Package: ggoncoplot (via r-universe)

May 11, 2026

Type Package

Title Easily Create Interactive Oncoplots

Version 0.1.0

Description Generate oncoplots from tabular mutational data. Optionally make these oncoplots interactive, with a fully customisable tooltip.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports assertions (>= 0.0.0.9000), cli, dplyr, forcats, ggEDA (>= 0.2.0.9000), ggiraph, ggplot2, grid, mutationtypes, patchwork (>= 1.3.0), rank (>= 0.2.0), RColorBrewer, rlang, scales

URL `https://github.com/selkamand/ggoncoplot`, `https://selkamand.github.io/ggoncoplot/`

BugReports `https://github.com/selkamand/ggoncoplot/issues`

Suggests covr, express, knitr, rmarkdown, somaticflags, testthat (>= 3.0.0), vdiffr

VignetteBuilder knitr

Config/testthat/edition 3

Remotes selkamand/mutationtypes, selkamand/express, CCICB/ggEDA

Roxygen list(markdown = TRUE)

Depends R (>= 3.5)

Config/Needs/website rmarkdown

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev

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

Date/Publication 2026-03-30 20:53:18 UTC

RemoteUrl https://github.com/selkamand/ggoncoplot

RemoteRef HEAD

RemoteSha 92f3d87c812db8c1b07386edda07fbd396d8ca02

Contents

`check_valid_dataframe_column` *data.frame has colnames*

Description

Assert that data.frame contains a set of user defined column names.

Usage

```
check_valid_dataframe_column(data, colnames, error_call = rlang::caller_env())
```

Arguments

`data`	dataframe that you want to assert contain specific columns (data.frame)
`colnames`	Name (character)
`error_call`	error call environment (do not change)

Details

data.frame may have any additional colnames. It just has to have AT LEAST the columns specified by `colnames`

Informs user about the missing columns one at a time. This may change in future

Value

invisibly returns TRUE. If data is missing columns, will throw error

Examples

```
# Check mtcars has columns 'mpg' and 'cyl'
ggoncoplot:::check_valid_dataframe_column(mtcars, c("mpg", "cyl"))
```

`combine_plots` *Combine margin plots with main plot*

Description

Combine margin plots with main plot

Usage

```
combine_plots(
  gg_main,
  gg_tmb = NULL,
  gg_gene = NULL,
  gg_metadata = NULL,
  gg_tmb_height,
  gg_gene_width,
  gg_metadata_height,
  metadata_position,
  buffer_metadata,
  buffer_tmb,
  buffer_genebar
)
```

Arguments

`gg_main`	main oncoplot tileplot (ggplot)
`gg_tmb`	barplot describing total mutations. Set to NULL to not draw barplot (ggplot)
`gg_gene`	barplot describing number of mutated samples per gene. Set to NULL to not draw barplot (ggplot)
`gg_metadata`	tile plot describing sample-level metadata
`gg_tmb_height`	percentage of plot height taken up by TMB plot (should be between 5-95) (number)
`gg_gene_width`	percentage of plot width taken up by genebar plot (should be between 5-95) (number)
`gg_metadata_height`	percentage of plot height taken up by metadata plot (should be between 5-95) (number)

`metadata_position`	should metadata plot be on the 'top' or the 'bottom' of the oncoplot?
`buffer_metadata`, `buffer_tmb`	amount of space to add between the main oncoplot and tmb/metadata marginal plots (number)
`buffer_genebar`	amount of space to add between the main oncoplot and tmb/metadata marginal plots (number)

Value

patchwork object (or ggplot obj if both `gg_tmb` and `gg_gene` are NULL)

`get_nonmutated_tiles` *Get data.frame o*

Description

Takes same data input as ggoncoplot and returns a dataframe with 'Sample' and 'Gene' columns ONLY for sample-gene pairs that are unmutated. This lets us colour render them separately (as grey)

Usage

```
get_nonmutated_tiles(data)
```

Arguments

`data`	transformed data from `ggoncoplot_prep_df()` (data.frame)

Value

a dataframe with 'Sample' and 'Gene' columns ONLY for sample-gene pairs that are unmutated. This lets us colour render them separately (as grey) (data.frame)

`ggoncoplot` *ggoncoplot*

Description

Creates an interactive oncoplot to visualize the mutation landscape of cancer cohorts.

Usage

```
ggoncoplot(
  data,
  col_genes,
  col_samples,
  col_mutation_type = NULL,
  genes_to_include = NULL,
  genes_to_ignore = NULL,
  col_tooltip = col_samples,
  topn = 10,
  return_extra_genes_if_tied = FALSE,
  draw_gene_barplot = FALSE,
  draw_tmb_barplot = FALSE,
  copy = c("sample", "gene", "tooltip", "mutation_type", "nothing"),
  palette = NULL,
  metadata = NULL,
  metadata_palette = NULL,
  col_samples_metadata = col_samples,
  cols_to_plot_metadata = NULL,
  metadata_require_mutations = TRUE,
  pathway = NULL,
  col_genes_pathway = col_genes,
  show_all_samples = FALSE,
  total_samples = c("any_mutations", "all", "oncoplot"),
  sample_order = NULL,
  metadata_sort_cols = NULL,
  metadata_sort_desc = TRUE,
  metadata_sort_by = "frequency",
  tmb_data = NULL,
  tmb_palette = NULL,
  interactive = TRUE,
  options = ggoncoplot_options(),
  verbose = TRUE
)
```

