

Package ‘tidyplots’

May 8, 2026

Title Tidy Plots for Scientific Papers

Version 0.4.0

Description The goal of ‘tidyplots’ is to streamline the creation of publication-ready plots for scientific papers. It allows to gradually add, remove and adjust plot components using a consistent and intuitive syntax.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports cli, dplyr, forcats, ggbeeswarm, ggplot2 (>= 4.0.1), ggpubr, ggrastr, ggrepel, glue, gtable, Hmisc, htmltools, lifecycle, purrr, rlang, scales, stringr, tidyr, tidyselect

Depends R (>= 4.1.0)

LazyData true

URL <https://github.com/jbengler/tidyplots>,
<https://jbengler.github.io/tidyplots/>

BugReports <https://github.com/jbengler/tidyplots/issues>

Suggests knitr, rmarkdown, testthat (>= 3.0.0), vdiff

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Jan Broder Engler [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0002-3169-2076>>)

Maintainer Jan Broder Engler <broder.engler@gmail.com>

Repository CRAN

Date/Publication 2026-01-08 08:40:07 UTC

Contents

add	3
add_annotation_text	4
add_areastack_absolute	5
add_barstack_absolute	7
add_boxplot	8
add_count_bar	10
add_curve_fit	13
add_data_labels	15
add_data_points	17
add_ellipse	21
add_heatmap	22
add_histogram	23
add_line	24
add_mean_bar	25
add_median_bar	28
add_pie	31
add_reference_lines	33
add_sem_errorbar	34
add_sem_ribbon	36
add_sum_bar	37
add_test_pvalue	41
add_title	46
add_violin	47
adjust_colors	48
adjust_font	50
adjust_legend_title	51
adjust_padding	53
adjust_size	55
adjust_theme_details	56
adjust_title	57
adjust_x_axis	59
all_rows	62
animals	64
climate	65
colors_continuous_viridis	65
colors_discrete_friendly	67
colors_diverging_blue2red	68
dinosaurs	69
distributions	70
energy	70
energy_week	71
eu_countries	71
flip_plot	72
format_p_value	73
gene_expression	73
new_color_scheme	74

pca	74
remove_legend	75
remove_padding	76
remove_title	76
remove_x_axis	77
remove_y_axis	78
rename_x_axis_levels	80
reorder_x_axis_levels	81
reverse_x_axis_levels	83
save_plot	84
sort_x_axis_levels	86
spendings	87
split_plot	88
study	90
theme_tidyplot	91
tidyplot	92
tidyplots_options	93
time_course	95
view_plot	95

Index	97
--------------	-----------

add	<i>Add ggplot2 code to a tidyplot</i>
-----	---------------------------------------

Description

Add ggplot2 code to a tidyplot

Usage

```
add()
```

Value

A tidyplot object.

Examples

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add(ggplot2::geom_point())
```

add_annotation_text *Add annotation*

Description

Add annotation

Usage

```
add_annotation_text(plot, text, x, y, fontsize = 7, ...)
```

```
add_annotation_rectangle(  
  plot,  
  xmin,  
  xmax,  
  ymin,  
  ymax,  
  fill = plot$tidyplot$ink,  
  color = NA,  
  alpha = 0.1,  
  ...  
)
```

```
add_annotation_line(plot, x, xend, y, yend, color = plot$tidyplot$ink, ...)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
text	String for annotation text.
x, xmin, xmax, xend, y, ymin, ymax, yend	Coordinates for the annotation.
fontsize	Font size in points. Defaults to 7.
...	Arguments passed on to ggplot2::annotate().
fill	A hex color for the fill color. For example, "#FFFFFF" for white.
color	A hex color for the stroke color. For example, "#FFFFFF" for white.
alpha	A number between 0 and 1 for the opacity of an object. A value of 0 is completely transparent, 1 is completely opaque.

Value

A tidyplot object.

Examples

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_boxplot() |>
  add_annotation_text("Look here!", x = 2, y = 25)

eu_countries |>
  tidyplot(x = area, y = population) |>
  add_data_points() |>
  add_annotation_rectangle(xmin = 2.5e5, xmax = Inf, ymin = 42, ymax = Inf)

eu_countries |>
  tidyplot(x = area, y = population) |>
  add_data_points() |>
  add_annotation_rectangle(xmin = 2.5e5, xmax = 6e5, ymin = 42, ymax = 90,
                          color = "#E69F00", fill = NA)

eu_countries |>
  tidyplot(x = area, y = population) |>
  add_data_points() |>
  add_annotation_line(x = 0, xend = Inf, y = 0, yend = Inf)
```

add_areastack_absolute

Add area stack

Description

Add area stack

Usage

```
add_areastack_absolute(
  plot,
  linewidth = 0.25,
  alpha = 0.4,
  reverse = FALSE,
  replace_na = FALSE,
  ...
)
```

```
add_areastack_relative(
  plot,
  linewidth = 0.25,
  alpha = 0.4,
  reverse = FALSE,
  replace_na = FALSE,
```

```
    ...
  )
```

Arguments

plot	A tidyplot generated with the function tidyplot().
linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.
alpha	A number between 0 and 1 for the opacity of an object. A value of 0 is completely transparent, 1 is completely opaque.
reverse	Whether the order should be reversed or not. Defaults to FALSE, meaning not reversed.
replace_na	Whether to replace count = NA with count = 0.
...	Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```
# for a `count` provide `x` and `color`
# `count` of the data points in each `energy_type` category
energy |>
  tidyplot(x = year, color = energy_type) |>
  add_areastack_absolute()

energy |>
  tidyplot(x = year, color = energy_type) |>
  add_areastack_relative()

# for a `sum` provide `x`, `y` and `color`
# `sum` of `energy` in each `energy_type` category
energy |>
  tidyplot(x = year, y = energy, color = energy_type) |>
  add_areastack_absolute()

energy |>
  tidyplot(x = year, y = energy, color = energy_type) |>
  add_areastack_relative()

# Flip x and y-axis
energy |>
  tidyplot(x = energy, y = year, color = energy_type) |>
  add_areastack_absolute(orientation = "y")

energy |>
  tidyplot(x = energy, y = year, color = energy_type) |>
  add_areastack_relative(orientation = "y")
```

add_barstack_absolute *Add bar stack*

Description

Add bar stack

Usage

```
add_barstack_absolute(plot, width = 0.8, reverse = FALSE, ...)
```

```
add_barstack_relative(plot, width = 0.8, reverse = FALSE, ...)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
width	Horizontal width of the plotted object (bar, error bar, boxplot, violin plot, etc). Typical values range between 0 and 1.
reverse	Whether the order should be reversed or not. Defaults to FALSE, meaning not reversed.
...	Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```
# for a `count` only provide `color`
# `count` of the data points in each `energy_type` category
energy |>
  tidyplot(color = energy_type) |>
  add_barstack_absolute()

energy |>
  tidyplot(color = energy_type) |>
  add_barstack_relative()

# for a `sum` provide `color` and `y`
# `sum` of `energy` in each `energy_type` category
energy |>
  tidyplot(y = energy, color = energy_type) |>
  add_barstack_absolute()

energy |>
  tidyplot(y = energy, color = energy_type) |>
  add_barstack_relative()
```

```
# Include variable on second axis
energy |>
  tidyplot(x = year, y = energy, color = energy_type) |>
  add_barstack_absolute()

energy |>
  tidyplot(x = year, y = energy, color = energy_type) |>
  add_barstack_relative()

# Flip x and y-axis
energy |>
  tidyplot(x = energy, y = year, color = energy_type) |>
  add_barstack_absolute(orientation = "y")

energy |>
  tidyplot(x = energy, y = year, color = energy_type) |>
  add_barstack_relative(orientation = "y")
```

add_boxplot

Add boxplot

Description

Add boxplot

Usage

```
add_boxplot(
  plot,
  dodge_width = NULL,
  alpha = 0.3,
  saturation = 1,
  show_whiskers = TRUE,
  show_outliers = TRUE,
  box_width = 0.6,
  whiskers_width = 0.8,
  outlier.size = 0.5,
  coef = 1.5,
  outlier.shape = 19,
  outlier.alpha = 1,
  linewidth = 0.25,
  preserve = "total",
  ...
)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
dodge_width	For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.
alpha	A number between 0 and 1 for the opacity of an object. A value of 0 is completely transparent, 1 is completely opaque.
saturation	A number between 0 and 1 for the color saturation of an object. A value of 0 is completely desaturated (white), 1 is the original color.
show_whiskers	Whether to show boxplot whiskers. Defaults to TRUE.
show_outliers	Whether to show outliers. Defaults to TRUE.
box_width	Width of the boxplot. Defaults to 0.6.
whiskers_width	Width of the whiskers. Defaults to 0.8.
outlier.size	Size of the outliers. Defaults to 0.5.
coef	Length of the whiskers as multiple of IQR. Defaults to 1.5.
outlier.shape	Shape of the outliers. Defaults to 19.
outlier.alpha	Opacity of the outliers. Defaults to 1.
linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.
preserve	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
...	Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_boxplot()

# Changing arguments:
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_boxplot(show_whiskers = FALSE)

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_boxplot(show_outliers = FALSE)

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_boxplot(box_width = 0.2)

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
```

```
add_boxplot(whiskers_width = 0.2)
```

add_count_bar	<i>Add count</i>
---------------	------------------

Description

Add count

Usage

```
add_count_bar(  
  plot,  
  dodge_width = NULL,  
  width = 0.6,  
  saturation = 1,  
  preserve = "total",  
  ...  
)
```

```
add_count_dash(  
  plot,  
  dodge_width = NULL,  
  width = 0.6,  
  linewidth = 0.25,  
  preserve = "total",  
  ...  
)
```

```
add_count_dot(plot, dodge_width = NULL, size = 2, preserve = "total", ...)
```

```
add_count_value(  
  plot,  
  dodge_width = NULL,  
  accuracy = 0.1,  
  scale_cut = NULL,  
  fontsize = 7,  
  extra_padding = 0.15,  
  vjust = NULL,  
  hjust = NULL,  
  preserve = "total",  
  ...  
)
```

```
add_count_line(  
  plot,
```

```

    group,
    dodge_width = NULL,
    linewidth = 0.25,
    preserve = "total",
    ...
)

add_count_area(
  plot,
  group,
  dodge_width = NULL,
  linewidth = 0.25,
  preserve = "total",
  ...
)

```

Arguments

plot	A tidyplot generated with the function <code>tidyplot()</code> .
dodge_width	For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.
width	Horizontal width of the plotted object (bar, error bar, boxplot, violin plot, etc). Typical values range between 0 and 1.
saturation	A number between 0 and 1 for the color saturation of an object. A value of 0 is completely desaturated (white), 1 is the original color.
preserve	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
...	Arguments passed on to the geom function.
linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.
size	A number representing the size of the plot symbol. Typical values range between 1 and 3.
accuracy	A number to round to. Use (e.g.) 0.01 to show 2 decimal places of precision. If NULL, the default, uses a heuristic that should ensure breaks have the minimum number of digits needed to show the difference between adjacent values. Applied to rescaled data.
scale_cut	Scale cut function to be applied. See <code>scales::cut_short_scale()</code> and friends.
fontsize	Font size in points. Defaults to 7.
extra_padding	Extra padding to create space for the value label.
vjust	Vertical position adjustment of the value label.
hjust	Horizontal position adjustment of the value label.
group	Variable in the dataset to be used for grouping.

Value

A tidyplot object.

Examples

```
dinosaurs |>
  tidyplot(x = time_lived, color = time_lived) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_bar()

dinosaurs |>
  tidyplot(x = time_lived, color = time_lived) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_dash()

dinosaurs |>
  tidyplot(x = time_lived, color = time_lived) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_dot()

dinosaurs |>
  tidyplot(x = time_lived, color = time_lived) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_value()

dinosaurs |>
  tidyplot(x = time_lived) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_line()

dinosaurs |>
  tidyplot(x = time_lived) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_area()

# Combination
dinosaurs |>
  tidyplot(x = time_lived) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_bar(alpha = 0.4) |>
  add_count_dash() |>
  add_count_dot() |>
  add_count_value() |>
  add_count_line()

# Changing arguments: alpha
# Makes objects transparent
dinosaurs |>
  tidyplot(x = time_lived, color = time_lived) |>
  theme_minimal_y() |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_bar(alpha = 0.4)

# Changing arguments: saturation
# Reduces fill color saturation without making the object transparent
dinosaurs |>
```

```

tidyplot(x = time_lived, color = time_lived) |>
  theme_minimal_y() |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_bar(saturation = 0.3)

# Changing arguments: accuracy
dinosaurs |>
  tidyplot(x = time_lived, color = time_lived) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_value(accuracy = 1)

# Changing arguments: fontsize
dinosaurs |>
  tidyplot(x = time_lived, color = time_lived) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_value(fontsize = 10)

# Changing arguments: color
dinosaurs |>
  tidyplot(x = time_lived, color = time_lived) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_count_value(color = "black")

```

add_curve_fit

Add curve fit

Description

Add curve fit

Usage

```

add_curve_fit(
  plot,
  dodge_width = NULL,
  method = "loess",
  linewidth = 0.25,
  alpha = 0.4,
  preserve = "total",
  ...
)

```

Arguments

plot A tidyplot generated with the function `tidyplot()`.

dodge_width For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.

method	<p>Smoothing method (function) to use, accepts either NULL or a character vector, e.g. "lm", "glm", "gam", "loess" or a function, e.g. MASS::rlm or mgcv::gam, stats::lm, or stats::loess. "auto" is also accepted for backwards compatibility. It is equivalent to NULL.</p> <p>For method = NULL the smoothing method is chosen based on the size of the largest group (across all panels). stats::loess() is used for less than 1,000 observations; otherwise mgcv::gam() is used with formula = y ~ s(x, bs = "cs") with method = "REML". Somewhat anecdotally, loess gives a better appearance, but is $O(N^2)$ in memory, so does not work for larger datasets.</p> <p>If you have fewer than 1,000 observations but want to use the same gam() model that method = NULL would use, then set method = "gam", formula = y ~ s(x, bs = "cs").</p>
linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.
alpha	A number between 0 and 1 for the opacity of an object. A value of 0 is completely transparent, 1 is completely opaque.
preserve	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
...	Arguments passed on to ggplot2::geom_smooth().

