dt_depmap`	`data.table` with dependent variables data load from DepMap. (rows are samples, columns are features or meta); outputted by one of `prep_dt_depmap_feat` for OmicsSomaticMutationsMatrixHotspot or OmicsSomaticMutationsMatrixDamaging

Details

OmicsSomaticMutationsMatrixDamaging is genotyped matrix determining for each cell line whether each gene has at least one damaging mutation. A variant is considered a damaging mutation if LikelyLoF is True

0 means no mutation; if there is one or more hot spot mutations or damaging mutations respectively, in the same gene for the same cell line, the allele frequencies are summed, and if the sum is greater than 0.95, a value of *2* is assigned (representing a likely homozygous mutation), otherwise a value of *1* is assigned (likely heterozygous).

This function transforms each gene column into binary columns: *0* indicates no mutation, and *1* indicates mutation, regardless of zygosity (the original value was *1* or *2*).

Value

`data.table` with OmicsSomaticMutationsMatrixHotspot or OmicsSomaticMutationsMatrixDamaging decoded as not mutated and mutated

Author(s)

Janina Smoła <janina.smola@external.roche.com>

`.resolve_group_colors` *Resolve a color mapping for group names*

Description

Builds a named color vector for `group_names`, resolving `colors_vec` by name wherever possible so each color maps to its intended group regardless of the order or length of the inputs.

Usage

```
.resolve_group_colors(group_names_requested, group_names, colors_vec = NULL)
```

Arguments

`group_names_requested`	character vector of group names in their originally requested order, used to pair an unnamed `colors_vec` by name; `NULL` if no `group_names` were requested. Duplicates are ignored (the first occurrence wins).
`group_names`	character vector (no duplicates) of group names to produce a color for, in their required order
`colors_vec`	character vector of colors (name or hex value). If fully named (e.g. `c(cell_DC = "blue")`), matched to `group_names` by name, independent of order; otherwise matched positionally against `group_names_requested` (partial naming is treated as unnamed). Invalid, missing, or extra entries fall back to the default palette (`get_qual_colors`); nothing is recycled.

Value

named character vector of colors, one per entry of `group_names`

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
gDRplots:::.resolve_group_colors(
 group_names_requested = c("cell_DC", "cell_XY", "cell_AB"),
 group_names = c("cell_AB", "cell_DC", "cell_XY"),
 colors_vec = c("blue", "red", "grey"))

gDRplots:::.resolve_group_colors(
 group_names_requested = c("cell_DC", "cell_XY", "cell_AB"),
 group_names = c("cell_AB", "cell_DC", "cell_XY"),
 colors_vec = c(cell_DC = "blue", cell_XY = "red", cell_AB = "grey"))

# missing entries fall back to the default palette
gDRplots:::.resolve_group_colors(
 group_names_requested = c("cell_DC", "cell_XY", "cell_AB"),
 group_names = c("cell_AB", "cell_DC", "cell_XY"),
 colors_vec = c("blue", "red"))
```

```
# partial naming is treated as unnamed
gDRplots:::.resolve_group_colors(
 group_names_requested = c("cell_DC", "cell_XY", "cell_AB"),
 group_names = c("cell_AB", "cell_DC", "cell_XY"),
 colors_vec = c(cell_AB = "grey", "blue", "red"))
```

.safe_as_numeric *Safely convert a character vector to numeric*

Description

Converts only values that match a numeric pattern to numeric; others become NA.

Usage

```
.safe_as_numeric(x)
```

Arguments

x	character vector to convert

Value

numeric vector

Examples

```
gDRplots:::.safe_as_numeric(c("0.01", "0.1", "untreated", NA))
```

.stop_on_aggregation *stop wrapper for 'data.table::dcast' to handle unexpected aggregation*

Description

idea from: https://github.com/Rdatatable/data.table/issues/5386

Usage

```
.stop_on_aggregation(fname, formula)
```

Arguments

fname	string with the name of the function that failed to the data.table::dcast aggregation
formula	string with the formula used in data.table::dcast

Value

NULL

Author(s)

Arkadiusz Gladki <arkadiusz.gladki@external.roche.com>

.table_to_ggplot *Render a data.table as a text-based ggplot table*

Description

Render a data.table as a text-based ggplot table

Usage

```
.table_to_ggplot(dt, base_size = 10, digits = 3)
```

Arguments

dt	data.table to render
base_size	numeric base font size (default 10)

Value

ggplot object displaying the table

Examples

```
dt <- data.table::data.table(drug = c("A", "B"), r2 = c(0.9, 0.8))
gDRplots:::.table_to_ggplot(dt)
```

.theme_bw *Get* ggplot2::theme_bw(), *built once per session*

Description

Get ggplot2::theme_bw(), built once per session

Usage

```
.theme_bw()
```

Value

ggplot2::theme_bw()

.trim_labels *Trim labels to the required number of characters*

Description

Trim labels to the required number of characters

Usage

```
.trim_labels(lbls_vec, max_lbl_length = Inf)
```

Arguments

lbls_vec	character vector with labels to be trimmed
max_lbl_length	numeric value for the maximum number of characters in the label; if set to Inf, no trimming will be performed

Value

character vectors with trimmed labels

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
ls_lbls <- c(
  "short_lbl", "short_lbl", "veryveryverylong",
  "long_duplicates|lbl_1AB", "long_duplicates|lbl_1AB", "long_duplicates|lbl_123"
)

gDRplots:::.trim_labels(lbls_vec = ls_lbls)
gDRplots:::.trim_labels(lbls_vec = ls_lbls, max_lbl_length = 15)
```

analyze_cgs *Analyze Chemical Genomics (CGS) Data and Perform GSEA*

Description

This function analyzes chemical genomics (CGS) data, filters by compound mechanism of action (MOA), prepares the data for Gene Set Enrichment Analysis (GSEA), and performs GSEA for specified cell lines and metrics.

Usage

```
analyze_cgs(
  dt_metrics,
  metrics,
  cl_name = NULL,
  cellline1 = NULL,
  cellline2 = NULL,
  normalization_type = "RV"
)
```

Arguments

`dt_metrics`	A data.table containing screening data. Requires columns: 'drug_moa', 'DrugName', 'CellLineName', and columns specified in the 'metrics' argument.
`metrics`	A character vector specifying the response metrics to analyze: "x_mean", "x_AOC_range", "xc50", "ec50", "x_max".
`cl_name`	An optional string specifying a single cell line to analyze. If NULL (default), all cell lines in the data are analyzed. Should be NULL if `cellline1` and `cellline2` are provided.
`cellline1`	An optional string representing the first cell line name. Should be specified alongside `cellline2`.
`cellline2`	An optional string representing the second cell line name. Should be specified alongside `cellline1`.
`normalization_type`	A string with normalization types to be selected, one of: "GR" ("GRvalue") or "RV" ("RelativeViability"). Passed to `prep_dt_response_metric_diff`.

Details

Each cell line is analysed at each co-treatment concentration it was screened at, so cell lines may use different concentrations. Drugs without a finite value of a metric are left out of the GSEA for that metric.

Value

A list of results, where each element corresponds to a cell line or cell line difference. A cell line screened at several co-treatment concentrations gets one element per concentration, named `"<cell line> (co-treatment <concentration>_<dilution drug>)"`. Each result contains:

- `fgsea`: GSEA results for the specified metrics,
- `metrics_diff`: The prepped data.table used for the GSEA analysis,
- `moa_list`: A list of DrugNames grouped by drug_moa.

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
dt_metrics <- qs2::qs_read(system.file("testdata/cgs_data.qs2", package = "gDRplots"))
analyze_cgs(dt_metrics, metrics = "xc50", cl_name = "CellLineName_1")
cellline1 <- "CellLineName_1"
cellline2 <- "CellLineName_2"
analyze_cgs(dt_metrics, "xc50", cl_name = NULL, cellline1, cellline2)
```

`analyze_cgs_conditions` *Compare two screening conditions of a chemical genomics screen and perform GSEA*

Description

Compares the response to each library drug between two conditions, each a cell line at one co-treatment concentration (`0` for the drug alone), and runs GSEA over the drug classes (`drug_moa`). The conditions may use different cell lines and different co-treatment concentrations, for example a resistant line kept in the anchor drug against its parental line in standard medium. The difference is `condition1` minus `condition2`.

Usage

```
analyze_cgs_conditions(
  dt_metrics,
  condition1,
  condition2,
  metrics = "x_mean",
  normalization_type = "RV"
)
```

Arguments

`dt_metrics`	A data.table of combination metrics, as for `analyze_cgs`, with `cotrt_value`.
`condition1, condition2`	Named lists with `cell_line` (a value of `CellLineName`) and `cotrt_value` (a co-treatment concentration of that cell line, `0` for the drug alone).
`metrics`	A character vector of metrics to compare: "x_mean", "x_AOC_range", "xc50", "ec50", "x_max". `xc50` is compared on the log10 scale.
`normalization_type`	A string, "RV" or "GR".

Value

A list of one element, named `"<condition1> vs. <condition2>"`, with `fgsea`, `metrics_diff` and `moa_list` as returned by `analyze_cgs`, so `plot_cgs_ranking` can draw it.

Examples

```
dt_metrics <- qs2::qs_read(system.file("testdata/cgs_data.qs2", package = "gDRplots"))
analyze_cgs_conditions(dt_metrics,
                       condition1 = list(cell_line = "CellLineName_1", cotrt_value = 1),
                       condition2 = list(cell_line = "CellLineName_2", cotrt_value = 0))
```

---

assert_RGB_format *Assert whether number may code color in rgb*

Description

Assert whether number may code color in rgb

Usage

```
assert_RGB_format(x)
```

Arguments

x	numeric vector describing rgb color

Value

x invisible, if the check is not successful throws an error message.

---

calc_assoc *Calculate linear associations*

Description

Calculate the linear model associations between dependent variables and response variable(s) of interest.

Usage

```
calc_assoc(X, Y)
```

Arguments

X	matrix dependent variables data matrix (rows are samples, columns are features). Must have the same number of rows as matrix Y or equal to length of vector Y
Y	vector or matrix experimental response data (rows are samples). When Y is a matrix must have the same number of rows as matrix X; when y is a vector - its length has to be equal to number of rows in matrix X.

Value

data.table with calculated linear associations

Note

inspired by the calc_assoc function written by James Hawley

Examples

```
X <- matrix(rep(1:13, length.out = 42), nrow = 6,
            dimnames = list(sprintf("row_%s", 1:6), sprintf("feat_%s", 1:7)))
Y <- matrix(c(10:15, 110:115, 210:215), ncol = 3,
            dimnames = list(sprintf("row_%s", 1:6), sprintf("met_%s", 1:3)))
tab_assoc <- calc_assoc(X, Y)
```

---

change_NA_into_char *Change NA into given string*

Description

Change NA into given string

Usage

```
change_NA_into_char(x, lbl_NA = "NA")
```

Arguments

x	vector with items suspected of being NA
lbl_NA	string - replacement for NA - as default "NA"

Value

character (for NA -> given string)

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
change_NA_into_char(c(1, NA, 3))
change_NA_into_char(c("a", NA, "b"), lbl_NA = "missing")
```

compute_distances *Compute distance between rows of a matrix.*

Description

This function was lifted from the package factoextra and slightly adjusted. The default method was changed to "spearman" and an option was added to replace missing values in the resulting distance matrix with an arbitrary value. This option defaults to 0 so that the function can be called by iheatmapr::add_row_clustering and iheatmapr::add_col_clustering.

Usage

```
compute_distances(
  x,
  method = "spearman",
  use = "pairwise.complete.obs",
  stand = FALSE,
  dummy = 0,
  ...
)
```

Arguments

x	numeric matrix
method	character string specifying the distance measure to be used; must be on of: "euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski", "pearson", "spearman" or "kendall"
use	character string specifying the method for computing covariances in the presence of missing values. Only used when "pearson", "spearman" or "kendall" chosen as a distance measure
stand	logical flag specifying whether the data should be standardized; if TRUE, columns are converted to z scores using scale
dummy	value to substitute for missing values; defaults to 0
...	arguments passed to internally called functions

Value

Object of class dist. NA and NaN are substituted by the value of dummy.

Control

The function offers different distance measures but since it is called internally, it will always use default arguments. As a result, should you want to do anything other than compute Spearman correlation distance and replace NAs with 0, you must edit this function definition accordingly. At the moment (December 2020) this is sufficient. If the control were to be given to the user, remove defaults for this definition and create a wrapper within plotly_metric_clustering.

The function was originally named computeDistances

Author(s)

Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

See Also

[factoextra::get_dist()]

Examples

```
x <- matrix(1:9, nrow = 3, ncol = 3)
rownames(x) <- letters[seq(NROW(x))]
compute_distances(x)
```

---

create_download_link *Prepare markdown chunk with download link*

Description

Generate markdown code with a html link item that, when clicked, downloads a file. The function output should be wrapped in `knitr::knit()`.

Usage

```
create_download_link(dwn_path, link_txt = "Download Table")
```

Arguments

dwn_path	string with relative path to file with plot to be downloaded
link_txt	string with text describing link

Value

string with html download code

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

knitr::knit

Examples

```
create_download_link("tables/results.xlsx")
create_download_link(NA)
```

create_log_seq *create a log-sequence*

Description

Create a sequence of numbers growing in log-domain.

Usage

```
create_log_seq(start, end, length)
```

Arguments

start, end	numeric, lower and upper margins of the sequence
length	integer, resulting sequence length

Details

The result is a numeric vector of length length. Differences between items are constant in logarithmic domain and therefore geometrically increase in linear domain.

Value

A numeric vector, see Details.

Examples

```
create_log_seq(1, 2, 5)
```

create_PRISM_plot_list_combo *Create a nested list of plots for PRISM data with combo metrics*

Description

Create a nested list of plots for PRISM data with combo metrics

Usage

```
create_PRISM_plot_list_combo(
  drug1_name_vec,
  drug2_name_vec,
  dt_metrics,
  dt_scores = NULL,
  normalization_type_vec = "RV",
  metric = c("xc50", "x_mean", "x_max"),
  metric_scores = c("hsa_score", "bliss_score"),
  fit_source = "gDR",
  meta_data_path,
```

```
  feat_data_path,
  feature_sets,
  metadata_columns = NULL,
  clear_taxonomy_info = TRUE,
  with_decoding = FALSE
)
```

Arguments

drug1_name_vec	character vector with drug names to be plotted (identifiers DrugName)
drug2_name_vec	character vector with co-drug names to be plotted (identifiers DrugName_2)
dt_metrics	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay "Metrics") and combo SummarizedExperiment
dt_scores	data.table representing data from the scores assay, outputted by gDRutils::convert_se_assay_ "scores") and combo SummarizedExperiment
normalization_type_vec	character vector with normalization types to be selected one of: "GR" ("GR-value") or "RV" ("RelativeViability") or both
metric	character vector with names of metric; chosen from: "xc50" ("GR50" or "IC50" - respectively depending on normalization_type), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean")
metric_scores	character vector with names of combo metric; chosen from: "hsa_score"("Bliss Excess GR" or "Bliss Excess RV" - respectively depending on normalization_type), "bliss_score" ("Bliss Score GR" or "Bliss Score RV")
fit_source	string source name for metrics
meta_data_path	string path to metadata file describing all cancer models/cell lines which are referenced by a dataset contained within the DepMap portal. It is usually a file named Model.csv or Model.csv.gz.
feat_data_path	string path to the directory containing the molecular feature set file to load from DepMap.
feature_sets	character vector containing the names of the molecular feature sets to load from DepMap. These names should also correspond to the file names containing the feature data (without the extension, which is assumed to be csv or csv.gz)
metadata_columns	character vector with the metadata columns to load for DepMap cell lines
clear_taxonomy_info	logical flag whether to remove taxonomy information for gene names in table with the molecular feature sets from DepMap.
with_decoding	logical whether the feature OmicsArmLevelCNA, OmicsSomaticMutationsMatrixHotspot and OmicsSomaticMutationsMatrixDamaging should be encoded into a 0-1 scheme

Value

A named list with elements:

- ls_plot nested list of plots for selected type of experiment
- ls_assoc_data nested list of table with association data

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- qs2::qs_read(system.file("testdata/finalMAE_combo_matrix_small.qs2",
                                package = "gDRtestData"))
se_combo <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se_combo, "Metrics")
drug1 <- unique(dt_metrics[[gDRutils::get_env_identifiers("drug_name")]])[1]
drug2 <- unique(dt_metrics[[gDRutils::get_env_identifiers("drug_name2")]])[1]
create_PRISM_plot_list_combo(
  drug1_name_vec = drug1,
  drug2_name_vec = drug2,
  dt_metrics = dt_metrics,
  meta_data_path = system.file("depmap_data/Model.csv.gz", package = "gDRtestData"),
  feat_data_path = system.file("depmap_data", package = "gDRtestData"),
  feature_sets = c("CRISPRGeneEffect")
)
```

---

create_PRISM_plot_list_sa

Create a nested list of plots for PRISM data with single-agent metrics

---

Description

Create a nested list of plots for PRISM data with single-agent metrics

Usage

```
create_PRISM_plot_list_sa(
  drug_name_vec,
  dt_metrics,
  dt_average = NULL,
  normalization_type_vec = "RV",
  metric = c("xc50", "x_mean", "x_max"),
  fit_source = "gDR",
  meta_data_path,
  feat_data_path,
  feature_sets,
  metadata_columns = NULL,
  clear_taxonomy_info = TRUE,
  with_decoding = FALSE
)
```

Arguments

drug_name_vec	character vector with drug names to be plotted (identifiers DrugName)
dt_metrics	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay "Metrics") and single-agent SummarizedExperiment

dt_average data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assa "Averaged") and single-agent SummarizedExperiment

normalization_type_vec
character vector with normalization types to be selected one of: "GR" ("GR-value") or "RV" ("RelativeViability") or both

metric character vector with names of metric; chosen from: "xc50" ("GR50" or "IC50" - respectively depending on normalization_type), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean")

fit_source string source name for metrics

meta_data_path string path to metadata file describing all cancer models/cell lines which are referenced by a dataset contained within the DepMap portal. It is usually a file named Model.csv or Model.csv.gz.

feat_data_path string path to the directory containing the molecular feature set file to load from DepMap.

feature_sets character vector containing the names of the molecular feature sets to load from DepMap. These names should also correspond to the file names containing the feature data (without the extension, which is assumed to be csv or csv.gz)

metadata_columns
character vector with the metadata columns to load for DepMap cell lines

clear_taxonomy_info
logical flag whether to remove taxonomy information for gene names in table with the molecular feature sets from DepMap.

with_decoding logical whether the feature OmicsArmLevelCNA, OmicsSomaticMutationsMatrixHotspot and OmicsSomaticMutationsMatrixDamaging should be encoded into a 0-1 scheme

Value

A named list with elements:

- ls_plot nested list of plots for selected type of experiment
- ls_assoc_data nested list of table with association data

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- qs2::qs_read(system.file("testdata/finalMAE_prism.qs2", package = "gDRtestData"))
se <- mae[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")
drug_name <- unique(dt_metrics[[gDRutils::get_env_identifiers("drug_name")]])[1]
create_PRISM_plot_list_sa(
  drug_name_vec = drug_name,
  dt_metrics = dt_metrics,
  meta_data_path = system.file("depmap_data/Model.csv.gz", package = "gDRtestData"),
  feat_data_path = system.file("depmap_data", package = "gDRtestData"),
  feature_sets = c("CRISPRGeneEffect")
)
```

create_PRISM_summary_list *Create a list of PRISM association table*

Description

Create a list of PRISM association table

Usage

```
create_PRISM_summary_list(assoc_summary_RV, assoc_summary_GR = NULL)
```

Arguments

assoc_summary_RV	A data.table with associations for normalization type of "Relative Viability", outputted by gDRplots::prep_assoc_summary()
assoc_summary_GR	A data.table with associations for normalization type of "GR Value", outputted by gDRplots::prep_assoc_summary()

Value

A list of table split by drug name and normalization type

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
dir_path <- system.file("testdata", package = "gDRplots")
ls_file <- list.files(dir_path, pattern = "[.]xlsx$")
assoc_tab <- prep_assoc_summary(dir_path = dir_path, ls_file = ls_file)
create_PRISM_summary_list(
  assoc_summary_RV = assoc_tab[grepl("RV", assoc_tab[["src"]]), ],
  assoc_summary_GR = assoc_tab[grepl("GR", assoc_tab[["src"]]), ]
)
```

create_zoom_link *Prepare markdown chunk with zoom link*

Description

Generate markdown code for html link item which when clicked opens plot in new browser. The function output should be wrapped in knitr::knit().

Usage

```
create_zoom_link(img_path, link_txt = "Zoom In for Details")
```

Arguments

img_path	string with relative path to file with plot to be shown
link_txt	string with text describing link

Value

string with html link code

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

knitr::knit

Examples

```
create_zoom_link("plots/scatter.svg")
create_zoom_link(NA)
```

---

escape_special_characters

Escape colon and hash

Description

Escape colon and hash

Usage

```
escape_special_characters(x)
```

Arguments

x	String

Value

Original string with :s and #s and /s escaped

Examples

```
escape_special_characters("ABC:123")
escape_special_characters("AD_12")
escape_special_characters("AD#12")
escape_special_characters("AD/12")
```

estimate_plot_size *Estimate the optimal plot size (either ggplot or pheatmap) for saving plots*

Description

For a panel of plots combined with `patchwork`, the size grows with the number of rows and columns in the panel.

Usage

```
estimate_plot_size(plt, base_width = 10, base_height = 6, scale_factor = 0.5)
```

Arguments

`plt`	a ggplot (including patchwork) or pheatmap object
`base_width`	an integer with default base_width
`base_height`	an integer with default base_height
`scale_factor`	an integer with default scale_factor

Value

named vector with optimal width and height used in the `ggplot2::ggsave` function

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
p <- ggplot2::ggplot(mtcars, ggplot2::aes(mpg, wt)) + ggplot2::geom_point()
estimate_plot_size(p)
```

fill_ann_color_map *Fill missing values in the color map for annotation*

Description

Fill missing values in the color map for annotation

Usage

```
fill_ann_color_map(dt_ann, map_ann)
```

Arguments

`dt_ann`	`data.table` with the annotations
`map_ann`	`list` with the annotations

Value

list with color mapping for the annotations with missing items filled in

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

pheatmap_with_anno_sa pheatmap_with_anno_combo

Examples

```
annotation_manual <- data.table::data.table(
 CellLineName = c("cellline_AA", "cellline_EA", "cellline_IB", "cellline_MC", "cellline_BC"),
  mut_A = c(0, 0, 1, 2, 3),
  mut_B = c("yes", "yes", "no", NA, NA),
  mut_C = c("AA", "AA", "AB", NA, "B")
)

annotation_map <- list(
  mut_A = c("1" = "coral", "0" = "cadetblue"),
  mut_B = c("yes" = "black", "no" = "grey90", "not_checked" = "lightblue"),
  mut_C = c("AA" = "red", "AB" = "orange", "B" = "yellow")
)

fill_ann_color_map(dt_ann = annotation_manual, map_ann = annotation_map)
```

---

generate_color_mappings

Helper function to generate color mappings for Gnumber and Gnumber_2

Description

Helper function to generate color mappings for Gnumber and Gnumber_2

Usage

```
generate_color_mappings(
  dt_plate_subset,
  untrt_tag = gDRutils::get_env_identifiers("untreated_tag")
)
```

Arguments

dt_plate_subset	The subset of data
untrt_tag	Untreated tag identifier

Value

A list containing color mappings for Gnumber and Gnumber_2

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

`generate_datatable` *Generate a customized datatable*

Description

This function creates a `DT::datatable` object with default settings for horizontal scrolling (`scrollX = TRUE`) and no table controls (`dom = "t"`). It supports input data of types `data.table` or `DFrame`. Additional options and arguments can be passed to customize the table further.

