Package ‘mlbplotR’

May 8, 2026

Type Package

Title Create 'ggplot2' and 'gt' Visuals with Major League Baseball Logos

Version 1.2.0

Description Tools to help visualize Major League Baseball analysis in 'ggplot2' and 'gt'. You provide team/player information and 'mlbplotR' will transform that information into team colors, logos, or player headshots for graphics.

URL https://github.com/camdenk/mlbplotR, https://camdenk.github.io/mlbplotR/

BugReports https://github.com/camdenk/mlbplotR/issues

Imports cli, data.table, ggplot2, gt, httr, lifecycle, magick, magrittr, rlang, scales

Suggests base64enc, dplyr, ggtext, gridtext, knitr, rsvg, rmarkdown, rstudioapi, sjmisc

Encoding UTF-8

License MIT + file LICENSE

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

NeedsCompilation no

Author Sebastian Carl [aut], Camden Kay [aut, cre, cph]

Maintainer Camden Kay <camden.kay23@gmail.com>

Repository CRAN

Date/Publication 2025-09-02 22:20:02 UTC

Contents

clean_team_abbrs 2
element 3

geom_cap_logos . . . 7
geom_from_path . . . 11
geom_lines . . . 14
geom_milb_logos . . . 17
geom_mlb_headshots . . . 20
geom_mlb_logos . . . 24
ggpreview . . . 28
gt_merge_stack_team_color . . . 30
gt_milb . . . 31
gt_mlb . . . 32
gt_mlb_column_labels . . . 33
gt_mlb_headshots . . . 34
load_headshots . . . 35
load_milb_teams . . . 36
load_mlb_teams . . . 37
mlb_player_tiers . . . 37
mlb_team_factor . . . 40
mlb_team_tiers . . . 41
scale_axes_mlb . . . 43
scale_mlb . . . 46
theme_mlb . . . 49
valid_team_names . . . 50

Index **51**

`clean_team_abbrs` *Standardize MLB Team Abbreviations*

Description

This function standardizes MLB team abbreviations to Baseball Savant defaults. This helps for joins and plotting

Usage

```
clean_team_abbrs(abbr, keep_non_matches = TRUE)
```

Arguments

`abbr`	a character vector of abbreviations
`keep_non_matches`	will non-matches be kept in the vector?

Value

A character vector with the length of `abbr` and cleaned team abbreviations if they are included in `team_data`. Non matches may be replaced with `NA` (depending on the value of `keep_non_matches`).

Examples

```
x <- c("PIE", "STL", "WSN", "CWS", "CHW")
# use current location and keep non matches
clean_team_abbrs(x)
```

`element` *Theme Elements for Image Grobs*

Description

In conjunction with the ggplot2::theme system, the following `element_` functions enable images in non-data components of the plot, e.g. axis text.

- `element_mlb_logo()`, `element_mlb_scoreboard_logo()`, `element_mlb_dot_logo()`: draws MLB team logos instead of their abbreviations
- `element_milb_logo`, `element_milb_light_cap_logo()`, and `element_milb_dot_logo`: draws MiLB team logos instead of their team names
- `element_mlb_dark_cap_logo()` and `element_mlb_light_cap_logo()`: draws MLB team cap logos instead of their abbreviations
- `element_mlb_headshot()` and `element_milb_dot_headshot()`: draws MLB player headshots instead of their MLB IDs
- `element_milb_dot_headshot()`: draws MiLB player headshots instead of their MLB IDs
- `element_path()`: draws images from valid image URLs instead of the URL.

Usage

```
element_mlb_logo(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)

element_mlb_scoreboard_logo(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)
```

```
element_mlb_dot_logo(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)

element_mlb_dark_cap_logo(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)

element_mlb_light_cap_logo(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)

element_mlb_headshot(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)

element_mlb_dot_headshot(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)

element_path(
  alpha = NULL,
  colour = NA,
```

```
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)

element_milb_logo(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)

element_milb_light_cap_logo(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)

element_milb_dot_logo(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)

element_milb_dot_headshot(
  alpha = NULL,
  colour = NA,
  hjust = NULL,
  vjust = NULL,
  color = NULL,
  size = 0.5
)
```

Arguments

`alpha`	The alpha channel, i.e. transparency level, as a numerical value between 0 and 1.
`colour`, `color`	The image will be colorized with this color. Use the special character `"b/w"` to set it to black and white. For more information on valid color names in ggplot2

see https://ggplot2.tidyverse.org/articles/ggplot2-specs.html?q=colour#colour-and-fill.

hjust, vjust — The horizontal and vertical adjustment respectively. Must be a numerical value between 0 and 1.

size — The output grob size in cm (!).