Value

A tidyplot object.

Examples

```
time_course |>
  tidyplot(x = day, y = score, color = treatment) |>
  add_curve_fit()

# Changing arguments
time_course |>
  tidyplot(x = day, y = score, color = treatment) |>
  add_curve_fit(linewidth = 1)

time_course |>
  tidyplot(x = day, y = score, color = treatment) |>
  add_curve_fit(alpha = 0.8)

# Remove confidence interval
time_course |>
  tidyplot(x = day, y = score, color = treatment) |>
  add_curve_fit(se = FALSE)
```

add_data_labels	<i>Add data labels</i>
-----------------	------------------------

Description

Add data labels

Usage

```
add_data_labels(  
  plot,  
  label,  
  data = all_rows(),  
  fontsize = 7,  
  dodge_width = NULL,  
  jitter_width = 0,  
  jitter_height = 0,  
  preserve = "total",  
  background = FALSE,  
  background_color = NULL,  
  background_alpha = 0.6,  
  label_position = c("below", "above", "left", "right", "center"),  
  ...  
)
```

```
add_data_labels_repel(  
  plot,  
  label,  
  data = all_rows(),  
  fontsize = 7,  
  dodge_width = NULL,  
  jitter_width = 0,  
  jitter_height = 0,  
  preserve = "total",  
  segment.size = 0.2,  
  box.padding = 0.2,  
  max.overlaps = Inf,  
  background = FALSE,  
  background_color = NULL,  
  background_alpha = 0.6,  
  ...  
)
```

Arguments

plot	A tidyplot generated with the function <code>tidyplot()</code> .
label	Variable in the dataset to be used for the text label.

<code>data</code>	The data to be displayed in this layer. There are three options: <ul style="list-style-type: none"> • If <code>all_rows()</code> (the default) the complete dataset is displayed. • A function to subset the plot data. See <code>filter_rows()</code> and friends. • A <code>data.frame</code> to override the plot data.
<code>fontsize</code>	Font size in points. Defaults to 7.
<code>dodge_width</code>	For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.
<code>jitter_width</code>	Amount of random noise to be added to the horizontal position of the of the data points. This can be useful to deal with overplotting. Typical values range between 0 and 1.
<code>jitter_height</code>	Amount of random noise to be added to the vertical position of the of the data points. This can be useful to deal with overplotting. Typical values range between 0 and 1.
<code>preserve</code>	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
<code>background</code>	Whether to include semitransparent background box behind the labels to improve legibility. Defaults to FALSE.
<code>background_color</code>	Hex color of the background box. The default (NULL) retrieves the setting from the paper color of the plot, which defaults to "#FFFFFF".
<code>background_alpha</code>	Opacity of the background box. Defaults to 0.6.
<code>label_position</code>	Position of the label in relation to the data point. Can be one of <code>c("below", "above", "left", "right", "center")</code> .
<code>...</code>	Arguments passed on to the <code>geom</code> function.
<code>segment.size</code>	Thickness of the line connecting the label with the data point. Defaults to 0.2.
<code>box.padding</code>	Amount of padding around bounding box, as unit or number. Defaults to 0.25. (Default unit is lines, but other units can be specified by passing <code>unit(x, "units")</code>).
<code>max.overlaps</code>	Exclude text labels when they overlap too many other things. For each text label, we count how many other text labels or other data points it overlaps, and exclude the text label if it has too many overlaps. Defaults to 10.

Details

- `add_data_labels_repel()` uses `ggrepel::geom_text_repel()`. Check there and in [ggrepel examples](#) for additional arguments.
- `add_data_labels()` and `add_data_labels_repel()` support data subsetting. See [Advanced plotting](#).

Value

A `tidyplot` object.

Examples

```
# Create plot and increase padding to make more space for labels
p <-
  animals |>
  dplyr::slice_head(n = 5) |>
  tidyplot(x = weight, y = speed) |>
  theme_ggplot2() |>
  add_data_points() |>
  adjust_padding(all = 0.3)

# Default label position is `below` the data point
p |> add_data_labels(label = animal)

# Alternative label positions
p |> add_data_labels(label = animal, label_position = "above")

p |> add_data_labels(label = animal, label_position = "right")

p |> add_data_labels(label = animal, label_position = "left")

# Include white background box
p |> add_data_labels(label = animal, background = TRUE)

p |> add_data_labels(label = animal, background = TRUE,
  background_color = "pink")

# Black labels
p |> add_data_labels(label = animal, color = "black")

# Use repelling data labels
p |> add_data_labels_repel(label = animal, color = "black")

p |> add_data_labels_repel(label = animal, color = "black",
  background = TRUE)

p |> add_data_labels_repel(label = animal, color = "black",
  background = TRUE, min.segment.length = 0)
```

add_data_points

Add data points

Description

Add data points

Usage

```
add_data_points(
```

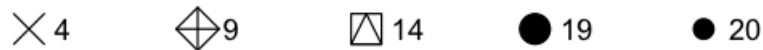
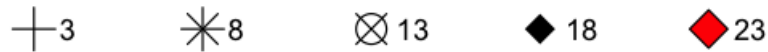
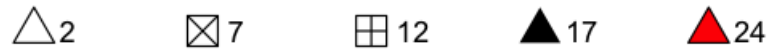
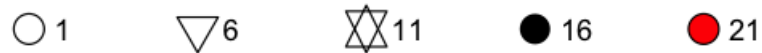
```
plot,  
data = all_rows(),  
shape = 19,  
size = 1,  
white_border = FALSE,  
dodge_width = NULL,  
preserve = "total",  
rasterize = FALSE,  
rasterize_dpi = 300,  
...  
)  
  
add_data_points_jitter(  
plot,  
data = all_rows(),  
shape = 19,  
size = 1,  
white_border = FALSE,  
dodge_width = NULL,  
jitter_width = 0.2,  
jitter_height = 0,  
preserve = "total",  
rasterize = FALSE,  
rasterize_dpi = 300,  
...  
)  
  
add_data_points_beeswarm(  
plot,  
data = all_rows(),  
shape = 19,  
size = 1,  
white_border = FALSE,  
cex = 3,  
corral = "wrap",  
corral.width = 0.5,  
dodge_width = NULL,  
preserve = "total",  
rasterize = FALSE,  
rasterize_dpi = 300,  
...  
)
```

Arguments

plot	A tidyplot generated with the function <code>tidyplot()</code> .
data	The data to be displayed in this layer. There are three options: <ul style="list-style-type: none">• If <code>all_rows()</code> (the default) the complete dataset is displayed.

- A function to subset the plot data. See `filter_rows()` and friends.
- A data.frame to override the plot data.

shape An integer between 0 and 24, representing the shape of the plot symbol.



size A number representing the size of the plot symbol. Typical values range between 1 and 3.

white_border Whether to include a white border around data points. Defaults to FALSE.

dodge_width For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.

preserve Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?

rasterize If FALSE (the default) the layer will be constructed of vector shapes. If TRUE the layer will be rasterized to a pixel image. This can be useful when plotting many individual objects (1,000 or more) compromises the performance of the generated PDF file.

rasterize_dpi The resolution in dots per inch (dpi) used for rastering the layer if rasterize is TRUE. The default is 300 dpi.

... Arguments passed on to the geom function.

jitter_width Amount of random noise to be added to the horizontal position of the of the data points. This can be useful to deal with overplotting. Typical values range between 0 and 1.

jitter_height	Amount of random noise to be added to the vertical position of the of the data points. This can be useful to deal with overplotting. Typical values range between 0 and 1.
cex	Scaling for adjusting point spacing (see <code>beeswarm::swarmx()</code>). Values between 1 (default) and 3 tend to work best.
corral	Method used to adjust points that would be placed too wide horizontally. Options are "none" (default), "gutter", "wrap", "random", and "omit". See Details below.
corral.width	Width of the corral, if not "none". Default is 0.9.

Details

- `add_data_points_beeswarm()` is based on `ggbeeswarm::geom_beeswarm()`. Check there for additional arguments.
- `add_data_points()` and friends support rasterization. See examples and [Advanced plotting](#).
- `add_data_points()` and friends support data subsetting. See examples and [Advanced plotting](#).

Value

A tidyplot object.

Examples

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points()
```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_jitter()
```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm()
```

Changing arguments

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_jitter(jitter_width = 1)
```

```
animals |>
  tidyplot(x = weight, y = size) |>
  add_data_points(white_border = TRUE)
```

```
animals |>
  tidyplot(x = weight, y = size) |>
  add_data_points(alpha = 0.4)
```

Rasterization

```
animals |>
  tidyplot(x = weight, y = size) |>
  add_data_points(rasterize = TRUE, rasterize_dpi = 50)

# Data subsetting
animals |>
  tidyplot(x = weight, y = size) |>
  add_data_points() |>
  add_data_points(data = filter_rows(size > 300), color = "red")
```

add_ellipse*Add ellipse*

Description

Add ellipse

Usage

```
add_ellipse(plot, ...)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
...	Arguments passed on to ggplot2::stat_ellipse().

Value

A tidyplot object.

Examples

```
pca |>
  tidyplot(x = pc1, y = pc2, color = group) |>
  add_data_points() |>
  add_ellipse()
```

```
pca |>
  tidyplot(x = pc1, y = pc2, color = group) |>
  add_data_points() |>
  add_ellipse(level = 0.75)
```

```
pca |>
  tidyplot(x = pc1, y = pc2, color = group) |>
  add_data_points() |>
  add_ellipse(type = "norm")
```

`add_heatmap`*Add heatmap*

Description

Add heatmap

Usage

```
add_heatmap(  
  plot,  
  scale = c("none", "row", "column"),  
  rotate_labels = 90,  
  rasterize = FALSE,  
  rasterize_dpi = 300,  
  ...  
)
```

Arguments

<code>plot</code>	A tidyplot generated with the function <code>tidyplot()</code> .
<code>scale</code>	Whether to compute row z scores for "row" or "column". Defaults to "none".
<code>rotate_labels</code>	Degree to rotate the x-axis labels. Defaults to 90.
<code>rasterize</code>	If FALSE (the default) the layer will be constructed of vector shapes. If TRUE the layer will be rasterized to a pixel image. This can be useful when plotting many individual objects (1,000 or more) compromises the performance of the generated PDF file.
<code>rasterize_dpi</code>	The resolution in dots per inch (dpi) used for rastering the layer if <code>rasterize</code> is TRUE. The default is 300 dpi.
<code>...</code>	Arguments passed on to the geom function.

Details

- `add_heatmap()` supports rasterization. See examples and [Advanced plotting](#).

Value

A tidyplot object.

Examples

```
climate |>  
  tidyplot(x = month, y = year, color = max_temperature) |>  
  add_heatmap()  
  
# Calculate row-wise z score
```

```

climate |>
  tidyplot(x = month, y = year, color = max_temperature) |>
  add_heatmap(scale = "row")

# Calculate column-wise z score
climate |>
  tidyplot(x = month, y = year, color = max_temperature) |>
  add_heatmap(scale = "column")

# Rasterize heatmap
climate |>
  tidyplot(x = month, y = year, color = max_temperature) |>
  add_heatmap(rasterize = TRUE, rasterize_dpi = 20)

```

add_histogram

Add histogram

Description

Add histogram

Usage

```
add_histogram(plot, binwidth = NULL, bins = NULL, ...)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
binwidth	The width of the bins. Can be specified as a numeric value or as a function that takes x after scale transformation as input and returns a single numeric value. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in bins, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data. The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.
bins	Number of bins. Overridden by binwidth. Defaults to 30.
...	Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```
energy |>
  tidyplot(x = energy) |>
  add_histogram()
```

```
energy |>
  tidyplot(x = energy, color = energy_type) |>
  add_histogram()
```

 add_line

Add line or area

Description

add_line() and add_area() connect individual data points, which is rarely needed. In most cases, you are probably looking for add_sum_line(), add_mean_line(), add_sum_area() or add_mean_area().

Usage

```
add_line(
  plot,
  group,
  dodge_width = NULL,
  linewidth = 0.25,
  preserve = "total",
  ...
)
```

```
add_area(
  plot,
  group,
  dodge_width = NULL,
  linewidth = 0.25,
  alpha = 0.4,
  preserve = "total",
  ...
)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
group	Variable in the dataset to be used for grouping.
dodge_width	For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.
linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.

preserve	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
...	Arguments passed on to the geom function.
alpha	A number between 0 and 1 for the opacity of an object. A value of 0 is completely transparent, 1 is completely opaque.

Value

A tidyplot object.