Usage

```
generate_datatable(
  tab,
  options = list(scrollX = TRUE, dom = "ftip"),
  width = "100%",
  col_to_round = NULL,
  digits = 3,
  ...
)
```

Arguments

`tab`	A dataset to be displayed in the datatable. Must be of class `data.table` or `DFrame`.
`options`	A list of options to customize the DataTable. Defaults to `list(scrollX = TRUE, dom = "ftip")` - to display the search box, the table itself, information about the number of rows and pagination
`width`	A character string specifying the width of the table. Defaults to "100%".
`col_to_round`	A character vector with the names of the columns whose values should be rounded.
`digits`	A numeric value indicating the number of decimal places to be used for rounding
`...`	Additional arguments passed to `DT::datatable`.

Value

A `DT::datatable` object.

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
generate_datatable(data.table::data.table(iris))
```

get_ann_color_map *Create color map for annotation*

Description

Create color map for annotation

Usage

```
get_ann_color_map(dt_ann)
```

Arguments

dt_ann	data.table with the annotations

Value

list with color mapping for the annotations

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

pheatmap_qc

Examples

```
mae <- gDRutils::get_synthetic_data("small")
se <- mae[[gDRutils::get_supported_experiments("sa")]][2:5, ]
dt_average <- gDRutils::convert_se_assay_to_dt(se = se, assay_name = "Averaged")
dt_ann <- dt_average[,.SD, .SDcols = c("Tissue", "ReferenceDivisionTime")]

get_ann_color_map(dt_ann)
```

get_col_luminance *Calculate the luminance of a color*

Description

Calculate the luminance of a color

Usage

```
get_col_luminance(col_name)
```

Arguments

col_name	string representing a valid color

Value

single element numeric vector

Examples

```
get_col_luminance("blue")
get_col_luminance("red")
get_col_luminance("#000000")
get_col_luminance("#906090")
get_col_luminance("#906090F2")
```

`get_hm_title` *Get Legend Title*

Description

Get Legend Title

Usage

```
get_hm_title(metric = "xc50", normalization_type = "GR", dataset_name = NULL)
```

Arguments

`metric`	string name of the metric one of: "xc50"("GR50" or "IC50" - respectively depending on `normalization_type`), "x_max" ("GR Max" or "E Max") or x_mean" ("GR Mean" or "RV Mean")
`normalization_type`	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
`dataset_name`	string name of dataset

Value

character title for heatmap

Examples

```
get_hm_title(dataset_name = "Dateset DX123",
             metric = "x_mean",
             normalization_type = "GR")

get_hm_title(metric = "xc50",
             normalization_type = "GR")
```

get_iso_colors *get_iso_colors*

Description

get_iso_colors

Usage

```
get_iso_colors(normalization_type = c("RV", "GR"))
```

Arguments

normalization_type	charvec normalization_types expected in the data

Value

named charvec with iso colors

Examples

```
get_iso_colors()
```

---

get_provenance_tables *Tables describing how the results of a gDR object were produced*

Description

Builds the tables shown in the "How These Numbers Were Produced" section of the processing report from the provenance that gDR records in each experiment (see get_SE_provenance): one row per fit, with the function, its version, its settings and the model, and one row per processing step, described in a sentence.

Usage

```
get_provenance_tables(x)
```

Arguments

x	MultiAssayExperiment or SummarizedExperiment.

Value

A list with two data.tables:

- fits: one row per fit, with columns assay, fit_source, fn, settings and model.
- steps: one row per processing step, with columns step and description.

For a MultiAssayExperiment, both tables start with an experiment column naming the experiment of each row; a SummarizedExperiment is a single experiment, so it has no such column.

Both tables are empty for objects without provenance.

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
mae <- gDRutils::get_synthetic_data("small")
tables <- get_provenance_tables(mae)
tables$fits
```

---

get_qual_colors *Create list of qualitative colors*

Description

Create list of qualitative colors

Usage

```
get_qual_colors(n = NULL)
```

Arguments

n	number of required colors

Value

vector with hex colors from qualitative palettes

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
get_qual_colors()
get_qual_colors(0)
get_qual_colors(5)
get_qual_colors(35)
```

get_r_file_path *Extract path of the executed R file*

Description

Extract path of the executed R file

Usage

```
get_r_file_path(test_mode = FALSE)
```

Arguments

test_mode	logical flag whether the function be run in the test mode

Value

string with the path to the executed Rscript file

Examples

```
## Not run:
get_r_file_path()

## End(Not run)
```

heatmap_combo_metrics *Plot heatmaps of fitted values for combination metrics data*

Description

Plot heatmaps of fitted values for combination metrics data

Usage

```
heatmap_combo_metrics(
  dt_excess,
  dt_isobolograms = NULL,
  drug1_name,
  drug2_name,
  cl_name,
  metric = "smooth",
  normalization_type = "GR",
  iso_levels = c("0.25", "0.5", "0.75"),
  colors_vec_smooth = NULL,
  colors_vec_excess = NULL,
  limit = NULL,
  no_breaks = 50,
  swap_axes = FALSE,
```

```
  show_values = FALSE,
  scale_type = NULL
)
```

Arguments

dt_excess	data.table representing data from the excess assay, outputted by gDRutils::convert_se_assay_to_ "excess") and combo SummarizedExperiment
dt_isobolograms	data.table representing data from the isobolograms assay, outputted by gDRutils::convert_se_ass "isobolograms") and combo SummarizedExperiment
drug1_name	string with drug name to be plotted (identifiers DrugName)
drug2_name	string with co-drug name to be plotted (identifiers DrugName_2)
cl_name	string with cell line to be plotted (identifiers CellLineName)
metric	string name of combo metric, any numeric column of dt_excess; e.g. "smooth"("Smooth GR" or "Smooth RV" - respectively depending on normalization_type), "hsa_excess"("Bliss Excess GR" or "Bliss Excess RV"), "bliss_excess" ("Bliss Excess GR" or "Bliss Excess RV") or a column written by a custom fit, such as a fitted MuSyC surface
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
iso_levels	character vector with isobologram levels to be selected
colors_vec_smooth	character vector of colors (valid names or hex codes) used in the heatmap for smooth values; the default is the dark purple-light grey palette
colors_vec_excess	character vector of colors (valid name or hex codes) used in the heatmap for excess values; the default is the blue-light grey-red color scale
limit	numeric vector of length two providing limits of the scale. Use NA to refer to the existing minimum or maximum
no_breaks	numeric number of breaks on scale
swap_axes	logical flag indicating whether to swap the axes with drugs of the heatmap
show_values	logical flag indicating whether to show values of the metric on the heatmap
scale_type	string, "smooth" for the response palette and limits or "excess" for the palette and symmetric limits of a difference; only "smooth" caps values at 1.1; NULL (default) picks the scale for "smooth", "hsa_excess" and "bliss_excess" and is an error for any other metric

Value

ggplot object containing heatmap with the values for the selected combo metric for selected drugs and cell line with selected isoline (when chosen)

Examples

```
cl_name <- "cellline_BC"
drug1_name <- "drug_001"
drug2_name <- "drug_026"

mae <- gDRutils::get_synthetic_data("combo_matrix")
```

```
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_excess <- gDRutils::convert_se_assay_to_dt(se, "excess")
dt_isobolograms <- gDRutils::convert_se_assay_to_dt(se, "isobolograms")

heatmap_combo_metrics(dt_excess,
                      dt_isobolograms,
                      drug1_name, drug2_name,
                      cl_name,
                      normalization_type = "GR")

heatmap_combo_metrics(dt_excess,
                      dt_isobolograms,
                      drug1_name, drug2_name,
                      cl_name,
                      normalization_type = "GR",
                      swap_axes = TRUE)

heatmap_combo_metrics(dt_excess,
                      dt_isobolograms,
                      drug1_name, drug2_name,
                      cl_name,
                      metric = "hsa_excess",
                      normalization_type = "GR",
                      limit = c(NA, 0.1))

cl_name <- "cellline_JE"
drug1_name <- "drug_011"
drug2_name <- "drug_026"

heatmap_combo_metrics(dt_excess,
                      dt_isobolograms,
                      drug1_name, drug2_name,
                      cl_name,
                      metric = "bliss_excess",
                      normalization_type = "RV",
                      iso_levels = "0.5")

heatmap_combo_metrics(dt_excess,
                      dt_isobolograms,
                      drug1_name, drug2_name,
                      cl_name,
                      metric = "bliss_excess",
                      normalization_type = "RV",
                      iso_levels = NULL)
```

`heatmap_combo_metrics_panel` *Plot panel of heatmaps of fitted values for combination metrics data*

Description

Plot panel of heatmaps of fitted values for combination metrics data

Usage

```
heatmap_combo_metrics_panel(
  dt_excess,
  dt_isobolograms = NULL,
  drug1_name,
  drug2_name,
  cl_name,
  normalization_type = "GR",
  iso_levels = c("0.25", "0.5", "0.75"),
  colors_vec_smooth = NULL,
  colors_vec_excess = NULL,
  no_breaks = 50,
  as_list = FALSE,
  one_row_panel = FALSE,
  swap_axes = FALSE,
  show_values = FALSE
)
```

Arguments

dt_excess	data.table representing data from the excess assay, outputted by gDRutils::convert_se_assay_to_ "excess") and combo SummarizedExperiment
dt_isobolograms	data.table representing data from the isobolograms assay, outputted by gDRutils::convert_se_ass "isobolograms") and combo SummarizedExperiment
drug1_name	string with drug name to be plotted (identifiers DrugName)
drug2_name	string with co-drug name to be plotted (identifiers DrugName_2)
cl_name	string with cell line to be plotted (identifiers CellLineName)
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
iso_levels	character vector with isobologram levels to be selected
colors_vec_smooth	character vector of colors (valid names or hex codes) used in the heatmap for smooth values; the default is the dark purple-light grey palette
colors_vec_excess	character vector of colors (valid name or hex codes) used in the heatmap for excess values; the default is the blue-light grey-red color scale
no_breaks	numeric number of breaks on scale
as_list	logical flag whether return list of plot or panel
one_row_panel	logical flag whether return panel 2x2 (containing heatmaps for combination metrics and CI plot) or 3x1 (containing only heatmaps for combination metrics); it is working only for as_list = TRUE
swap_axes	logical flag indicating whether to swap the axes with drugs of the heatmap
show_values	logical flag indicating whether to show values of the metric on the heatmap

Value

ggplot object containing panel with heatmaps with value for excess assays for selected drugs and cell line with selected isoline and comparison of iso levels or list of ggplot object containing these plots.

Examples

```
cl_name <- "cellline_BC"
drug1_name <- "drug_001"
drug2_name <- "drug_026"

mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_excess <- gDRutils::convert_se_assay_to_dt(se, "excess")
dt_isobolograms <- gDRutils::convert_se_assay_to_dt(se, "isobolograms")

heatmap_combo_metrics_panel(dt_excess,
                            dt_isobolograms,
                            drug1_name, drug2_name,
                            cl_name,
                            normalization_type = "GR")

cl_name <- "cellline_JE"
drug1_name <- "drug_011"
drug2_name <- "drug_026"

heatmap_combo_metrics_panel(dt_excess,
                            dt_isobolograms,
                            drug1_name, drug2_name,
                            cl_name,
                            normalization_type = "RV",
                            iso_levels = "0.5",
                            as_list = TRUE)

heatmap_combo_metrics_panel(dt_excess,
                            dt_isobolograms,
                            drug1_name, drug2_name,
                            cl_name,
                            normalization_type = "RV",
                            iso_levels = "0.5",
                            one_row_panel = TRUE)

heatmap_combo_metrics_panel(dt_excess,
                            dt_isobolograms,
                            drug1_name, drug2_name,
                            cl_name,
                            normalization_type = "RV",
                            iso_levels = NULL,
                            as_list = FALSE,
                            swap_axes = FALSE)

heatmap_combo_metrics_panel(dt_excess,
                            dt_isobolograms,
                            drug1_name, drug2_name,
                            cl_name,
                            normalization_type = "RV",
                            iso_levels = NULL,
                            as_list = FALSE,
                            swap_axes = TRUE)
```

`heatmap_combo_with_isoref` *Plot heatmaps of averaged values for combination data*

Description

Plot heatmaps of averaged values for combination data

Usage

```
heatmap_combo_with_isoref(
  dt_excess,
  dt_isobolograms,
  drug1_name,
  drug2_name,
  cl_name,
  normalization_type = "GR",
  metric = "smooth",
  iso_levels = "0.5",
  colors_vec = NULL,
  colors_vec_iso = NULL,
  no_breaks = 50,
  swap_axes = FALSE
)
```

Arguments

`dt_excess`	data.table representing data from the `excess` assay, outputted by `gDRutils::convert_se_assay_to_` `"excess"` and combo `SummarizedExperiment`
`dt_isobolograms`	data.table representing data from the `isobolograms` assay, outputted by `gDRutils::convert_se_ass` `"isobolograms"` and combo `SummarizedExperiment`
`drug1_name`	string with drug name to be plotted (identifiers `DrugName`)
`drug2_name`	string with co-drug name to be plotted (identifiers `DrugName_2`)
`cl_name`	string with cell line to be plotted (identifiers `CellLineName`)
`normalization_type`	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
`metric`	string name of the combo metric; one of: "smooth" ("Smooth GR" or "Smooth RV" - respectively depending on `normalization_type`) "hsa_excess" ("Bliss Excess GR" or "Bliss Excess RV") or "bliss_excess" ("Bliss Excess GR" or "Bliss Excess RV")
`iso_levels`	character vector with isobologram levels to be selected; when `NULL` - no isolines will be displayed
`colors_vec`	character vector of colors (valid name or hex) used in heatmap; the default depends on `metric`: for "smooth" - the dark purple-light grey palette and for "hsa_excess" and "bliss_excess" - the blue-light grey-red color scale
`colors_vec_iso`	character vector of colors (valid name or hex) used for the isolines; the default is the dark red-orange palette

no_breaks	numeric number of breaks on scale
swap_axes	logical flag whether to swap the axes with drugs of the heatmap

Value

ggplot object containing heatmap for fitted values and reference data for isobolograms for selected drug and co-drug and selected cell line

Examples

```
cl_name <- "cellline_BC"
drug1_name <- "drug_001"
drug2_name <- "drug_026"

mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_excess <- gDRutils::convert_se_assay_to_dt(se, "excess")
dt_isobolograms <- gDRutils::convert_se_assay_to_dt(se, "isobolograms")

heatmap_combo_with_isoref(dt_excess,
                          dt_isobolograms,
                          drug1_name, drug2_name,
                          cl_name)

heatmap_combo_with_isoref(dt_excess,
                          dt_isobolograms,
                          drug1_name, drug2_name,
                          cl_name,
                          metric = "hsa_excess",
                          iso_levels = c("-0.2", "0.2"))

heatmap_combo_with_isoref(dt_excess,
                          dt_isobolograms,
                          drug1_name, drug2_name,
                          cl_name,
                          iso_levels = NULL,
                          colors_vec = c("darkcyan", "snow", "darkorange"))

heatmap_combo_with_isoref(dt_excess,
                          dt_isobolograms,
                          drug1_name, drug2_name,
                          cl_name,
                          normalization_type = "RV",
                          iso_levels = c("0.25", "0.75"),
                          swap_axes = FALSE)

heatmap_combo_with_isoref(dt_excess,
                          dt_isobolograms,
                          drug1_name, drug2_name,
                          cl_name,
                          normalization_type = "RV",
                          iso_levels = c("0.25", "0.75"),
                          swap_axes = TRUE)

heatmap_combo_with_isoref(dt_excess,
                          dt_isobolograms,
                          drug1_name, drug2_name,
```

```
                        cl_name,
                        metric = "hsa_excess",
                        iso_levels = c("0.25", "0.75"),
                        colors_vec_iso = c("0.25" = "darkcyan",
                                           "0.75" = "darkblue"),
                        swap_axes = FALSE)
```

`heatmap_combo_with_isoref_panel` *Plot panel of heatmaps with fitted and reference data for isobolograms*

Description

Plot panel of heatmaps with fitted and reference data for isobolograms

Usage

```
heatmap_combo_with_isoref_panel(
  dt_excess,
  dt_isobolograms,
  drug1_name,
  drug2_name,
  cl_names,
  normalization_type = "GR",
  metric = "smooth",
  iso_levels = "0.5",
  colors_vec = NULL,
  no_breaks = 50,
  swap_axes = FALSE
)
```

Arguments

`dt_excess`	data.table representing data from the `excess` assay, outputted by `gDRutils::convert_se_assay_to_` `"excess"` and combo `SummarizedExperiment`
`dt_isobolograms`	data.table representing data from the `isobolograms` assay, outputted by `gDRutils::convert_se_ass` `"isobolograms"` and combo `SummarizedExperiment`
`drug1_name`	string with drug name to be plotted (identifiers `DrugName`)
`drug2_name`	string with co-drug name to be plotted (identifiers `DrugName_2`)
`cl_names`	character vector with cell line names to be plotted (Cell Line Name); if `NULL` - all available cell lines will be plotted
`normalization_type`	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
`metric`	string name of the combo metric; one of: "smooth" ("Smooth GR" or "Smooth RV" - respectively depending on `normalization_type`) "hsa_excess" ("Bliss Excess GR" or "Bliss Excess RV") or "bliss_excess" ("Bliss Excess GR" or "Bliss Excess RV")

iso_levels	character vector with isobologram levels to be selected; when NULL - no isolines will be displayed
colors_vec	character vector of colors (valid name or hex) used in heatmap; the default depends on metric: for "smooth" - the dark purple-light grey palette and for "hsa_excess" and "bliss_excess" - the blue-light grey-red color scale
no_breaks	numeric number of breaks on scale
swap_axes	logical flag whether to swap the axes with drugs of the heatmap

Value

ggplot object containing panel with heatmaps for fitted values and reference data for isobolograms for selected drug and co-drug and all selected cell line

Examples

```
cl_names <-
  c("cellline_AA", "cellline_EA", "cellline_IB",
  "cellline_MC", "cellline_BC", "cellline_FD")

drug1_name <- "drug_001"
drug2_name <- "drug_026"

mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_excess <- gDRutils::convert_se_assay_to_dt(se, "excess")
dt_isobolograms <- gDRutils::convert_se_assay_to_dt(se, "isobolograms")

heatmap_combo_with_isoref_panel(dt_excess,
                                dt_isobolograms,
                                drug1_name, drug2_name,
                                cl_names)

dt_excess_2 <- data.table::copy(dt_excess)
invisible(dt_excess_2[CellLineName %in% cl_names[1:2],
                      Concentration := Concentration / 10])

heatmap_combo_with_isoref_panel(dt_excess_2,
                                dt_isobolograms,
                                drug1_name, drug2_name,
                                cl_names,
                                iso_levels = c("0.25", "0.75"),
                                colors_vec = c("darkcyan", "snow", "coral"))
```

---

heatmap_combo_with_isoref_panel_common

#' Plot panel of heatmaps with fitted and reference data for isobolograms

---

Description

#' Plot panel of heatmaps with fitted and reference data for isobolograms

Usage