Details

The elements translate MLB team abbreviations or MLB player IDs into logo images or headshots, respectively.

Value

An S3 object of class element.

See Also

geom_mlb_logos(), geom_mlb_headshots(), and geom_from_path() for more information on valid team abbreviations, player ids, and other parameters.

Examples

```
library(mlbplotR)
library(ggplot2)

team_abbr <- valid_team_names()
# remove league logos from this example
team_abbr <- team_abbr[!team_abbr %in% c("AL", "NL", "MLB")]

df <- data.frame(
  random_value = runif(length(team_abbr), 0, 1),
  teams = team_abbr
)

# use logos for x-axis
ggplot(df, aes(x = teams, y = random_value)) +
  geom_col(aes(color = teams, fill = teams), width = 0.5) +
  scale_color_mlb(type = "secondary") +
  scale_fill_mlb(alpha = 0.4) +
  theme_minimal() +
  theme(axis.text.x = element_mlb_logo())

# use logos for y-axis
ggplot(df, aes(y = teams, x = random_value)) +
  geom_col(aes(color = teams, fill = teams), width = 0.5) +
  scale_color_mlb(type = "secondary") +
  scale_fill_mlb(alpha = 0.4) +
  theme_minimal() +
  theme(axis.text.y = element_mlb_logo())

##############################################################################
```

```
# Headshot Examples
##############################################################################
library(mlbplotR)
library(ggplot2)


dfh <- data.frame(
  random_value = runif(9, 0, 1),
  player_id = c("594798",
                "592450",
                "605141",
                "665742",
                "545361",
                "665487",
                "571448",
                "0",
                "543037")
)

# use headshots for x-axis
ggplot(dfh, aes(x = player_id, y = random_value)) +
  geom_col(width = 0.5) +
  theme_minimal() +
  theme(axis.text.x = element_mlb_headshot())

# use headshots for y-axis
ggplot(dfh, aes(y = player_id, x = random_value)) +
  geom_col(width = 0.5) +
  theme_minimal() +
  theme(axis.text.y = element_mlb_headshot())
```

`geom_cap_logos` *ggplot2 Layer for Visualizing MLB Team Cap Logos*

Description

`geom_mlb_dark_cap_logos()` and `geom_mlb_light_cap_logos()` are used to plot MLB team cap and league logos instead of points in a ggplot. It requires x, y aesthetics as well as a valid MLB team abbreviation. The latter can be checked with `valid_team_names()` but is also cleaned before being plotted.

Usage

```
geom_mlb_dark_cap_logos(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
```

```
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)

geom_mlb_light_cap_logos(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)
```

Arguments

`mapping` Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

`data` The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `ggplot2::layer()`. These are often aesthetics, used to set an aesthetic to a fixed value. See the below section "Aesthetics" for a full list of possible arguments.

`nudge_x, nudge_y` Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with `position`.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

Value

A ggplot2 layer (`ggplot2::layer()`) that can be added to a plot created with `ggplot2::ggplot()`.

Aesthetics

`geom_mlb_dark_cap_logos()` and `geom_mlb_light_cap_logos()` understand the following aesthetics:

`x` - The x-coordinate. Required.

`y` - The y-coordinate. Required.

`team_abbr` - The team abbreviation. Need to use Savant's abbreviation. Required.

`alpha = NULL` - The alpha channel, i.e. transparency level, as a numerical value between 0 and 1.

`colour = NULL` - The image will be colourized with this colour. Use the special character `"b/w"` to set it to black and white. For more information on valid colour names in ggplot2 see https://ggplot2.tidyverse.org/articles/ggplot2-specs.html?q=colour#colour-and-fill

`angle = 0` - The angle of the image as a numerical value between 0° and 360°.

`hjust = 0.5` - The horizontal adjustment relative to the given x coordinate. Must be a numerical value between 0 and 1.

`vjust = 0.5` - The vertical adjustment relative to the given y coordinate. Must be a numerical value between 0 and 1.

height = 1.0 - The desired height of the image in npc (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is height = 0.1 (see below examples). For cap logos, the scaling works better when adjusting height and not width.

width = 1.0 - The desired width of the image in npc (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is height = 0.075 (see below examples). For cap logos, the scaling works better when adjusting height and not width.

