Package: gDRplots (via r-universe)

November 24, 2026

Title 'gDRplots'

Version 1.1.1

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

Config/pak/sysreqs cmake make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libnode-dev zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-11-24 10:09:14 UTC

RemoteUrl https://github.com/bioc/gDRplots

RemoteRef HEAD

RemoteSha 248018c6c1432ed9db807ed27d805dc2d840a080

Contents

analyze_cgs . . . 3
analyze_cgs_conditions . . . 5
compute_distances . . . 6
create_log_seq . . . 7
create_PRISM_plot_list_combo . . . 8
create_PRISM_plot_list_sa . . . 10
create_PRISM_summary_list . . . 11
fill_ann_color_map . . . 12
get_ann_color_map . . . 13
get_col_luminance . . . 14
get_hm_title . . . 15
get_iso_colors . . . 15
get_qual_colors . . . 16
heatmap_combo_metrics . . . 17
heatmap_combo_metrics_panel . . . 19
heatmap_combo_with_isoref . . . 21
heatmap_combo_with_isoref_panel . . . 24
heatmap_combo_with_isoref_panel_common . . . 25
heatmap_combo_with_isoref_panel_independent . . . 28
heatmap_control_mapping_qc . . . 30
is_color_dark . . . 31
is_valid_color . . . 32
pheatmap_qc . . . 32
pheatmap_with_anno_cd . . . 34
pheatmap_with_anno_combo . . . 37
pheatmap_with_anno_combo_metrics . . . 40
pheatmap_with_anno_sa . . . 42
plot_boxplot_meta . . . 46
plot_boxplot_metric_combo . . . 47
plot_boxplot_metric_combo_by_CLs . . . 49
plot_boxplot_metric_combo_by_drugs . . . 50
plot_boxplot_metric_combo_by_grp . . . 52
plot_boxplot_metric_sa . . . 54

plot_boxplot_metric_sa_by_CLs . . . 56
plot_boxplot_metric_sa_by_drugs . . . 57
plot_boxplot_metric_sa_by_grp . . . 59
plot_boxplot_num . . . 62
plot_boxplot_num_panel . . . 63
plot_cgs_ranking . . . 64
plot_combination_index . . . 65
plot_dose_response_combo . . . 67
plot_dose_response_combo_panel . . . 68
plot_dose_response_sa . . . 69
plot_dose_response_sa_by_CLs . . . 72
plot_dose_response_sa_by_drugs . . . 73
plot_dose_response_sa_qc . . . 75
plot_dose_response_sa_qc_panel . . . 76
plot_fitting_acc . . . 77
plot_plate . . . 78
plot_plate_stack_info . . . 79
plot_scatter_with_corr . . . 80
plot_scatter_with_corr_panel . . . 81
plot_single_plate_stack_info . . . 83
plot_var_distribution_qc . . . 84
plot_var_stat_qc . . . 86
plot_volcano_assoc . . . 87
plot_volcano_assoc_panel . . . 88
prep_dt_assoc . . . 89
prep_dt_response_dose_sa . . . 90
prep_dt_response_metric_diff . . . 91
prep_dt_response_metric_sa . . . 93
prep_dt_response_scores . . . 94
prep_pheatmap_matrix . . . 95
round_to_unique_string . . . 97

Index **98**

`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))
```

`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_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_to_` `"Metrics"`) and combo `SummarizedExperiment`
`dt_scores`	`data.table` representing data from the `scores` assay, outputted by `gDRutils::convert_se_assay_to_c` `"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_to_ "Metrics") and single-agent SummarizedExperiment
dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assay_to "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"]]), ]
)
```

---

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)
```

---

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_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)
```