```
heatmap_combo_with_isoref_panel_common(
  dt_excess,
  dt_isobolograms,
  drug1_name,
  drug2_name,
  cl_names,
  normalization_type = "GR",
  metric = "smooth",
  iso_levels = "0.5",
  colors_vec = NULL,
  no_breaks = 50,
  swap_axes = FALSE
)
```

Arguments

dt_excess	data.table representing data from the excess assay, outputted by gDRutils::convert_se_assay_to_ "excess") and combo SummarizedExperiment
dt_isobolograms	data.table representing data from the isobolograms assay, outputted by gDRutils::convert_se_ass "isobolograms") and combo SummarizedExperiment
drug1_name	string with drug name to be plotted (identifiers DrugName)
drug2_name	string with co-drug name to be plotted (identifiers DrugName_2)
cl_names	character vector with cell line names to be plotted (Cell Line Name); if NULL - all available cell lines will be plotted
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the combo metric; one of: "smooth" ("Smooth GR" or "Smooth RV" - respectively depending on normalization_type) "hsa_excess" ("Bliss Excess GR" or "Bliss Excess RV") or "bliss_excess" ("Bliss Excess GR" or "Bliss Excess RV")
iso_levels	character vector with isobologram levels to be selected; when NULL - no isolines will be displayed
colors_vec	character vector of colors (valid name or hex) used in heatmap; the default depends on metric: for "smooth" - the dark purple-light grey palette and for "hsa_excess" and "bliss_excess" - the blue-light grey-red color scale
no_breaks	numeric number of breaks on scale
swap_axes	logical flag whether to swap the axes with drugs of the heatmap

Value

ggplot object containing panel with heatmaps for fitted values and reference data for isobolograms for selected drug and co-drug by list of cell lines

Examples

```
cl_names <-
  c("cellline_AA", "cellline_EA", "cellline_IB",
```

```
  "cellline_MC", "cellline_BC", "cellline_FD")

drug1_name <- "drug_001"
drug2_name <- "drug_026"

mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_excess <- gDRutils::convert_se_assay_to_dt(se, "excess")
dt_isobolograms <- gDRutils::convert_se_assay_to_dt(se, "isobolograms")

heatmap_combo_with_isoref_panel_common(dt_excess,
                                       dt_isobolograms,
                                       drug1_name, drug2_name,
                                       cl_names)

heatmap_combo_with_isoref_panel_common(dt_excess,
                                       dt_isobolograms,
                                       drug1_name, drug2_name,
                                       cl_names,
                                       iso_levels = c("0.25", "0.5"))

heatmap_combo_with_isoref_panel_common(dt_excess,
                                       dt_isobolograms,
                                       drug1_name, drug2_name,
                                       cl_names,
                                       metric = "hsa_excess",
                                       iso_levels = c("-0.25", "0.25"))

heatmap_combo_with_isoref_panel_common(dt_excess,
                                       dt_isobolograms,
                                       drug1_name, drug2_name,
                                       cl_names,
                                       normalization_type = "RV",
                                       iso_levels = NULL,
                                       colors_vec = c("darkcyan", "snow", "darkorange"),
                                       swap_axes = FALSE)

heatmap_combo_with_isoref_panel_common(dt_excess,
                                       dt_isobolograms,
                                       drug1_name, drug2_name,
                                       cl_names,
                                       normalization_type = "RV",
                                       iso_levels = NULL,
                                       colors_vec = c("darkcyan", "snow", "darkorange"),
                                       swap_axes = TRUE)

heatmap_combo_with_isoref_panel_common(dt_excess,
                                       dt_isobolograms,
                                       drug1_name, drug2_name,
                                       cl_names,
                                       metric = "hsa_excess",
                                       iso_levels = NULL,
                                       swap_axes = FALSE)
```

`heatmap_combo_with_isoref_panel_independent` *Plot panel of heatmaps with fitted and reference data for isobolograms*

Description

This function is dedicated to cases in which given cell lines are exposed to drugs of different concentrations and have almost no or no common values.

Usage

```
heatmap_combo_with_isoref_panel_independent(
  dt_excess,
  dt_isobolograms,
  drug1_name,
  drug2_name,
  cl_names,
  normalization_type = "GR",
  metric = "smooth",
  iso_levels = "0.5",
  colors_vec = NULL,
  no_breaks = 50,
  swap_axes = FALSE
)
```

Arguments

`dt_excess`	data.table representing data from the `excess` assay, outputted by `gDRutils::convert_se_assay_to_` `"excess")` and combo `SummarizedExperiment`
`dt_isobolograms`	data.table representing data from the `isobolograms` assay, outputted by `gDRutils::convert_se_ass` `"isobolograms")` and combo `SummarizedExperiment`
`drug1_name`	string with drug name to be plotted (identifiers `DrugName`)
`drug2_name`	string with co-drug name to be plotted (identifiers `DrugName_2`)
`cl_names`	character vector with cell line names to be plotted (Cell Line Name); if `NULL` - all available cell lines will be plotted
`normalization_type`	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
`metric`	string name of the combo metric; one of: "smooth" ("Smooth GR" or "Smooth RV" - respectively depending on `normalization_type`) "hsa_excess" ("Bliss Excess GR" or "Bliss Excess RV") or "bliss_excess" ("Bliss Excess GR" or "Bliss Excess RV")
`iso_levels`	character vector with isobologram levels to be selected; when `NULL` - no isolines will be displayed
`colors_vec`	character vector of colors (valid name or hex) used in heatmap; the default depends on `metric`: for "smooth" - the dark purple-light grey palette and for "hsa_excess" and "bliss_excess" - the blue-light grey-red color scale
`no_breaks`	numeric number of breaks on scale
`swap_axes`	logical flag whether to swap the axes with drugs of the heatmap

Value

ggplot object containing panel with heatmaps for fitted values and reference data for isobolograms for selected drug and co-drug by list of cell lines

Examples

```
cl_names <-
  c("cellline_AA", "cellline_EA", "cellline_IB",
  "cellline_MC", "cellline_BC", "cellline_FD")

drug1_name <- "drug_001"
drug2_name <- "drug_026"

mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_excess <- gDRutils::convert_se_assay_to_dt(se, "excess")
dt_isobolograms <- gDRutils::convert_se_assay_to_dt(se, "isobolograms")

heatmap_combo_with_isoref_panel_independent(dt_excess,
                                            dt_isobolograms,
                                            drug1_name, drug2_name,
                                            cl_names)

heatmap_combo_with_isoref_panel_independent(dt_excess,
                                            dt_isobolograms,
                                            drug1_name, drug2_name,
                                            cl_names,
                                            iso_levels = c("-0.25", "0.25"))

heatmap_combo_with_isoref_panel_independent(
  dt_excess,
  dt_isobolograms,
  drug1_name, drug2_name,
  cl_names = c("cellline_FD", "cellline_MC", "cellline_AA", "cellline_EA"),
  iso_levels =  c("-0.25", "-0.05", "0.2", "0.65"))

heatmap_combo_with_isoref_panel_independent(dt_excess,
                                            dt_isobolograms,
                                            drug1_name, drug2_name,
                                            cl_names,
                                            normalization_type = "RV",
                                            iso_levels = NULL,
                                       colors_vec = c("darkcyan", "snow", "darkorange"),
                                            swap_axes = TRUE)
```

---

heatmap_control_mapping_qc

Plot heatmap of mapping controls to treated for single-agent and combo data to control quality of the data

---

Description

Plot heatmap of mapping controls to treated for single-agent and combo data to control quality of the data

Usage

```
heatmap_control_mapping_qc(dt_treat, dt_controls)
```

Arguments

dt_treat	data.table representation of the data in RawTreated assay, outputted by gDRutils::convert_se_ass "RawTreated") and SummarizedExperiment with chosen data type: single-agent or combo
dt_controls	data.table representation of the data in Controls assay, outputted by gDRutils::convert_se_assay "Controls") and SummarizedExperiment with chosen data type: single-agent or combo

Value

pheatmap object containing hetamap of mapping controls to treated

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
mae <- gDRutils::get_synthetic_data("small")
se <- mae[[gDRutils::get_supported_experiments("sa")]]

dt_treat <- gDRutils::convert_se_assay_to_dt(se, "RawTreated")
dt_controls <- gDRutils::convert_se_assay_to_dt(se, "Controls")

heatmap_control_mapping_qc(dt_treat = dt_treat,
                           dt_controls = dt_controls)

heatmap_control_mapping_qc(dt_treat = dt_treat[1:1350, ],
                           dt_controls = dt_controls)

dt_treat_NA <- dt_treat[-c(1:135, 270:405),]
heatmap_control_mapping_qc(dt_treat = dt_treat_NA,
                           dt_controls = dt_controls)

dt_controls_NA <- dt_controls[-c(1:305, 611:763, 1221:1750),]
heatmap_control_mapping_qc(dt_treat = dt_treat,
                           dt_controls = dt_controls_NA)

heatmap_control_mapping_qc(dt_treat = dt_treat,
                           dt_controls = dt_controls[1:3660, ])
```

---

is_color_dark *Determine whether or not a color is dark*

Description

Determine whether or not a color is dark

Usage

```
is_color_dark(col_name)
```

Arguments

col_name	string representing a valid color

Value

logical flag

Examples

```
is_color_dark("blue")
is_color_dark("red")
is_color_dark("#000000")
```

---

is_valid_color *Determine whether or not a color name is valid*

Description

A name of color is valid when either is a color name listed by grDevices::colors or a hexadecimal string of the form #rrggbb

Usage

```
is_valid_color(col_name)
```

Arguments

col_name	string representing a valid color

Value

logical flag

Examples

```
is_valid_color("darkblue")
is_valid_color("#FF8C00")
is_valid_color("#FF8C00DC")
is_valid_color("RED")
```

neutralize_spaces *Replace spaces with another character*

Description

Replace spaces with another character

Usage

```
neutralize_spaces(x, replacement = "_")
```

Arguments

x	String where matches are sought
replacement	String replacement for spaces

Value

String with spaces replaced by the specified character

Examples

```
neutralize_spaces("GDC-123|Abc x G01234")
neutralize_spaces("MNO-321P 789R YY#1 ")
neutralize_spaces("drug_001 x drug_002", ".")
```

pheatmap_qc *Plot pretty heatmap for single-agent or combo data to control quality of the data*

Description

Plot pretty heatmap for single-agent or combo data to control quality of the data

Usage

```
pheatmap_qc(
  dt_average,
  normalization_type = "GR",
  metric = "x",
  fit_source = "gDR",
  hm_title = NA,
  colors_vec = c("black", "grey99"),
  no_breaks = 50,
  cluster_rows = TRUE,
  distfun = compute_distances,
  lbl_by_CellLineName = FALSE,
  lbl_by_DrugName = FALSE
)
```

Arguments

dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assa and SummarizedExperiment with chosen data type: single-agent or combo
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the metric; one of: "x" (value of "GR" or "RV" itself - respectively depending on normalization_type), or "x_std" (standard deviation)
fit_source	string source name for metrics
hm_title	string plot title
colors_vec	character vector of colors (valid name or hex) used in heatmap note that for metric "x" the first color will be assigned to the min value, and the last one - to the max; for "x_std" - that will be reversed
no_breaks	numeric number of breaks on scale used for mapping values to colors
cluster_rows	logical flag whether rows should be clustered; the dendrogram will not be shown for the matrix with any dimension greater than 200.
distfun	function used to compute the distance (dissimilarity) between rows; used for the dendrogram when cluster_rows is set to TRUE; the default is compute_distances using Spearman method.
lbl_by_CellLineName	logical flag whether heatmap should be described by CellLineNames instead of clid
lbl_by_DrugName	logical flag whether heatmap should be described by DrugName instead of Gnumber

Value

pheatmap object containing heatmap for selected metric with annotation - if given

See Also

pheatmap::pheatmap

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("sa")]][2:5, ]
dt_average <- gDRutils::convert_se_assay_to_dt(se = se,
                                               assay_name = "Averaged")

hm_1 <- pheatmap_qc(dt_average = dt_average)

hm_2 <- pheatmap_qc(dt_average = dt_average,
                    normalization_type = "RV",
                    colors_vec = c("darkblue", "grey90"),
                    lbl_by_CellLineName = TRUE,
                    lbl_by_DrugName = TRUE)

patchwork::wrap_elements(hm_1[["gtable"]])
patchwork::wrap_elements(hm_2[["gtable"]])
```

```
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_average <- gDRutils::convert_se_assay_to_dt(se = se,
                                               assay_name = "Averaged")
hm_3 <- pheatmap_qc(dt_average = dt_average,
                    cluster_rows = FALSE)
hm_4 <- pheatmap_qc(dt_average = dt_average,
                    metric = "x_std",
                    cluster_rows = FALSE)

patchwork::wrap_elements(hm_3[["gtable"]])
patchwork::wrap_elements(hm_4[["gtable"]])
```

`pheatmap_with_anno_cd` *Plot pretty heatmap with annotations for co-dilution data*

Description

Plot pretty heatmap with annotations for co-dilution data

Usage

```
pheatmap_with_anno_cd(
  dt_metrics,
  normalization_type = "GR",
  metric = "xc50",
  fit_source = "gDR",
  hm_title = NA,
  colors_vec = NULL,
  no_breaks = 50,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  distfun = compute_distances,
  annotation_row = NULL,
  annotation_col = NULL,
  annotation_colors = NULL
)
```

Arguments

`dt_metrics`	`data.table` representing data from the `Metrics` assay, outputted by `gDRutils::convert_se_assay` and co-dilution `SummarizedExperiment`
`normalization_type`	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
`metric`	string name of the metric; one of: "xc50" ("GR50" or "IC50" - respectively depending on `normalization_type`), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean")
`fit_source`	string source name for metrics
`hm_title`	string plot title

colors_vec	character vector of colors (valid name or hex) used in heatmap; note that the first color will be assigned to the min value, and the last one - to the max
no_breaks	numeric number of breaks on scale used for mapping values to colors
cluster_rows	logical flag whether indicating rows should be clustered; the dendrogram will not be shown for the matrix with any dimension greater than 200.
cluster_cols	logical flag indicating whether columns should be clustered; the dendrogram will not be shown for the matrix with any dimension greater than 200.
distfun	function used to compute the distance (dissimilarity) between both rows and columns; used for the dendrogram when cluster_rows or cluster_cols is set to TRUE the default is compute_distances using Spearman method.
annotation_row	data.table that specifies the annotations shown on the left side of the heatmap. Each row defines the features for a specific row. The rows in the data and in the annotation are matched using corresponding names from the required DrugName, DrugName_2 and Concentration_2 columns. Note that color schemes takes into account if the variable is continuous or discrete.
annotation_col	data.table that specifies the annotations shown above the heatmap. Each row defines the features for a specific column. The columns in the data and in the annotation are matched using corresponding names from the required CellLineName column. Note that color schemes takes into account if the variable is continuous or discrete.
annotation_colors	named list for specifying annotation_col and annotation_row track colors manually; note list is named with annotation name (column names of annotation_row - without DrugName and column names of annotation_col - without CellLineName), each list item is named vector with valid color name for each value described in annotation_row and in annotation_col, respectively. Not described elements will be colored by default.

Value

A named list with elements:

- data a list containing the information visualized in the heatmap:
  - matrix data shown in the heatmap for the selected metric.
  - annotation_row a table with row annotations (for DrugName), if provided.
  - annotation_col a table with column annotations (for CellLineName), if provided.
- pheatmap object containing the heatmap itself.

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

pheatmap::pheatmap

Examples

```
mae <- gDRutils::get_synthetic_data("combo_codilution_small")
se <- mae[[gDRutils::get_supported_experiments("cd")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se = se,
```

```
                                                assay_name = "Metrics")

output <- pheatmap_with_anno_cd(dt_metrics = dt_metrics)
hm_1 <- output[["heatmap"]]
patchwork::wrap_elements(hm_1[["gtable"]])

annotation_manual_col <-
  unique(dt_metrics[, c("CellLineName", "Tissue"), with = FALSE])
annotation_manual_row <-
  unique(dt_metrics[, c("DrugName", "DrugName_2", "Concentration_2",
                        "drug_moa", "drug_moa_2"),
                    with = FALSE])
annotation_map <-
 get_ann_color_map(unique(dt_metrics[, c("Tissue", "drug_moa", "drug_moa_2"), with = FALSE]))

output <- pheatmap_with_anno_cd(dt_metrics = dt_metrics,
                                normalization_type = "RV",
                                metric = "x_mean",
                                colors_vec = c("darkblue", "grey90"),
                                annotation_row = annotation_manual_row,
                                annotation_col = annotation_manual_col,
                                annotation_colors = annotation_map)
hm_2 <- output[["heatmap"]]
patchwork::wrap_elements(hm_2[["gtable"]])

annotation_manual <- data.table::data.table(
  CellLineName =
    c("cellline_AA", "cellline_EA", "cellline_IB", "cellline_MC", "cellline_BC"),
  mut_A = c(1, 1, 1, 0, 0),
  mut_B = c("yes", "yes", "no", "no", "no")
)
annotation_map <- list(
  mut_A = c("1" = "coral", "0" = "cadetblue"),
  mut_B = c("yes" = "black", "no" = "grey90")
)

output <- pheatmap_with_anno_cd(dt_metrics = dt_metrics,
                                annotation_col = annotation_manual,
                                annotation_colors = annotation_map,
                                hm_title = get_hm_title(
                                  normalization_type = "GR",
                                  metric = "hsa_score",
                                  dataset_name = "Co-dilution data"))
hm_3 <- output[["heatmap"]]
patchwork::wrap_elements(hm_3[["gtable"]])
```

---

pheatmap_with_anno_combo

Plot pretty heatmap with annotations for combo data

Description

Plot pretty heatmap with annotations for combo data

Usage

```
pheatmap_with_anno_combo(
  dt_scores,
  normalization_type = "GR",
  metric = "hsa_score",
  fit_source = "gDR",
  hm_title = NA,
  colors_vec = NULL,
  no_breaks = 50,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  distfun = compute_distances,
  annotation_row = NULL,
  annotation_col = NULL,
  annotation_colors = NULL,
  max_hm_lbl_length = .get_setting("MAX_HM_LBL_LENGTH")
)
```

Arguments

dt_scores	data.table representing data from the scores assay, outputted by gDRutils::convert_se_assay_ and combo SummarizedExperiment
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the combo metric, any numeric column of dt_scores; e.g. "hsa_score" ("Bliss Excess GR" or "Bliss Excess RV" - respectively depending on normalization_type), "bliss_score" ("Bliss Score GR" or "Bliss Score RV") or a column written by a custom fit, such as beta from MuSyC
fit_source	string source name for metrics
hm_title	string plot title
colors_vec	character vector of colors (valid name or hex) used in heatmap; note that the first color will be assigned to the min value, and the last one - to the max
no_breaks	numeric number of breaks on scale used for mapping values to colors
cluster_rows	logical flag whether rows should be clustered; the dendrogram will not be shown for the matrix with any dimension greater than 200.
cluster_cols	logical flag whether columns should be clustered; the dendrogram will not be shown for the matrix with any dimension greater than 200.
distfun	function used to compute the distance (dissimilarity) between both rows and columns; used for the dendrogram when cluster_rows or cluster_cols is set to TRUE the default is compute_distances using Spearman method.
annotation_row	data.table that specifies the annotations shown on left side of the heatmap. Each row defines the features for a specific row. The rows in the data and in the annotation are matched using corresponding combination of names from DrugName and DrugName_2 columns. Both columns are required. Note that color schemes takes into account if variable is continuous or discrete.
annotation_col	data.table that specifies the annotations shown above the heatmap. Each row defines the features for a specific column. The columns in the data and in the annotation are matched using corresponding names from the required

CellLineName column. Note that color schemes takes into account if variable is continuous or discrete.

annotation_colors
: named list for specifying annotation_col and annotation_row track colors manually; note list is named with annotation name (column names of annotation_row - without DrugName and DrugName_2, and column names of annotation_col - without CellLineName), each list item is named vector with a valid color name for each value described in annotation_row) and in annotation_col) - respectively. Not described elements will be colored in default.

max_hm_lbl_length
: numeric value for the maximum number of characters in the label; if set to Inf, no trimming will be performed; for better readability, it is recommended to use the default number.

Value

A named list with elements:

- data a list containing the information visualized in the heatmap:
  - matrix data shown in the heatmap for the selected metric.
  - annotation_row a table with row annotations (for DrugName), if provided.
  - annotation_col a table with column annotations (for CellLineName), if provided.
- pheatmap object containing the heatmap itself.

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

pheatmap

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_scores <- gDRutils::convert_se_assay_to_dt(se = se,
                                              assay_name = "scores")
annotation_manual_col <-
  unique(dt_scores[, c("CellLineName", "Tissue"), with = FALSE])
annotation_manual_row <-
 unique(dt_scores[, c("DrugName", "DrugName_2", "drug_moa", "drug_moa_2"), with = FALSE])
annotation_map <-
 get_ann_color_map(unique(dt_scores[, c("Tissue", "drug_moa", "drug_moa_2"), with = FALSE]))

output <- pheatmap_with_anno_combo(dt_scores = dt_scores,
                                   normalization_type = "RV",
                                   metric = "bliss_score",
                                   colors_vec = c("darkblue", "grey90", "darkred"),
                                   annotation_row = annotation_manual_row,
                                   annotation_col = annotation_manual_col,
                                   annotation_colors = annotation_map)
hm_1 <- output[["heatmap"]]
patchwork::wrap_elements(hm_1[["gtable"]])
```

```
annotation_manual <- data.table::data.table(
  CellLineName =
    c("cellline_AA", "cellline_EA", "cellline_IB", "cellline_MC", "cellline_BC"),
  mut_A = c(1, 1, 1, 0, 0),
  mut_B = c("yes", "yes", "no", "no", "no")
)

annotation_map <- list(
  mut_A = c("1" = "coral", "0" = "cadetblue"),
  mut_B = c("yes" = "black", "no" = "grey90")
)

output <- pheatmap_with_anno_combo(dt_scores = dt_scores,
                                   cluster_cols = FALSE,
                                   annotation_col = annotation_manual,
                                   annotation_colors = annotation_map,
                                   hm_title = get_hm_title(
                                     normalization_type = "GR",
                                     metric = "hsa_score",
                                     dataset_name = "Combo Matrix - combo data"))
hm_2 <- output[["heatmap"]]
patchwork::wrap_elements(hm_2[["gtable"]])
```

`pheatmap_with_anno_combo_metrics`
Plot heatmap with annotations for combo metrics data

Description

Plots a heatmap for metrics data (e.g., "xc50", "x_inf") covering both Single Agents and Combinations. Rows are sorted alphabetically by the primary drug, prioritizing Single Agent conditions first, followed by combinations sorted by co-treatment name and concentration.