Examples

```
# Paired data points
study |>
  tidyplot(x = treatment, y = score, color = group) |>
  reorder_x_axis_labels("A", "C", "B", "D") |>
  add_data_points() |>
  add_line(group = participant, color = "grey")

study |>
  tidyplot(x = treatment, y = score) |>
  reorder_x_axis_labels("A", "C", "B", "D") |>
  add_data_points() |>
  add_area(group = participant)
```

add_mean_bar	<i>Add mean</i>
--------------	-----------------

Description

Add mean

Usage

```
add_mean_bar(
  plot,
  dodge_width = NULL,
  width = 0.6,
  saturation = 1,
  preserve = "total",
  ...
)

add_mean_dash(
  plot,
  dodge_width = NULL,
  width = 0.6,
```

```

    linewidth = 0.25,
    preserve = "total",
    ...
)

add_mean_dot(plot, dodge_width = NULL, size = 2, preserve = "total", ...)

add_mean_value(
  plot,
  dodge_width = NULL,
  accuracy = 0.1,
  scale_cut = NULL,
  fontsize = 7,
  extra_padding = 0.15,
  vjust = NULL,
  hjust = NULL,
  preserve = "total",
  ...
)

add_mean_line(
  plot,
  group,
  dodge_width = NULL,
  linewidth = 0.25,
  preserve = "total",
  ...
)

add_mean_area(
  plot,
  group,
  dodge_width = NULL,
  linewidth = 0.25,
  preserve = "total",
  ...
)

```

Arguments

plot	A tidyplot generated with the function <code>tidyplot()</code> .
dodge_width	For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.
width	Horizontal width of the plotted object (bar, error bar, boxplot, violin plot, etc). Typical values range between 0 and 1.
saturation	A number between 0 and 1 for the color saturation of an object. A value of 0 is completely desaturated (white), 1 is the original color.

preserve	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
...	Arguments passed on to the geom function.
linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.
size	A number representing the size of the plot symbol. Typical values range between 1 and 3.
accuracy	A number to round to. Use (e.g.) 0.01 to show 2 decimal places of precision. If NULL, the default, uses a heuristic that should ensure breaks have the minimum number of digits needed to show the difference between adjacent values. Applied to rescaled data.
scale_cut	Scale cut function to be applied. See <code>scales::cut_short_scale()</code> and friends.
fontsize	Font size in points. Defaults to 7.
extra_padding	Extra padding to create space for the value label.
vjust	Vertical position adjustment of the value label.
hjust	Horizontal position adjustment of the value label.
group	Variable in the dataset to be used for grouping.

Value

A `tidyplot` object.

Examples

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar()
```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_dash()
```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_dot()
```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_value()
```

```
study |>
  tidyplot(x = treatment, y = score) |>
  add_mean_line()
```

```
study |>
  tidyplot(x = treatment, y = score) |>
  add_mean_area()
```

```
# Combination
```

```
study |>
  tidyplot(x = treatment, y = score) |>
  add_mean_bar(alpha = 0.4) |>
  add_mean_dash() |>
  add_mean_dot() |>
  add_mean_value() |>
  add_mean_line()

# Changing arguments: alpha
# Makes objects transparent
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  theme_minimal_y() |>
  add_mean_bar(alpha = 0.4)

# Changing arguments: saturation
# Reduces fill color saturation without making the object transparent
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  theme_minimal_y() |>
  add_mean_bar(saturation = 0.3)

# Changing arguments: accuracy
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_value(accuracy = 0.01)

# Changing arguments: fontsize
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_value(fontsize = 10)

# Changing arguments: color
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_value(color = "black")
```

add_median_bar

Add median

Description

Add median

Usage

```
add_median_bar(
  plot,
  dodge_width = NULL,
```

```
    width = 0.6,
    saturation = 1,
    preserve = "total",
    ...
)

add_median_dash(
  plot,
  dodge_width = NULL,
  width = 0.6,
  linewidth = 0.25,
  preserve = "total",
  ...
)

add_median_dot(plot, dodge_width = NULL, size = 2, preserve = "total", ...)

add_median_value(
  plot,
  dodge_width = NULL,
  accuracy = 0.1,
  scale_cut = NULL,
  fontsize = 7,
  extra_padding = 0.15,
  vjust = NULL,
  hjust = NULL,
  preserve = "total",
  ...
)

add_median_line(
  plot,
  group,
  dodge_width = NULL,
  linewidth = 0.25,
  preserve = "total",
  ...
)

add_median_area(
  plot,
  group,
  dodge_width = NULL,
  linewidth = 0.25,
  preserve = "total",
  ...
)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
dodge_width	For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.
width	Horizontal width of the plotted object (bar, error bar, boxplot, violin plot, etc). Typical values range between 0 and 1.
saturation	A number between 0 and 1 for the color saturation of an object. A value of 0 is completely desaturated (white), 1 is the original color.
preserve	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
...	Arguments passed on to the geom function.
linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.
size	A number representing the size of the plot symbol. Typical values range between 1 and 3.
accuracy	A number to round to. Use (e.g.) 0.01 to show 2 decimal places of precision. If NULL, the default, uses a heuristic that should ensure breaks have the minimum number of digits needed to show the difference between adjacent values. Applied to rescaled data.
scale_cut	Scale cut function to be applied. See scales::cut_short_scale() and friends.
fontsize	Font size in points. Defaults to 7.
extra_padding	Extra padding to create space for the value label.
vjust	Vertical position adjustment of the value label.
hjust	Horizontal position adjustment of the value label.
group	Variable in the dataset to be used for grouping.

Value

A tidyplot object.

Examples

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_median_bar()
```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_median_dash()
```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_median_dot()
```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
```

```
    add_median_value()

study |>
  tidyplot(x = treatment, y = score) |>
  add_median_line()

study |>
  tidyplot(x = treatment, y = score) |>
  add_median_area()

# Combination
study |>
  tidyplot(x = treatment, y = score) |>
  add_median_bar(alpha = 0.4) |>
  add_median_dash() |>
  add_median_dot() |>
  add_median_value() |>
  add_median_line()

# Changing arguments: alpha
# Makes objects transparent
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  theme_minimal_y() |>
  add_median_bar(alpha = 0.4)

# Changing arguments: saturation
# Reduces fill color saturation without making the object transparent
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  theme_minimal_y() |>
  add_median_bar(saturation = 0.3)

# Changing arguments: accuracy
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_median_value(accuracy = 0.01)

# Changing arguments: fontsize
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_median_value(fontsize = 10)

# Changing arguments: color
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_median_value(color = "black")
```

Description

Add pie or donut chart

Usage

```
add_pie(plot, width = 1, reverse = FALSE, ...)
```

```
add_donut(plot, width = 1, reverse = FALSE, ...)
```

Arguments

<code>plot</code>	A tidyplot generated with the function <code>tidyplot()</code> .
<code>width</code>	Width of the donut ring.
<code>reverse</code>	Whether the order should be reversed or not. Defaults to <code>FALSE</code> , meaning not reversed.
<code>...</code>	Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```
# for a `count` only provide `color`
# `count` of the data points in each `energy_type` category
energy |>
  tidyplot(color = energy_type) |>
  add_pie()

energy |>
  tidyplot(color = energy_type) |>
  add_donut()

energy |>
  tidyplot(color = energy_type) |>
  add_donut(width = 0.5)

# for a `sum` provide `color` and `y`
# `sum` of `energy` in each `energy_type` category
energy |>
  tidyplot(y = energy, color = energy_type) |>
  add_pie()

energy |>
  tidyplot(y = energy, color = energy_type) |>
  add_donut()

energy |>
  tidyplot(y = energy, color = energy_type) |>
  add_donut(width = 0.5)
```

add_reference_lines *Add reference lines*

Description

Add reference lines

Usage

```
add_reference_lines(  
  plot,  
  x = NULL,  
  y = NULL,  
  linetype = "dashed",  
  linewidth = 0.25,  
  ...  
)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
x	Numeric values where the reference lines should meet the x-axis. For example, x = 4 or x = c(2, 3, 4).
y	Numeric values where the reference lines should meet the y-axis. For example, y = 4 or y = c(2, 3, 4).
linetype	Either an integer (0-6) or a name (0 = blank, 1 = solid, 2 = dashed, 3 = dotted, 4 = dotdash, 5 = longdash, 6 = twodash).
linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.
...	Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```
animals |>  
  tidyplot(x = weight, y = speed) |>  
  add_reference_lines(x = 4000, y = c(100, 200)) |>  
  add_data_points()  
  
animals |>  
  tidyplot(x = weight, y = speed) |>  
  add_reference_lines(x = 4000, y = c(100, 200), linetype = "dotdash") |>  
  add_data_points()
```

add_sem_errorbar *Add error bar*

Description

- add_sem_errorbar() adds the standard error of mean.
- add_range_errorbar() adds the range from smallest to largest value.
- add_sd_errorbar() adds the standard deviation.
- add_ci95_errorbar() adds the 95% confidence interval.

Usage

```
add_sem_errorbar(  
  plot,  
  dodge_width = NULL,  
  width = 0.4,  
  linewidth = 0.25,  
  preserve = "total",  
  ...  
)
```

```
add_range_errorbar(  
  plot,  
  dodge_width = NULL,  
  width = 0.4,  
  linewidth = 0.25,  
  preserve = "total",  
  ...  
)
```

```
add_sd_errorbar(  
  plot,  
  dodge_width = NULL,  
  width = 0.4,  
  linewidth = 0.25,  
  preserve = "total",  
  ...  
)
```

```
add_ci95_errorbar(  
  plot,  
  dodge_width = NULL,  
  width = 0.4,  
  linewidth = 0.25,  
  preserve = "total",  
  ...  
)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
dodge_width	For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.
width	Width of the error bar.
linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.
preserve	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
...	Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```
# Standard error of the mean
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_sem_errorbar()

# Range from minimum to maximum value
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_range_errorbar()

# Standard deviation
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_sd_errorbar()

# 95% confidence interval
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_ci95_errorbar()

# Changing arguments: error bar width
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_sem_errorbar(width = 0.8)

# Changing arguments: error bar line width
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
```

```
add_sem_errorbar(linewidth = 1)
```

```
add_sem_ribbon          Add ribbon
```

Description

- `add_sem_ribbon()` adds the standard error of mean.
- `add_range_ribbon()` adds the range from smallest to largest value.
- `add_sd_ribbon()` adds the standard deviation.
- `add_ci95_ribbon()` adds the 95% confidence interval.

Usage

```
add_sem_ribbon(plot, dodge_width = NULL, alpha = 0.4, color = NA, ...)
```

```
add_range_ribbon(plot, dodge_width = NULL, alpha = 0.4, color = NA, ...)
```

```
add_sd_ribbon(plot, dodge_width = NULL, alpha = 0.4, color = NA, ...)
```

```
add_ci95_ribbon(plot, dodge_width = NULL, alpha = 0.4, color = NA, ...)
```

Arguments

<code>plot</code>	A tidyplot generated with the function <code>tidyplot()</code> .
<code>dodge_width</code>	For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.
<code>alpha</code>	A number between 0 and 1 for the opacity of an object. A value of 0 is completely transparent, 1 is completely opaque.
<code>color</code>	A hex color for the stroke color. For example, "#FFFFFF" for white.
<code>...</code>	Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```
# Standard error of the mean
time_course |>
  tidyplot(x = day, y = score, color = treatment) |>
  add_mean_line() |>
  add_sem_ribbon()

# Range from minimum to maximum value
```

```
time_course |>
  tidyplot(x = day, y = score, color = treatment) |>
  add_mean_line() |>
  add_range_ribbon()

# Standard deviation
time_course |>
  tidyplot(x = day, y = score, color = treatment) |>
  add_mean_line() |>
  add_sd_ribbon()

# 95% confidence interval
time_course |>
  tidyplot(x = day, y = score, color = treatment) |>
  add_mean_line() |>
  add_ci95_ribbon()

# Changing arguments: alpha
time_course |>
  tidyplot(x = day, y = score, color = treatment) |>
  add_mean_line() |>
  add_sem_ribbon(alpha = 0.7)
```

add_sum_bar

Add sum

Description

Add sum

Usage

```
add_sum_bar(
  plot,
  dodge_width = NULL,
  width = 0.6,
  saturation = 1,
  preserve = "total",
  ...
)
```

```
add_sum_dash(
  plot,
  dodge_width = NULL,
  width = 0.6,
  linewidth = 0.25,
  preserve = "total",
  ...
)
```

```

)

add_sum_dot(plot, dodge_width = NULL, size = 2, preserve = "total", ...)

add_sum_value(
  plot,
  dodge_width = NULL,
  accuracy = 0.1,
  scale_cut = NULL,
  fontsize = 7,
  extra_padding = 0.15,
  vjust = NULL,
  hjust = NULL,
  preserve = "total",
  ...
)

add_sum_line(
  plot,
  group,
  dodge_width = NULL,
  linewidth = 0.25,
  preserve = "total",
  ...
)

add_sum_area(
  plot,
  group,
  dodge_width = NULL,
  linewidth = 0.25,
  preserve = "total",
  ...
)

```

Arguments

plot	A tidyplot generated with the function <code>tidyplot()</code> .
dodge_width	For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.
width	Horizontal width of the plotted object (bar, error bar, boxplot, violin plot, etc). Typical values range between 0 and 1.
saturation	A number between 0 and 1 for the color saturation of an object. A value of 0 is completely desaturated (white), 1 is the original color.
preserve	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
...	Arguments passed on to the geom function.

linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.
size	A number representing the size of the plot symbol. Typical values range between 1 and 3.
accuracy	A number to round to. Use (e.g.) 0.01 to show 2 decimal places of precision. If NULL, the default, uses a heuristic that should ensure breaks have the minimum number of digits needed to show the difference between adjacent values. Applied to rescaled data.
scale_cut	Scale cut function to be applied. See <code>scales::cut_short_scale()</code> and friends.
fontsize	Font size in points. Defaults to 7.
extra_padding	Extra padding to create space for the value label.
vjust	Vertical position adjustment of the value label.
hjust	Horizontal position adjustment of the value label.
group	Variable in the dataset to be used for grouping.

Value

A tidyplot object.

Examples

```
spendings |>
  tidyplot(x = category, y = amount, color = category) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_bar()
```

```
spendings |>
  tidyplot(x = category, y = amount, color = category) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_dash()
```

```
spendings |>
  tidyplot(x = category, y = amount, color = category) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_dot()
```