Examples

```
library(mlbplotR)
library(ggplot2)

team_abbr <- valid_team_names()
# remove conference logos from this example
team_abbr <- team_abbr[!team_abbr %in% c("NL", "AL", "MLB")]

df <- data.frame(
  a = rep(1:6, 5),
  b = sort(rep(1:5, 6), decreasing = TRUE),
  teams = team_abbr
)

# keep alpha == 1 for all teams including an "A"
matches <- grepl("A", team_abbr)
df$alpha <- ifelse(matches, 1, 0.2)
# also set a custom fill colour for the non "A" teams
df$colour <- ifelse(matches, NA, "gray")

# scatterplot of all logos
ggplot(df, aes(x = a, y = b)) +
  geom_mlb_dark_cap_logos(aes(team_abbr = teams), height = 0.075) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  theme_void()

# apply alpha via an aesthetic from inside the dataset `df`
# please note that you have to add scale_alpha_identity() to use the alpha
# values in your dataset!
ggplot(df, aes(x = a, y = b)) +
  geom_mlb_dark_cap_logos(aes(team_abbr = teams, alpha = alpha), height = 0.075) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  scale_alpha_identity() +
  theme_void()

# apply alpha and colour via an aesthetic from inside the dataset `df`
# please note that you have to add scale_alpha_identity() as well as
# scale_colour_identity() to use the alpha and colour values in your dataset!
ggplot(df, aes(x = a, y = b)) +
 geom_mlb_light_cap_logos(aes(team_abbr = teams, alpha = alpha, colour = colour), height = 0.075) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  scale_alpha_identity() +
```

```
  scale_colour_identity() +
  theme_void()

# apply alpha as constant for all logos
ggplot(df, aes(x = a, y = b)) +
  geom_mlb_light_cap_logos(aes(team_abbr = teams), height = 0.075, alpha = 0.6) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  theme_void()

# it's also possible to plot league logos
league <- data.frame(a = 1:3, b = 0, teams = c("AL", "NL", "MLB"))
ggplot(league, aes(x = a, y = b)) +
  geom_mlb_dark_cap_logos(aes(team_abbr = teams), width = 0.3) +
  geom_label(aes(label = teams), nudge_y = -0.4, alpha = 0.5) +
  coord_cartesian(xlim = c(0.5,3.5), ylim = c(-0.75,.75)) +
  theme_void()
```

---

geom_from_path *ggplot2 Layer for Visualizing Images from URLs or Local Paths*

Description

This geom is used to plot MLB images instead of points in a ggplot. It requires x, y aesthetics as well as a path.

Usage

```
geom_from_path(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

`data` The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `ggplot2::layer()`. These are often aesthetics, used to set an aesthetic to a fixed value. See the below section "Aesthetics" for a full list of possible arguments.

`nudge_x, nudge_y`

Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with `position`.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

Value

A ggplot2 layer (`ggplot2::layer()`) that can be added to a plot created with `ggplot2::ggplot()`.

Aesthetics

`geom_mlb_logos()` understands the following aesthetics:

`x` - The x-coordinate. Required.

`y` - The y-coordinate. Required.

`path` - a file path, url, raster object or bitmap array. See `magick::image_read()` for further information. Required.

`alpha = NULL` - The alpha channel, i.e. transparency level, as a numerical value between 0 and 1.

`colour = NULL` - The image will be colorized with this colour. Use the special character `"b/w"` to set it to black and white. For more information on valid colour names in ggplot2 see https://ggplot2.tidyverse.org/articles/ggplot2-specs.html?q=colour#colour-and-fill

`angle = 0` - The angle of the image as a numerical value between 0° and 360°.

`hjust = 0.5` - The horizontal adjustment relative to the given x coordinate. Must be a numerical value between 0 and 1.

`vjust = 0.5` - The vertical adjustment relative to the given y coordinate. Must be a numerical value between 0 and 1.

`width = 1.0` - The desired width of the image in `npc` (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is `width = 0.1` (see below examples).

`height = 1.0` - The desired height of the image in `npc` (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is `height = 0.1` (see below examples).

Examples

```
library(mlbplotR)
library(ggplot2)