Usage

```
pheatmap_with_anno_combo_metrics(
  dt_metrics,
  dt_metrics_capped = NULL,
  normalization_type = "GR",
  metric = "x_mean",
  fit_source = "gDR",
  hm_title = NA,
  colors_vec = NULL,
  no_breaks = 50,
  cluster_rows = FALSE,
  cluster_cols = TRUE,
  distfun = compute_distances,
  annotation_col = NULL,
  annotation_colors = NULL,
  max_hm_lbl_length = .get_setting("MAX_HM_LBL_LENGTH")
)
```

Arguments

dt_metrics	data.table representing data from the metrics assay, outputted by gDRutils::convert_se_assay Must contain DrugName, DrugName_2, cotrt_value, and CellLineName.
dt_metrics_capped	data.table representing data from the "Metrics" assay, the same as dt_metrics but with capped values with gDRutils::cap_assay_infinities
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the metric; one of: "xc50" ("GR50" or "IC50" - respectively depending on normalization_type), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean"); but the values from any numeric column can be displayed.
fit_source	string source name for metrics
hm_title	string plot title
colors_vec	character vector of colors (valid name or hex) used in heatmap; note that the first color will be assigned to the min value, and the last one - to the max
no_breaks	numeric number of breaks on scale used for mapping values to colors
cluster_rows	logical flag whether rows should be clustered; the dendrogram will not be shown for the matrix with any dimension greater than 200.
cluster_cols	logical flag whether columns should be clustered; the dendrogram will not be shown for the matrix with any dimension greater than 200.
distfun	function used to compute the distance (dissimilarity) between both rows and columns; used for the dendrogram when cluster_rows or cluster_cols is set to TRUE the default is compute_distances using Spearman method.
annotation_col	data.table that specifies the annotations shown above the heatmap. Each row defines the features for a specific column. The columns in the data and in the annotation are matched using corresponding names from the required CellLineName column. Note that color schemes takes into account if variable is continuous or discrete.
annotation_colors	named list for specifying annotation_col and annotation_row track colors manually; note list is named with annotation name (column names of annotation_row - without DrugName and column names of annotation_col - without CellLineName), each list item is named vector with valid color name for each value described in annotation_row and in annotation_col - respectively. Not described elements will be colored in default.
max_hm_lbl_length	numeric value for the maximum number of characters in the label; if set to Inf, no trimming will be performed; for better readability, it is recommended to use the default number.

Value

A named list with elements:

- data a list containing the information visualized in the heatmap:
  - matrix data shown in the heatmap for the selected metric.
  - annotation_row a table with row annotations (auto-generated).
  - annotation_col a table with column annotations, if provided.
- heatmap the pheatmap object.

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix_small")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se = se,
                                               assay_name = "Metrics")

output <- pheatmap_with_anno_combo_metrics(dt_metrics = dt_metrics,
                                           normalization_type = "RV",
                                           metric = "x_mean")
hm_1 <- output[["heatmap"]]
patchwork::wrap_elements(hm_1[["gtable"]])

annotation_manual <- data.table::data.table(
  CellLineName =
    c("cellline_AA", "cellline_GB", "cellline_HB"),
  mut_A = c(0, 1, 1),
  mut_B = c("yes", "yes", "no")
)
annotation_map <- list(
  mut_A = c("1" = "coral", "0" = "cadetblue"),
  mut_B = c("yes" = "black", "no" = "grey90")
)

output <- pheatmap_with_anno_combo_metrics(dt_metrics = dt_metrics,
                                           metric = "ec50",
                                           normalization_type = "GR",
                                           annotation_col = annotation_manual,
                                           annotation_colors = annotation_map,
                                           hm_title = get_hm_title(
                                             normalization_type = "GR",
                                             metric = "EC50 for combo data ",
                                             dataset_name = "Combo data"))



mae <- gDRutils::get_synthetic_data("combo_triple")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se = se,
                                               assay_name = "Metrics",
                                               merge_additional_variables = TRUE)

output <- pheatmap_with_anno_combo_metrics(dt_metrics = dt_metrics,
                                           normalization_type = "RV",
                                           metric = "x_mean")
hm_4 <- output[["heatmap"]]
patchwork::wrap_elements(hm_4[["gtable"]])
```

---

pheatmap_with_anno_sa *Plot pretty heatmap with annotations for single-agent data*

Description

Plot pretty heatmap with annotations for single-agent data

Usage

```
pheatmap_with_anno_sa(
  dt_metrics,
  dt_metrics_capped = NULL,
  normalization_type = "GR",
  metric = "xc50",
  fit_source = "gDR",
  hm_title = NA,
  colors_vec = NULL,
  no_breaks = 50,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  distfun = compute_distances,
  annotation_row = NULL,
  annotation_col = NULL,
  annotation_colors = NULL,
  max_hm_lbl_length = .get_setting("MAX_HM_LBL_LENGTH")
)
```

Arguments

dt_metrics	data.table representing data from the "Metrics" assay, outputted by gDRutils::convert_se_ass and single-agent SummarizedExperiment
dt_metrics_capped	data.table representing data from the "Metrics" assay, the same as dt_metrics but with capped values with gDRutils::cap_assay_infinities
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the metric; one of: "xc50" ("GR50" or "IC50" - respectively depending on normalization_type), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean"); but the values from any numeric column can be displayed.
fit_source	string source name for metrics
hm_title	string plot title
colors_vec	character vector of colors (valid name or hex) used in heatmap; note that the first color will be assigned to the min value, and the last one - to the max
no_breaks	numeric number of breaks on scale used for mapping values to colors
cluster_rows	logical flag whether rows should be clustered; the dendrogram will not be shown for the matrix with any dimension greater than 200.
cluster_cols	logical flag whether columns should be clustered; the dendrogram will not be shown for the matrix with any dimension greater than 200.
distfun	function used to compute the distance (dissimilarity) between both rows and columns; used for the dendrogram when cluster_rows or cluster_cols is set to TRUE the default is compute_distances using Spearman method.

annotation_row data.table that specifies the annotations shown on left side of the heatmap. Each row defines the features for a specific row. The rows in the data and in the annotation are matched using corresponding names from the required DrugName column. Note that color schemes takes into account if variable is continuous or discrete.

annotation_col data.table that specifies the annotations shown above the heatmap. Each row defines the features for a specific column. The columns in the data and in the annotation are matched using corresponding names from the required CellLineName column. Note that color schemes takes into account if variable is continuous or discrete.

annotation_colors named list for specifying annotation_col and annotation_row track colors manually; note list is named with annotation name (column names of annotation_row - without DrugName and column names of annotation_col - without CellLineName), each list item is named vector with valid color name for each value described in annotation_row and in annotation_col - respectively. Not described elements will be colored in default.

max_hm_lbl_length numeric value for the maximum number of characters in the label; if set to Inf, no trimming will be performed; for better readability, it is recommended to use the default number.

Value

A named list with elements:

- data a list containing the information visualized in the heatmap:
  - matrix data shown in the heatmap for the selected metric.
  - annotation_row a table with row annotations (for DrugName), if provided.
  - annotation_col a table with column annotations (for CellLineName), if provided.
- pheatmap object containing the heatmap itself.

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

pheatmap::pheatmap

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se = se,
                                               assay_name = "Metrics")
dt_averaged <- gDRutils::convert_se_assay_to_dt(se = se,
                                                assay_name = "Averaged")
dt_metrics_capped <-
  gDRutils::cap_assay_infinities(
    conc_assay_dt = dt_averaged,
    assay_dt = dt_metrics,
    experiment_name = gDRutils::get_supported_experiments("sa"),
```

```
    col = "xc50",
    capping_fold = 5)

output <- pheatmap_with_anno_sa(dt_metrics = dt_metrics)
hm_0 <- output[["heatmap"]]
patchwork::wrap_elements(hm_0[["gtable"]])

output <- pheatmap_with_anno_sa(dt_metrics = dt_metrics,
                                dt_metrics_capped = dt_metrics_capped)
hm_1 <- output[["heatmap"]]
patchwork::wrap_elements(hm_1[["gtable"]])

annotation_manual_col <-
  unique(dt_metrics[, c("CellLineName", "Tissue"), with = FALSE])
annotation_manual_row <-
  unique(dt_metrics[, c("DrugName", "drug_moa"), with = FALSE])
annotation_map <-
  get_ann_color_map(unique(dt_metrics[, c("Tissue", "drug_moa"), with = FALSE]))

output <- pheatmap_with_anno_sa(dt_metrics = dt_metrics,
                                normalization_type = "RV",
                                metric = "x_mean",
                                colors_vec = c("darkblue", "grey90"),
                                annotation_row = annotation_manual_row,
                                annotation_col = annotation_manual_col,
                                annotation_colors = annotation_map)
hm_2 <- output[["heatmap"]]
patchwork::wrap_elements(hm_2[["gtable"]])

annotation_manual <- data.table::data.table(
  CellLineName =
    c("cellline_AA", "cellline_EA", "cellline_IB", "cellline_MC", "cellline_BC"),
  mut_A = c(1, 1, 1, 0, 0),
  mut_B = c("yes", "yes", "no", "no", "no")
)
annotation_map <- list(
  mut_A = c("1" = "coral", "0" = "cadetblue"),
  mut_B = c("yes" = "black", "no" = "grey90")
)

output <- pheatmap_with_anno_sa(dt_metrics = dt_metrics,
                                annotation_col = annotation_manual,
                                annotation_colors = annotation_map,
                                hm_title = get_hm_title(
                                  normalization_type = "GR",
                                  metric = "hsa_score",
                                  dataset_name = "Combo Matrix - combo data"))
hm_3 <- output[["heatmap"]]
patchwork::wrap_elements(hm_3[["gtable"]])
```

`plot_boxplot_meta` *Plot boxplot for metric values grouped by metadata from DepMap*

Description

Plot boxplot for metric values grouped by metadata from DepMap

Usage

```
plot_boxplot_meta(
  dt_response,
  dt_depmap,
  selected_feat_meta_col,
  with_1_item_grp = TRUE,
  max_x_lbl_length = 60,
  with_inf = FALSE
)
```

Arguments

dt_response	data.table with experimental response data (rows are samples) for one metric outputted by one of functions: prep_dt_response_metric_sa, prep_dt_response_dose_sa, prep_dt_response_scores or prep_dt_response_metric_diff,
dt_depmap	data.table with dependent variables data load from DepMap - for one metadata; (rows are samples, columns are metadata levels); outputted by prep_dt_depmap_meta
selected_feat_meta_col	string with the name of the selected metadata from dt_depmap (will be used as a plot title)
with_1_item_grp	logical flag indicating whether to show group with only one item
max_x_lbl_length	numeric value for the maximum number of characters in the x-axis label
with_inf	a logical flag indicating whether infinite values should be shown on boxplots

Value

ggplot object containing boxplots for variable levels

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae_prism <- gDRutils::get_synthetic_data("prism")
se_prism <- mae_prism[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se_prism, "Metrics")
dt_response <- prep_dt_response_metric_sa(
  dt_metrics = dt_metrics,
  d_name = unique(dt_metrics[["DrugName"]])[1],
  normalization_type = "RV", metric = "x_mean")
meta_data_path <- system.file("depmap_data/Model.csv.gz", package = "gDRtestData")
obj_depmap_meta <- prep_dt_depmap_meta(meta_data_path, metadata_col = "OncotreeLineage")
plot_boxplot_meta(
  dt_response = dt_response,
  dt_depmap = obj_depmap_meta[["dt_depmap"]],
  selected_feat_meta_col = obj_depmap_meta[["selected_feat_meta_col"]])
```

plot_boxplot_metric_combo *Plot box plots for metric for combo data grouped by selected variable*

Description

Plot box plots for metric for combo data grouped by selected variable

Usage

```
plot_boxplot_metric_combo(
  dt_scores,
  group_var,
  normalization_type = "GR",
  metric = "hsa_score",
  fit_source = "gDR",
  grouped_flag = FALSE,
  colored_pts_flag = FALSE,
  colors_vec = NULL
)
```

Arguments

dt_scores	data.table representing data from the scores assay, outputted by gDRutils::convert_se_assay_ "scores") and combo SummarizedExperiment
group_var	string name of group variable; one of: "CellLineName" or "DrugName"; for group_var set as "DrugName" points will be grouped by drug combinations "DrugName" x "DrugName_2"
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the combo metric, any numeric column of dt_scores; e.g. "hsa_score" ("Bliss Excess GR" or "Bliss Excess RV" - respectively depending on normalization_type), "bliss_score" ("Bliss Score GR" or "Bliss Score RV") or a column written by a custom fit, such as beta from MuSyC
fit_source	string source name for metrics
grouped_flag	a logical flag whether the boxplots should be grouped and colored by Tissue for group_var set as "CellLineName" and drug_moa - for "DrugName"
colored_pts_flag	a logical flag whether the points should be colored by grouped variable - for group_var equal "CellLineName" points will be colored by "DrugName" and similarly vice versa
colors_vec	character vector with colors (name or hex value) to color boxplots; for grouped_flag set as FALSE only first from vector will be used

Value

ggplot object containing boxplots for selected combo metric grouped by group_var

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]

dt_scores <- gDRutils::convert_se_assay_to_dt(se = se,
                                              assay_name = "scores")

plot_boxplot_metric_combo(dt_scores,
                          group_var = "DrugName")

plot_boxplot_metric_combo(dt_scores,
                          group_var = "DrugName",
                          colored_pts_flag = TRUE,
                          colors_vec = "grey")

plot_boxplot_metric_combo(dt_scores,
                          group_var = "CellLineName",
                          metric = "hsa_score",
                          normalization_type = "RV",
                          grouped_flag = TRUE)

plot_boxplot_metric_combo(
  dt_scores,
  group_var = "CellLineName",
  metric = "hsa_score",
  grouped_flag = TRUE,
  colors_vec = c("deeppink", "darkblue"))
```

---

plot_boxplot_metric_combo_by_CLs

Plot box plots for metric for combo data grouped by cell line names

---

Description

Plot box plots for metric for combo data grouped by cell line names

Usage

```
plot_boxplot_metric_combo_by_CLs(
  dt_scores,
  normalization_type = "GR",
  metric = "hsa_score",
  fit_source = "gDR",
  grouped_flag = FALSE,
  colored_pts_flag = FALSE,
  colors_vec = NULL
)
```

Arguments

dt_scores	data.table representing data from the scores assay, outputted by gDRutils::convert_se_assay_ "scores") and combo SummarizedExperiment
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the combo metric, any numeric column of dt_scores; e.g. "hsa_score" ("Bliss Excess GR" or "Bliss Excess RV" - respectively depending on normalization_type), "bliss_score" ("Bliss Score GR" or "Bliss Score RV") or a column written by a custom fit, such as beta from MuSyC
fit_source	string source name for metrics
grouped_flag	a logical flag whether the boxplots should be grouped and colored by Tissue for group_var set as "CellLineName" and drug_moa - for "DrugName"
colored_pts_flag	a logical flag whether the points should be colored by grouped variable - for group_var equal "CellLineName" points will be colored by "DrugName" and similarly vice versa
colors_vec	character vector with colors (name or hex value) to color boxplots; for grouped_flag set as FALSE only first from vector will be used

Value

ggplot object containing boxplots for selected combo metric grouped by cell line names

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]

dt_scores <- gDRutils::convert_se_assay_to_dt(se = se,
                                              assay_name = "scores")

plot_boxplot_metric_combo_by_CLs(dt_scores)

plot_boxplot_metric_combo_by_CLs(dt_scores,
                                 normalization_type = "RV",
                                 grouped_flag = TRUE)

plot_boxplot_metric_combo_by_CLs(dt_scores,
                                 metric = "bliss_score",
                                 colors_vec = "gold")

plot_boxplot_metric_combo_by_CLs(
  dt_scores,
  metric = "bliss_score",
  colored_pts_flag = TRUE)

plot_boxplot_metric_combo_by_CLs(
  dt_scores,
```

```
  metric = "hsa_score",
  normalization_type = "RV",
  grouped_flag = TRUE,
  colors_vec = c("tomato", "darkgreen", "orange", "darkblue"))
```

`plot_boxplot_metric_combo_by_drugs` *Plot box plots for metric for combo data grouped by drug names*

Description

Plot box plots for metric for combo data grouped by drug names

Usage

```
plot_boxplot_metric_combo_by_drugs(
  dt_scores,
  normalization_type = "GR",
  metric = "hsa_score",
  fit_source = "gDR",
  colored_pts_flag = FALSE,
  colors_vec = NULL
)
```

Arguments

`dt_scores`	`data.table` representing data from the `scores` assay, outputted by `gDRutils::convert_se_assay_` `"scores")` and combo `SummarizedExperiment`
`normalization_type`	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
`metric`	string name of the combo metric, any numeric column of `dt_scores`; e.g. "hsa_score" ("Bliss Excess GR" or "Bliss Excess RV" - respectively depending on `normalization_type`), "bliss_score" ("Bliss Score GR" or "Bliss Score RV") or a column written by a custom fit, such as `beta` from MuSyC
`fit_source`	string source name for metrics
`colored_pts_flag`	a logical flag whether the points should be colored by grouped variable - for `group_var` equal `"CellLineName"` points will be colored by `"DrugName"` and similarly vice versa
`colors_vec`	character vector with colors (name or hex value) to color boxplots; for `grouped_flag` set as `FALSE` only first from vector will be used

Value

`ggplot` object containing boxplots for selected combo metric grouped by drug names

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]

dt_scores <- gDRutils::convert_se_assay_to_dt(se = se,
                                              assay_name = "scores")

plot_boxplot_metric_combo_by_drugs(dt_scores)

plot_boxplot_metric_combo_by_drugs(dt_scores,
                                   normalization_type = "RV",
                                   colors_vec = "gold")

plot_boxplot_metric_combo_by_drugs(dt_scores,
                                   metric = "bliss_score",
                                   normalization_type = "RV",
                                   colored_pts_flag = TRUE)
```

---

plot_boxplot_metric_combo_by_grp

Plot box plots for metric for combo data grouped by selected variable

---

Description

Plot box plots for metric for combo data grouped by selected variable

Usage

```
plot_boxplot_metric_combo_by_grp(
  dt_scores,
  selection_var,
  selection_name,
  group_var,
  group_names = NULL,
  normalization_type = "GR",
  metric = "hsa_score",
  fit_source = "gDR",
  named_n = 5,
  named_n_mode = "top",
  grouped_flag = FALSE,
  colors_vec = NULL
)
```

Arguments

dt_scores	data.table representing data from the scores assay, outputted by gDRutils::convert_se_assay_ "scores") and combo SummarizedExperiment
selection_var	string name of selected main variable - one value from column "CellLineName" or "DrugName"
selection_name	string name of selected variable value from column selection_var to filter data for plotting

group_var	string name of group variable; should to be not numeric variable from dt_metrics different than selection_var and not containing unique values for each row
group_names	character vector with names to subset from column group_var; if NULL then all values will be plotted
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the combo metric, any numeric column of dt_scores; e.g. "hsa_score" ("Bliss Excess GR" or "Bliss Excess RV" - respectively depending on normalization_type), "bliss_score" ("Bliss Score GR" or "Bliss Score RV") or a column written by a custom fit, such as beta from MuSyC
fit_source	string source name for metrics
named_n	number of points to label based on the highest or lowest metric values; if group_var is "DrugName", points are labeled by "CellLineName" and similarly vice versa
named_n_mode	string determines whether the labels are applied to the highest or lowest values of metric; one of: "top" or "bottom"
grouped_flag	logical flag whether the boxplots should be colored by group_var
colors_vec	character vector of colors (name or hex value), matched to group_names by position (1st color = 1st name) when grouped_flag = TRUE, independent of final plot order; missing/invalid/extra entries fall back to the default palette per group. If grouped_flag = FALSE, only the first color is used.