Arguments

`data`	data for oncoplot. A data.frame with 1 row per mutation in your cohort. Must contain columns describing gene_symbols and sample_identifiers (data.frame)
`col_genes`	name of **data** column containing gene names/symbols (string)
`col_samples`	name of **data** column containing sample identifiers (string)
`col_mutation_type`	name of **data** column describing mutation types (string, optional)
`genes_to_include`	specific genes to include in the oncoplot (character, optional)
`genes_to_ignore`	names of the genes that should be ignored (character, optional)

`col_tooltip`	name of **data** column containing whatever information you want to display in (string, defaults to col_samples)
`topn`	how many of the top genes to visualize. Ignored if `genes_to_include` is supplied (number, default 10)
`return_extra_genes_if_tied`	instead of strictly returning `topn` genes, in the case of ties (where multiple genes are mutated in the exact same number of samples, complicating selection of top n genes), return all tied genes (potentially more than topn). If FALSE, will return strictly `topn` genes, breaking ties based on order of appearance in dataset (flag, default FALSE)
`draw_gene_barplot`	add a barplot describing number of samples with each gene mutated (right side) (flag, default FALSE)
`draw_tmb_barplot`	add a barplot describing total number of mutations in each sample (above main plot). If a single gene is mutated multiple times, all mutations are counted towards total (flag, default FALSE)
`copy`	value to copy to clipboard when an oncoplot tile is clicked (string, one of 'sample', 'gene', 'tooltip', 'mutation_type', 'nothing', default 'sample')
`palette`	a named vector mapping all possible mutation types (vector names) to colors (vector values, optional)
`metadata`	dataframe describing sample level metadata. One column must contain unique sample identifiers. Other columns can describe numeric / categorical metadata (data.frame, optional)
`metadata_palette`	A list of named vectors. List names correspond to metadata column names (categorical only). Vector names to levels of columns. Vector values are colors, the vector names are used to map values in data to a color. (optional)
`col_samples_metadata`	which column in metadata data.frame describes sample identifiers (string, defaults to col_samples)
`cols_to_plot_metadata`	names of columns in metadata that should be plotted (character, optional)
`metadata_require_mutations`	filter out samples from metadata lacking any mutations in data (flag, default TRUE)
`pathway`	a two column dataframe describing pathway. The column containing gene names should have the same name as **col_gene** (data.frame, optional)
`col_genes_pathway`	which column in pathway data.frame describes gene names (string, defaults to col_genes)
`show_all_samples`	show all samples in oncoplot, even if they don't have mutations in the selected genes. Samples only described in metadata but with no mutations at all are still filtered out by default, but you can show these too by setting `metadata_require_mutations = FALSE` (flag, default FALSE)

`total_samples`	Strategy for calculating the total number of samples. This value is used to compute the proportion of mutation recurrence displayed in the tooltip when hovering over the gene barplot, or as a text annotation when `ggoncoplot_options(show_genebar_labels = TRUE)` is set to TRUE. Possible values: • **any_mutations**: All the samples that are in `data` (the mutation dataset), irrespective of whether they are on the oncoplot or not. • **oncoplot**: Only the samples that are present on the oncoplot. • **all**: All the samples in either `data` or `metadata`.
`sample_order`	sample IDs in the order they should be shown on oncoplot (left to right). Overrides gene-based auto-ranking. (character vector).
`metadata_sort_cols`	A character vector of metadata columns to sort on. If `NULL` will default to typical gene-based sort unless `sample_order` is specified.
`metadata_sort_desc`	Logical scalar or vector indicating whether to rank each column in descending order. If a single value is supplied it is recycled for all columns.
`metadata_sort_by`	Character vector specifying how to rank each non-numeric column. Valid values include "alphabetical" or "frequency". If a single value is supplied it is recycled for all columns. For numeric columns, sort_by is ignored and ranking is always based on numeric order.
`tmb_data`	Optional custom TMB dataset. A data.frame with 2–3 columns including `col_samples`. Column mapping is inferred as follows: 1. sample column: the column named by `col_samples` 2. TMB column: the first numeric non-sample column 3. subtype column (optional): if a third column is present, it is treated as a stacking/colouring subtype No missing values are permitted. Note: stacked bars are disabled when `log10_transform_tmb = TRUE` (totals are shown).
`tmb_palette`	a named vector mapping all possible tmb sub types (vector names) to colors (vector values). If `tmb_palette` and `tmb_data` are NULL, will be set to match `palette`.
`interactive`	should plot be interactive (boolean, default TRUE)
`options`	a list of additional visual parameters created by calling `ggoncoplot_options()`. See `ggoncoplot_options` for details.
`verbose`	verbose mode (flag, default TRUE)

Details

This function generates a customizable oncoplot that displays the most frequently mutated genes (default top 10) along with interactive tooltips and clickable elements.