```
spendings |>
  tidyplot(x = category, y = amount, color = category) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_value()
```

```
spendings |>
  tidyplot(x = category, y = amount) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_line()
```

```
spendings |>
  tidyplot(x = category, y = amount) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_area()
```

```
# Combination
spendings |>
  tidyplot(x = category, y = amount) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_median_bar(alpha = 0.4) |>
  add_median_dash() |>
  add_median_dot() |>
  add_median_value() |>
  add_median_line()

# Changing arguments: alpha
# Makes objects transparent
spendings |>
  tidyplot(x = category, y = amount, color = category) |>
  theme_minimal_y() |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_bar(alpha = 0.4)

# Changing arguments: saturation
# Reduces fill color saturation without making the object transparent
spendings |>
  tidyplot(x = category, y = amount, color = category) |>
  theme_minimal_y() |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_bar(saturation = 0.3)

# Changing arguments: accuracy
spendings |>
  tidyplot(x = category, y = amount, color = category) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_value(accuracy = 1)

# Changing arguments: fontsize
spendings |>
  tidyplot(x = category, y = amount, color = category) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_value(fontsize = 10)

# Changing arguments: color
spendings |>
  tidyplot(x = category, y = amount, color = category) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_value(color = "black")

# Changing arguments: extra_padding
spendings |>
  tidyplot(x = category, y = amount, color = category) |>
  adjust_x_axis(rotate_labels = TRUE) |>
  add_sum_value(extra_padding = 0.5)
```

add_test_pvalue	<i>Add statistical test</i>
-----------------	-----------------------------

Description

Add statistical test

Usage

```
add_test_pvalue(  
  plot,  
  padding_top = 0.15,  
  method = "t_test",  
  p.adjust.method = "none",  
  ref.group = NULL,  
  comparisons = NULL,  
  paired_by = NULL,  
  label = "{format_p_value(p.adj, 0.0001)}",  
  label.size = 7/ggplot2::.pt,  
  step.increase = 0.15,  
  vjust = -0.25,  
  bracket.nudge.y = 0.1,  
  hide.ns = FALSE,  
  p.adjust.by = "panel",  
  symnum.args = list(cutpoints = c(0, 0.001, 0.01, 0.05, Inf), symbols = c("***", "**",  
    "*", "ns")),  
  color = plot$tidyplot$ink,  
  hide_info = FALSE,  
  ...  
)  
  
add_test_asterisks(  
  plot,  
  padding_top = 0.1,  
  method = "t_test",  
  p.adjust.method = "none",  
  ref.group = NULL,  
  comparisons = NULL,  
  paired_by = NULL,  
  label = "p.adj.signif",  
  label.size = 10/ggplot2::.pt,  
  step.increase = 0.2,  
  vjust = 0.3,  
  bracket.nudge.y = 0.15,  
  hide.ns = TRUE,  
  p.adjust.by = "panel",  
  symnum.args = list(cutpoints = c(0, 0.001, 0.01, 0.05, Inf), symbols = c("***", "**",
```

```

    "*", "ns")),
  color = plot$tidyplot$ink,
  hide_info = FALSE,
  ...
)

```

Arguments

plot	A tidyplot generated with the function tidyplot().
padding_top	Extra padding above the data points to accommodate the statistical comparisons.
method	a character string indicating which method to be used for pairwise comparisons. Default is "t_test". Allowed methods include pairwise comparisons methods implemented in the rstatix R package. These methods are: "wilcox_test", "t_test", "sign_test", "dunn_test", "emmeans_test", "tukey_hsd", "games_howell_test".
p.adjust.method	method for adjusting p values (see p.adjust). Has impact only in a situation, where multiple pairwise tests are performed; or when there are multiple grouping variables. Ignored when the specified method is "tukey_hsd" or "games_howell_test" because they come with internal p adjustment method. Allowed values include "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". If you don't want to adjust the p value (not recommended), use p.adjust.method = "none".
ref.group	<p>a character string or a numeric value specifying the reference group. If specified, for a given grouping variable, each of the group levels will be compared to the reference group (i.e. control group).</p> <p>ref.group can be also "all". In this case, each of the grouping variable levels is compared to all (i.e. basemean).</p> <p>Allowed values can be:</p> <ul style="list-style-type: none"> • numeric value: specifying the rank of the reference group. For example, use ref.group = 1 when the first group is the reference; use ref.group = 2 when the second group is the reference, and so on. This works for all situations, including i) when comparisons are performed between x-axis groups and ii) when comparisons are performed between legend groups. • character value: For example, you can use ref.group = "ctrl" instead of using the numeric rank value of the "ctrl" group. • "all": In this case, each of the grouping variable levels is compared to all (i.e. basemean).
comparisons	A list of length-2 vectors. The entries in the vector are 2 integers that correspond to the index of the groups of interest, to be compared.
paired_by	Variable to be used for paired analysis.
label	<p>character string specifying label. Can be:</p> <ul style="list-style-type: none"> • the column containing the label (e.g.: label = "p" or label = "p.adj"), where p is the p-value. Other possible values are "p.signif", "p.adj.signif", "p.format", "p.adj.format".

- an expression that can be formatted by the `glue()` package. For example, when specifying `label = "Wilcoxon, $p = \{p\}$ "`, the expression `{p}` will be replaced by its value.
- a combination of plotmath expressions and glue expressions. You may want some of the statistical parameter in italic; for example: `label = "Wilcoxon, italic(p)={p}"`

<code>label.size</code>	change the size of the label text
<code>step.increase</code>	numeric vector with the increase in fraction of total height for every additional comparison to minimize overlap.
<code>vjust</code>	move the text up or down relative to the bracket.
<code>bracket.nudge.y</code>	Vertical adjustment to nudge brackets by (in fraction of the total height). Useful to move up or move down the bracket. If positive value, brackets will be moved up; if negative value, brackets are moved down.
<code>hide.ns</code>	can be logical value (TRUE or FALSE) or a character vector ("p.adj" or "p").
<code>p.adjust.by</code>	possible value is one of <code>c("group", "panel")</code> . Default is "group": for a grouped data, if pairwise test is performed, then the p-values are adjusted for each group level independently. P-values are adjusted by panel when <code>p.adjust.by = "panel"</code> .
<code>symnum.args</code>	a list of arguments to pass to the function <code>symnum</code> for symbolic number coding of p-values. For example, <code>symnum.args = list(cutpoints = c(0, 0.0001, 0.001, 0.01, 0.05, Inf), symbols = c("****", "***", "**", "*", "ns"))</code> . In other words, we use the following convention for symbols indicating statistical significance: <ul style="list-style-type: none"> • ns: $p > 0.05$ • *: $p \leq 0.05$ • **: $p \leq 0.01$ • ***: $p \leq 0.001$ • ****: $p \leq 0.0001$
<code>color</code>	A hex color for the stroke color. For example, "#FFFFFF" for white.
<code>hide_info</code>	Whether to hide details about the statistical testing as caption. Defaults to FALSE.
<code>...</code>	Arguments passed on to <code>ggpubr::geom_pwc()</code> .

Details

- `add_test_pvalue()` and `add_test_asterisks()` use `ggpubr::geom_pwc()`. Check there for additional arguments.
- Known limitation: `add_test_pvalue()` and `add_test_asterisks()` expect a discrete variable on the x-axis and a continuous variable on the y-axis. To produce horizontal plots, use `flip_plot()`.

Value

A tidyplot object.

Examples

```
# Add p value
study |>
  tidyplot(x = dose, y = score, color = group) |>
  add_mean_dash() |>
  add_sem_errorbar() |>
  add_data_points() |>
  add_test_pvalue()

# Add asterisks
study |>
  tidyplot(x = dose, y = score, color = group) |>
  add_mean_dash() |>
  add_sem_errorbar() |>
  add_data_points() |>
  add_test_asterisks()

# Change stat method
study |>
  tidyplot(x = dose, y = score, color = group) |>
  add_mean_dash() |>
  add_sem_errorbar() |>
  add_data_points() |>
  add_test_pvalue(method = "wilcoxon")

# Change p.adjust method
study |>
  tidyplot(x = dose, y = score, color = group) |>
  add_mean_dash() |>
  add_sem_errorbar() |>
  add_data_points() |>
  add_test_pvalue(p.adjust.method = "bonferroni")

# Define reference group to test against
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_dash() |>
  add_sem_errorbar() |>
  add_data_points() |>
  add_test_asterisks(ref.group = 1)

# Define selected comparisons
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_dash() |>
  add_sem_errorbar() |>
  add_data_points() |>
  add_test_pvalue(comparisons = list(c(1,3),c(2,4)))

# Paired analysis
x <- c(2.3, 4.5, 6.3, 3.4, 7.8, 6.7)
df <- data.frame(
```

```

x = c(x, x + c(0.8, 0.75)),
group = paste0("g", rep(c(1, 2), each = 6)),
batch = paste0("b", c(1:6, 1:6)),
shuffle = paste0("c", c(1:6, 6:1))
)

df |>
  tidyplot(group, x, color = group) |>
  add_boxplot() |>
  add_data_points() |>
  add_test_pvalue(paired_by = shuffle) |>
  add_line(group = shuffle, color = "black")

df |>
  tidyplot(group, x, color = group) |>
  add_boxplot() |>
  add_data_points() |>
  add_test_pvalue(paired_by = batch) |>
  add_line(group = batch, color = "black")

# hide non-significant p values
gene_expression |>
  # filter to one gene
  dplyr::filter(external_gene_name == "Apol6") |>
  # start plotting
  tidyplot(x = condition, y = expression, color = sample_type) |>
  add_mean_dash() |>
  add_sem_errorbar() |>
  add_data_points() |>
  add_test_pvalue(hide.ns = TRUE)

# Flip plot
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_dash() |>
  add_sem_errorbar() |>
  add_data_points() |>
  add_test_asterisks(comparisons = list(c(1,4),c(2,3))) |>
  flip_plot()

# Adjust top padding for statistical comparisons
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_dash() |>
  add_sem_errorbar() |>
  add_data_points() |>
  add_test_pvalue(padding_top = 0.08)

# Hide stats information
study |>
  tidyplot(x = dose, y = score, color = group) |>
  add_mean_dash() |>
  add_sem_errorbar() |>

```

```
add_data_points() |>
add_test_pvalue(hide_info = TRUE)
```

add_title	<i>Add plot title or caption</i>
-----------	----------------------------------

Description

Add plot title or caption

Usage

```
add_title(plot, title = ggplot2::waiver())

add_caption(plot, caption = ggplot2::waiver())
```

Arguments

plot	A tidyplot generated with the function tidyplot().
title	Title of the plot.
caption	Caption of the plot.

Details

- add_title() and add_caption() support [plotmath expressions](#) to include special characters. See examples and [Advanced plotting](#).

Value

A tidyplot object.

Examples

```
study |>
  tidyplot(x = treatment, y = score) |>
  add_data_points_beeswarm() |>
  add_title("This is my title")

study |>
  tidyplot(x = treatment, y = score) |>
  add_data_points_beeswarm() |>
  add_caption("This is the fine print in the caption")

# Plotmath expression
study |>
  tidyplot(x = treatment, y = score) |>
  add_data_points_beeswarm() |>
  add_title("$H[2]*0~and~E==m*c^{2}$")
```

add_violin	<i>Add violin plot</i>
------------	------------------------

Description

Add violin plot

Usage

```
add_violin(  
  plot,  
  dodge_width = NULL,  
  alpha = 0.3,  
  saturation = 1,  
  trim = TRUE,  
  linewidth = 0.25,  
  scale = "width",  
  ...  
)
```

Arguments

plot	A tidyplot generated with the function <code>tidyplot()</code> .
dodge_width	For adjusting the distance between grouped objects. Defaults to 0.8 for plots with at least one discrete axis and 0 for plots with two continuous axes.
alpha	A number between 0 and 1 for the opacity of an object. A value of 0 is completely transparent, 1 is completely opaque.
saturation	A number between 0 and 1 for the color saturation of an object. A value of 0 is completely desaturated (white), 1 is the original color.
trim	If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, don't trim the tails.
linewidth	Thickness of the line in points (pt). Typical values range between 0.25 and 1.
scale	if "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.
...	Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_violin()

# Changing arguments:
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_violin(saturation = 0.6)

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_violin(draw_quantiles = c(0.25, 0.5, 0.75))

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_violin(trim = FALSE)

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_violin(linewidth = 1)

```

`adjust_colors`*Adjust colors*

Description

Adjust colors

Usage

```

adjust_colors(
  plot,
  new_colors = NULL,
  saturation = 1,
  labels = tidyplot_parse_labels(),
  downsample = c("evenly", "first", "last", "middle"),
  ...
)

```

Arguments

<code>plot</code>	A tidyplot generated with the function <code>tidyplot()</code> .
<code>new_colors</code>	A character vector of new hex colors to use. Can be a named character vector of hex colors to assign certain data labels to specific colors.
<code>saturation</code>	A number between 0 and 1 for the color saturation of an object. A value of 0 is completely desaturated (white), 1 is the original color.

labels	<p>One of the options below. Please note that when labels is a vector, it is highly recommended to also set the breaks argument as a vector to protect against unintended mismatches.</p> <ul style="list-style-type: none"> • NULL for no labels • waiver() for the default labels computed by the transformation object • A character vector giving labels (must be same length as breaks) • An expression vector (must be the same length as breaks). See ?plotmath for details. • A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.
downsample	If too many colors are provided, whether to downsample evenly, or use the first, the last or the middle colors of the color vector. Defaults to evenly.
...	Arguments passed on to the ggplot2 scale function.

Value

A tidyplot object.