Value

ggplot object containing boxplots for selected combo metric grouped by selected variable

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]

dt_scores <- gDRutils::convert_se_assay_to_dt(se = se,
                                              assay_name = "scores")
invisible(dt_scores[, Tissue_grp := data.table::fifelse(Tissue == "tissue_w",
                                                        "tissue_w",
                                                        "tissue_other")])

plot_boxplot_metric_combo_by_grp(dt_scores,
                                 selection_var = "DrugName",
                                 selection_name = c("drug_001", "drug_021"),
                                 group_var = "Tissue")

plot_boxplot_metric_combo_by_grp(dt_scores,
                                 selection_var = "DrugName",
                                 selection_name = c("drug_001", "drug_021"),
                                 group_var = "Tissue_grp",
                                 grouped_flag = TRUE,
```

```
                                colors_vec = c("darkblue", "deeppink"))

plot_boxplot_metric_combo_by_grp(dt_scores,
                                selection_var = "DrugName",
                                selection_name = c("drug_011", "drug_021"),
                                group_var = "Tissue",
                                grouped_flag = TRUE)

plot_boxplot_metric_combo_by_grp(dt_scores,
                                selection_var = "DrugName",
                                selection_name = c("drug_011", "drug_021"),
                                group_var = "Tissue",
                                colors_vec = c("darkblue", "orange"))

plot_boxplot_metric_combo_by_grp(dt_scores,
                                selection_var = "DrugName",
                                selection_name = c("drug_001", "drug_021"),
                                group_var = "Tissue_grp",
                                grouped_flag = TRUE,
                                group_names = c("tissue_w", "tissue_other"),
                                colors_vec = c("black", "gold"))
```

`plot_boxplot_metric_sa` *Plot box plots for metric for single-agent data grouped by selected variable*

Description

Plot box plots for metric for single-agent data grouped by selected variable

Usage

```
plot_boxplot_metric_sa(
  dt_metrics,
  group_var,
  normalization_type = "GR",
  metric = "xc50",
  fit_source = "gDR",
  grouped_flag = FALSE,
  colored_pts_flag = FALSE,
  colors_vec = NULL,
  with_inf = FALSE
)
```

Arguments

`dt_metrics`	data.table representing data from the `Metrics` assay, outputted by `gDRutils::convert_se_assay_to "Metrics"`) and single-agent `SummarizedExperiment`
`group_var`	string name of group variable; one of: `"CellLineName"` or `"DrugName"`

normalization_type
: string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")

metric
: string name of the metric; one of: "xc50" ("GR50" or "IC50" - respectively depending on normalization_type), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean"), but the values from any numeric column can be displayed

fit_source
: string source name for metrics

grouped_flag
: a logical flag whether the boxplots should be grouped and colored by Tissue for group_var set as "CellLineName" and drug_moa - for "DrugName"

colored_pts_flag
: a logical flag whether the points should be colored by grouped variable - for group_var equal "CellLineName" points will be colored by "DrugName" and similarly vice versa

colors_vec
: character vector with colors (name or hex value) to color boxplots; for grouped_flag set as FALSE only first from vector will be used

with_inf
: a logical flag indicating whether infinite values should be shown on boxplots

Value

ggplot object containing boxplots for selected single-agent metric grouped by group_var

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("sa")]]

dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")

plot_boxplot_metric_sa(dt_metrics,
                       group_var = "CellLineName")

plot_boxplot_metric_sa(dt_metrics,
                       group_var = "DrugName")

plot_boxplot_metric_sa(dt_metrics,
                       group_var = "DrugName",
                       metric = "x_AOC_range",
                       colors_vec = "grey",
                       colored_pts_flag = TRUE)

plot_boxplot_metric_sa(dt_metrics,
                       group_var = "CellLineName",
                       normalization_type = "RV",
                       metric = "x_max",
                       colors_vec = c("gold", "darkorange", "darkcyan", "darkblue"),
                       grouped_flag = TRUE)
```

plot_boxplot_metric_sa_by_CLs *Plot box plots for metric for single-agent data grouped by cell line names*

Description

Plot box plots for metric for single-agent data grouped by cell line names

Usage

```
plot_boxplot_metric_sa_by_CLs(
  dt_metrics,
  normalization_type = "GR",
  metric = "xc50",
  fit_source = "gDR",
  grouped_flag = FALSE,
  colored_pts_flag = FALSE,
  colors_vec = NULL,
  with_inf = FALSE
)
```

Arguments

dt_metrics	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay_to "Metrics") and single-agent SummarizedExperiment
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the metric; one of: "xc50" ("GR50" or "IC50" - respectively depending on normalization_type), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean"), but the values from any numeric column can be displayed
fit_source	string source name for metrics
grouped_flag	a logical flag whether the boxplots should be grouped and colored by Tissue for group_var set as "CellLineName" and drug_moa - for "DrugName"
colored_pts_flag	a logical flag whether the points should be colored by grouped variable - for group_var equal "CellLineName" points will be colored by "DrugName" and similarly vice versa
colors_vec	character vector with colors (name or hex value) to color boxplots; for grouped_flag set as FALSE only first from vector will be used
with_inf	a logical flag indicating whether infinite values should be shown on boxplots

Value

ggplot object containing boxplots for selected single-agent metric grouped by cellline names

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("sa")]]

dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")

plot_boxplot_metric_sa_by_CLs(dt_metrics)

plot_boxplot_metric_sa_by_CLs(dt_metrics,
                              grouped_flag = TRUE)

plot_boxplot_metric_sa_by_CLs(dt_metrics,
                              with_inf = TRUE)

plot_boxplot_metric_sa_by_CLs(dt_metrics,
                              metric = "x_AOC",
                              colors_vec = "gold")

plot_boxplot_metric_sa_by_CLs(
  dt_metrics,
  metric = "x_max",
  colored_pts_flag = TRUE)

plot_boxplot_metric_sa_by_CLs(
  dt_metrics,
  metric = "x_mean",
  grouped_flag = TRUE,
  colors_vec = c("deeppink", "darkcyan", "orange", "darkblue"))
```

---

plot_boxplot_metric_sa_by_drugs

Plot box plots for metric for single-agent data grouped by drug names

---

Description

Plot box plots for metric for single-agent data grouped by drug names

Usage

```
plot_boxplot_metric_sa_by_drugs(
  dt_metrics,
  normalization_type = "GR",
  metric = "xc50",
  fit_source = "gDR",
  grouped_flag = FALSE,
  colored_pts_flag = FALSE,
  colors_vec = NULL,
  with_inf = FALSE
)
```

Arguments

dt_metrics	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay_to "Metrics") and single-agent SummarizedExperiment
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the metric; one of: "xc50" ("GR50" or "IC50" - respectively depending on normalization_type), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean"), but the values from any numeric column can be displayed
fit_source	string source name for metrics
grouped_flag	a logical flag whether the boxplots should be grouped and colored by Tissue for group_var set as "CellLineName" and drug_moa - for "DrugName"
colored_pts_flag	a logical flag whether the points should be colored by grouped variable - for group_var equal "CellLineName" points will be colored by "DrugName" and similarly vice versa
colors_vec	character vector with colors (name or hex value) to color boxplots; for grouped_flag set as FALSE only first from vector will be used
with_inf	a logical flag indicating whether infinite values should be shown on boxplots

Value

ggplot object containing boxplots for selected single-agent metric grouped by drug names

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("sa")]]

dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")

plot_boxplot_metric_sa_by_drugs(dt_metrics,
                                normalization_type = "RV")

plot_boxplot_metric_sa_by_drugs(dt_metrics,
                                normalization_type = "RV",
                                grouped_flag = TRUE)

plot_boxplot_metric_sa_by_drugs(dt_metrics,
                                metric = "x_AOC",
                                colors_vec = "gold")

plot_boxplot_metric_sa_by_drugs(
  dt_metrics,
  metric = "x_max",
  colored_pts_flag = TRUE)

plot_boxplot_metric_sa_by_drugs(
```

```
  dt_metrics,
  metric = "x_mean",
  grouped_flag = TRUE,
  colors_vec = c("deeppink", "darkcyan", "orange", "darkblue", "limegreen"))
```

---

plot_boxplot_metric_sa_by_grp *Plot box plots for metric for single-agent data grouped by selected variable*

Description

Plot box plots for metric for single-agent data grouped by selected variable

Usage

```
plot_boxplot_metric_sa_by_grp(
  dt_metrics,
  selection_var,
  selection_name,
  group_var,
  group_names = NULL,
  normalization_type = "GR",
  metric = "xc50",
  fit_source = "gDR",
  named_n = 5,
  named_n_mode = "bottom",
  grouped_flag = FALSE,
  colors_vec = NULL,
  with_inf = FALSE
)
```

Arguments

dt_metrics	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay_to "Metrics") and single-agent SummarizedExperiment
selection_var	string name of selected main variable - one value from column "CellLineName" or "DrugName"
selection_name	string name of selected variable value from column selection_var to filter data for plotting
group_var	string name of group variable; should to be not numeric variable from dt_metrics different than selection_var and not containing unique values for each row
group_names	character vector with names to subset from column group_var; if NULL then all values will be plotted
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")

metric	string name of the metric; one of: "xc50" ("GR50" or "IC50" - respectively depending on normalization_type), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean"), but the values from any numeric column can be displayed
fit_source	string source name for metrics
named_n	number of points to label based on the highest or lowest metric values; if group_var is "DrugName", points are labeled by "CellLineName" and similarly vice versa
named_n_mode	string determines whether the labels are applied to the highest or lowest values of metric; one of: "top" or "bottom"
grouped_flag	logical flag whether the boxplots should be colored by group_var
colors_vec	character vector of colors (name or hex value), matched to group_names by position (1st color = 1st name) when grouped_flag = TRUE, independent of final plot order; missing/invalid/extra entries fall back to the default palette per group. If grouped_flag = FALSE, only the first color is used.
with_inf	a logical flag indicating whether infinite values should be shown on boxplots

Value

ggplot object containing boxplots for selected single-agent metric grouped by selected variable

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("sa")]]

dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")
invisible(dt_metrics[, Tissue_grp := data.table::fifelse(Tissue == "tissue_w",
                                                         "tissue_w",
                                                         "tissue_other")])

plot_boxplot_metric_sa_by_grp(dt_metrics,
                              selection_var = "DrugName",
                              selection_name = "drug_011",
                              group_var = "Tissue")

plot_boxplot_metric_sa_by_grp(dt_metrics,
                              selection_var = "DrugName",
                              selection_name = "drug_001",
                              group_var = "Tissue_grp")

plot_boxplot_metric_sa_by_grp(dt_metrics,
                              selection_var = "DrugName",
                              selection_name = "drug_011",
                              group_var = "Tissue",
                              with_inf = TRUE)

plot_boxplot_metric_sa_by_grp(dt_metrics,
                              selection_var = "DrugName",
                              selection_name = "drug_011",
```

```
                              group_var = "Tissue",
                              named_n = 0)

plot_boxplot_metric_sa_by_grp(dt_metrics,
                              selection_var = "DrugName",
                              selection_name = "drug_011",
                              group_var = "Tissue",
                              grouped_flag = TRUE)

plot_boxplot_metric_sa_by_grp(dt_metrics,
                              selection_var = "DrugName",
                              selection_name = "drug_011",
                              group_var = "Tissue",
                              colors_vec = c("darkblue", "deeppink"))

plot_boxplot_metric_sa_by_grp(dt_metrics,
                              selection_var = "DrugName",
                              selection_name = "drug_001",
                              group_var = "Tissue_grp",
                              grouped_flag = TRUE,
                              group_names = c("tissue_w", "tissue_other"),
                              colors_vec = c("black", "gold"))
```

`plot_boxplot_num` *Plot boxplot for categorical features*

Description

Plot boxplot for categorical features

Usage

```
plot_boxplot_num(
  dt_response,
  dt_depmap,
  selected_feat,
  selected_feat_meta_col = NULL
)
```

Arguments

`dt_response`	`data.table` with experimental response data (rows are samples) for one metric outputted by one of functions: `prep_dt_response_metric_sa`, `prep_dt_response_dose_sa`, `prep_dt_response_scores` or `prep_dt_response_metric_diff`,
`dt_depmap`	`data.table` with dependent variables data loaded from DepMap - for one feature; (rows are samples, columns are features). outputted by `prep_dt_depmap_feat`
`selected_feat`	string with name of selected feature from `dt_depmap`
`selected_feat_meta_col`	string with the name of a feature column in DepMap (will be used as a plot title) that has 0-1 values only (categorical character but without relation one-to-one for ids and feature)

Value

ggplot object containing boxplots for variable levels

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae_prism <- gDRutils::get_synthetic_data("prism")
se_prism <- mae_prism[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se_prism, "Metrics")
dt_response <- prep_dt_response_metric_sa(
  dt_metrics = dt_metrics,
  d_name = unique(dt_metrics[["DrugName"]])[1],
  normalization_type = "RV", metric = "x_mean")
meta_data_path <- system.file("depmap_data/Model.csv.gz", package = "gDRtestData")
obj_depmap_meta <- prep_dt_depmap_meta(meta_data_path, metadata_col = "OncotreeLineage")
plot_boxplot_num(
  dt_response = dt_response,
  dt_depmap = obj_depmap_meta[["dt_depmap"]],
  selected_feat = names(obj_depmap_meta[["dt_depmap"]])[3])
```

---

plot_boxplot_num_panel

Plot panel with boxplots

Description

Plot panel with boxplots

Usage

```
plot_boxplot_num_panel(
  dt_response,
  dt_depmap,
  selected_feats,
  selected_feat_meta_col = NULL,
  ncol = NULL
)
```

Arguments

dt_response	data.table with experimental response data (rows are samples) for one metric outputted by one of functions: prep_dt_response_metric_sa, prep_dt_response_dose_sa, prep_dt_response_scores or prep_dt_response_metric_diff,
dt_depmap	data.table with dependent variables data loaded from DepMap - for one feature; (rows are samples, columns are features). outputted by prep_dt_depmap_feat
selected_feats	character vector with names of selected features from dt_depmap

selected_feat_meta_col
: string with the name of a feature column in DepMap (will be used as a plot title) that has 0-1 values only (categorical character but without relation one-to-one for ids and feature)

ncol
: number of plot column in panel

Value

ggplot object containing boxplots for variable levels (0-1)

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae_prism <- gDRutils::get_synthetic_data("prism")
se_prism <- mae_prism[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se_prism, "Metrics")
dt_response <- prep_dt_response_metric_sa(
  dt_metrics = dt_metrics,
  d_name = unique(dt_metrics[["DrugName"]])[1],
  normalization_type = "RV", metric = "x_mean")
meta_data_path <- system.file("depmap_data/Model.csv.gz", package = "gDRtestData")
obj_depmap_meta <- prep_dt_depmap_meta(meta_data_path, metadata_col = "OncotreeLineage")
plot_boxplot_num_panel(
  dt_response = dt_response,
  dt_depmap = obj_depmap_meta[["dt_depmap"]],
  selected_feats = names(obj_depmap_meta[["dt_depmap"]])[3:5])
```

---

plot_cgs_ranking *Plot Chemical Genomics Screen GSEA Results*

Description

Generates a ggplot2 visualization of chemical genomics screening data, highlighting GSEA results.

Usage

```
plot_cgs_ranking(
  results,
  cl_name,
  metric,
  padj_threshold = 0.1,
  top_results_no_sig = 5,
  max_results_with_sig = 15
)
```

Arguments

results	A list object returned from 'analyze_cgs'.
cl_name	A string specifying the cell line included in the results to prepare a visualization.
metric	A string specifying the metric included in the results to prepare a visualization.
padj_threshold	A numeric value specifying the threshold for filtering significant GSEA results based on adjusted p-value.
top_results_no_sig	A numeric value specifying the number of top results to plot if there are no significant values.
max_results_with_sig	A numeric value specifying the maximum number of results to plot when there are more than this number of significant values.

Value

A ggplot2 object with cgs results

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
dt_metrics <- qs2::qs_read(system.file("testdata/cgs_data.qs2", package = "gDRplots"))
results <- analyze_cgs(dt_metrics, metrics = c("xc50"), cl_name = "CellLineName_1")
plot_cgs_ranking(results,
  cl_name = "CellLineName_1",
  metric = "xc50",
  padj_threshold = 0.1,
  top_results_no_sig = 5,
  max_results_with_sig = 15)
```

---

plot_combination_index

Plot line plot of combination index

Description

Plot line plot of combination index

Usage

```
plot_combination_index(
  dt_excess,
  dt_isobolograms,
  drug1_name,
  drug2_name,
  cl_name,
  normalization_type = "GR",
```

```
  iso_levels = c("0.25", "0.5", "0.75"),
  colors_vec_iso = NULL
)
```

Arguments

dt_excess	data.table representing data from the excess assay, outputted by gDRutils::convert_se_assay_to_ "excess") and combo SummarizedExperiment
dt_isobolograms	data.table representing data from the isobolograms assay, outputted by gDRutils::convert_se_ass "isobolograms") and combo SummarizedExperiment
drug1_name	string with drug name to be plotted (identifiers DrugName)
drug2_name	string with co-drug name to be plotted (identifiers DrugName_2)
cl_name	string with cell line to be plotted (identifiers CellLineName)
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
iso_levels	character vector with isobologram levels to be selected
colors_vec_iso	character vector of colors (valid name or hex) used for the isolines; the default is the dark red-orange palette

Value

ggplot object containing combination index plot at different ratios of the two drugs

Examples

```
cl_name <- "cellline_BC"
drug1_name <- "drug_001"
drug2_name <- "drug_026"

mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_excess <- gDRutils::convert_se_assay_to_dt(se, "excess")
dt_isobolograms <- gDRutils::convert_se_assay_to_dt(se, "isobolograms")

plot_combination_index(dt_excess,
                       dt_isobolograms,
                       drug1_name, drug2_name,
                       cl_name,
                       normalization_type = "GR")

plot_combination_index(dt_excess,
                       dt_isobolograms,
                       drug1_name, drug2_name,
                       cl_name,
                       normalization_type = "RV",
                       colors_vec_iso = c("darkblue", "darkcyan"))

cl_name <- "cellline_JE"
drug1_name <- "drug_011"
drug2_name <- "drug_026"

plot_combination_index(dt_excess,
```

```
                    dt_isobolograms,
                    drug1_name, drug2_name,
                    cl_name,
                    normalization_type = "RV",
                    iso_levels = "0.5")
```

`plot_dose_response_combo` *Plot drug response curves for combo data*

Description

Plot drug response curves for combo data

Usage

```
plot_dose_response_combo(
  dt_average,
  drug1_name,
  drug2_name,
  cl_name,
  normalization_type = "GR",
  colors_vec = NULL,
  split_by_conc = FALSE
)
```

Arguments

`dt_average`	data.table representing data from the `Averaged` assay, outputted by `gDRutils::convert_se_assay_` `"Averaged")` and combo `SummarizedExperiment`
`drug1_name`	string with drug name to be plotted (identifiers `DrugName`)
`drug2_name`	string with co-drug name to be plotted (identifiers `DrugName_2`)
`cl_name`	string with cell line name to be plotted (identifiers `CellLineName`)
`normalization_type`	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
`colors_vec`	character vector with colors for `Concentration_2` - name or hex value note that the first color will be assigned to the min value of `Concentration_2`, and the last one - to the max of `Concentration_2`; the default is the orange-black palette
`split_by_conc`	split_by_conc logical flag indicating whether curves for `Concentration_2` should be plotted on a single plot or separately

Value

`ggplot` object containing plot with dose-response curves for combo data

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_average <- gDRutils::convert_se_assay_to_dt(se, "Averaged")

cl_name <- "cellline_BC"
drug1_name <- "drug_011"
drug2_name <- "drug_021"

plot_dose_response_combo(dt_average = dt_average,
                         drug1_name = drug1_name,
                         drug2_name = drug2_name,
                         cl_name = cl_name)

plot_dose_response_combo(dt_average = dt_average,
                         drug1_name = drug1_name,
                         drug2_name = drug2_name,
                         cl_name = cl_name,
                         split_by_conc = TRUE)

plot_dose_response_combo(dt_average = dt_average,
                         drug1_name = drug1_name,
                         drug2_name = drug2_name,
                         cl_name = cl_name,
                         normalization_type = "RV",
                         colors_vec = c("lightblue", "darkblue"))
```

---

plot_dose_response_combo_panel

Plot panel with drug response curves for single-agent data to control quality of the data

---

Description

Plot panel with drug response curves for single-agent data to control quality of the data

Usage

```
plot_dose_response_combo_panel(
  dt_average,
  cl_name,
  d_names = NULL,
  normalization_type = "GR",
  colors_vec = NULL
)
```

Arguments

dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assay_t "Averaged") and combo SummarizedExperiment
cl_name	string with cell line name to be plotted (identifiers CellLineName)

d_names — character vector with drug names to be plotted (Drug Name); if NULL - all available drugs will be plotted

normalization_type — string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")

colors_vec — character vector with colors for Concentration_2 - name or hex value note that the first color will be assigned to the min value of Concentration_2, and the last one - to the max of Concentration_2; the default is the orange-black palette

Value

ggplot object containing panel with plot with dose-response curves for selected cell line by drugs

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_average <- gDRutils::convert_se_assay_to_dt(se, "Averaged")

cl_name <- "cellline_IB"

plot_dose_response_combo_panel(dt_average = dt_average,
                               cl_name = cl_name)

d_names <- c("drug_001", "drug_002")
plot_dose_response_combo_panel(dt_average = dt_average,
                               cl_name = cl_name,
                               d_names = d_names)
```