Value

ggplot or girafe object if `interactive=TRUE`

Examples

```
# ===== GBM =====
gbm_csv <- system.file(
  package = "ggoncoplot",
  "testdata/GBM_tcgamutations_mc3_maf.csv.gz"
)

gbm_clinical_csv <- system.file(
  package = "ggoncoplot",
  "testdata/GBM_tcgamutations_mc3_clinical.csv"
)

gbm_df <- read.csv(file = gbm_csv, header = TRUE)
gbm_clinical_df <- read.csv(file = gbm_clinical_csv, header = TRUE)

# Plot Basic Oncoplot
ggoncoplot(
  gbm_df,
  "Hugo_Symbol",
  "Tumor_Sample_Barcode",
  col_mutation_type = "Variant_Classification",
  metadata = gbm_clinical_df,
  cols_to_plot_metadata = "gender"
)

# Customise how the Oncoplot looks
ggoncoplot(
  gbm_df,
  "Hugo_Symbol",
  "Tumor_Sample_Barcode",
  col_mutation_type = "Variant_Classification",
  metadata = gbm_clinical_df,
  cols_to_plot_metadata = "gender",

  # Customise Visual Options
  options = ggoncoplot_options(
    xlab_title = "Glioblastoma Samples",
    ylab_title = "Top 10 mutated genes"
  )
)
```

`ggoncoplot_gene_barplot` *Gene barplot*

Description

Gene barplot

Usage

```
ggoncoplot_gene_barplot(
  data,
  fontsize_count = 14,
  palette = NULL,
  colour_mutation_type_unspecified = "grey10",
  show_axis,
  total_samples,
  show_genebar_labels = TRUE,
  genebar_label_nudge = 2,
  genebar_label_padding = 0.2,
  only_pad_if_labels_shown = TRUE,
  digits_to_round_to = 0,
  genebar_scale_n_breaks = 3,
  genebar_scale_breaks = ggplot2::waiver()
)
```

Arguments

`data`	data frame output by ggoncoplot_prep_df
`fontsize_count`	fontsize of gene mutation count x axis (number)
`palette`	a named vector mapping all possible mutation types (vector names) to colors (vector values, optional)
`colour_mutation_type_unspecified`	colour of mutations in oncoplot and margin plots if `col_mutation_type` is not supplied (string)
`show_axis`	show axis text/ticks/line (flag)
`total_samples`	Strategy for calculating the total number of samples. This value is used to compute the proportion of mutation recurrence displayed in the tooltip when hovering over the gene barplot, or as a text annotation when `ggoncoplot_options(show_genebar_labels = TRUE)` is set to TRUE. Possible values: • **any_mutations**: All the samples that are in `data` (the mutation dataset), irrespective of whether they are on the oncoplot or not. • **oncoplot**: Only the samples that are present on the oncoplot. • **all**: All the samples in either `data` or `metadata`.
`show_genebar_labels`	should gene barplot be labelled with % of samples the gene is mutated in (flag)
`genebar_label_nudge`	how much padding to add between the gene barplot and bar annotations (number)
`genebar_label_padding`	how much padding to add to the x axis of the gene barplot (number)
`only_pad_if_labels_shown`	should expansion to x axis be applied if bar labels aren't shown?
`digits_to_round_to`	how many digits to round recurrence proportions to

`genebar_scale_n_breaks`	an integer guiding the number of breaks The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `genebar_scale_breaks = ggplot2::waiver()`. Use `NULL` to use the default
`genebar_scale_breaks`	fine-grained control over the x axis breaks on the gene barplot. One of: • `NULL` for no minor breaks • `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous) • A numeric vector of positions • A function that given the limits returns a vector of minor breaks. When the function has two arguments, it will be given the limits and major break positions.

Value

ggplot showing gene mutation counts

`ggoncoplot_options` *ggoncoplot options*

Description

Customise the look of your oncoplot.