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_dt("excess") and combo SummarizedExperiment
dt_isobolograms	data.table representing data from the isobolograms assay, outputted by gDRutils::convert_se_assay_ "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_dt("excess") and combo SummarizedExperiment
dt_isobolograms	data.table representing data from the isobolograms assay, outputted by gDRutils::convert_se_assay_ "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_dt("excess") and combo SummarizedExperiment
dt_isobolograms	data.table representing data from the isobolograms assay, outputted by gDRutils::convert_se_assay_ "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_dt( "excess")` and combo `SummarizedExperiment`
`dt_isobolograms`	data.table representing data from the `isobolograms` assay, outputted by `gDRutils::convert_se_assay_ "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_dt("excess") and combo SummarizedExperiment
dt_isobolograms	data.table representing data from the isobolograms assay, outputted by gDRutils::convert_se_assay_ "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_dt( "excess")` and combo `SummarizedExperiment`
`dt_isobolograms`	data.table representing data from the `isobolograms` assay, outputted by `gDRutils::convert_se_assay_ "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_assay_t "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_to_ "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")
```

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_assay_to 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_to_ 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_to_c 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_to_ 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_assay_t 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_to_d "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_to_d "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_to_d "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_to_d "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_dt "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_dt "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_dt "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_dt "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_dt("excess") and combo SummarizedExperiment
dt_isobolograms	data.table representing data from the isobolograms assay, outputted by gDRutils::convert_se_assay_ "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_to_d "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_to_d "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_dt "Metrics") and single-agent SummarizedExperiment
dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assay_to_d "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_dt "Metrics")` and single-agent `SummarizedExperiment`
`dt_average`	data.table representing data from the `Averaged` assay, outputted by `gDRutils::convert_se_assay_to_d "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_dt "Metrics") and single-agent SummarizedExperiment
dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assay_to_d "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_dt "Metrics")` and single-agent `SummarizedExperiment`
`dt_average`	data.table representing data from the `Averaged` assay, outputted by `gDRutils::convert_se_assay_to_d "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_dt "Metrics") and single-agent SummarizedExperiment
dt_average	data.table representing data from the Averaged assay, outputted by gDRutils::convert_se_assay_to_d "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_dt "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_dt "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_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_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_assay_to 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_to_ 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_to_` 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_to_d 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_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_to_ 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"))
```