---

plot_dose_response_sa *Plot drug response curves for single-agent data*

Description

Plot drug response curves for single-agent data

Usage

```
plot_dose_response_sa(
  dt_metrics,
  dt_average,
  selection_name,
  group_var,
  group_names = NULL,
  normalization_type = "GR",
  colors_vec = NULL,
  plot_averaged_flag = TRUE,
  plot_fit_flag = TRUE,
  fit_source = "gDR"
)
```

Arguments

dt_metrics	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay_to "Metrics") and single-agent SummarizedExperiment
dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assay_ "Averaged") and single-agent SummarizedExperiment
selection_name	string name of selected main variable - one value from column "CellLineName" or "DrugName"
group_var	string name of group variable; one of: "CellLineName" or "DrugName"
group_names	character vector with names to subset from se (the same dim as group_var); if NULL then all values will be plotted
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
colors_vec	character vector of colors (name or hex value), matched to group_names by position, or by name if fully named. Invalid, missing, or extra entries fall back to the default palette (get_qual_colors); nothing is recycled.
plot_averaged_flag	logical flag whether plot points with average values
plot_fit_flag	logical flag whether plot points with fitted values
fit_source	string source name for metrics

Value

ggplot object containing plot of dose-response curves

Note

inspired by the grob_SA function written by Marc Hafner

Examples

```
mae <- gDRutils::get_synthetic_data("small")
se <- mae[[gDRutils::get_supported_experiments("sa")]]
selected_drug <- "drug_002"
group_var <- "CellLineName"
dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")
dt_average <- gDRutils::convert_se_assay_to_dt(se, "Averaged")
celline_names <- unique(dt_metrics[[group_var]])[1:3]

plot_dose_response_sa(dt_metrics = dt_metrics,
                      dt_average = dt_average,
                      selection_name = selected_drug,
                      group_var = group_var,
                      group_names = celline_names)

plot_dose_response_sa(dt_metrics = dt_metrics,
                      dt_average = NULL,
                      selection_name = selected_drug,
                      group_var = group_var,
                      group_names = celline_names)

plot_dose_response_sa(dt_metrics = dt_metrics,
```

```
                      dt_average = NULL,
                      selection_name = selected_drug,
                      group_var = group_var,
                      group_names = celline_names,
                      colors_vec = c("red", "blue"))

selected_cellline <- "cellline_HB"
group_var <- "DrugName"
dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")
dt_average <- gDRutils::convert_se_assay_to_dt(se, "Averaged")
group_names <- unique(dt_metrics[[group_var]])[1:3]

plot_dose_response_sa(dt_metrics = dt_metrics,
                      dt_average = dt_average,
                      selection_name = selected_cellline,
                      group_var = group_var,
                      group_names = group_names)

plot_dose_response_sa(dt_metrics = dt_metrics,
                      dt_average = dt_average,
                      selection_name = selected_cellline,
                      group_var = group_var,
                      group_names = rev(group_names),
                      colors_vec = c("lightblue", "orange"))

my_colors <- c("drug_004" = "black")
my_widths <- c("drug_002" = 0.5, "drug_003" = 0.5, "drug_004" = 2)
suppressMessages(suppressWarnings(
  plot_dose_response_sa(dt_metrics = dt_metrics,
                        dt_average = dt_average,
                        selection_name = selected_cellline,
                        group_var = group_var,
                        group_names = group_names) +
    ggplot2::scale_color_manual(values = my_colors) +
    ggplot2::aes(linewidth = group_var) +
    ggplot2::scale_linewidth_manual(values = my_widths) +
    ggplot2::labs(color = "Drugs by color", linewidth = "Drugs by linewidth")
))

suppressMessages(suppressWarnings(print(
  plot_dose_response_sa(dt_metrics = dt_metrics,
                        dt_average = dt_average,
                        selection_name = selected_cellline,
                        group_var = group_var,
                        group_names = group_names,
                        colors_vec = "black") +
    ggplot2::aes(linetype = group_var, color = NULL)
)))
```

`plot_dose_response_sa_by_CLs` *Plot drug response curves for single-agent data for selected call lines and drugs*

Description

Plot drug response curves for single-agent data for selected call lines and drugs

Usage

```
plot_dose_response_sa_by_CLs(
  dt_metrics,
  dt_average,
  cellline_name_vec = NULL,
  drug_name_vec = NULL,
  normalization_type = "GR",
  colors_vec = NULL,
  plot_averaged_flag = TRUE,
  plot_fit_flag = TRUE,
  fit_source = "gDR"
)
```

Arguments

dt_metrics	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay_to "Metrics") and single-agent SummarizedExperiment
dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assay_ "Averaged") and single-agent SummarizedExperiment
cellline_name_vec	character vector with cell line names to be plotted (Cell Line Name)
drug_name_vec	character vector with drug names to be plotted (Drug Name)
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
colors_vec	character vector of colors (name or hex value), matched to group_names by position, or by name if fully named. Invalid, missing, or extra entries fall back to the default palette (get_qual_colors); nothing is recycled.
plot_averaged_flag	logical flag whether plot points with average values
plot_fit_flag	logical flag whether plot points with fitted values
fit_source	string source name for metrics

Value

list of ggplot objects containing plots of dose-response curves

Examples

```
mae <- gDRutils::get_synthetic_data("small")
se <- mae[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")
dt_average <- gDRutils::convert_se_assay_to_dt(se, "Averaged")
cellline_name_vec <- unique(dt_metrics[["CellLineName"]])[2:5]
drug_name_vec <- unique(dt_metrics[["DrugName"]])[5:7]

plot_dose_response_sa_by_CLs(dt_metrics = dt_metrics,
```

```
                              dt_average = dt_average,
                              cellline_name_vec = cellline_name_vec,
                              drug_name_vec = drug_name_vec,
                              normalization_type = "RV",
                              colors_vec = c("#00008B", "#FF6347", "#4CBB17"))

plot_dose_response_sa_by_CLs(dt_metrics = dt_metrics,
                              dt_average = dt_average,
                              cellline_name_vec = cellline_name_vec,
                              drug_name_vec = drug_name_vec,
                              normalization_type = "RV")
```

`plot_dose_response_sa_by_drugs` *Plot drug response curves for single-agent data for selected call lines and drugs*

Description

Plot drug response curves for single-agent data for selected call lines and drugs

Usage

```
plot_dose_response_sa_by_drugs(
  dt_metrics,
  dt_average,
  cellline_name_vec = NULL,
  drug_name_vec = NULL,
  normalization_type = "GR",
  colors_vec = NULL,
  plot_averaged_flag = TRUE,
  plot_fit_flag = TRUE,
  fit_source = "gDR"
)
```

Arguments

`dt_metrics`	data.table representing data from the `Metrics` assay, outputted by `gDRutils::convert_se_assay_to "Metrics")` and single-agent `SummarizedExperiment`
`dt_average`	data.table representing data from the `Averaged` assay, outputted by `gDRutils::convert_se_assay_ "Averaged")` and single-agent `SummarizedExperiment`
`cellline_name_vec`	character vector with cell line names to be plotted (Cell Line Name)
`drug_name_vec`	character vector with drug names to be plotted (Drug Name)
`normalization_type`	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
`colors_vec`	character vector of colors (name or hex value), matched to `group_names` by position, or by name if fully named. Invalid, missing, or extra entries fall back to the default palette (`get_qual_colors`); nothing is recycled.

plot_averaged_flag
: logical flag whether plot points with average values

plot_fit_flag
: logical flag whether plot points with fitted values

fit_source
: string source name for metrics

Value

list of ggplot objects containing plots of dose-response curves

Examples

```
mae <- gDRutils::get_synthetic_data("small")
se <- mae[[gDRutils::get_supported_experiments("sa")]]

dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")
dt_average <- gDRutils::convert_se_assay_to_dt(se, "Averaged")
cellline_name_vec <- unique(dt_metrics[["CellLineName"]])[2:5]
drug_name_vec <- unique(dt_metrics[["DrugName"]])[5:7]

plot_dose_response_sa_by_drugs(dt_metrics = dt_metrics,
                               dt_average = dt_average,
                               cellline_name_vec = cellline_name_vec,
                               drug_name_vec = drug_name_vec,
                               normalization_type = "RV",
                               colors_vec = c("#00008B", "#FF6347", "#4CBB17"))
```

---

plot_dose_response_sa_qc

Plot drug response curves for single-agent data to control quality of the data

---

Description

Plot drug response curves for single-agent data to control quality of the data

Usage

```
plot_dose_response_sa_qc(
  dt_metrics,
  dt_average,
  cl_name,
  d_name,
  normalization_type = "GR",
  fit_source = "gDR"
)
```

Arguments

dt_metrics
: data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay_to "Metrics") and single-agent SummarizedExperiment

dt_average
: data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assay_ "Averaged") and single-agent SummarizedExperiment

cl_name	string cell line name to be plotted (Cell Line Name)
d_name	string vector with drug name to be plotted (Drug Name)
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
fit_source	string source name for metrics

Value

ggplot object containing plot of dose-response curves (observed and fitted values)

Examples

```
mae <- gDRutils::get_synthetic_data("small")
se <- mae[[gDRutils::get_supported_experiments("sa")]]

dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")
dt_average <- gDRutils::convert_se_assay_to_dt(se, "Averaged")
cl_name <- dt_metrics[["CellLineName"]][1]
d_name <- dt_metrics[["DrugName"]][1]

plot_dose_response_sa_qc(dt_metrics = dt_metrics,
                         dt_average = dt_average,
                         cl_name = cl_name,
                         d_name = d_name)
```

---

plot_dose_response_sa_qc_panel

Plot panel with drug response curves for single-agent data to control quality of the data

---

Description

Plot panel with drug response curves for single-agent data to control quality of the data

Usage

```
plot_dose_response_sa_qc_panel(
  dt_metrics,
  dt_average,
  cl_name,
  d_names = NULL,
  normalization_type = "GR",
  fit_source = "gDR"
)
```

Arguments

dt_metrics	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay_to "Metrics") and single-agent SummarizedExperiment
dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assay_ "Averaged") and single-agent SummarizedExperiment
cl_name	string cell line name to be plotted (Cell Line Name)
d_names	character vector with drug names to be plotted (Drug Name); if NULL - all available drugs will be plotted
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
fit_source	string source name for metrics

Value

ggplot object with panel with plots of dose-response curves for selected cell line by drugs (observed and fitted values)

Examples

```
mae <- gDRutils::get_synthetic_data("small")
se <- mae[[1]]

dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")
dt_average <- gDRutils::convert_se_assay_to_dt(se, "Averaged")
cl_name <- dt_metrics[["CellLineName"]][1]
d_names <- unique(dt_metrics[["DrugName"]])[1:5]

plot_dose_response_sa_qc_panel(dt_metrics = dt_metrics,
                               dt_average = dt_average,
                               cl_name = cl_name)

plot_dose_response_sa_qc_panel(dt_metrics = dt_metrics,
                               dt_average = dt_average,
                               cl_name = cl_name,
                               d_names = d_names)
```

---

plot_fitting_acc *Visualization for the quality control of the fitting for single-agent data*

Description

Visualization for the quality control of the fitting for single-agent data

Usage

```
plot_fitting_acc(dt_assay, cl_name, normalization_type = "GR")
```

Arguments

dt_assay	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay_to "Metrics") and single-agent SummarizedExperiment
cl_name	string cell line name to be plotted (Cell Line Name)
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")

Value

ggplot object containing panel with lollipop plots with r2 and rss values for each drug

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
mae <- gDRutils::get_synthetic_data("small")
se <- mae[[gDRutils::get_supported_experiments("sa")]]

dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")
cl_name <- dt_metrics[["CellLineName"]][1]

plot_fitting_acc(dt_assay = dt_metrics,
                 cl_name = cl_name,
                 normalization_type = "RV")
```

---

plot_plate *Plot data from a specific column*

Description

Plot data from a specific column

Usage

```
plot_plate(dt_plate, column_name)
```

Arguments

dt_plate	The data table containing plate information
column_name	The name of the column to plot

Value

A named list of ggplot objects for each barcode and column

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
test_data <- data.table::data.table(
  WellColumn = rep(1:12, each = 8),
  WellRow = rep(LETTERS[1:8], times = 12),
  clid = "A",
  Gnumber = c(rep("untreated", 48), sample(1:5, size = 48, replace = TRUE)),
  Gnumber_2 = c(rep("untreated", 48), sample(1:5, size = 48, replace = TRUE)),
  Concentration = runif(96, min = 0, max = 100),
  ReadoutValue = runif(96, min = 0, max = 100),
  Barcode = rep(c("A", "B"), 48)
  )
plot_plate(test_data, "Gnumber")[["A"]]
```

---

plot_plate_stack_info *Plot plate data (Heatmap + Dose Ranks + Smart Legend + Explicit QC)*

Description

Plot plate data (Heatmap + Dose Ranks + Smart Legend + Explicit QC)

Usage

```
plot_plate_stack_info(
  dt_plate,
  ctrl_fail_threshold = 0.6,
  n_sd = 1,
  use_sd_threshold = TRUE,
  items_per_line = 6
)
```

Arguments

dt_plate	`data.table`. Input data containing plate layout, measurements, and standard gDR identifiers.
ctrl_fail_threshold	`numeric` (Default: 0.6). Flags control wells with a readout below `Mean(Controls) * ctrl_fail_threshold`.
n_sd	`numeric` (Default: 1). Sets the upper limit for flagging high signals using the formula `Mean(Controls) + (n_sd * SD(Controls))`.
use_sd_threshold	`logical` (Default: TRUE). If `TRUE`, uses the dynamic SD-based limit defined by `n_sd`. If `FALSE`, uses a fixed limit of `Mean(Controls) * 1.1`.
items_per_line	`integer`. Number of doses to show per line in the legend. Default 6.

Value

A named list of ggplot objects for each barcode

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
conc_series <- c(0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10)

test_data <- data.table::data.table(
   WellColumn = sprintf("%02d", rep(1:12, each = 8)),
   WellRow = rep(LETTERS[1:8], times = 12),
   Barcode = rep(c("Plate_1", "Plate_2"), each = 48),
   clid = "CellLineA"
)

invisible(test_data[, Concentration := rep(rep(conc_series, length.out = 48), 2)])
invisible(test_data[, Gnumber := ifelse(Concentration == 0, "vehicle", "Drug_A")])
invisible(test_data[Gnumber == "vehicle", Concentration := 0])

invisible(test_data[, ReadoutValue := ifelse(Gnumber == "vehicle",
                                    rnorm(.N, 1000, 50),
                                    rnorm(.N, 500, 100))])
library(ggtext)
plots <- plot_plate_stack_info(test_data)
plots[[1]]

combo_data <- data.table::copy(test_data)
invisible(combo_data[, Barcode := "Plate_Combo"])
invisible(combo_data[, Concentration_2 := rep(rev(conc_series), length.out = .N)])
invisible(combo_data[, Gnumber_2 := ifelse(Concentration_2 == 0, "vehicle", "Drug_B")])
invisible(combo_data[Gnumber_2 == "vehicle", Concentration_2 := 0])

combo_plots <- plot_plate_stack_info(combo_data)
combo_plots[[1]]
```

`plot_scatter_with_corr` *Plot scatter with correlation*

Description

Plot scatter with correlation

Usage

```
plot_scatter_with_corr(
  dt_response,
  dt_depmap,
  selected_feat,
  selected_feat_meta_col = NULL
)
```

Arguments

`dt_response`	`data.table` with experimental response data (rows are samples) for one metric outputted by one of functions: `prep_dt_response_metric_sa`, `prep_dt_response_dose_sa`, `prep_dt_response_scores` or `prep_dt_response_metric_diff`,

dt_depmap — data.table with dependent variables data loaded from DepMap - for one feature; (rows are samples, columns are features). outputted by prep_dt_depmap_feat

selected_feat — string with name of selected feature from dt_depmap

selected_feat_meta_col — string with the name of a feature column in DepMap (will be used as a plot title)

Value

ggplot object containing a scatter plot with correlation

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae_prism <- gDRutils::get_synthetic_data("prism")
se_prism <- mae_prism[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se_prism, "Metrics")
dt_response <- prep_dt_response_metric_sa(
  dt_metrics = dt_metrics,
  d_name = unique(dt_metrics[["DrugName"]])[1],
  normalization_type = "RV", metric = "x_mean")
feat_data_path <- system.file("depmap_data", package = "gDRtestData")
meta_data_path <- system.file("depmap_data/Model.csv.gz", package = "gDRtestData")
obj_depmap_feat <- prep_dt_depmap_feat(feat_data_path, meta_data_path,
                                       feature_set = "OmicsCNGene")
plot_scatter_with_corr(
  dt_response = dt_response,
  dt_depmap = obj_depmap_feat[["dt_depmap"]],
  selected_feat = names(obj_depmap_feat[["dt_depmap"]])[3])
```

---

plot_scatter_with_corr_panel

Plot panel with scatter with correlation

Description

Plot panel with scatter with correlation

Usage

```
plot_scatter_with_corr_panel(
  dt_response,
  dt_depmap,
  selected_feats,
  selected_feat_meta_col = NULL,
  ncol = NULL
)
```

Arguments

dt_response	data.table with experimental response data (rows are samples) for one metric outputted by one of functions: prep_dt_response_metric_sa, prep_dt_response_dose_sa, prep_dt_response_scores or prep_dt_response_metric_diff,
dt_depmap	data.table with dependent variables data loaded from DepMap - for one feature; (rows are samples, columns are features). outputted by prep_dt_depmap_feat
selected_feats	character vector with names of selected features from dt_depmap
selected_feat_meta_col	string with the name of a feature column in DepMap (will be used as a plot title)
ncol	number of plot column in panel

Value

ggplot object containing panel of scatter plot with correlation for selected features

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae_prism <- gDRutils::get_synthetic_data("prism")
se_prism <- mae_prism[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se_prism, "Metrics")
dt_response <- prep_dt_response_metric_sa(
  dt_metrics = dt_metrics,
  d_name = unique(dt_metrics[["DrugName"]])[1],
  normalization_type = "RV", metric = "x_mean")
feat_data_path <- system.file("depmap_data", package = "gDRtestData")
meta_data_path <- system.file("depmap_data/Model.csv.gz", package = "gDRtestData")
obj_depmap_feat <- prep_dt_depmap_feat(feat_data_path, meta_data_path,
                                       feature_set = "OmicsCNGene")
plot_scatter_with_corr_panel(
  dt_response = dt_response,
  dt_depmap = obj_depmap_feat[["dt_depmap"]],
  selected_feats = names(obj_depmap_feat[["dt_depmap"]])[3:5])
```

---

plot_single_plate_stack_info

Plot a single plate's stack info

Description

Generates the QC and dose-rank heatmap for a single plate subset. Handles all necessary data transformations (factor conversion, ranking, color mapping) internally.

Usage

```
plot_single_plate_stack_info(
  dt_subset,
  plate_id = NULL,
  drug_color_mapping = NULL,
  ctrl_fail_threshold = 0.6,
  n_sd = 1,
  use_sd_threshold = TRUE,
  items_per_line = 6
)
```

Arguments

dt_subset	data.table. Subset of the plate data corresponding to a single barcode/plate.
plate_id	character (Optional). The identifier (barcode) of the plate. If NULL, attempts to extract it from the data.
drug_color_mapping	character (Optional). Named vector of color codes. If NULL, generated automatically.
ctrl_fail_threshold	numeric (Default: 0.6). Flags control wells with a readout below Mean(Controls) * ctrl_fail_threshold.
n_sd	numeric (Default: 1). Sets the upper limit for flagging high signals using the formula Mean(Controls) + (n_sd * SD(Controls)).
use_sd_threshold	logical (Default: TRUE). If TRUE, uses the dynamic SD-based limit defined by n_sd. If FALSE, uses a fixed limit of Mean(Controls) * 1.1.
items_per_line	integer. Number of doses to show per line in the legend. Default 6.

Value

A ggplot object.

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
conc_series <- c(0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10)

test_data <- data.table::data.table(
   WellColumn = rep(1:12, each = 8),
   WellRow = rep(LETTERS[1:8], times = 12),
   clid = "CellLineA",
   Barcode = "Plate_1"
)

invisible(test_data[, Concentration := rep(conc_series, length.out = .N)])
invisible(test_data[, Gnumber := ifelse(Concentration == 0, "vehicle", "Drug_A")])
invisible(test_data[Gnumber == "vehicle", Concentration := 0])

invisible(test_data[, ReadoutValue := ifelse(Gnumber == "vehicle",
```

```
                                    rnorm(.N, 1000, 50),
                                    rnorm(.N, 1000 * (1 / (1 + Concentration)), 50))])

library(ggtext)
plot_single_plate_stack_info(test_data)

invisible(test_data[, Concentration_2 := rep(rev(conc_series), length.out = .N)])
invisible(test_data[, Gnumber_2 := ifelse(Concentration_2 == 0, "vehicle", "Drug_B")])

invisible(test_data[Gnumber_2 == "vehicle", Concentration_2 := 0])

plot_single_plate_stack_info(test_data)
```

`plot_var_distribution_qc` *Plot a violin plot for normalized or averaged data (single-agent and combo) to control the quality of the data*

Description

Plot a violin plot for normalized or averaged data (single-agent and combo) to control the quality of the data

Usage

```
plot_var_distribution_qc(
  dt_assay,
  cl_name,
  metric = "x",
  normalization_type = "GR"
)
```