Usage

```
ggoncoplot_options(
  interactive_svg_width = 12,
  interactive_svg_height = 6,
  selection_type = c("none", "multiple", "single"),
  plotsize_tmb_rel_height = 10,
  plotsize_gene_rel_width = 20,
  plotsize_metadata_rel_height = 20,
  buffer_metadata = 2,
  buffer_tmb = 1,
  buffer_genebar = 2,
  xlab_title = "Sample",
  ylab_title = "Gene",
  sample_id_position = c("bottom", "top"),
  sample_id_angle = 90,
  fontsize_xlab = 26,
  fontsize_ylab = 26,
  fontsize_genes = 16,
  fontsize_samples = 12,
  fontsize_count = 14,
```

```
  fontsize_tmb_title = 14,
  fontsize_tmb_axis = 11,
  fontsize_pathway = 16,
  fontsize_legend_title = 12,
  fontsize_legend_text = 12,
  fontface_genes = c("plain", "italic", "bold", "bold.italic"),
  fontface_samples = c("plain", "italic", "bold", "bold.italic"),
  fontface_metadata_text = c("plain", "italic", "bold", "bold.italic"),
  tile_height = 1,
  tile_width = 1,
  colour_backround = "grey90",
  colour_mutation_type_unspecified = "grey10",
  show_sample_ids = FALSE,
  show_ylab_title = FALSE,
  show_xlab_title = FALSE,
  show_ylab_title_tmb = FALSE,
  show_legend = TRUE,
  show_legend_titles = TRUE,
  show_axis_gene = TRUE,
  show_genebar_labels = FALSE,
  show_axis_tmb = TRUE,
  log10_transform_tmb = TRUE,
  scientific_tmb = FALSE,
  genebar_label_padding = 0.3,
  genebar_only_pad_when_labels_shown = TRUE,
  genebar_label_nudge = 2,
  genebar_label_round = 0,
  genebar_scale_breaks = ggplot2::waiver(),
  genebar_scale_n_breaks = 3,
  colour_pathway_text = "white",
  colour_pathway_bg = "grey10",
  colour_pathway_outline = "black",
  pathway_text_angle = 0,
  ggoncoplot_guide_ncol = 2,
  legend_key_size = 0.4,
  prettify_legend_titles = TRUE,
  prettify_legend_values = TRUE,
  prettify_function = prettify,
  metadata_position = c("bottom", "top"),
  fontsize_metadata_text = 12,
  fontsize_metadata_legend_title = fontsize_legend_title,
  fontsize_metadata_legend_text = fontsize_legend_text,
  fontsize_metadata_barplot_y_numbers = 8,
  metadata_legend_nrow = NULL,
  metadata_legend_ncol = NULL,
  metadata_legend_key_size = legend_key_size,
  metadata_na_marker = "!",
  metadata_na_marker_size = 8,
```

```
  metadata_maxlevels = 6,
  metadata_numeric_plot_type = c("bar", "heatmap"),
  metadata_legend_orientation_heatmap = c("horizontal", "vertical"),
  metadata_colours_default = c("#66C2A5", "#FC8D62", "#8DA0CB", "#E78AC3", "#A6D854",
    "#FFD92F", "#E5C494")
)
```

Arguments

`interactive_svg_width`	dimensions of interactive plot (number)
`interactive_svg_height`	dimensions of interactive plot (number)
`selection_type`	Defines the type of data point selection allowed when the ggplot is interactive. Options include 'none' (default), 'multiple' (enables lasso-select tool), and 'single' (supports single-click selection).
`plotsize_tmb_rel_height`	percentage of vertical space TMB margin plot should take up. Must be some value between 5-90 (number)
`plotsize_gene_rel_width`	percentage of horizontal space the gene barplot should take up. Must be some value between 5-90 (number)
`plotsize_metadata_rel_height`	percentage of vertical space the metadata tile plot should take up. Must be some value between 5-90 (number)
`buffer_metadata`, `buffer_tmb`	amount of space to add between the main oncoplot and tmb/metadata marginal plots (number)
`buffer_genebar`	amount of space to add between the main oncoplot and tmb/metadata marginal plots (number)
`xlab_title`	x axis label. Set `xlab_title = NULL` to remove title (string)
`ylab_title`	y axis of interactive plot. Set `ylab_title = NULL` to remove title (string)
`sample_id_position`	should sample names on the x axis be on the **top** or **bottom** of the main oncoplot (string)
`sample_id_angle`	angle of the sample names (number)
`fontsize_xlab`	size of x axis title (number)
`fontsize_ylab`	size of y axis title (number)
`fontsize_genes`	size of y axis text (gene names) (number)
`fontsize_samples`	size of x axis text (sample names). Ignored unless show_sample_ids is set to true (number)
`fontsize_count`	fontsize of gene mutation count x axis (number)

`fontsize_tmb_title`	fontsize of y axis title for TMB marginal plot (number)
`fontsize_tmb_axis`	fontsize of y axis text for TMB marginal plot (number)
`fontsize_pathway`	fontsize of y axis strip text describing gene pathways (number)
`fontsize_legend_title`	fontsize of the legend titles (number)
`fontsize_legend_text`	fontsize of the legend text (number)
`fontface_genes`	font face of the gene names. One of ("plain", "italic", "bold", "bold.italic").
`fontface_samples`	font face of the sample names. One of ("plain", "italic", "bold", "bold.italic").
`fontface_metadata_text`	font face of the metadata columns. One of ("plain", "italic", "bold", "bold.italic").
`tile_height`	proportion of available vertical space each tile will take up (0-1) (number)
`tile_width`	proportion of available horizontal space each tile take up (0-1) (number)
`colour_backround`	colour used for background non-mutated tiles (string)
`colour_mutation_type_unspecified`	colour of mutations in oncoplot and margin plots if `col_mutation_type` is not supplied (string)
`show_sample_ids`	show sample_ids_on_x_axis (flag)
`show_ylab_title`	show y axis title of oncoplot (flag)
`show_xlab_title`	show x axis title of oncoplot (flag)
`show_ylab_title_tmb`	show y axis title of TMB margin plot (flag)
`show_legend`	show the oncoplot legend
`show_legend_titles`	show legend titles (flag)
`show_axis_gene`	show x axis line/ticks/labels for gene barplot (flag)
`show_genebar_labels`	should gene barplot be labelled with % of samples the gene is mutated in (flag)
`show_axis_tmb`	show y axis line/ticks/labels for TMB barplot (flag)
`log10_transform_tmb`	log10 transform total number of mutations for TMB marginal plot (flag)
`scientific_tmb`	display tmb counts in scientific notation (flag)
`genebar_label_padding`	how much padding to add to the x axis of the gene barplot (number)

`genebar_only_pad_when_labels_shown`
: only apply `genebar_label_padding` when labels are shown (flag)

`genebar_label_nudge`
: how much padding to add between the gene barplot and bar annotations (number)

`genebar_label_round`
: how many digits to round the genebar labels to (number)

`genebar_scale_breaks`
: fine-grained control over the x axis breaks on the gene barplot. One of:

  - `NULL` for no minor breaks
  - `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
  - A numeric vector of positions
  - A function that given the limits returns a vector of minor breaks. When the function has two arguments, it will be given the limits and major break positions.