See Also

[colors_discrete_friendly\(\)](#), [colors_continuous_viridis\(\)](#), [colors_diverging_blue2brown\(\)](#), and [new_color_scheme\(\)](#)

Examples

```
# Plot without adjustments
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Provide hex colors
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_colors(new_colors = c("#644296", "#F08533", "#3B78B0", "#D1352C"))

# Provide discrete color scheme
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_colors(new_colors = colors_discrete_seaside)

# Provide named vector
study |>
```

```

tidyplot(x = treatment, y = score, color = treatment) |>
add_data_points() |>
add_mean_bar(alpha = 0.4) |>
add_sem_errorbar() |>
adjust_colors(new_colors = c(
  "A" = "pink",
  "B" = "purple",
  "C" = "grey",
  "D" = "blue"))

# Provide continuous color scheme
climate |>
tidyplot(x = month, y = year, color = max_temperature) |>
add_heatmap() |>
adjust_colors(new_colors = colors_continuous_turbo)

```

adjust_font

Adjust font

Description

Adjust font

Usage

```
adjust_font(plot, fontsize = 7, family = NULL, face = NULL)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
fontsize	Font size in points. Defaults to 7.
family	The typeface to use. The validity of this value will depend on the graphics device being used for rendering the plot. See the systemfonts vignette for guidance on the best way to access fonts installed on your computer. The values "sans", "serif", and "mono" should always be valid and will select the default typeface for the respective styles. However, what is considered default is dependant on the graphics device and the operating system.
face	Font face ("plain", "italic", "bold", "bold.italic")

Value

A tidyplot object.

Examples

```
# Plot without adjustments
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Increase font size
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_font(fontsize = 16)

# Change font family
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_font(family = "mono")

# Change font face
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_font(face = "bold")
```

adjust_legend_title *Adjust legend*

Description

Adjust legend

Usage

```
adjust_legend_title(
  plot,
  title = ggplot2::waiver(),
  fontsize = NULL,
  family = NULL,
  face = NULL,
  color = NULL,
```

```

    ...
  )

adjust_legend_position(plot, position = "right")

```

Arguments

plot	A tidyplot generated with the function tidyplot().
title	Legend title.
fontsize	Font size in points. Defaults to 7.
family	The typeface to use. The validity of this value will depend on the graphics device being used for rendering the plot. See the systemfonts vignette for guidance on the best way to access fonts installed on your computer. The values "sans", "serif", and "mono" should always be valid and will select the default typeface for the respective styles. However, what is considered default is dependant on the graphics device and the operating system.
face	Font face ("plain", "italic", "bold", "bold.italic")
color	A hex color for the stroke color. For example, "#FFFFFF" for white.
...	Arguments passed on to ggplot2::element_text().
position	The position of the legend. Can be one of c("right", "left", "bottom", "top", "none"). Defaults to "right".

Details

- The title argument of adjust_legend_title() supports [plotmath expressions](#) to include special characters. See examples and [Advanced plotting](#).

Value

A tidyplot object.

Examples

```

# Plot without adjustments
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# New title
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_legend_title("My new legend title")

# New title with plotmath expression

```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_legend_title("$E=mc^2$")

# Alternative legend positions
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_legend_position("left")

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_legend_position("top")

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_legend_position("bottom")

# `position = "none"` hides the legend
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_legend_position("none")
```

adjust_padding

Adjust plot area padding

Description

Adjust plot area padding

Usage

```
adjust_padding(  
  plot,  
  top = NA,
```

```

    right = NA,
    bottom = NA,
    left = NA,
    all = NA,
    force_continuous = FALSE,
    ...
  )

```

Arguments

<code>plot</code>	A tidyplot generated with the function <code>tidyplot()</code> .
<code>top</code>	Extra space between the data points and the top. Defaults to NA, which does not change the padding.
<code>right</code>	Extra space between the data points and the right. Defaults to NA, which does not change the padding.
<code>bottom</code>	Extra space between the data points and the bottom. Defaults to NA, which does not change the padding.
<code>left</code>	Extra space between the data points and the left. Defaults to NA, which does not change the padding.
<code>all</code>	Extra space around the data pointst. Overwrites <code>top</code> , <code>right</code> , <code>bottom</code> , <code>left</code> if set. Defaults to NA, which does not change the padding.
<code>force_continuous</code>	Whether to force the axis to be continuous. Defaults to FALSE.
<code>...</code>	Arguments passed on to the <code>geom</code> function.

Value

A tidyplot object.

Examples

```

# Plot without adjustments
animals |>
  tidyplot(x = weight, y = size, color = family) |>
  add_data_points() |>
  adjust_padding()

# Increase plot area padding
animals |>
  tidyplot(x = weight, y = size, color = family) |>
  add_data_points() |>
  adjust_padding(all = 0.2)

animals |>
  tidyplot(x = weight, y = size, color = family) |>
  add_data_points() |>
  adjust_padding(top = 0.8)

animals |>

```

```

tidyplot(x = weight, y = size, color = family) |>
add_data_points() |>
adjust_padding(bottom = 0.8)

animals |>
tidyplot(x = weight, y = size, color = family) |>
add_data_points() |>
adjust_padding(right = 0.8)

animals |>
tidyplot(x = weight, y = size, color = family) |>
add_data_points() |>
adjust_padding(left = 0.8)

```

adjust_size

Adjust plot area size

Description

Adjust plot area size

Usage

```

adjust_size(
  plot,
  width = NULL,
  height = NULL,
  unit = NULL,
  overall_width = NULL,
  overall_height = NULL
)

```

Arguments

plot	A tidyplot generated with the function <code>tidyplot()</code> .
width, height	Dimensions of the plot area. The default (NULL) retrieves the setting from the tidyplots options , which defaults to 50. NA results in taking all available space (ggplot2 default).
unit	Unit of the plot area width and height. The default (NULL) retrieves the setting from the tidyplots options , which defaults to "mm".
overall_width, overall_height	The overall dimensions of a multiplot layout generated with <code>split_plot()</code> .

Value

A tidyplot object.

Examples

```
# Plot without adjustments
study |>
  tidypplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm(shape = 1) |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Resize to 15 x 15 mm
study |>
  tidypplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm(shape = 1) |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_size(width = 15, height = 15)

# Resize to 4 x 4 cm
study |>
  tidypplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm(shape = 1) |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_size(width = 4, height = 4, unit = "cm")

# Remove absolute dimensions and take all available space.
# This is the ggplot2 default.
study |>
  tidypplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm(shape = 1) |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_size(width = NA, height = NA)
```

adjust_theme_details *Adjust theme details*

Description

This function is a wrapper around `ggplot2::theme()`. To use the required theme helper functions `ggplot2::element_blank()`, `ggplot2::element_rect()`, `ggplot2::element_line()`, and `ggplot2::element_text()` you need to either load the `ggplot2` package via `library(ggplot2)` or use the `ggplot2::` prefix as shown above.

Usage

```
adjust_theme_details(plot, ...)
```

Arguments

plot A tidyplot generated with the function tidyplot().
... Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```
study |>  
  tidyplot(x = treatment, y = score, color = treatment) |>  
  add_data_points_beeswarm() |>  
  add_mean_bar(alpha = 0.4) |>  
  adjust_theme_details(plot.background = ggplot2::element_rect(fill = "#FFEBFF"))
```

adjust_title	<i>Adjust titles and caption</i>
--------------	----------------------------------

Description

Adjust titles and caption

Usage

```
adjust_title(  
  plot,  
  title = ggplot2::waiver(),  
  fontsize = NULL,  
  family = NULL,  
  face = NULL,  
  color = NULL,  
  ...  
)
```

```
adjust_x_axis_title(  
  plot,  
  title = ggplot2::waiver(),  
  fontsize = NULL,  
  family = NULL,  
  face = NULL,  
  color = NULL,  
  ...  
)
```

```
adjust_y_axis_title(  
  plot,  
  title = ggplot2::waiver(),  
  fontsize = NULL,  
  family = NULL,  
  face = NULL,  
  color = NULL,  
  ...  
)
```

```

plot,
title = ggplot2::waiver(),
fontsize = NULL,
family = NULL,
face = NULL,
color = NULL,
...
)

adjust_caption(
plot,
caption = ggplot2::waiver(),
fontsize = NULL,
family = NULL,
face = NULL,
color = NULL,
...
)

```

Arguments

plot	A tidyplot generated with the function <code>tidyplot()</code> .
title	Plot or axes title.
fontsize	Font size in points. Defaults to 7.
family	The typeface to use. The validity of this value will depend on the graphics device being used for rendering the plot. See the systemfonts vignette for guidance on the best way to access fonts installed on your computer. The values "sans", "serif", and "mono" should always be valid and will select the default typeface for the respective styles. However, what is considered default is dependant on the graphics device and the operating system.
face	Font face ("plain", "italic", "bold", "bold.italic")
color	A hex color for the stroke color. For example, "#FFFFFF" for white.
...	Arguments passed on to <code>ggplot2::element_text()</code> .
caption	Plot caption.

Details

Adjust the plot title, axis titles and caption

- All functions support [plotmath expressions](#) to include special characters. See examples and [Advanced plotting](#).

Value

A tidyplot object.

Examples

```

# Plot without adjustments
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Adjust description
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_title("This is my fantastic plot title") |>
  adjust_x_axis_title("Treatment group") |>
  adjust_y_axis_title("Disease score") |>
  adjust_legend_title("Legend title") |>
  adjust_caption("Here goes the caption")

# Plotmath expressions
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  adjust_title("$H[2]*0$") |>
  adjust_x_axis_title("$H[2]*0$") |>
  adjust_y_axis_title("$H[2]*0$") |>
  adjust_legend_title("$H[2]*0$") |>
  adjust_caption("$H[2]*0$")

```

adjust_x_axis

Adjust axes

Description

Adjust axes

Usage

```

adjust_x_axis(
  plot,
  title = ggplot2::waiver(),
  breaks = ggplot2::waiver(),
  labels = NULL,
  limits = NULL,
  padding = c(NA, NA),

```

```

    rotate_labels = FALSE,
    transform = "identity",
    cut_short_scale = FALSE,
    force_continuous = FALSE,
    ...
  )

adjust_y_axis(
  plot,
  title = ggplot2::waiver(),
  breaks = ggplot2::waiver(),
  labels = NULL,
  limits = NULL,
  padding = c(NA, NA),
  rotate_labels = FALSE,
  transform = "identity",
  cut_short_scale = FALSE,
  force_continuous = FALSE,
  ...
)

```

Arguments

plot	A tidyplot generated with the function <code>tidyplot()</code> .
title	Axis title.
breaks	One of: <ul style="list-style-type: none"> • <code>NULL</code> for no breaks • <code>waiver()</code> for the default breaks computed by the transformation object • A numeric vector of positions • A function that takes the limits as input and returns breaks as output (e.g., a function returned by <code>scales::extended_breaks()</code>). Note that for position scales, limits are provided after scale expansion. Also accepts rlang lambda function notation.
labels	One of the options below. Please note that when <code>labels</code> is a vector, it is highly recommended to also set the <code>breaks</code> argument as a vector to protect against unintended mismatches. <ul style="list-style-type: none"> • <code>NULL</code> for no labels • <code>waiver()</code> for the default labels computed by the transformation object • A character vector giving labels (must be same length as <code>breaks</code>) • An expression vector (must be the same length as <code>breaks</code>). See <code>?plotmath</code> for details. • A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.
limits	Axis limits. For example, with <code>limits = c(20, 90)</code> the axis starts at 20 and ends at 90.

padding	Extra space between the data points and the axes. Defaults to <code>c(NA, NA)</code> , which does not change the padding.
rotate_labels	Whether to rotate axis labels. If <code>TRUE</code> is set to 45 degrees. You can also provide custom degree values, for example, <code>rotate_labels = 90</code> . Defaults to <code>FALSE</code> .
transform	For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time". A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called <code>transform_<name></code> . If transformations require arguments, you can call them from the scales package, e.g. <code>scales::transform_boxcox(p = 2)</code> . You can create your own transformation with <code>scales::new_transform()</code> .
cut_short_scale	Whether to shorten axis labels using K for thousand, M for million, and so on. Defaults to <code>FALSE</code> .
force_continuous	Whether to force the axis to be continuous. Defaults to <code>FALSE</code> .
...	Arguments passed on to <code>ggplot2</code> scale function.

Details

- The title argument of `adjust_x_axis()` and `adjust_y_axis()` supports **plotmath expressions** to include special characters. See examples and **Advanced plotting**.

Value

A `tidyplot` object.

Examples

```
# Plot without adjustments
animals |>
  tidyplot(x = weight, y = size, color = family) |>
  add_data_points()

# New titles
animals |>
  tidyplot(x = weight, y = size, color = family) |>
  add_data_points() |>
  adjust_x_axis(title = "My new x-axis title") |>
  adjust_y_axis(title = "My new y-axis title")

# New titles with plotmath expressions
animals |>
  tidyplot(x = weight, y = size, color = family) |>
  add_data_points() |>
  adjust_x_axis(title = "$H[2]*0$") |>
  adjust_y_axis(title = "$E==m*c^{2}$")
```

```
# Axes limits
animals |>
  tidyplot(x = weight, y = size, color = family) |>
  add_data_points() |>
  adjust_x_axis(limits = c(-1000, 4000)) |>
  adjust_y_axis(limits = c(-200, 600))

# Rotate labels
animals |>
  tidyplot(x = weight, y = size, color = family) |>
  add_data_points() |>
  adjust_x_axis(rotate_labels = 90) |>
  adjust_y_axis(rotate_labels = 90)

# Increase plot area padding
animals |>
  tidyplot(x = weight, y = size, color = family) |>
  add_data_points() |>
  adjust_x_axis(padding = c(0.2, 0.2)) |>
  adjust_y_axis(padding = c(0.2, 0.2))