# create x-y-coordinates of a triangle and add league logo urls
df <- data.frame(
  a = c(sin(2 * pi * (0:4) / 5), 0),
  b = c(cos(2 * pi * (0:4) / 5), 0),
  url = c(
   "https://i.turner.ncaa.com/sites/default/files/images/logos/schools/bgl/virginia.svg",
   "https://i.turner.ncaa.com/sites/default/files/images/logos/schools/bgl/michigan-st.svg",
    "https://i.turner.ncaa.com/sites/default/files/images/logos/schools/bgl/lsu.svg",
    "https://i.turner.ncaa.com/sites/default/files/images/logos/schools/bgl/texas.svg",
    "https://i.turner.ncaa.com/sites/default/files/images/logos/schools/bgl/oregon.svg",
   "https://i.turner.ncaa.com/sites/default/files/images/logos/schools/bgl/james-madison.svg"
  )
)

# plot images directly from url
```

```
ggplot(df, aes(x = a, y = b)) +
  geom_from_path(aes(path = url), width = 0.15) +
  coord_cartesian(xlim = c(-2, 2), ylim = c(-1.3, 1.5)) +
  theme_void()

# plot images directly from url and apply transparency
ggplot(df, aes(x = a, y = b)) +
  geom_from_path(aes(path = url), width = 0.15, alpha = 0.5) +
  coord_cartesian(xlim = c(-2, 2), ylim = c(-1.3, 1.5)) +
  theme_void()

# It is also possible and recommended to use the underlying Geom inside a
# ggplot2 annotation
ggplot() +
  annotate(
    mlbplotR::GeomFromPath,
    x = 0,
    y = 0,
    path = "https://a.espncdn.com/combiner/i?img=/i/teamlogos/leagues/500/mlb.png",
    width = 0.4
  ) +
  theme_minimal()
```

---

geom_lines *ggplot2 Layer for Horizontal and Vertical Reference Lines*

Description

These geoms can be used to draw horizontal or vertical reference lines in a ggplot. They use the data in the aesthetics `v_var` and `h_var` to compute their `median` or `mean` and draw the as a line.

Usage

```
geom_median_lines(
  mapping = NULL,
  data = NULL,
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_mean_lines(
  mapping = NULL,
  data = NULL,
  ...,
  na.rm = FALSE,
  show.legend = NA,
```

```
  inherit.aes = TRUE
)
```

Arguments

`mapping` Set of aesthetic mappings created by `aes()`.

`data` The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behavior from the default plot specification.

Value

A ggplot2 layer (ggplot2::layer()) that can be added to a plot created with ggplot2::ggplot().

Aesthetics

geom_median_lines() and geom_mean_lines() understand the following aesthetics (at least one of v_var and h_var are required):

v_var - The variable for which to compute the median/mean that is drawn as vertical line.

h_var - The variable for which to compute the median/mean that is drawn as horizontal line.

alpha = NA - The alpha channel, i.e. transparency level, as a numerical value between 0 and 1.

color = "black" - The color of the drawn lines.

linetype = 2 - The linetype of the drawn lines.

linewidth = 0.5 - The size of the drawn lines.

See Also

The underlying ggplot2 geoms ggplot2::geom_hline() and ggplot2::geom_vline()

Examples

```
library(mlbplotR)
library(ggplot2)

# inherit top level aesthetics
ggplot(mtcars, aes(x = disp, y = mpg, h_var = mpg, v_var = disp)) +
  geom_point() +
  geom_median_lines() +
  geom_mean_lines(color = "blue") +
  theme_minimal()

# draw horizontal line only
ggplot(mtcars, aes(x = disp, y = mpg, h_var = mpg)) +
  geom_point() +
  geom_median_lines() +
  geom_mean_lines(color = "blue") +
  theme_minimal()

# draw vertical line only
ggplot(mtcars, aes(x = disp, y = mpg, v_var = disp)) +
  geom_point() +
  geom_median_lines() +
  geom_mean_lines(color = "blue") +
  theme_minimal()

# choose your own value
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  geom_median_lines(v_var = 400, h_var = 15) +
  geom_mean_lines(v_var = 150, h_var = 30, color = "blue") +
```

```
  theme_minimal()
```

---

geom_milb_logos *ggplot2 Layer for Visualizing MiLB Team Logos*

Description

geom_milb_logos(), geom_milb_light_cap_logos(), geom_milb_dot_logos() are used to plot MiLB team instead of points in a ggplot. It requires x, y aesthetics as well as a valid MiLB team name

Usage