`genebar_scale_n_breaks`
: an integer guiding the number of breaks The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `genebar_scale_breaks = ggplot2::waiver()`. Use `NULL` to use the default

`colour_pathway_text`
: colour of text describing pathways (string)

`colour_pathway_bg`
: background fill colour of pathway strips (string)

`colour_pathway_outline`
: outline colour of pathway strips (string)

`pathway_text_angle`
: angle of pathway text label (typically 0 or 90 degrees) (number)

`ggoncoplot_guide_ncol`
: how many columns to use when describing oncoplot legend (number)

`legend_key_size`
: width of the legend key block (number)

`prettify_legend_titles`
: Should legend titles be prettified to more human-readable forms (e.g. converting 'my_title' to 'My Title'). Prettification can be customised using the 'prettify_function' argument (flag)

`prettify_legend_values`
: Should legend values be prettified to more human-readable forms (e.g. converting 'my_title' to 'My Title'). Prettification can be customised using the 'prettify_function' argument (flag)

`prettify_function`
: a function that takes a string and returns a nicely formatted string. Used to prettify legend titles and values (function)

`metadata_position`
: should the metadata plot be at the **top** or **bottom** of the oncoplot.

`fontsize_metadata_text`	fontsize of the y axis text for in the sample metadata plot (number)
`fontsize_metadata_legend_title`	fontsize of the titles of metadata legends. Will default to `fontsize_legend_text` (number)
`fontsize_metadata_legend_text`	fontsize of the text in metadata legends. Will default to `fontsize_legend_title` (number)
`fontsize_metadata_barplot_y_numbers`	fontsize of the text describing numeric barplot max & min values (number)
`metadata_legend_nrow`	number of rows allowed per metadata legend (number)
`metadata_legend_ncol`	number of columns allowed per metadata legend (number)
`metadata_legend_key_size`	width of the legend key block (number). Defaults to `legend_key_size`
`metadata_na_marker`	character used to indicate data is missing (string)
`metadata_na_marker_size`	size of character used when data is missing (number)
`metadata_maxlevels`	or categorical variables, what is the maximum number of distinct values to allow (too many will make it hard to find a palette that suits) (number)
`metadata_numeric_plot_type`	visual representation of numeric properties. One of 'bar', for bar charts, or 'heatmap' for heatmaps
`metadata_legend_orientation_heatmap`	the orientation of heatmaps in legends. One of "horizontal" or "vertical" number of breaks given by the transformation.
`metadata_colours_default`	Default colors for categorical variables without a custom palette.

Value

ggoncoplot options object ready to be passed to `ggoncoplot()` `options` argument

Examples

```
# Read GBM MAF file
gbm_csv <- system.file(
  package = "ggoncoplot",
  "testdata/GBM_tcgamutations_mc3_maf.csv.gz"
)
gbm_df <- read.csv(file = gbm_csv, header = TRUE)

# Plot Oncoplot and Customise Options
gbm_df |>
```

```
ggoncoplot(
  col_genes = "Hugo_Symbol",
  col_samples = "Tumor_Sample_Barcode",
  col_mutation_type = "Variant_Classification",

  # Customise Visual Options
  options = ggoncoplot_options(

    # Interactive Plot Options
    interactive_svg_width = 12,
    interactive_svg_height = 6,

    # Relative height of different plotsizes
    plotsize_tmb_rel_height = 10,
    plotsize_gene_rel_width = 20,
    plotsize_metadata_rel_height = 20,

    # Axis Titles
    xlab_title = "Glioblastoma Samples",
    ylab_title = "Top 10 mutated genes",

    # Fontsizes
    fontsize_xlab = 40,
    fontsize_ylab = 40,
    fontsize_genes = 16,
    fontsize_samples = 12,
    fontsize_count = 14,
    fontsize_tmb_title = 14,
    fontsize_tmb_axis = 11,
    fontsize_pathway = 16,

    # Customise Tiles
    tile_height = 1,
    tile_width = 1,
    colour_backround = "grey90",
    colour_mutation_type_unspecified = "grey10",

    # Show different elements
    show_sample_ids = FALSE,
    show_ylab_title = FALSE,
    show_xlab_title = FALSE,
    show_ylab_title_tmb = FALSE,
    show_axis_gene = TRUE,
    show_axis_tmb = TRUE,

    # Transformation and label scales
    log10_transform_tmb = TRUE,
    scientific_tmb = FALSE,

    # Gene Barplot Specific Options
    show_genebar_labels = TRUE,
    genebar_label_padding = 0.2,
    genebar_only_pad_when_labels_shown = TRUE,
```

```
    genebar_label_nudge = 2,
    genebar_label_round = 1,

    # Pathway Faceting Colours / Text
    colour_pathway_text = "white",
    colour_pathway_bg = "grey10",
    colour_pathway_outline = "black",
    pathway_text_angle = 0,

    # Legend number of columns
    ggoncoplot_guide_ncol = 2
  )
)
```

`ggoncoplot_plot` *Plot oncoplot*

Description

This function takes the output from **ggoncoplot_prep_df** and plots it. Should not be exposed since it makes some assumptions about structure of input data.