Arguments

`dt_assay`	data.table representing data from the assay, outputted by `gDRutils::convert_se_assay_to_dt(se, <assay_name>)` for assay_name like `Normalized` or `Averaged` and `SummarizedExperiment` with chosen data type: single-agent or combo
`cl_name`	string cell line name to be plotted (Cell Line Name)
`metric`	string with variable name to be plotted; it has to be in `dt_assay`
`normalization_type`	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")

Value

`ggplot` object containing plot with violin for each drug

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("sa")]]

dt_norm <- gDRutils::convert_se_assay_to_dt(se, "Normalized")
cl_name <- dt_norm[["CellLineName"]][1]

plot_var_distribution_qc(dt_assay = dt_norm,
                         cl_name = cl_name)

dt_average <- gDRutils::convert_se_assay_to_dt(se, "Averaged")
plot_var_distribution_qc(dt_assay = dt_average,
                         cl_name = cl_name,
                         normalization_type = "RV")

plot_var_distribution_qc(dt_assay = dt_average,
                         cl_name = cl_name,
                         metric = "x_std",
                         normalization_type = "RV")

mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]

dt_norm <- gDRutils::convert_se_assay_to_dt(se, "Normalized")
cl_name <- dt_norm[["CellLineName"]][1]

plot_var_distribution_qc(dt_assay = dt_norm,
                         cl_name = cl_name)

dt_average <- gDRutils::convert_se_assay_to_dt(se, "Averaged")
plot_var_distribution_qc(dt_assay = dt_average,
                         cl_name = cl_name,
                         normalization_type = "RV")
```

---

plot_var_stat_qc *Lollipop plot for metric single-agent data to control quality of the data*

Description

Lollipop plot for metric single-agent data to control quality of the data

Usage

```
plot_var_stat_qc(
  dt_assay,
  cl_name,
  metric = "r2",
  normalization_type = "GR",
  with_table = FALSE
)
```

Arguments

dt_assay	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay_to "Metrics") and single-agent SummarizedExperiment
cl_name	string cell line name to be plotted (Cell Line Name)
metric	string with variable name to be plotted; it has to be in dt_assay
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
with_table	logical whether table with metric values should be shown next to the plot

Value

ggplot object containing lollipop plot with stat value for each drug

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
mae <- gDRutils::get_synthetic_data("small")
se <- mae[[gDRutils::get_supported_experiments("sa")]]

dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")
cl_name <- dt_metrics[["CellLineName"]][1]

plot_var_stat_qc(dt_assay = dt_metrics,
                 cl_name = cl_name)

plot_var_stat_qc(dt_assay = dt_metrics,
                 cl_name = cl_name,
                 metric = "r2",
                 normalization_type = "RV")

plot_var_stat_qc(dt_assay = dt_metrics,
                 cl_name = cl_name,
                 metric = "x_AOC",
                 normalization_type = "RV",
                 with_table = TRUE)
```

---

plot_volcano_assoc *Volcano plot with association*

Description

Volcano plot with association

Usage

```
plot_volcano_assoc(
  dt_assoc,
  selected_feat_meta_col,
  selected_metric,
  condition_info = NULL,
  alpha = 0.05,
  named_p_top = 10,
  max_N = NULL
)
```

Arguments

dt_assoc	data.table with the calculated linear association between DepMap and metrics outputted by calc_assoc
selected_feat_meta_col	string describing the name of the associated feature/metadata from DepMap
selected_metric	string describing the name of the selected metric used the association calculation
condition_info	string describing experiment condition (preferred: "DrugName"_"Gnumber"_"drug_moa"_"Duration
alpha	numeric cutoff to identify statistically significant correlations
named_p_top	numeric value for p-top statistically significant correlations to be labeled on the plot
max_N	numeric value for limit the maximum number of non-statistically significant points to plot; for default NULL all points will be plotted.

Value

ggplot object containing a volcano plot with association

Examples

```
Y <- matrix(seq(0.5, 2, length.out = 50), nrow = 50,
            dimnames = list(sprintf("row_%s", 1:50), "met_1"))
X <- matrix(
  withr::with_seed(42, sample(c(NA, seq(0.35, 23.5, 1.25)),
                              size = 50*20, replace = TRUE)),
  nrow = 50,
  dimnames = list(sprintf("row_%s", 1:50), sprintf("feat_%s", 1:20)))
tab_assoc <- calc_assoc(X, Y)
plot_volcano_assoc(tab_assoc,
                   selected_feat_meta_col = "feat_XY",
                   selected_metric = "met_RV")
```

plot_volcano_assoc_panel *Plot panel with volcano plot and according to the data type - scatter plots or box plots*

Description

Plot panel with volcano plot and according to the data type - scatter plots or box plots

Usage

```
plot_volcano_assoc_panel(
  dt_response,
  dt_depmap,
  selected_metric,
  selected_feat_meta_col
)
```

Arguments

dt_response	data.table with the experimental response data (rows are samples) for one metric outputted by one of functions: prep_dt_response_metric_sa, prep_dt_response_dose_sa prep_dt_response_scores or prep_dt_response_metric_diff, must have at least a column with CellLineName and a numeric column with metric values.
dt_depmap	data.table with dependent variables data loaded from DepMap where rows are samples, columns are features/metadata levels; one of: data for one feature outputted by prep_dt_depmap_feat or data or data for one metadata outputted by prep_dt_depmap_meta
selected_metric	string name of the metric in dt_response
selected_feat_meta_col	string with name of selected feature from dt_depmap or the name of the selected metadata from dt_depmap - respectively

Value

A named list with elements:

- assoc_data table with association data
- panel ggplot object containing a panel with volcano plot and depending on data type: a scatter plots with correlation for top 4 variables or boxplots for variable levels

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae_prism <- gDRutils::get_synthetic_data("prism")
se_prism <- mae_prism[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se_prism, "Metrics")
dt_response <- prep_dt_response_metric_sa(
  dt_metrics = dt_metrics,
  d_name = unique(dt_metrics[["DrugName"]])[1],
  normalization_type = "RV", metric = "x_mean")
feat_data_path <- system.file("depmap_data", package = "gDRtestData")
meta_data_path <- system.file("depmap_data/Model.csv.gz", package = "gDRtestData")
obj_depmap_feat <- prep_dt_depmap_feat(feat_data_path, meta_data_path,
                                       feature_set = "OmicsCNGene")
plot_volcano_assoc_panel(
  dt_response = dt_response,
  dt_depmap = obj_depmap_feat[["dt_depmap"]],
  selected_metric = "RV_gDR_x_mean",
  selected_feat_meta_col = obj_depmap_feat[["selected_feat_meta_col"]])
```

---

prep_assoc_summary *Prepare summary table with statistically significant associations*

Description

Prepare summary table with statistically significant associations

Usage

```
prep_assoc_summary(
  dir_path,
  ls_file,
  alpha = 0.05,
  n_stat_sig_row = 10,
  read_file_fun = readxl::read_excel,
  as_list = FALSE
)
```

Arguments

dir_path	A string path to the directory containing files with associations data.
ls_file	A character vector with names of files containing associations data.
alpha	A numeric cutoff to identify statistically significant correlations
n_stat_sig_row	A numeric value specifying the maximum number of statistically significant associations (rows) to include from each file.
read_file_fun	A function to read the data from file; default is `readxl::read_excel`
as_list	A logical flag indicating whether the result should be returned as a list or as a table.

Value

A `DT::datatable` object.

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
dir_path <- system.file("testdata", package = "gDRplots")
ls_file <- list.files(dir_path, pattern = "[.]xlsx$")
prep_assoc_summary(dir_path = dir_path, ls_file = ls_file)
```

---

prep_double_table_chunk

Prepare markdown chunk based on a doubly nested list of tables

---

Description

Generates markdown code for displaying tables in a document using ‘knitr::knit()‘. Handles doubly nested lists, allowing for tabbed sections for cell lines and then metrics. The inner header level (for metrics) is automatically set to one level greater than the outer header level.

Usage

```
prep_double_table_chunk(
  tbl_list,
  chunk_name,
  dwn_list = NULL,
  header_level = 3,
  tabset_options = c("tabset", "tabset-dropdown"),
  sorting_opts = NULL
)
```

Arguments

tbl_list	A doubly nested named list of tables. The outer list represents cell lines, and the inner lists represent metrics. Names are used as headings.
chunk_name	A character string specifying the base name for the generated code chunks. Avoid spaces.
dwn_list	A named list of links to location (relative paths) where table or plots are saved, which when clocked, will be downloaded. It must have the same structure as plt_list.
header_level	An integer specifying the markdown header level to use (e.g., 1 for ‘#‘, 2 for ‘##‘, etc.).
tabset_options	A character vector of options for the tabset. This is only used when plt_list is a nested list. Possible values are "unnumbered", "tabset", and "tabset-dropdown".
sorting_opts	A vector specifying global sorting options for all tables. Column names can be preceded by "-" to indicate descending order.

Value

A list of character vectors. Each element corresponds to a cell line. Each character vector represents markdown code for the cell line’s tabset.

Author(s)

Bartosz Czech <czech.bartosz@external.gene.com>

Examples

```
nested_tables <- list(
  CellLine1 = list(MetricA = mtcars[1:5, ], MetricB = mtcars[6:10, ]),
  CellLine2 = list(MetricC = iris[1:5, ], MetricD = iris[6:10, ])
)
sorting_options <- c("cyl", "-hp")
prep_double_table_chunk(nested_tables, "nested_tables", header_level = 2,
  tabset_options = "tabset", sorting_opts = sorting_options)
```

---

prep_dt_assoc *Prep table with calculated linear associations*

Description

Prep table with calculated linear associations

Usage

```
prep_dt_assoc(dt_response, dt_depmap, selected_feat_meta_col = NULL)
```

Arguments

dt_response	data.table with experimental response data (rows are samples) for one metric
dt_depmap	data.table with dependent variables data load from DepMap. (rows are samples, columns are features or meta); outputted by one of prep_dt_depmap_feat or prep_dt_depmap_meta
selected_feat_meta_col	string name of feature/meta column in DepMap

Value

A named list with elements, that may be input to plot_volcano_assoc

- dt_assoc data.table with calculated association values between feature/meta of DepMap and selected metric,
- condition_info string describing experiment condition (drugs),
- selected_feat_meta_col string name of feature/meta.

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae_prism <- gDRutils::get_synthetic_data("prism")
se_prism <- mae_prism[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se_prism, "Metrics")
dt_response <- prep_dt_response_metric_sa(
  dt_metrics = dt_metrics,
  d_name = unique(dt_metrics[["DrugName"]])[1],
  normalization_type = "RV", metric = "x_mean")
feat_data_path <- system.file("depmap_data", package = "gDRtestData")
meta_data_path <- system.file("depmap_data/Model.csv.gz", package = "gDRtestData")
obj_depmap_feat <- prep_dt_depmap_feat(feat_data_path, meta_data_path,
                                       feature_set = "OmicsCNGene")
prep_dt_assoc(
  dt_response = dt_response,
  dt_depmap = obj_depmap_feat[["dt_depmap"]],
  selected_feat_meta_col = obj_depmap_feat[["selected_feat_meta_col"]])
```

---

prep_dt_depmap_feat *Load DepMap merged data for one selected feature*

Description

Load DepMap merged data for one selected feature

Usage

```
prep_dt_depmap_feat(
  feat_data_path,
  meta_data_path,
  feature_set = "CRISPRGeneEffect",
  with_decoding = FALSE
)
```

Arguments

`feat_data_path`	string with path to the directory containing the molecular feature set file to load from DepMap.
`meta_data_path`	string with path to metadata file describing all cancer models/cell lines which are referenced by a dataset contained within the DepMap portal. It is usually a file named `Model.csv` or `Model.csv.gz`.
`feature_set`	string containing the name of the molecular feature set to load from DepMap. This name should also correspond to the file containing the feature data (without the extension, which is assumed to be `csv` or `csv.gz`)
`with_decoding`	logical whether the feature OmicsArmLevelCNA, OmicsSomaticMutationsMatrixHotspot and OmicsSomaticMutationsMatrixDamaging should be encoded into a 0-1 scheme

Value

A named list with elements, that may be input to prep_dt_assoc

- dt_depmap data.table with feature data from DepMap (wide format),.
- selected_feat_meta_col string name of feature.

Examples

```
feat_data_path <- system.file("depmap_data", package = "gDRtestData")
meta_data_path <- system.file("depmap_data/Model.csv.gz", package = "gDRtestData")
dt_depmap_feat <- prep_dt_depmap_feat(feat_data_path = feat_data_path,
                                      meta_data_path = meta_data_path)
```

---

prep_dt_depmap_meta *Load DepMap merged data for one selected metadata*

Description

Load DepMap merged data for one selected metadata

Usage

```
prep_dt_depmap_meta(meta_data_path, metadata_col = "PatientRace")
```

Arguments

meta_data_path string with path to metadata file describing all cancer models/cell lines

Value

A named list with elements, that may be input to prep_dt_assoc

- dt_depmap data.table with feature data from DepMap (wide format),
- selected_feat_meta_col string name of metadata column.

Examples

```
meta_data_path <- system.file("depmap_data/Model.csv.gz", package = "gDRtestData")
dt_depmap_meta <- prep_dt_depmap_meta(meta_data_path)
```

prep_dt_response_dose_sa *Prep table with metric values by doses for single-agent experiment*

Description

Prep table with metric values by doses for single-agent experiment

Usage

```
prep_dt_response_dose_sa(
  dt_average,
  d_name,
  normalization_type = "RV",
  metric = "x",
  fit_source = "gDR"
)
```

Arguments

dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assa and SummarizedExperiment with chosen data type: single-agent or combo
d_name	string with drug name to be plotted (identifiers DrugName)
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the metric; one of: "x" (value of "GR" or "RV" itself - respectively depending on normalization_type), or "x_std" (standard deviation)
fit_source	string source name for metrics

Value

data.table with selected metric, input to prep_dt_assoc. Note: identifier columns (CellLineName, and DrugName/DrugName_2 where applicable) are always returned as character, regardless of their type in the input.

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix_small")
se <- mae[[gDRutils::get_supported_experiments("sa")]]
dt_average <- gDRutils::convert_se_assay_to_dt(se = se,
                                               assay_name = "Averaged")
d_name <- "drug_004"
dt_response <- prep_dt_response_dose_sa(dt_average, d_name)
```

`prep_dt_response_metric_diff` *Prep table with metric values for combination experiment*

Description

Prep table with metric values for combination experiment

Usage

```
prep_dt_response_metric_diff(
  dt_metrics,
  d_name,
  d_name2,
  cellline1 = NULL,
  cellline2 = NULL,
  normalization_type = "RV",
  metric = "xc50",
  fit_source = "gDR",
  additional_cols = NULL
)
```

Arguments

`dt_metrics`	`data.table` representing data from the `Metrics` assay, outputted by `gDRutils::convert_se_assay` and combo `SummarizedExperiment`
`d_name`	string representing the drug name to be plotted (identifier `DrugName`). If set to NULL, the function will return a table for all available DrugName
`d_name2`	string representing the drug name to be plotted (identifier `DrugName_2`). If set to NULL, the function will return a table for all available DrugName_2
`cellline1`	string representing the first cell line name. If set to NULL, the function will return a table for all available cell lines.
`cellline2`	string representing the second cell line name. If set to NULL, the function will return a table for all available cell lines.
`normalization_type`	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
`metric`	string name of the metric; one of: "xc50" ("GR50" or "IC50" - respectively depending on `normalization_type`), "x_max" ("GR Max" or "E Max"), "x_mean" ("GR Mean" or "RV Mean") "ec50" ("GEC50" or "EC50") or "x_AOC_range" ("GR AOC within set range" or "RV AOC within set range")
`fit_source`	string source name for metrics
`additional_cols`	character vector with additional cols that should be included in the output

Value

`data.table` with selected metric, input to `prep_dt_assoc`. Note: identifier columns (`CellLineName`, and `DrugName`/`DrugName_2` where applicable) are always returned as `character`, regardless of their type in the input.

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix_small")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se = se,
                                               assay_name = "Metrics")
d_name <- "drug_004"
d_name2 <- "drug_026"
dt_response <-
  prep_dt_response_metric_diff(dt_metrics, d_name, d_name2,
  metric = c("xc50", "x_mean", "x_max"))

cellline1 <- "cellline_GB"
cellline2 <- "cellline_HB"

dt_response <-
  prep_dt_response_metric_diff(dt_metrics, d_name = NULL, d_name2 = NULL,
  cellline1, cellline2,
  metric = c("xc50", "x_mean", "x_max"))
```

---

prep_dt_response_metric_sa

Prep table with metric values for single-agent experiment

Description

Prep table with metric values for single-agent experiment

Usage

```
prep_dt_response_metric_sa(
  dt_metrics,
  d_name,
  normalization_type = "RV",
  metric = "xc50",
  fit_source = "gDR"
)
```

Arguments

dt_metrics	data.table representing data from the Metrics assay, outputted by gDRutils::convert_se_assay and single-agent SummarizedExperiment
d_name	string with drug name to be plotted (identifiers DrugName)
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the metric; one of: "xc50" ("GR50" or "IC50" - respectively depending on normalization_type), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean")
fit_source	string source name for metrics

Value

data.table with selected metric, input to prep_dt_assoc. Note: identifier columns (CellLineName, and DrugName/DrugName_2 where applicable) are always returned as character, regardless of their type in the input.

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix_small")
se <- mae[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se = se,
                                               assay_name = "Metrics")
d_name <- "drug_004"
dt_response <- prep_dt_response_metric_sa(dt_metrics, d_name)
dt_response <-
  prep_dt_response_metric_sa(dt_metrics, d_name,
                             metric = c("xc50", "x_mean", "x_max"))

dt_averaged <- gDRutils::convert_se_assay_to_dt(se = se,
                                                assay_name = "Averaged")
dt_metrics_capped <-
  gDRutils::cap_assay_infinities(conc_assay_dt = dt_averaged,
                                 assay_dt = dt_metrics,
                               experiment_name = gDRutils::get_supported_experiments("sa"),
                                 capping_fold = 5)
d_name <- "drug_026"
dt_response <- prep_dt_response_metric_sa(dt_metrics, d_name)
dt_response <-
  prep_dt_response_metric_sa(dt_metrics, d_name,
                             metric = c("xc50", "x_mean", "x_max"))
```

---

prep_dt_response_scores

Prep table with metric values for combination experiment

Description

Prep table with metric values for combination experiment

Usage

```
prep_dt_response_scores(
  dt_scores,
  d_name,
  d_name2,
  normalization_type = "RV",
  metric = "hsa_score",
  fit_source = "gDR"
)
```

Arguments

dt_scores	data.table representing data from the scores assay, outputted by gDRutils::convert_se_assay_ and combo SummarizedExperiment
d_name	string with drug name to be plotted (identifiers DrugName)
d_name2	string with drug name to be plotted (identifiers DrugName_2)
normalization_type	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
metric	string name of the combo metric; one of: "hsa_score"("Bliss Excess GR" or "Bliss Excess RV" - respectively depending on normalization_type), "bliss_score" ("Bliss Score GR" or "Bliss Score RV")
fit_source	string source name for metrics

Value

data.table with selected metric, input to prep_dt_assoc. Note: identifier columns (CellLineName, and DrugName/DrugName_2 where applicable) are always returned as character, regardless of their type in the input.