# Scale transformation
animals |>
  tidyplot(x = weight, y = size, color = family) |>
  add_data_points() |>
  adjust_x_axis(transform = "log10") |>
  adjust_y_axis(transform = "log2")
```

all_rows

Subset data rows

Description

Subset data rows

Usage

```
all_rows()
```

```
filter_rows(..., .by = NULL)
```

```
max_rows(order_by, n, by = NULL, with_ties = TRUE, na_rm = FALSE)
```

```
min_rows(order_by, n, by = NULL, with_ties = TRUE, na_rm = FALSE)
```

```
first_rows(n, by = NULL)
```

```
last_rows(n, by = NULL)
```

```
sample_rows(n, by = NULL)
```

Arguments

...	<data-masking> Expressions that return a logical value, and are defined in terms of the variables in <code>.data</code> . If multiple expressions are included, they are combined with the <code>&</code> operator. Only rows for which all conditions evaluate to <code>TRUE</code> are kept.
<code>.by, by</code>	[Experimental] <tidy-select> Optionally, a selection of columns to group by for just this operation, functioning as an alternative to <code>group_by()</code> . For details and examples, see <code>?dplyr_by</code> .
<code>order_by</code>	<data-masking> Variable or function of variables to order by. To order by multiple variables, wrap them in a data frame or tibble.
<code>n</code>	The number of rows to select. If not are supplied, <code>n = 1</code> will be used. If <code>n</code> is greater than the number of rows in the group, the result will be silently truncated to the group size. A negative value of <code>n</code> will be subtracted from the group size. For example, <code>n = -2</code> with a group of 5 rows will select $5 - 2 = 3$ rows.
<code>with_ties</code>	Should ties be kept together? The default, <code>TRUE</code> , may return more rows than you request. Use <code>FALSE</code> to ignore ties, and return the first <code>n</code> rows.
<code>na_rm</code>	Should missing values in <code>order_by</code> be removed from the result? If <code>FALSE</code> , <code>NA</code> values are sorted to the end (like in <code>dplyr::arrange()</code>), so they will only be included if there are insufficient non-missing values to reach <code>n</code> .

Value

A function to achieve the desired data subsetting.

Examples

```
# Highlight all animals
animals |>
  tidyplot(x = weight, y = size) |>
  add_data_points() |>
  add_data_points(data = all_rows(),
    color = "red", shape = 1, size = 3)

# Highlight 3 animals with the highest weight
animals |>
  tidyplot(x = weight, y = size) |>
  add_data_points() |>
  add_data_points(data = max_rows(weight, n = 3),
    color = "red", shape = 1, size = 3)

# Highlight 3 animals with the lowest weight
animals |>
```

```
tidyplot(x = weight, y = size) |>
add_data_points() |>
add_data_points(data = min_rows(weight, n = 3),
  color = "red", shape = 1, size = 3)

# Highlight the first 3 animals in the dataset
animals |>
tidyplot(x = weight, y = size) |>
add_data_points() |>
add_data_points(data = first_rows(n = 3),
  color = "red", shape = 1, size = 3)

# Highlight the last 3 animals in the dataset
animals |>
tidyplot(x = weight, y = size) |>
add_data_points() |>
add_data_points(data = last_rows(n = 3),
  color = "red", shape = 1, size = 3)

# Highlight 3 random animals
animals |>
tidyplot(x = weight, y = size) |>
add_data_points() |>
add_data_points(data = sample_rows(n = 3),
  color = "red", shape = 1, size = 3)
```

animals

Animals data

Description

Animals data

Usage

```
animals
```

Format

A data frame.

Source

ChatGPT-3.5, *Caution: The accuracy of the data has not been verified.*

Examples

```
dplyr::glimpse(animals)
```

climate	<i>Climate data</i>
---------	---------------------

Description

Climate data

Usage

```
climate
```

Format

A data frame.

Source

National Oceanic and Atmospheric Administration, Temperature data, weather station Hamburg Fuhlsbüttel, Germany

Examples

```
dplyr::glimpse(climate)
```

colors_continuous_viridis	<i>Continuous color schemes</i>
---------------------------	---------------------------------

Description

For more information about the use of color schemes in tidypLOTS, check out this article: [Color schemes](#)

Usage

```
colors_continuous_viridis
```

```
colors_continuous_magma
```

```
colors_continuous_inferno
```

```
colors_continuous_plasma
```

```
colors_continuous_cividis
```

```
colors_continuous_rocket
```

`colors_continuous_mako`

`colors_continuous_turbo`

`colors_continuous_bluepinkyellow`

Format

An object of class `tidycolor` (inherits from `character`) of length 265.

An object of class `tidycolor` (inherits from `character`) of length 265.

An object of class `tidycolor` (inherits from `character`) of length 265.

An object of class `tidycolor` (inherits from `character`) of length 265.

An object of class `tidycolor` (inherits from `character`) of length 265.

An object of class `tidycolor` (inherits from `character`) of length 265.

An object of class `tidycolor` (inherits from `character`) of length 265.

An object of class `tidycolor` (inherits from `character`) of length 265.

An object of class `tidycolor` (inherits from `character`) of length 11.

Details

Color schemes can be conveniently previewed by using the `print` method of the `tidycolor` class. This will send a `html` preview to the RStudio Viewer pane.

`colors_continuous_viridis`

`colors_continuous_magma`

`colors_continuous_inferno`

`colors_continuous_plasma`

`colors_continuous_cividis`

`colors_continuous_rocket`

`colors_continuous_mako`

`colors_continuous_turbo`

`colors_continuous_bluepinkyellow`

`colors_discrete_friendly`*Discrete color schemes*

Description

For more information about the use of color schemes in tidypLOTS, check out this article: [Color schemes](#)

Usage`colors_discrete_friendly``colors_discrete_seaside``colors_discrete_apple``colors_discrete_friendly_long``colors_discrete_okabeito``colors_discrete_ibm``colors_discrete_metro``colors_discrete_candy``colors_discrete_alger``colors_discrete_rainbow`**Format**

An object of class `tidycolor` (inherits from `character`) of length 6.

An object of class `tidycolor` (inherits from `character`) of length 5.

An object of class `tidycolor` (inherits from `character`) of length 7.

An object of class `tidycolor` (inherits from `character`) of length 7.

An object of class `tidycolor` (inherits from `character`) of length 7.

An object of class `tidycolor` (inherits from `character`) of length 5.

An object of class `tidycolor` (inherits from `character`) of length 5.

An object of class `tidycolor` (inherits from `character`) of length 5.

An object of class `tidycolor` (inherits from `character`) of length 5.

An object of class `tidycolor` (inherits from `character`) of length 9.

Details

The signature theme of tidypLOTS colors_discrete_friendly was adapted from the [Okabe & Ito](#) color palette that was designed to work well for people with color vision deficiency.

Color schemes can be conveniently previewed by using the print method of the tidycolor class. This will send a html preview to the RStudio Viewer pane.

colors_discrete_friendly

colors_discrete_seaside

colors_discrete_apple

colors_discrete_friendly_long

colors_discrete_okabeito

colors_discrete_ibm

colors_discrete_metro

colors_discrete_candy

colors_discrete_alger

colors_discrete_rainbow

colors_diverging_blue2red

Diverging color schemes

Description

For more information about the use of color schemes in tidypLOTS, check out this article: [Color schemes](#)

Usage

colors_diverging_blue2red

colors_diverging_blue2brown

colors_diverging_BuRd

colors_diverging_BuYlRd

colors_diverging_spectral

colors_diverging_icefire

Format

An object of class `tidycolor` (inherits from `character`) of length 17.
An object of class `tidycolor` (inherits from `character`) of length 17.
An object of class `tidycolor` (inherits from `character`) of length 11.
An object of class `tidycolor` (inherits from `character`) of length 11.
An object of class `tidycolor` (inherits from `character`) of length 96.
An object of class `tidycolor` (inherits from `character`) of length 96.

Details

Color schemes can be conveniently previewed by using the `print` method of the `tidycolor` class. This will send a `html` preview to the RStudio Viewer pane.

```
colors_diverging_blue2red  
colors_diverging_blue2brown  
colors_diverging_BuRd  
colors_diverging_BuYlRd  
colors_diverging_spectral  
colors_diverging_icefire
```

dinosaurus	<i>Dinosaurus data</i>
------------	------------------------

Description

Dinosaurus data

Usage

```
dinosaurus
```

Format

A data frame.

Source

ChatGPT-3.5, *Caution: The accuracy of the data has not been verified.*

Examples

```
dplyr::glimpse(dinosaurus)
```

distributions	<i>Distributions data</i>
---------------	---------------------------

Description

Distributions data

Usage

```
distributions
```

Format

A data frame.

Source

tidyplots package

Examples

```
dplyr::glimpse(distributions)
```

energy	<i>Energy data</i>
--------	--------------------

Description

Energy data

Usage

```
energy
```

Format

A data frame.

Source

[Energy-Charts](#), Energy production data, Germany

Examples

```
dplyr::glimpse(energy)
```

energy_week	<i>Energy week data</i>
-------------	-------------------------

Description

Energy week data

Usage

```
energy_week
```

Format

A data frame.

Source

[Energy-Charts](#), Energy production data, Germany

Examples

```
dplyr::glimpse(energy_week)
```

eu_countries	<i>EU countries data</i>
--------------	--------------------------

Description

EU countries data

Usage

```
eu_countries
```

Format

A data frame.

Source

ChatGPT-3.5, *Caution: The accuracy of the data has not been verified.*

Examples

```
dplyr::glimpse(eu_countries)
```

`flip_plot`*Flip x and y-axis*

Description

[Superseded]

This function is superseded because in many cases, `flip_plot()` can easily be replaced by swapping the x and y axis. Some plot components additionally require to set the orientation argument to "y".

Usage

```
flip_plot(plot, ...)
```

Arguments

<code>plot</code>	A tidyplot generated with the function <code>tidyplot()</code> .
<code>...</code>	Arguments passed on to <code>ggplot2::coord_flip()</code> .

Value

A tidyplot object.

Examples

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  flip_plot()

energy |>
  tidyplot(x = year, y = energy, color = energy_type) |>
  add_barstack_absolute() |>
  flip_plot()

# Better solutions without `flip_plot()`
study |>
  tidyplot(x = score, y = treatment, color = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

energy |>
  tidyplot(x = energy, y = year, color = energy_type) |>
  add_barstack_absolute(orientation = "y")
```

format_p_value	<i>Format p values</i>
----------------	------------------------

Description

Format p values

Usage

```
format_p_value(x, accuracy = 1e-04)
```

Arguments

x	A number to format.
accuracy	A number to round to. For example, use 0.01 to show 2 decimal places of precision. Defaults to 0.0001, corresponding to 4 decimal places of precision.

Value

Formatted number as character string.

Examples

```
format_p_value(0.03445553)
format_p_value(0.0003445553)
format_p_value(0.00003445553)
```

gene_expression	<i>RNA-Seq expression data</i>
-----------------	--------------------------------

Description

RNA-Seq expression data

Usage

```
gene_expression
```

Format

A data frame.

Source

[Bassoon proteinopathy drives neurodegeneration in multiple sclerosis](#), Nature Neuroscience 2019
[GSE104899](#), Gene Expression Omnibus

Examples

```
dplyr::glimpse(gene_expression)
```

new_color_scheme	<i>New color scheme</i>
------------------	-------------------------

Description

For more information about the use of color schemes in tidypLOTS, check out this article: [Color schemes](#)

Usage

```
new_color_scheme(x, name = "Untitled color scheme", reverse = FALSE)
```

Arguments

x	Character vector of hex colors. For example <code>x = c("#FF00FF", "#00FFFF")</code> .
name	Name of the custom color scheme.
reverse	Whether the order should be reversed or not. Defaults to FALSE, meaning not reversed.

Value

A tidypLOT object.

Examples

```
new_color_scheme(c("#ECA669", "#E06681", "#8087E2", "#E2D269"))
new_color_scheme(c("#ECA669", "#E06681", "#8087E2", "#E2D269"),
  name = "my_custom_color_scheme")
```

pca	<i>Principle component analysis data</i>
-----	--

Description

Principle component analysis data

Usage

```
pca
```

Format

A data frame.

Source

[Bassoon proteinopathy drives neurodegeneration in multiple sclerosis](#), Nature Neuroscience 2019

Examples

```
dplyr::glimpse(pca)
```

remove_legend	<i>Remove legend or legend title</i>
---------------	--------------------------------------

Description

Remove legend or legend title

Usage

```
remove_legend(plot)
```

```
remove_legend_title(plot)
```

Arguments

plot A tidyplot generated with the function tidyplot().

Value

A tidyplot object.

Examples

```
# Before removing
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar()

# After removing
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_legend_title()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_legend()
```

remove_padding	<i>Remove plot area padding</i>
----------------	---------------------------------

Description

Remove plot area padding

Usage

```
remove_padding(plot, force_continuous = FALSE)
```

Arguments

`plot` A tidyplot generated with the function `tidyplot()`.
`force_continuous` Whether to force the axis to be continuous. Defaults to FALSE.

Value

A tidyplot object.

Examples

```
# Before removing  
animals |>  
  tidyplot(x = weight, y = speed, color = family) |>  
  add_data_points()  
  
# After removing  
animals |>  
  tidyplot(x = weight, y = speed, color = family) |>  
  add_data_points() |>  
  remove_padding()
```

remove_title	<i>Remove plot title or caption</i>
--------------	-------------------------------------

Description

Remove plot title or caption

Usage

```
remove_title(plot)  
  
remove_caption(plot)
```

Arguments

plot A tidyplot generated with the function tidyplot().

Value

A tidyplot object.