Usage

```
ggoncoplot_plot(
  data,
  show_sample_ids = FALSE,
  palette = NULL,
  show_ylab_title = FALSE,
  show_xlab_title = FALSE,
  xlab_title = "Sample",
  ylab_title = "Gene",
  sample_id_position = c("bottom", "top"),
  sample_id_angle = 90,
  fontsize_xlab = 16,
  fontsize_ylab = 16,
  fontsize_genes = 14,
  fontsize_samples = 10,
  fontface_genes = "plain",
  fontface_samples = "plain",
  fontsize_legend_title = 12,
  fontsize_legend_text = 12,
  tile_height = 1,
  tile_width = 1,
  copy = c("sample", "gene", "tooltip", "mutation_type", "nothing"),
  colour_backround = "grey90",
  colour_mutation_type_unspecified = "grey10",
  fontsize_pathway = 16,
```

```
  colour_pathway_text = "white",
  colour_pathway_bg = "grey10",
  colour_pathway_outline = "black",
  pathway_text_angle = 0,
  legend_title = "Mutation Type",
  show_legend_titles = TRUE,
  ggoncoplot_guide_ncol = 2,
  legend_key_size = 0.3,
  margin_t = 0.2,
  margin_r = 0.3,
  margin_b = 0.2,
  margin_l = 0.3,
  margin_unit = "cm",
  mutation_type_supplied = TRUE,
  prettify_legend_values = TRUE,
  prettify_function = prettify
)
```

Arguments

`data`	transformed data from `ggoncoplot_prep_df()` (data.frame)
`show_sample_ids`	show sample_ids_on_x_axis (flag)
`palette`	a named vector mapping all possible mutation types (vector names) to colors (vector values, optional)
`show_ylab_title`	show y axis title of oncoplot (flag)
`show_xlab_title`	show x axis title of oncoplot (flag)
`xlab_title`	x axis label. Set `xlab_title` = `NULL` to remove title (string)
`ylab_title`	y axis of interactive plot. Set `ylab_title` = `NULL` to remove title (string)
`sample_id_position`	should sample names on the x axis be on the **top** or **bottom** of the main oncoplot (string)
`sample_id_angle`	angle of the sample names (number)
`fontsize_xlab`	size of x axis title (number)
`fontsize_ylab`	size of y axis title (number)
`fontsize_genes`	size of y axis text (gene names) (number)
`fontsize_samples`	size of x axis text (sample names). Ignored unless show_sample_ids is set to true (number)
`fontface_genes`	font face of the gene names. One of ("plain", "italic", "bold", "bold.italic").
`fontface_samples`	font face of the sample names. One of ("plain", "italic", "bold", "bold.italic").

`fontsize_legend_title`	fontsize of the legend titles (number)
`fontsize_legend_text`	fontsize of the legend text (number)
`tile_height`	proportion of available vertical space each tile will take up (0-1) (number)
`tile_width`	proportion of available horizontal space each tile take up (0-1) (number)
`copy`	value to copy to clipboard when an oncoplot tile is clicked (string, one of 'sample', 'gene', 'tooltip', 'mutation_type', 'nothing', default 'sample')
`colour_backround`	colour used for background non-mutated tiles (string)
`colour_mutation_type_unspecified`	colour of mutations in oncoplot and margin plots if `col_mutation_type` is not supplied (string)
`fontsize_pathway`	fontsize of y axis strip text describing gene pathways (number)
`colour_pathway_text`	colour of text describing pathways (string)
`colour_pathway_bg`	background fill colour of pathway strips (string)
`colour_pathway_outline`	outline colour of pathway strips (string)
`pathway_text_angle`	angle of pathway text label (typically 0 or 90 degrees) (number)
`legend_title`	name of legend title (string)
`show_legend_titles`	show legend titles (flag)
`ggoncoplot_guide_ncol`	how many columns to use when describing oncoplot legend (number)
`legend_key_size`	width of the legend key block (number)
`margin_t`, `margin_r`, `margin_b`, `margin_l`	margin for top, right, bottom, and left side of plot. By default, unit is 'cm' but can be changed by setting `margin_unit` to any value `ggplot2::margin()` will understand (number)
`margin_unit`	Unit of margin specification. By default is 'cm' but can be changed by setting `margin_unit` to any value `ggplot2::margin()` will understand (string)
`mutation_type_supplied`	did user supply a mutation_type column? If not, will turn off legend.
`prettify_legend_values`	Should legend values be prettified to more human-readable forms (e.g. converting 'my_title' to 'My Title'). Prettification can be customised using the 'prettify_function' argument (flag)
`prettify_function`	a function that takes a string and returns a nicely formatted string. Used to prettify legend titles and values (function)