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix_small")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_scores <- gDRutils::convert_se_assay_to_dt(se = se,
                                              assay_name = "scores")
d_name <- "drug_004"
d_name2 <- "drug_026"
dt_response <- prep_dt_response_scores(dt_scores, d_name, d_name2)
dt_response <-
  prep_dt_response_scores(dt_scores, d_name, d_name2,
                          metric = c("hsa_score", "bliss_score"))
```

---

prep_filename_path *Prepare list of names or paths*

Description

This function can creates lists that can be used as an input to prep_plot_chunk - param link_list and dwn_list. Depending on the inputs, outputted string will have a format: <*path_file*><*prefix*><*name of item on the list*>.<*file_format*> Note: An unnamed item in the list will be numbered.

Usage

```
prep_filename_path(
  plt_list,
  prefix = NULL,
  path_file = NULL,
  file_format = NULL
)
```

Arguments

plt_list	A named list of plots. Names will be used as headings for plots/tab groups. If unnamed, ordinal numbers will be used. Can be nested lists for tabbed output. If a nested list is provided, the inner lists should also be named.
prefix	string to be added as prefix to the name of file
path_file	string path to directory, where data will be read from
file_format	string specifying the format of file

Value

list of strings describing names or paths depending on the input; the list is structured like the input plt_list and retains the same names as plt_list.

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

prep_plot_chunk

Examples

```
# Simple list of plots
plotlist <- lapply(unique(iris$Species), function(iris_name) {
  ggplot2::ggplot(iris[iris$Species == iris_name, c("Sepal.Length", "Sepal.Width")]) +
    ggplot2::geom_point(ggplot2::aes(x = Sepal.Length, y = Sepal.Width))
})

prep_filename_path(plt_list = plotlist,
                   prefix = "iris__",
                   path_file = file.path(".", "plots"))

# Nested list of plots for tabbed output
nested_plotlist <- list()
for (species in unique(iris$Species)) {
  nested_plotlist[[species]] <- list()
  nested_plotlist[[species]][["Sepal"]] <-
    ggplot2::ggplot(iris[iris$Species == species, ],
                ggplot2::aes(x = Sepal.Length, y = Sepal.Width)) + ggplot2::geom_point()
  nested_plotlist[[species]][["Petal"]] <-
    ggplot2::ggplot(iris[iris$Species == species, ],
                ggplot2::aes(x = Petal.Length, y = Petal.Width)) + ggplot2::geom_point()
}

prep_filename_path(plt_list = nested_plotlist,
                   file_format = "png")
```

prep_hm_limits *Calculate limit for combo heatmap with gDR assumptions*

Description

Calculate limit for combo heatmap with gDR assumptions

Usage

```
prep_hm_limits(
  num_vec,
  scale_type = "smooth",
  normalization_type = "GR",
  symmetric = FALSE
)
```

Arguments

num_vec	numeric vector
scale_type	string, "smooth" for response values or "excess" for differences
normalization_type	string with normalization_types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
symmetric	logical indicating if limits should be symmetric around 0

Value

capped limits (min and max) for given numeric vector

Examples

```
vec <- c(-0.1, -0.3, 0, 0.5, Inf, NA)
gDRplots:::prep_hm_limits(vec)
gDRplots:::prep_hm_limits(vec, scale_type = "excess", symmetric = TRUE)
```

prep_nested_plot_chunk *Prepare markdown chunk based on the nested plots list*

Description

Function output should be generated with knitr::knit(text = unlist(<result>))

Usage

```
prep_nested_plot_chunk(
  plt_list,
  chunk_name,
  link_list = NULL,
  dwn_list = NULL,
  header_level = 2
)
```

Arguments

plt_list	named list with generated plots to be shown in tabs; list of plots in nested hierarchy, where last 4th level is plot and 3rd level is normalization_type described by one of: "GR" ("GR Value") or "RV" ("Relative Viability")#'
chunk_name	A character string specifying the base name for the generated code chunks. Avoid spaces.
link_list	A named list of links to the location (relative paths) where plots are saved, which when clicked, will be displayed in a new browser tab. It must have the same structure as plt_list.
dwn_list	A named list of links to location (relative paths) where table or plots are saved, which when clocked, will be downloaded. It must have the same structure as plt_list.
header_level	An integer specifying the markdown header level to use (e.g., 1 for '#', 2 for '##', etc.).

Value

list of character vectors - input for knitr::knit

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

knitr::knit

Examples

```
mae <- gDRutils::get_synthetic_data("small")
se <- mae[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se, "Metrics")

# help function
plot_col <- function(tab_plt, norm_type, col = "red") {
  tab_plt <- data.table::melt(
   data = tab_plt[normalization_type == norm_type][, c("rId", "xc50", "x_mean", "x_max")],
    id = "rId")
  plt <- ggplot2::ggplot(tab_plt, ggplot2::aes(x = variable, y = value)) +
    ggplot2::geom_col(fill = col)
  return(plt)
}
```

```
# creating nested list with plots
plotlist <- list()
ls_color <- c("darkred", "orange", "darkcyan")
for (drug in unique(dt_metrics$DrugName)) {
  for (cl in unique(dt_metrics$CellLineName)) {
    tab_plot <- dt_metrics[DrugName == drug & CellLineName == cl]

    plt_GR <- lapply(ls_color, function(col) plot_col(tab_plot, "RV", col))
    names(plt_GR) <- sprintf("%s_%s", "GR", ls_color)
    plt_RV <- lapply(ls_color, function(col) plot_col(tab_plot, "RV", col))
    names(plt_RV) <- sprintf("%s_%s", "RV", ls_color)

    plotlist[[drug]][[cl]][["RV"]] <- plt_RV
    plotlist[[drug]][[cl]][["GR"]] <- plt_GR
  }
}

prep_nested_plot_chunk(plotlist, "metric_value")

prep_nested_plot_chunk(plotlist, "metric_value")
```

`prep_pheatmap_matrix` *Prep matrix with metric value based on the Metrics assay*

Description

Prep matrix with metric value based on the Metrics assay

Usage

```
prep_pheatmap_matrix(
  dt_response,
  normalization_type = "GR",
  metric = "xc50",
  fit_source = "gDR",
  experiment_type = gDRutils::get_supported_experiments("sa"),
  remove_na_only = TRUE
)
```

Arguments

`dt_response`	`data.table` representing data from the `Metrics` assay, outputted by `gDRutils::convert_se_assay` and single-agent `SummarizedExperiment` or `data.table` representing data from the `scores` assay, outputted by `gDRutils::convert_se_assay_to_dt` and combo `SummarizedExperiment`
`normalization_type`	string with normalization types to be selected one of: "GR" ("GRvalue") or "RV" ("RelativeViability")
`metric`	string name of the metric; one of: "xc50" ("GR50" or "IC50" - respectively depending on `normalization_type`), "x_max" ("GR Max" or "E Max") or "x_mean" ("GR Mean" or "RV Mean"); but the values from any numeric column can be displayed.

fit_source	string source name for metrics
experiment_type	string with experiment name; one of: "single-agent", "combination" or "co-dilution"
remove_na_only	logical flag; if TRUE (default), rows and columns with only missing values are removed from the matrix

Value

matrix with values for selected metric with CellLinName in the rows and DrugName (or combination of DrugName and DrugName_2) in the columns

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("sa")]]
dt_metrics <- gDRutils::convert_se_assay_to_dt(se = se,
                                               assay_name = "Metrics")

mat <- prep_pheatmap_matrix(dt_response = dt_metrics,
                            experiment_type = gDRutils::get_supported_experiments("sa"))

mae <- gDRutils::get_synthetic_data("combo_matrix")
se <- mae[[gDRutils::get_supported_experiments("combo")]]
dt_scores <- gDRutils::convert_se_assay_to_dt(se = se,
                                              assay_name = "scores")

mat <- prep_pheatmap_matrix(dt_response = dt_scores,
                            metric = "hsa_score",
                            normalization_type = "RV",
                           experiment_type = gDRutils::get_supported_experiments("combo"))
```

---

prep_plot_chunk *Prepare markdown chunk based on a list of plots*

Description

Generates markdown code for displaying plots in a document using knitr::knit(). The function handles both simple lists of plots and nested lists, allowing for the creation of tabbed sections for grouped plots.

Usage

```
prep_plot_chunk(
  plt_list,
  chunk_name,
  link_list = NULL,
  dwn_list = NULL,
```

```
  saved_plot_list = NULL,
  header_level = 3,
  tabset_options = c("tabset", "tabset-dropdown")
)
```

Arguments

plt_list	A named list of plots. Names will be used as headings for plots/tab groups. If unnamed, ordinal numbers will be used. Can be nested lists for tabbed output. If a nested list is provided, the inner lists should also be named.
chunk_name	A character string specifying the base name for the generated code chunks. Avoid spaces.
link_list	A named list of links to the location (relative paths) where plots are saved, which when clicked, will be displayed in a new browser tab. It must have the same structure as plt_list.
dwn_list	A named list of links to location (relative paths) where table or plots are saved, which when clocked, will be downloaded. It must have the same structure as plt_list.
saved_plot_list	A character vector or list of absolute paths to pre-saved plot files (e.g. SVG). When provided, plots are embedded from these files instead of re-rendering the ggplot objects, which avoids expensive double-rendering and can speed up report generation. Must have the same structure as plt_list.
header_level	An integer specifying the markdown header level to use (e.g., 1 for '#', 2 for '##', etc.).
tabset_options	A character vector of options for the tabset. This is only used when plt_list is a nested list. Possible values are "unnumbered", "tabset", and "tabset-dropdown".

Value

A list of character vectors. Each element of the list corresponds to a plot or a group of plots (if plt_list is nested). Each character vector within the list represents the markdown code to be processed by knitr::knit(). For nested lists, each element will contain a "header" element and an "items" element. The "header" element is the markdown for the tabset header, and the "items" element is a list of markdown chunks for each tab.

Author(s)

Janina Smoła <janina.smola@external.roche.com>

See Also

knitr::knit

Examples

```
# Simple list of plots
plotlist <- lapply(unique(iris$Species), function(iris_name) {
  ggplot2::ggplot(iris[iris$Species == iris_name, c("Sepal.Length", "Sepal.Width")]) +
  ggplot2::geom_point(ggplot2::aes(x = Sepal.Length, y = Sepal.Width))
})
names(plotlist) <- unique(iris$Species)
```

```
prep_plot_chunk(plotlist, "iris")

# Nested list of plots for tabbed output
nested_plotlist <- list()
for (species in unique(iris$Species)) {
  nested_plotlist[[species]] <- list()
 nested_plotlist[[species]][["Sepal"]] <- ggplot2::ggplot(iris[iris$Species == species, ],
    ggplot2::aes(x = Sepal.Length, y = Sepal.Width)) + ggplot2::geom_point()
 nested_plotlist[[species]][["Petal"]] <- ggplot2::ggplot(iris[iris$Species == species, ],
    ggplot2::aes(x = Petal.Length, y = Petal.Width)) + ggplot2::geom_point()
}

prep_plot_chunk(nested_plotlist, "iris_nested", tabset_options = c("tabset", "unnumbered"))
```

`round_to_unique_string` *Round numbers to unique string*

Description

Rounds a numeric vector to the minimum precision needed to ensure uniqueness

Usage

```
round_to_unique_string(num_vec, initial_digits = 4)
```

Arguments

`num_vec`	a numeric vector to be rounded and change into character
`initial_digits`	numeric value for number of decimal places to start rounding with

Value

a character vector of unique numeric strings.

Author(s)

Janina Smoła <janina.smola@external.roche.com>

Examples

```
vec <- c(0.00000000, 0.00000256, 0.00001280, 0.00006400, 0.00032000,
         0.00160000, 0.00800000, 0.04000000, 0.20000000, 1.00000000)

round_to_unique_string(vec)
```

save_plot *Save gDR plots to a specified path*

Description

Save gDR plots to a specified path

Usage

```
save_plot(plt, path, format = "svg")
```

Arguments

plt	A plot object; either a ggplot2 or pheatmap object.
path	A string specifying the path where the plot should be saved.
format	A string specifying the format for saving the plot; either "svg", "png", or "pdf". Default is "svg".

Value

NULL

See Also

ggplot2::ggsave

Examples

```
tmp_dir <- file.path(tempdir(), "plot_dir")
dir.create(tmp_dir, showWarnings = FALSE)
p <- ggplot2::ggplot(mtcars, ggplot2::aes(mpg, wt)) + ggplot2::geom_point()
save_plot(plt = p, path = paste(tmp_dir, "mtcars_scatter", sep = "/"), format = "png")
```

transform_log_conc *Transform concentrations values with log10*

Description

Transform concentrations values with log10

Usage

```
transform_log_conc(conc_vec)
```

Arguments

conc_vec	numeric vector with concentration values

Value

numeric vector with log10 concentration values; log10 for concentration equal 0 (-Inf) is replaced with one step less in the dose dilution. When all concentrations are 0, all positions are 0.

Examples

```
vec <- c(0, 0.003, 0.01, 0.03)
gDRplots:::transform_log_conc(vec)
vec <- c(0, 0, 0)
gDRplots:::transform_log_conc(vec)
```

Index

∗ **QC_plot**
heatmap_control_mapping_qc, 63
pheatmap_qc, 66
plot_dose_response_sa_qc, 105
plot_dose_response_sa_qc_panel, 106
plot_fitting_acc, 107
plot_plate, 108
plot_plate_stack_info, 109
plot_single_plate_stack_info, 112
plot_var_distribution_qc, 114
plot_var_stat_qc, 115
∗ **cgs_plots**
analyze_cgs, 31
analyze_cgs_conditions, 33
plot_cgs_ranking, 95
∗ **combo_plots**
heatmap_combo_metrics, 51
heatmap_combo_metrics_panel, 53
heatmap_combo_with_isoref, 56
heatmap_combo_with_isoref_panel, 58
heatmap_combo_with_isoref_panel_common, 59
heatmap_combo_with_isoref_panel_independent, 62
plot_boxplot_metric_combo, 80
plot_boxplot_metric_combo_by_CLs, 81
plot_boxplot_metric_combo_by_drugs, 83
plot_boxplot_metric_combo_by_grp, 84
plot_combination_index, 96
plot_dose_response_combo, 98
plot_dose_response_combo_panel, 99
∗ **internal**
.calc_dose_rank, 6
.cgs_diff_by_cotrt, 6
.cgs_fgsea, 7
.create_PRISM_plot_list, 7
.describe_fn_sentence, 9
.describe_gr, 9
.describe_import, 10
.describe_masked, 10
.describe_plates_without_barcode, 11
.describe_provenance_step, 11
.describe_settings, 12
.fill_pheatmap_annotation, 12
.format_dose_list, 13
.get_combo_curves_colors, 13
.get_combo_panel_type, 14
.get_combo_score_breaks, 15
.get_data_type, 16
.get_excess_palette, 17
.get_gradient_colors, 17
.get_info_from_name, 18
.get_info_from_names, 19
.get_iso_colors, 19
.get_n_top_asssoc, 20
.get_pheatmap_cluster_param, 20
.get_pheatmap_fontsize, 22
.get_pheatmap_number_color, 22
.get_scale_type, 23
.get_smooth_palette, 24
.get_tile_size, 24
.lin_associations, 25
.order_assoc_result, 26
.prep_dt_OmicsArmLevelCNA, 26
.prep_dt_OmicsSomaticMutationsMatrix, 27
.resolve_group_colors, 28
.safe_as_numeric, 29
.stop_on_aggregation, 29
.table_to_ggplot, 30
.theme_bw, 30
.trim_labels, 31
assert_RGB_format, 34
calc_assoc, 34
change_NA_into_char, 35
create_download_link, 37
create_zoom_link, 42
escape_special_characters, 43
estimate_plot_size, 44
gDRplots-package, 4

generate_color_mappings, 45
generate_datatable, 46
get_provenance_tables, 49
get_r_file_path, 51
neutralize_spaces, 66
prep_assoc_summary, 119
prep_double_table_chunk, 120
prep_dt_depmap_feat, 122
prep_dt_depmap_meta, 123
prep_filename_path, 128
prep_hm_limits, 130
prep_nested_plot_chunk, 130
prep_plot_chunk, 133
save_plot, 136
transform_log_conc, 136
∗ **pheat_ann**
get_hm_title, 48
pheatmap_with_anno_cd, 68
pheatmap_with_anno_combo, 70
pheatmap_with_anno_combo_metrics, 73
pheatmap_with_anno_sa, 75
prep_pheatmap_matrix, 132
∗ **prism_plots**
.create_PRISM_plot_list, 7
.get_data_type, 16
.get_n_top_asssoc, 20
create_PRISM_plot_list_combo, 38
create_PRISM_plot_list_sa, 40
create_PRISM_summary_list, 42
plot_boxplot_meta, 78
plot_boxplot_num, 93
plot_boxplot_num_panel, 94
plot_scatter_with_corr, 110
plot_scatter_with_corr_panel, 111
plot_volcano_assoc, 116
plot_volcano_assoc_panel, 118
prep_dt_assoc, 121
prep_dt_response_dose_sa, 124
prep_dt_response_metric_diff, 125
prep_dt_response_metric_sa, 126
prep_dt_response_scores, 127
∗ **single-agent_plots**
plot_boxplot_metric_sa, 86
plot_boxplot_metric_sa_by_CLs, 88
plot_boxplot_metric_sa_by_drugs, 89
plot_boxplot_metric_sa_by_grp, 91
plot_dose_response_sa, 100
plot_dose_response_sa_by_CLs, 102
plot_dose_response_sa_by_drugs, 104
∗ **utils_color**
fill_ann_color_map, 44
get_ann_color_map, 47
get_col_luminance, 47
get_iso_colors, 49
get_qual_colors, 50
is_color_dark, 64
is_valid_color, 65
∗ **utils**
compute_distances, 36
create_log_seq, 38
round_to_unique_string, 135
.calc_dose_rank, 6
.cgs_diff_by_cotrt, 6
.cgs_fgsea, 7
.create_PRISM_plot_list, 7
.describe_fn_sentence, 9
.describe_gr, 9
.describe_import, 10
.describe_masked, 10
.describe_plates_without_barcode, 11
.describe_provenance_step, 11
.describe_settings, 12
.fill_pheatmap_annotation, 12
.format_dose_list, 13
.get_combo_curves_colors, 13
.get_combo_panel_type, 14
.get_combo_score_breaks, 15
.get_data_type, 16
.get_excess_palette, 17
.get_gradient_colors, 17
.get_info_from_name, 18
.get_info_from_names, 19
.get_iso_colors, 19
.get_n_top_asssoc, 20
.get_pheatmap_cluster_param, 20
.get_pheatmap_fontsize, 22
.get_pheatmap_number_color, 22
.get_scale_type, 23
.get_smooth_palette, 24
.get_tile_size, 24
.lin_associations, 25
.order_assoc_result, 26
.prep_dt_OmicsArmLevelCNA, 26
.prep_dt_OmicsSomaticMutationsMatrix, 27
.resolve_group_colors, 28
.safe_as_numeric, 29
.stop_on_aggregation, 29
.table_to_ggplot, 30
.theme_bw, 30
.trim_labels, 31

analyze_cgs, 31, *33*
analyze_cgs_conditions, 33
ashr::ash, *25*
assert_RGB_format, 34

calc_assoc, 34
change_NA_into_char, 35
compute_distances, *21*, 36, *67*, *69*, *71*, *74*, *76*
create_download_link, 37
create_log_seq, 38
create_PRISM_plot_list_combo, 38
create_PRISM_plot_list_sa, 40
create_PRISM_summary_list, 42
create_zoom_link, 42

escape_special_characters, 43
estimate_plot_size, 44

fill_ann_color_map, 44

gDRplots *(*gDRplots-package*)*, 4
gDRplots-package, 4
gDRutils::cap_assay_infinities, *74*, *76*
gDRutils::convert_se_assay_to_dt, *67*, *68*, *71*, *74*, *76*, *124–126*, *128*, *132*
generate_color_mappings, 45
generate_datatable, 46
get_ann_color_map, 47
get_col_luminance, 47
get_hm_title, 48
get_iso_colors, 49
get_provenance_tables, 49
get_qual_colors, *28*, 50, *101*, *103*, *104*
get_r_file_path, 51
get_SE_provenance, *49*
ggplot2::ggsave, *44*, *136*
grDevices::colors, *65*

heatmap_combo_metrics, 51
heatmap_combo_metrics_panel, 53
heatmap_combo_with_isoref, 56
heatmap_combo_with_isoref_panel, 58
heatmap_combo_with_isoref_panel_common, *14*, *15*, 59
heatmap_combo_with_isoref_panel_independent, *14*, *15*, 62
heatmap_control_mapping_qc, 63

is_color_dark, 64
is_valid_color, 65

knitr::knit, *37*, *43*, *131*, *134*

neutralize_spaces, 66

pheatmap, *72*
pheatmap::pheatmap, *67*, *69*, *77*
pheatmap_qc, *47*, 66
pheatmap_with_anno_cd, 68
pheatmap_with_anno_combo, *45*, 70
pheatmap_with_anno_combo_metrics, 73
pheatmap_with_anno_sa, *45*, 75
plot_boxplot_meta, 78
plot_boxplot_metric_combo, 80
plot_boxplot_metric_combo_by_CLs, 81
plot_boxplot_metric_combo_by_drugs, 83
plot_boxplot_metric_combo_by_grp, 84
plot_boxplot_metric_sa, 86
plot_boxplot_metric_sa_by_CLs, 88
plot_boxplot_metric_sa_by_drugs, 89
plot_boxplot_metric_sa_by_grp, 91
plot_boxplot_num, 93
plot_boxplot_num_panel, 94
plot_cgs_ranking, *33*, 95
plot_combination_index, 96
plot_dose_response_combo, 98
plot_dose_response_combo_panel, 99
plot_dose_response_sa, 100
plot_dose_response_sa_by_CLs, 102
plot_dose_response_sa_by_drugs, 104
plot_dose_response_sa_qc, 105
plot_dose_response_sa_qc_panel, 106
plot_fitting_acc, 107
plot_plate, 108
plot_plate_stack_info, 109
plot_scatter_with_corr, 110
plot_scatter_with_corr_panel, 111
plot_single_plate_stack_info, 112
plot_var_distribution_qc, 114
plot_var_stat_qc, 115
plot_volcano_assoc, 116, *121*
plot_volcano_assoc_panel, 118
prep_assoc_summary, 119
prep_double_table_chunk, 120
prep_dt_assoc, 121, *123–125*, *127*, *128*
prep_dt_depmap_feat, *26*, *27*, *93*, *94*, *111*, *112*, *118*, *121*, 122
prep_dt_depmap_meta, *79*, *118*, *121*, 123
prep_dt_response_dose_sa, *79*, *93*, *94*, *110*, *112*, *118*, 124
prep_dt_response_metric_diff, *32*, *79*, *93*, *94*, *110*, *112*, *118*, 125
prep_dt_response_metric_sa, *79*, *93*, *94*, *110*, *112*, *118*, 126
prep_dt_response_scores, *79*, *93*, *94*, *110*, *112*, *118*, 127
prep_filename_path, 128

prep_hm_limits, 130
prep_nested_plot_chunk, 130
prep_pheatmap_matrix, 132
prep_plot_chunk, *128*, *129*, 133

round_to_unique_string, 135

save_plot, 136
stats::dist, *21*

transform_log_conc, 136

WGCNA::cor, *25*