`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)
```

Index

∗ **QC_plot**
heatmap_control_mapping_qc, 30
pheatmap_qc, 32
plot_dose_response_sa_qc, 75
plot_dose_response_sa_qc_panel, 76
plot_fitting_acc, 77
plot_plate, 78
plot_plate_stack_info, 79
plot_single_plate_stack_info, 83
plot_var_distribution_qc, 84
plot_var_stat_qc, 86
∗ **cgs_plots**
analyze_cgs, 3
analyze_cgs_conditions, 5
plot_cgs_ranking, 64
∗ **combo_plots**
heatmap_combo_metrics, 17
heatmap_combo_metrics_panel, 19
heatmap_combo_with_isoref, 21
heatmap_combo_with_isoref_panel, 24
heatmap_combo_with_isoref_panel_common, 25
heatmap_combo_with_isoref_panel_independent, 28
plot_boxplot_metric_combo, 47
plot_boxplot_metric_combo_by_CLs, 49
plot_boxplot_metric_combo_by_drugs, 50
plot_boxplot_metric_combo_by_grp, 52
plot_combination_index, 65
plot_dose_response_combo, 67
plot_dose_response_combo_panel, 68
∗ **pheat_ann**
get_hm_title, 15
pheatmap_with_anno_cd, 34
pheatmap_with_anno_combo, 37
pheatmap_with_anno_combo_metrics, 40
pheatmap_with_anno_sa, 42
prep_pheatmap_matrix, 95
∗ **prism_plots**
create_PRISM_plot_list_combo, 8
create_PRISM_plot_list_sa, 10
create_PRISM_summary_list, 11
plot_boxplot_meta, 46
plot_boxplot_num, 62
plot_boxplot_num_panel, 63
plot_scatter_with_corr, 80
plot_scatter_with_corr_panel, 81
plot_volcano_assoc, 87
plot_volcano_assoc_panel, 88
prep_dt_assoc, 89
prep_dt_response_dose_sa, 90
prep_dt_response_metric_diff, 91
prep_dt_response_metric_sa, 93
prep_dt_response_scores, 94
∗ **single-agent_plots**
plot_boxplot_metric_sa, 54
plot_boxplot_metric_sa_by_CLs, 56
plot_boxplot_metric_sa_by_drugs, 57
plot_boxplot_metric_sa_by_grp, 59
plot_dose_response_sa, 69
plot_dose_response_sa_by_CLs, 72
plot_dose_response_sa_by_drugs, 73
∗ **utils_color**
fill_ann_color_map, 12
get_ann_color_map, 13
get_col_luminance, 14
get_iso_colors, 15
get_qual_colors, 16
is_color_dark, 31
is_valid_color, 32
∗ **utils**
compute_distances, 6

create_log_seq, 7
round_to_unique_string, 97

analyze_cgs, 3, *5*
analyze_cgs_conditions, 5

compute_distances, 6, *33*, *35*, *38*, *41*, *43*
create_log_seq, 7
create_PRISM_plot_list_combo, 8
create_PRISM_plot_list_sa, 10
create_PRISM_summary_list, 11

fill_ann_color_map, 12

gDRutils::cap_assay_infinities, *40*, *43*
gDRutils::convert_se_assay_to_dt, *33*, *35*, *37*, *40*, *43*, *91–93*, *95, 96*
get_ann_color_map, 13
get_col_luminance, 14
get_hm_title, 15
get_iso_colors, 15
get_qual_colors, 16, *70*, *73, 74*
grDevices::colors, *32*

heatmap_combo_metrics, 17
heatmap_combo_metrics_panel, 19
heatmap_combo_with_isoref, 21
heatmap_combo_with_isoref_panel, 24
heatmap_combo_with_isoref_panel_common, 25
heatmap_combo_with_isoref_panel_independent, 28
heatmap_control_mapping_qc, 30

is_color_dark, 31
is_valid_color, 32

pheatmap, *39*
pheatmap::pheatmap, *33*, *36*, *44*
pheatmap_qc, *14*, 32
pheatmap_with_anno_cd, 34
pheatmap_with_anno_combo, *13*, 37
pheatmap_with_anno_combo_metrics, 40
pheatmap_with_anno_sa, *13*, 42
plot_boxplot_meta, 46
plot_boxplot_metric_combo, 47
plot_boxplot_metric_combo_by_CLs, 49
plot_boxplot_metric_combo_by_drugs, 50
plot_boxplot_metric_combo_by_grp, 52
plot_boxplot_metric_sa, 54
plot_boxplot_metric_sa_by_CLs, 56
plot_boxplot_metric_sa_by_drugs, 57
plot_boxplot_metric_sa_by_grp, 59
plot_boxplot_num, 62
plot_boxplot_num_panel, 63
plot_cgs_ranking, *5*, 64
plot_combination_index, 65
plot_dose_response_combo, 67
plot_dose_response_combo_panel, 68
plot_dose_response_sa, 69
plot_dose_response_sa_by_CLs, 72
plot_dose_response_sa_by_drugs, 73
plot_dose_response_sa_qc, 75
plot_dose_response_sa_qc_panel, 76
plot_fitting_acc, 77
plot_plate, 78
plot_plate_stack_info, 79
plot_scatter_with_corr, 80
plot_scatter_with_corr_panel, 81
plot_single_plate_stack_info, 83
plot_var_distribution_qc, 84
plot_var_stat_qc, 86
plot_volcano_assoc, 87, *90*
plot_volcano_assoc_panel, 88
prep_dt_assoc, 89, *91, 92*, *94, 95*
prep_dt_depmap_feat, *62, 63*, *81, 82*, *88*, *90*
prep_dt_depmap_meta, *46*, *88*, *90*
prep_dt_response_dose_sa, *46*, *62, 63*, *81, 82*, *88*, 90
prep_dt_response_metric_diff, *4*, *46*, *62, 63*, *81, 82*, *88*, 91
prep_dt_response_metric_sa, *46*, *62, 63*, *81, 82*, *88*, 93
prep_dt_response_scores, *46*, *62, 63*, *81, 82*, *88*, 94
prep_pheatmap_matrix, 95

round_to_unique_string, 97