Value

ggplot or girafe object if `interactive=TRUE`

Examples

```
# ===== GBM =====
gbm_csv <- system.file(
  package = "ggoncoplot",
  "testdata/GBM_tcgamutations_mc3_maf.csv.gz"
)

gbm_clinical_csv <- system.file(
  package = "ggoncoplot",
  "testdata/GBM_tcgamutations_mc3_clinical.csv"
)

gbm_df <- read.csv(file = gbm_csv, header = TRUE)
gbm_clinical_df <- read.csv(file = gbm_clinical_csv, header = TRUE)

# Plot Basic Oncoplot
ggoncoplot(
  gbm_df,
  "Hugo_Symbol",
  "Tumor_Sample_Barcode",
  col_mutation_type = "Variant_Classification",
  metadata = gbm_clinical_df,
  cols_to_plot_metadata = "gender"
)

# Customise how the Oncoplot looks
ggoncoplot(
  gbm_df,
  "Hugo_Symbol",
  "Tumor_Sample_Barcode",
  col_mutation_type = "Variant_Classification",
  metadata = gbm_clinical_df,
  cols_to_plot_metadata = "gender",

  # Customise Visual Options
  options = ggoncoplot_options(
    xlab_title = "Glioblastoma Samples",
    ylab_title = "Top 10 mutated genes"
  )
)
```

`ggoncoplot_prep_df` *Prep data for oncoplot*

Description

Prep data for oncoplot

Usage

```
ggoncoplot_prep_df(
  data,
  col_genes,
  col_samples,
  genes_for_oncoplot,
  col_mutation_type = NULL,
  col_tooltip = col_samples,
  pathway = NULL,
  verbose = TRUE
)
```

Arguments

`data`	data for oncoplot. A data.frame with 1 row per mutation in your cohort. Must contain columns describing gene_symbols and sample_identifiers (data.frame)
`col_genes`	name of **data** column containing gene names/symbols (string)
`col_samples`	name of **data** column containing sample identifiers (string)
`genes_for_oncoplot`	a list of genes to include in the oncoplot (character).
`col_mutation_type`	name of **data** column describing mutation types (string)
`col_tooltip`	name of **data** column containing whatever information you want to display in (string)
`pathway`	a two column dataframe describing pathway. The column containing gene names should have the same name as **col_gene** (data.frame, optional)
`verbose`	verbose mode (flag, default TRUE)

Value

dataframe with the following columns: 'Gene', 'Sample', 'MutationType', 'Tooltip'. Sample is a factor with levels sorted in appropriate order for oncoplot vis. Genes represents either topn genes or specific genes set by `genes_to_include`

Examples

```
#' # ===== GBM =====
gbm_csv <- system.file(
  package = "ggoncoplot",
  "testdata/GBM_tcgamutations_mc3_maf.csv.gz"
)

gbm_df <- read.csv(file = gbm_csv, header = TRUE)

# Get genes in appropriate order for oncoplot
genes_for_oncoplot <- ggoncoplot:::get_genes_for_oncoplot(
  data = gbm_df,
  col_samples = "Tumor_Sample_Barcode",
```

```
  col_genes = "Hugo_Symbol",
  topn = 20,
  verbose = FALSE
)

# Create dataframe basis of oncoplot (1 row per sample-gene combo)
ggoncoplot:::ggoncoplot_prep_df(
  gbm_df,
  col_genes = "Hugo_Symbol",
  col_samples = "Tumor_Sample_Barcode",
  col_mutation_type = "Variant_Classification",
  genes_for_oncoplot = genes_for_oncoplot
)
```

`identify_topn_genes` *Identify top genes from a mutation df*

Description

Identify top genes from a mutation df

Usage

```
identify_topn_genes(
  data,
  col_samples,
  col_genes,
  topn,
  genes_to_ignore = NULL,
  return_extra_genes_if_tied = FALSE,
  verbose = TRUE
)
```

Arguments

`data`	data for oncoplot. A data.frame with 1 row per mutation in your cohort. Must contain columns describing gene_symbols and sample_identifiers (data.frame)
`col_samples`	name of **data** column containing sample identifiers (string)
`col_genes`	name of **data** column containing gene names/symbols (string)
`topn`	how many of the top genes to visualize. Ignored if `genes_to_include` is supplied (number, default 10)
`genes_to_ignore`	names of the genes that should be ignored (character, optional)

`return_extra_genes_if_tied`	instead of strictly returning `topn` genes, in the case of ties (where multiple genes are mutated in the exact same number of samples, complicating selection of top n genes), return all tied genes (potentially more than topn). If FALSE, will return strictly `topn` genes, breaking ties based on order of appearance in dataset (flag, default FALSE)
`verbose`	verbose mode (flag, default TRUE)

Value

vector of topn genes. Their order will be their rank (most mutated = first) (character)

`oncosim` *Simulated Cancer Genome Dataset*

Description

An artificial cancer dataset describing mutations found in 9 different tumour samples. Rows represent mutations.

Usage

```
oncosim
```

Format

`oncosim`**:**

A data frame with 143 rows and 3 columns:

Samples Sample containing mutation in the specified gene

Genes Mutated gene

VariantType Type of mutation in gene

Source

not applicable, simulated

`oncosim_metadata` *Simulated Cancer Dataset Metadata*

Description

A sample-level metadata table for the `oncosim` simulated cancer dataset. Contains assorted numeric, categorical, clinical, and logical features for each sample.