Examples

```
# Before removing
animals |>
  tidyplot(x = weight, y = speed, color = family) |>
  add_data_points() |>
  add_title("Name of the plot") |>
  add_caption("This is the caption")

# After removing
animals |>
  tidyplot(x = weight, y = speed, color = family) |>
  add_data_points() |>
  add_title("Name of the plot") |>
  add_caption("This is the caption") |>
  remove_title()

animals |>
  tidyplot(x = weight, y = speed, color = family) |>
  add_data_points() |>
  add_title("Name of the plot") |>
  add_caption("This is the caption") |>
  remove_caption()
```

remove_x_axis

Remove x-axis or parts of it

Description

Remove x-axis or parts of it

Usage

```
remove_x_axis(plot)

remove_x_axis_line(plot)

remove_x_axis_ticks(plot)

remove_x_axis_labels(plot)

remove_x_axis_title(plot)
```

Arguments

plot A tidyplot generated with the function tidyplot().

Value

A tidyplot object.

Examples

```
# Before removing
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar()

# After removing
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_x_axis_line()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_x_axis_ticks()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_x_axis_labels()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_x_axis_title()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_x_axis()
```

remove_y_axis

Remove y-axis or parts of it

Description

Remove y-axis or parts of it

Usage

```
remove_y_axis(plot)

remove_y_axis_line(plot)

remove_y_axis_ticks(plot)

remove_y_axis_labels(plot)

remove_y_axis_title(plot)
```

Arguments

`plot` A tidyplot generated with the function `tidyplot()`.

Value

A tidyplot object.

Examples

```
# Before removing
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar()

# After removing
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_y_axis_line()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_y_axis_ticks()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_y_axis_labels()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_y_axis_title()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar() |>
  remove_y_axis()
```

rename_x_axis_levels *Rename axis or color levels*

Description

Rename axis or color levels

Usage

```
rename_x_axis_levels(plot, new_names)
```

```
rename_y_axis_levels(plot, new_names)
```

```
rename_color_levels(plot, new_names)
```

Arguments

`plot` A tidyplot generated with the function `tidyplot()`.
`new_names` Named character vector in the format `c("old1" = "new1", "old2" = "new2")`.

Value

A tidyplot object.

Examples

```
# Before adjustments
study |>
  tidyplot(x = treatment, y = score) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Rename x-axis levels
study |>
  tidyplot(x = treatment, y = score) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  rename_x_axis_levels(new_names = c(
    "A" = "This",
    "B" = "is",
    "C" = "totally",
    "D" = "new"))

# Before adjustments
study |>
```

```
tidyplot(x = score, y = treatment) |>
add_data_points() |>
add_mean_bar(alpha = 0.4) |>
add_sem_errorbar()

# Rename y-axis levels
study |>
tidyplot(x = score, y = treatment) |>
add_data_points() |>
add_mean_bar(alpha = 0.4) |>
add_sem_errorbar() |>
rename_y_axis_levels(new_names = c(
  "A" = "This",
  "B" = "is",
  "C" = "totally",
  "D" = "new"))

# Before adjustment
study |>
tidyplot(x = group, y = score, color = dose) |>
add_data_points() |>
add_mean_bar(alpha = 0.4) |>
add_sem_errorbar()

# Rename color levels
study |>
tidyplot(x = group, y = score, color = dose) |>
add_data_points() |>
add_mean_bar(alpha = 0.4) |>
add_sem_errorbar() |>
rename_color_levels(new_names = c(
  "high" = "Sky high",
  "low" = "Deep low"))
```

reorder_x_axis_levels *Reorder axis or color levels*

Description

Reorder axis or color levels

Usage

```
reorder_x_axis_levels(plot, ...)
```

```
reorder_y_axis_levels(plot, ...)
```

```
reorder_color_levels(plot, ...)
```

Arguments

`plot` A tidyplot generated with the function `tidyplot()`.
`...` Arguments passed on to `forcats::fct_relevel()`.

Value

A tidyplot object.

Examples

```
# Before adjustments
study |>
  tidyplot(x = treatment, y = score) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Reorder x-axis levels
study |>
  tidyplot(x = treatment, y = score) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  reorder_x_axis_levels("D", "B", "A")

# Before adjustments
study |>
  tidyplot(x = score, y = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Reorder y-axis levels
study |>
  tidyplot(x = score, y = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  reorder_y_axis_levels("D", "B", "A")

# Before adjustment
study |>
  tidyplot(x = group, y = score, color = dose) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Reorder color levels
study |>
  tidyplot(x = group, y = score, color = dose) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
```

```
add_sem_errorbar() |>  
reorder_color_levels("low")
```

reverse_x_axis_levels *Reverse axis or color levels*

Description

Reverse axis or color levels

Usage

```
reverse_x_axis_levels(plot)  
  
reverse_y_axis_levels(plot)  
  
reverse_color_levels(plot)
```

Arguments

`plot` A tidyplot generated with the function `tidyplot()`.

Value

A tidyplot object.

Examples

```
# Before adjustments  
study |>  
  tidyplot(x = treatment, y = score) |>  
  add_data_points() |>  
  add_mean_bar(alpha = 0.4) |>  
  add_sem_errorbar()  
  
# Reverse x-axis levels  
study |>  
  tidyplot(x = treatment, y = score) |>  
  add_data_points() |>  
  add_mean_bar(alpha = 0.4) |>  
  add_sem_errorbar() |>  
  reverse_x_axis_levels()  
  
# Before adjustments  
study |>  
  tidyplot(x = score, y = treatment) |>  
  add_data_points() |>  
  add_mean_bar(alpha = 0.4) |>  
  add_sem_errorbar()
```

```
# Reverse y-axis levels
study |>
  tidyplot(x = score, y = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  reverse_y_axis_levels()

# Before adjustment
study |>
  tidyplot(x = group, y = score, color = dose) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Reverse color levels
study |>
  tidyplot(x = group, y = score, color = dose) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  reverse_color_levels()
```

save_plot

Save plots to file

Description

This function takes a plot or list of plots and writes them to a (multipage) file.

Usage

```
save_plot(
  plot = ggplot2::last_plot(),
  filename,
  width = NA,
  height = NA,
  units = c("mm", "cm", "in"),
  padding = 0.1,
  multiple_files = FALSE,
  view_plot = TRUE,
  ...
)
```

Arguments

plot Plot to save, defaults to last plot displayed.

filename	File name to create on disk.
width, height	Dimensions of the graphic device to save the plot. Defaults to NA. In case of NA, the dimensions are inferred from the incoming plot object (see Details).
units	Units of length. Defaults to "mm".
padding	Extra space around the saved plot. Defaults to 0.1 meaning 10%.
multiple_files	Whether to save multiple pages as individual files.
view_plot	Whether to view the plot on screen after saving.
...	Other arguments passed on to the graphics device function, as specified by device.

Details

Handling of file dimensions. Output file dimensions are determined according the the following precedence.

1. The width and height arguments.
2. Dimensions inferred from the incoming plot object with absolute dimensions.
3. System default device dimensions.

Value

A tidyplot object.

Examples

```
# Save plot to file
study |>
  tidyplot(treatment, score) |>
  add_data_points() |>
  save_plot("single_plot.pdf")

# Save intermediate stages to file
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  add_data_points_beeswarm() |>
  save_plot("before.pdf") |>
  adjust_colors(colors_discrete_seaside) |>
  save_plot("after.pdf")

# Save multipage PDF file
gene_expression |>
  dplyr::slice_head(n = 160) |>
  tidyplot(group, expression, color = sample_type) |>
  add_data_points() |>
  adjust_size(width = 30, height = 25) |>
```

```

split_plot(by = external_gene_name, nrow = 2, ncol = 2) |>
save_plot("multipage_plot.pdf")

# Save multiple PDF files
gene_expression |>
  dplyr::slice_head(n = 160) |>
  tidyplot(group, expression, color = sample_type) |>
  add_data_points() |>
  adjust_size(width = 30, height = 25) |>
  split_plot(by = external_gene_name, nrow = 2, ncol = 2) |>
  save_plot("plot.pdf", multiple_files = TRUE)

```

sort_x_axis_levels *Sort axis or color levels*

Description

Sort axis or color levels

Usage

```
sort_x_axis_levels(plot, ..., .fun = NULL, .reverse = FALSE)
```

```
sort_y_axis_levels(plot, ..., .fun = NULL, .reverse = FALSE)
```

```
sort_color_levels(plot, ..., .fun = NULL, .reverse = FALSE)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
...	Optional variables to use for sorting.
.fun	Override the function used for sorting. Is automatically determined from the plot.
.reverse	Whether the order should be reversed or not. Defaults to FALSE, meaning not reversed.

Value

A tidyplot object.

Examples

```
# Before adjustments
study |>
  tidyplot(x = treatment, y = score) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Sort x-axis levels by score
study |>
  tidyplot(x = treatment, y = score) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  sort_x_axis_levels()

# Before adjustments
study |>
  tidyplot(x = score, y = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Sort y-axis levels by score
study |>
  tidyplot(x = score, y = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  sort_y_axis_levels()

# Before adjustment
study |>
  tidyplot(x = group, y = score, color = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar()

# Sort color levels by score
study |>
  tidyplot(x = group, y = score, color = treatment) |>
  add_data_points() |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  sort_color_levels()
```

Description

Spending data

Usage

```
spendings
```

Format

A data frame.

Source

tidyplots package

Examples

```
dplyr::glimpse(spendings)
```

split_plot

Split plot into multiple subplots

Description

Split plot into multiple subplots

Usage

```
split_plot(  
  plot,  
  by = NULL,  
  rows = NULL,  
  cols = NULL,  
  ncol = NULL,  
  nrow = NULL,  
  axes = "all",  
  axis.titles = "all",  
  scales = NULL,  
  ...  
)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
by	One variable that should be used for splitting.
rows, cols	Two variables that should be used for splitting, representing rows and columns, respectively.
ncol, nrow	The number of columns and rows per page. Only takes effect when using by to split by a single variable.
axes	Determines which axes will be drawn in case of fixed scales. When "margins", axes will be drawn at the exterior margins. "all_x" and "all_y" will draw the respective axes at the interior panels too, whereas "all" (the default) will draw all axes at all panels.
axis.titles	Determines which axis titles will be drawn. When "margins", axis titles will be drawn at the exterior margins. "all" (the default) will draw all axis titles at all panels. "single" will draw a single axis title per multiplot layout.
scales	Should scales be fixed "fixed", free ("free"), or free in one dimension ("free_x", "free_y")? Defaults to "free" when providing one splitting variable via by. Defaults to "fixed" when providing two splitting variables via rows and cols.
...	Arguments passed on to the geom function.

Value

A tidyplot object.

Examples

```
# Before splitting
energy |>
  dplyr::filter(year %in% c(2005, 2010, 2015, 2020)) |>
  tidyplot(y = energy, color = energy_source) |>
  add_donut() |>
  adjust_size(width = 25, height = 25)

# Split by year
energy |>
  dplyr::filter(year %in% c(2005, 2010, 2015, 2020)) |>
  tidyplot(y = energy, color = energy_source) |>
  add_donut() |>
  adjust_size(width = 25, height = 25) |>
  split_plot(by = year)

# Change dimensions of subplots
energy |>
  dplyr::filter(year %in% c(2005, 2010, 2015, 2020)) |>
  tidyplot(y = energy, color = energy_source) |>
  add_donut() |>
  adjust_size(width = 15, height = 15) |>
  split_plot(by = year)
```

```
# Spread plots across multiple pages
energy |>
  dplyr::filter(year %in% c(2005, 2010, 2015, 2020)) |>
  tidypplot(y = energy, color = energy_source) |>
  add_donut() |>
  adjust_size(width = 25, height = 25) |>
  split_plot(by = year, ncol = 2, nrow = 1)

# Split by two variables
energy |>
  dplyr::mutate(decade = paste0(floor(year / 10) * 10, "s")) |>
  tidypplot(y = energy, color = energy_source) |>
  add_donut() |>
  adjust_size(14,14) |>
  split_plot(rows = decade, cols = energy_type)
```

study

Study data

Description

Study data

Usage

study

Format

A data frame.

Source

tidypplots package

Examples

```
dplyr::glimpse(study)
```

theme_tidyplot	<i>Themes</i>
----------------	---------------

Description

Themes

Usage

```
theme_tidyplot(plot, fontsize = 7)
```

```
theme_ggplot2(plot, fontsize = 7)
```

```
theme_minimal_xy(plot, fontsize = 7)
```

```
theme_minimal_x(plot, fontsize = 7)
```

```
theme_minimal_y(plot, fontsize = 7)
```

Arguments

`plot` A tidyplot generated with the function `tidyplot()`.

`fontsize` Font size in points. Defaults to 7.

Value

A tidyplot object.