Usage

```
oncosim_metadata
```

Format

`oncosim_metadata`**:**

A data frame with 11 rows and 6 columns:

Samples Unique sample identifiers

numeric_feature Numeric variable including zeros, positive and negative values, `NA`, and `Inf`/`-Inf`

categorical_feature4levels Categorical variable with four levels (`"cat"`, `"dog"`, `"magpie"`, `"giraffe"`), may contain empty strings or `NA`

clinical_feature2levels Clinical categorical variable indicating biological sex with two levels (`"male"`, `"female"`), may contain `NA`

logical_feature Logical variable with `TRUE`, `FALSE`, or `NA`

numeric_that_could_be_logical Integer variable coded as `0` or `1` (and `NA`) that could be interpreted as logical

Source

not applicable, simulated

`prettify` *Make strings prettier for printing*

Description

Takes an input string and ’prettify’ by converting underscores to spaces, capitalising each word, etc.

Usage

```
prettify(string, space_after_apostrophe = TRUE, autodetect_units = TRUE)
```

Arguments

`string`	input string
`space_after_apostrophe`	add a space after any apostrophe so long as its after an alphanumeric character and followed by anything but a space (flag)
`autodetect_units`	automatically detect units (e.g. mm, kg, etc) and wrap in brackets.

Value

string

`rank_genes_based_on_pathways`
Calculate Pathway-informed Genes Rankings

Description

Which genes should appear at the top of the oncoplot? This function takes pathway and gene ranks and returns a list of genes sorted first by pathway then by gene rank. Gene & pathway rankings can be calculated upstream. By default will use their order in gene_pathway_map.

Usage

```
rank_genes_based_on_pathways(
  gene_pathway_map,
  generanks = unique(as.character(gene_pathway_map[[1]])),
  pathwayranks = unique(as.character(gene_pathway_map[[2]]))
)
```

Arguments

`gene_pathway_map`	dataframe where column 1 = gene names and column 2 = pathway names
`generanks`	gene names in the order they should be ranked, where earlier in vector = further up in oncoplot. (character)
`pathwayranks`	pathway names in the order they should be ranked, where earlier in vector = further up in oncoplot (character)

Value

gene names, sorted based on order they should appear in oncoplot (first = top). Only returns genes present in generanks (character)

`score_based_on_gene_rank` *Generate score based on genes*

Description

Score used to sort samples based on which genes are mutated. Make sure to run on one sample at once (use grouping)

Usage

```
score_based_on_gene_rank(
  mutated_genes,
  genes_informing_score,
  gene_rank,
  debug_mode = FALSE
)
```

Arguments

`mutated_genes`	vector of genes that are mutated for a single sample (character)
`genes_informing_score`	which genes determine the sort order? (character)
`gene_rank`	what is the order of importance of genes used to determine sort order. Higher number = higher in sort order (character)
`debug_mode`	debug mode (flag)

Value

a score (higher = should be higher in the sorting order) (number)

Examples

```
## Not run:
# First set of genes has a high rank since both BRCA2 and EGFR are mutated
score_based_on_gene_rank(c("TERT", "EGFR", "PTEN", "BRCA2"), c("EGFR", "BRCA2"), gene_rank = 1:2)

# If EGFR is mutated without BRCA2, we get a lower score
score_based_on_gene_rank(c("TERT", "EGFR", "PTEN", "IDH1"), c("EGFR", "BRCA2"), gene_rank = 1:2)

# If BRCA2 is mutated without EGFR,
# we get a score lower than BRCA2+EGFR but higher than EGFR alone due to higher gene_rank of BRCA2
score_based_on_gene_rank(c("TERT", "IDH1", "PTEN", "BRCA2"), c("EGFR", "BRCA2"), gene_rank = 1:2)

## End(Not run)
```

`theme_oncoplot_default` *Oncoplot Theme: default*

Description

Oncoplot Theme: default

Usage

```
theme_oncoplot_default(
  show_legend_titles = TRUE,
  fontsize_legend_title = 12,
  fontsize_legend_text = 12,
  ...
)
```

Arguments

`show_legend_titles`	show legend titles (flag)
`fontsize_legend_title`	fontsize of the legend titles (number)
`fontsize_legend_text`	fontsize of the legend text (number)
`...`	passed to `ggplot2::theme()` theme

`unify_samples` *Prepare dataset for plotting*

Description

Take a dataframe containing a column describing sample IDs (`col_sample`) Filter on `col_sample` %in% samples_to_show. Add any missing samples_to_show not present DF as levels of `col_sample`. This way, when plotting we can use scale_x_discrete(drop=FALSE) to display all the samples we care about

Usage

```
unify_samples(data, col_samples, samples_to_show)
```

Arguments

`data`	dataframe with a column describing sample IDs (data.frame)
`col_samples`	name of column in `data` containing sample IDs (character)
`samples_to_show`	the samples we want to show in plots. These samples should be the only ones represented in data.frame content, and any missing ones will be added as factor levels (character)

Value

data.frame

Index