Examples

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_sem_errorbar() |>
  add_mean_dash() |>
  theme_tidyplot()
```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_sem_errorbar() |>
  add_mean_dash() |>
  theme_ggplot2()
```

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_sem_errorbar() |>
  add_mean_dash() |>
```

```

theme_minimal_xy()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_sem_errorbar() |>
  add_mean_dash() |>
  theme_minimal_x()

study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points() |>
  add_sem_errorbar() |>
  add_mean_dash() |>
  theme_minimal_y()

```

tidyplot*Create a new tidyplot*

Description

Create a new tidyplot

Usage

```

tidyplot(
  data,
  ...,
  width = NULL,
  height = NULL,
  unit = NULL,
  dodge_width = NULL,
  my_style = NULL,
  paper = NULL,
  ink = NULL
)

```

Arguments

<code>data</code>	A tidy data frame to use for plotting.
<code>...</code>	Mappings for the x axis, y axis and color, see examples. Additional arguments are passed to <code>ggplot2::aes()</code> .
<code>width, height</code>	Dimensions of the plot area. The default (NULL) retrieves the setting from the tidyplots options , which defaults to 50. NA results in taking all available space (ggplot2 default).
<code>unit</code>	Unit of the plot area width and height. The default (NULL) retrieves the setting from the tidyplots options , which defaults to "mm".

dodge_width	For adjusting the distance between grouped objects. The default (NULL) retrieves the setting from the tidyplots options , which defaults to 0.8 for plots with at least one discrete axis and to 0 for plots with two continuous axes.
my_style	Styling function to apply to the plot. The default (NULL) retrieves the setting from the tidyplots options , which default to no additional styling.
paper	Background color. The default (NULL) retrieves the setting from the tidyplots options , which defaults to "#FFFFFF".
ink	Foreground color. The default (NULL) retrieves the setting from the tidyplots options , which defaults to "#000000".

Examples

```
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_data_points_beeswarm()

study |>
  tidyplot(x = group, y = score, color = dose) |>
  add_mean_bar()

# Change plot area size
study |>
  tidyplot(x = treatment, y = score, color = treatment,
           width = 25, height = 25) |>
  add_data_points_beeswarm()

# Change dodge_width
study |>
  tidyplot(x = group, y = score, color = dose, dodge_width = 0.3) |>
  add_mean_bar()
```

tidyplots_options *Tidyplots options*

Description

Control the settings for formatting tidyplots globally.

Usage

```
tidyplots_options(
  width = NULL,
  height = NULL,
  unit = NULL,
  dodge_width = NULL,
  my_style = NULL,
  paper = NULL,
  ink = NULL
)
```

Arguments

width, height	Dimensions of the plot area. The default (NULL) retrieves the setting from the tidyplots options , which defaults to 50. NA results in taking all available space (ggplot2 default).
unit	Unit of the plot area width and height. The default (NULL) retrieves the setting from the tidyplots options , which defaults to "mm".
dodge_width	For adjusting the distance between grouped objects. The default (NULL) retrieves the setting from the tidyplots options , which defaults to 0.8 for plots with at least one discrete axis and to 0 for plots with two continuous axes.
my_style	Styling function to apply to the plot. The default (NULL) retrieves the setting from the tidyplots options , which default to no additional styling.
paper	Background color. The default (NULL) retrieves the setting from the tidyplots options , which defaults to "#FFFFFF".
ink	Foreground color. The default (NULL) retrieves the setting from the tidyplots options , which defaults to "#000000".

Value

The old options invisibly

Examples

```
# Define custom style
my_style <- function(x) x |>
  adjust_colors(colors_discrete_candy) |>
  adjust_font(family = "mono")

# Set tidyplots options
tidyplots_options(
  width = 3,
  height = 4,
  unit = "cm",
  dodge_width = 1,
  my_style = my_style
)

# Plot
study |>
  tidyplot(x = group, y = score, color = dose) |>
  add_mean_bar()

# Reset tidyplots options
tidyplots_options()

# Same plot
study |>
  tidyplot(x = group, y = score, color = dose) |>
  add_mean_bar()
```

time_course	<i>Time course data</i>
-------------	-------------------------

Description

Time course data

Usage

```
time_course
```

Format

A data frame.

Source

tidyplots package

Examples

```
dplyr::glimpse(time_course)
```

view_plot	<i>View plot on screen</i>
-----------	----------------------------

Description

View plot on screen

Usage

```
view_plot(plot, data = all_rows(), title = ggplot2::waiver(), ...)
```

Arguments

plot	A tidyplot generated with the function tidyplot().
data	The data to be displayed in this layer. There are three options: <ul style="list-style-type: none">• If <code>all_rows()</code> (the default) the complete dataset is displayed.• A function to subset the plot data. See <code>filter_rows()</code> and friends.• A data.frame to override the plot data.
title	Plot title.
...	Arguments passed on to <code>print()</code> .

Details

- `view_plot()` supports data subsetting. See examples and [Advanced plotting](#).

Value

A tidyplot object.

Examples

```
# View intermediate stages on screen
study |>
  tidyplot(x = treatment, y = score, color = treatment) |>
  add_mean_bar(alpha = 0.4) |>
  add_sem_errorbar() |>
  add_data_points_beeswarm() |>
  view_plot(title = "Before changing color scheme") |>
  adjust_colors(colors_discrete_seaside) |>
  view_plot(title = "After changing color scheme")

# View data subsets on screen
gene_expression |>
  tidyplot(x = condition, y = expression, color = sample_type) |>
  add_mean_dash() |>
  add_sem_errorbar() |>
  add_data_points_beeswarm() |>
  view_plot(data = filter_rows(external_gene_name == "Apol6"),
            title = "Apol6") |>
  view_plot(data = filter_rows(external_gene_name == "Bsn"),
            title = "Bsn")
```

Index

- * **datasets**
 - animals, [64](#)
 - climate, [65](#)
 - colors_continuous_viridis, [65](#)
 - colors_discrete_friendly, [67](#)
 - colors_diverging_blue2red, [68](#)
 - dinosaurs, [69](#)
 - distributions, [70](#)
 - energy, [70](#)
 - energy_week, [71](#)
 - eu_countries, [71](#)
 - gene_expression, [73](#)
 - pca, [74](#)
 - spendings, [87](#)
 - study, [90](#)
 - time_course, [95](#)
- ?dplyr_by, [63](#)
- add, [3](#)
- add_annotation_line
 - (add_annotation_text), [4](#)
- add_annotation_rectangle
 - (add_annotation_text), [4](#)
- add_annotation_text, [4](#)
- add_area(add_line), [24](#)
- add_areastack_absolute, [5](#)
- add_areastack_relative
 - (add_areastack_absolute), [5](#)
- add_barstack_absolute, [7](#)
- add_barstack_relative
 - (add_barstack_absolute), [7](#)
- add_boxplot, [8](#)
- add_caption(add_title), [46](#)
- add_ci95_errorbar(add_sem_errorbar), [34](#)
- add_ci95_ribbon(add_sem_ribbon), [36](#)
- add_count_area(add_count_bar), [10](#)
- add_count_bar, [10](#)
- add_count_dash(add_count_bar), [10](#)
- add_count_dot(add_count_bar), [10](#)
- add_count_line(add_count_bar), [10](#)
- add_count_value(add_count_bar), [10](#)
- add_curve_fit, [13](#)
- add_data_labels, [15](#)
- add_data_labels_repel
 - (add_data_labels), [15](#)
- add_data_points, [17](#)
- add_data_points_beeswarm
 - (add_data_points), [17](#)
- add_data_points_jitter
 - (add_data_points), [17](#)
- add_donut(add_pie), [31](#)
- add_ellipse, [21](#)
- add_heatmap, [22](#)
- add_histogram, [23](#)
- add_line, [24](#)
- add_mean_area(add_mean_bar), [25](#)
- add_mean_bar, [25](#)
- add_mean_dash(add_mean_bar), [25](#)
- add_mean_dot(add_mean_bar), [25](#)
- add_mean_line(add_mean_bar), [25](#)
- add_mean_value(add_mean_bar), [25](#)
- add_median_area(add_median_bar), [28](#)
- add_median_bar, [28](#)
- add_median_dash(add_median_bar), [28](#)
- add_median_dot(add_median_bar), [28](#)
- add_median_line(add_median_bar), [28](#)
- add_median_value(add_median_bar), [28](#)
- add_pie, [31](#)
- add_range_errorbar(add_sem_errorbar), [34](#)
- add_range_ribbon(add_sem_ribbon), [36](#)
- add_reference_lines, [33](#)
- add_sd_errorbar(add_sem_errorbar), [34](#)
- add_sd_ribbon(add_sem_ribbon), [36](#)
- add_sem_errorbar, [34](#)
- add_sem_ribbon, [36](#)
- add_sum_area(add_sum_bar), [37](#)
- add_sum_bar, [37](#)
- add_sum_dash(add_sum_bar), [37](#)

- add_sum_dot (add_sum_bar), 37
- add_sum_line (add_sum_bar), 37
- add_sum_value (add_sum_bar), 37
- add_test_asterisks (add_test_pvalue), 41
- add_test_pvalue, 41
- add_title, 46
- add_violin, 47
- adjust_caption (adjust_title), 57
- adjust_colors, 48
- adjust_font, 50
- adjust_legend_position
 - (adjust_legend_title), 51
- adjust_legend_title, 51
- adjust_padding, 53
- adjust_size, 55
- adjust_theme_details, 56
- adjust_title, 57
- adjust_x_axis, 59
- adjust_x_axis_title (adjust_title), 57
- adjust_y_axis (adjust_x_axis), 59
- adjust_y_axis_title (adjust_title), 57
- all_rows, 62
- animals, 64

- beeswarm::swarmx(), 20

- climate, 65
- colors_continuous_bluepinkyellow
 - (colors_continuous_viridis), 65
- colors_continuous_cividis
 - (colors_continuous_viridis), 65
- colors_continuous_inferno
 - (colors_continuous_viridis), 65
- colors_continuous_magma
 - (colors_continuous_viridis), 65
- colors_continuous_mako
 - (colors_continuous_viridis), 65
- colors_continuous_plasma
 - (colors_continuous_viridis), 65
- colors_continuous_rocket
 - (colors_continuous_viridis), 65
- colors_continuous_turbo
 - (colors_continuous_viridis), 65
- colors_continuous_viridis, 65
- colors_continuous_viridis(), 49
- colors_discrete_alger
 - (colors_discrete_friendly), 67
- colors_discrete_apple
 - (colors_discrete_friendly), 67
- colors_discrete_candy
 - (colors_discrete_friendly), 67
- colors_discrete_friendly, 67
- colors_discrete_friendly(), 49
- colors_discrete_friendly_long
 - (colors_discrete_friendly), 67
- colors_discrete_ibm
 - (colors_discrete_friendly), 67
- colors_discrete_metro
 - (colors_discrete_friendly), 67
- colors_discrete_okabeito
 - (colors_discrete_friendly), 67
- colors_discrete_rainbow
 - (colors_discrete_friendly), 67
- colors_discrete_seaside
 - (colors_discrete_friendly), 67
- colors_diverging_blue2brown
 - (colors_diverging_blue2red), 68
- colors_diverging_blue2brown(), 49
- colors_diverging_blue2red, 68
- colors_diverging_BuRd
 - (colors_diverging_blue2red), 68
- colors_diverging_BuYlRd
 - (colors_diverging_blue2red), 68
- colors_diverging_icefire
 - (colors_diverging_blue2red), 68
- colors_diverging_spectral
 - (colors_diverging_blue2red), 68

- dinosaurs, 69
- distributions, 70
- dplyr::arrange(), 63

- energy, 70
- energy_week, 71
- eu_countries, 71

- filter_rows (all_rows), 62
- first_rows (all_rows), 62
- flip_plot, 72
- format_p_value, 73

- gene_expression, 73
- glue, 43
- group_by(), 63

- lambda, 49, 60
- last_rows (all_rows), 62
- max_rows (all_rows), 62

- mgcv::gam(), 14
- min_rows (all_rows), 62
- new_color_scheme, 74
- new_color_scheme(), 49
- p.adjust, 42
- pca, 74
- remove_caption (remove_title), 76
- remove_legend, 75
- remove_legend_title (remove_legend), 75
- remove_padding, 76
- remove_title, 76
- remove_x_axis, 77
- remove_x_axis_labels (remove_x_axis), 77
- remove_x_axis_line (remove_x_axis), 77
- remove_x_axis_ticks (remove_x_axis), 77
- remove_x_axis_title (remove_x_axis), 77
- remove_y_axis, 78
- remove_y_axis_labels (remove_y_axis), 78
- remove_y_axis_line (remove_y_axis), 78
- remove_y_axis_ticks (remove_y_axis), 78
- remove_y_axis_title (remove_y_axis), 78
- rename_color_labels
 (rename_x_axis_levels), 80
- rename_color_levels
 (rename_x_axis_levels), 80
- rename_x_axis_labels
 (rename_x_axis_levels), 80
- rename_x_axis_levels, 80
- rename_y_axis_labels
 (rename_x_axis_levels), 80
- rename_y_axis_levels
 (rename_x_axis_levels), 80
- reorder_color_labels
 (reorder_x_axis_levels), 81
- reorder_color_levels
 (reorder_x_axis_levels), 81
- reorder_x_axis_labels
 (reorder_x_axis_levels), 81
- reorder_x_axis_levels, 81
- reorder_y_axis_labels
 (reorder_x_axis_levels), 81
- reorder_y_axis_levels
 (reorder_x_axis_levels), 81
- reverse_color_labels
 (reverse_x_axis_levels), 83
- reverse_color_levels
 (reverse_x_axis_levels), 83
- reverse_x_axis_labels
 (reverse_x_axis_levels), 83
- reverse_x_axis_levels, 83
- reverse_y_axis_labels
 (reverse_x_axis_levels), 83
- reverse_y_axis_levels
 (reverse_x_axis_levels), 83
- sample_rows (all_rows), 62
- save_plot, 84
- scales::extended_breaks(), 60
- scales::new_transform(), 61
- sort_color_labels (sort_x_axis_levels),
 86
- sort_color_levels (sort_x_axis_levels),
 86
- sort_x_axis_labels
 (sort_x_axis_levels), 86
- sort_x_axis_levels, 86
- sort_y_axis_labels
 (sort_x_axis_levels), 86
- sort_y_axis_levels
 (sort_x_axis_levels), 86
- spendings, 87
- split_plot, 88
- stats::loess(), 14
- study, 90
- symnum, 43
- theme_ggplot2 (theme_tidyplot), 91
- theme_minimal_x (theme_tidyplot), 91
- theme_minimal_xy (theme_tidyplot), 91
- theme_minimal_y (theme_tidyplot), 91
- theme_tidyplot, 91
- tidyplot, 92
- tidyplots options, 55, 92–94
- tidyplots_options, 93
- time_course, 95
- transformation object, 60
- view_plot, 95