```
geom_milb_logos(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)

geom_milb_light_cap_logos(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)

geom_milb_dot_logos(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
```

```
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)
```

Arguments

`mapping` Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

`data` The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `ggplot2::layer()`. These are often aesthetics, used to set an aesthetic to a fixed value. See the below section "Aesthetics" for a full list of possible arguments.

`nudge_x, nudge_y` Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with `position`.

na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Value

A ggplot2 layer (ggplot2::layer()) that can be added to a plot created with ggplot2::ggplot().

Aesthetics

geom_milb_logos(), geom_milb_light_cap_logos(), geom_milb_dot_logos() understand the following aesthetics:

x - The x-coordinate. Required.

y - The y-coordinate. Required.

team_name - The team name. Need to use the full team name. Required.

alpha = NULL - The alpha channel, i.e. transparency level, as a numerical value between 0 and 1.

colour = NULL - The image will be colourized with this colour. Use the special character "b/w" to set it to black and white. For more information on valid colour names in ggplot2 see https://ggplot2.tidyverse.org/articles/ggplot2-specs.html?q=colour#colour-and-fill

angle = 0 - The angle of the image as a numerical value between 0° and 360°.

hjust = 0.5 - The horizontal adjustment relative to the given x coordinate. Must be a numerical value between 0 and 1.

vjust = 0.5 - The vertical adjustment relative to the given y coordinate. Must be a numerical value between 0 and 1.

height = 1.0 - The desired height of the image in npc (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is height = 0.1 (see below examples). For cap logos, the scaling works better when adjusting height and not width.

width = 1.0 - The desired width of the image in npc (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is height = 0.075 (see below examples). For cap logos, the scaling works better when adjusting height and not width.

Examples

```
library(mlbplotR)
library(ggplot2)

team_names <- c("Kannapolis Cannon Ballers", "Charlotte Knights",
           "Chesapeake Baysox", "Durham Bulls", "Montgomery Biscuits", "Las Vegas Aviators",
              "Lehigh Valley IronPigs", "Richmond Flying Squirrels", "Round Rock Express",
```

```
                "Frisco RoughRiders", "Hub City Spartanburgers", "Hickory Crawdads")

df <- data.frame(
  a = rep(1:4, 3),
  b = sort(rep(1:3, 4), decreasing = TRUE),
  teams = team_names
)

# keep alpha == 1 for all teams including an "A"
matches <- grepl("A|a", team_names)
df$alpha <- ifelse(matches, 1, 0.2)
# also set a custom fill colour for the non "A" teams
df$colour <- ifelse(matches, NA, "gray")

# scatterplot of all logos
ggplot(df, aes(x = a, y = b)) +
  geom_milb_logos(aes(team_name = teams), height = 0.1) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  theme_void()


# apply alpha and colour via an aesthetic from inside the dataset `df`
# please note that you have to add scale_alpha_identity() as well as
# scale_colour_identity() to use the alpha and colour values in your dataset!
ggplot(df, aes(x = a, y = b)) +
 geom_milb_light_cap_logos(aes(team_name = teams, alpha = alpha, colour = colour), height = 0.1) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  scale_alpha_identity() +
  scale_colour_identity() +
  theme_void()

# apply alpha as constant for all logos
ggplot(df, aes(x = a, y = b)) +
  geom_milb_dot_logos(aes(team_name = teams), height = 0.15, alpha = 0.6) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  theme_void()
```

`geom_mlb_headshots` *ggplot2 Layer for Visualizing MLB/MiLB Player Headshots*

Description

This geom is used to plot MLB/MiLB player headshots instead of points in a ggplot. It requires x, y aesthetics as well as a valid MLBAM id (The same ID associated with their Baseball Savant page).

Usage

```
geom_mlb_headshots(
```

```
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)

geom_mlb_dot_headshots(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)

geom_milb_dot_headshots(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)
```

Arguments

`mapping` Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

`data` The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be

created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `ggplot2::layer()`. These are often aesthetics, used to set an aesthetic to a fixed value. See the below section "Aesthetics" for a full list of possible arguments.

`nudge_x, nudge_y` Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with `position`.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

Value

A ggplot2 layer (`ggplot2::layer()`) that can be added to a plot created with `ggplot2::ggplot()`.

Aesthetics

`geom_mlb_headshots()`, `geom_mlb_dot_headshots()`, `geom_milb_dot_headshots()`, understand the following aesthetics:

`x` - The x-coordinate. Required.

`y` - The y-coordinate. Required.

`player_id` - The players' MLBAM (Baseball Savant) id. Required.

`alpha = NULL` - The alpha channel, i.e. transparency level, as a numerical value between 0 and 1.

`colour = NULL` - The image will be colorized with this colour. Use the special character `"b/w"` to set it to black and white. For more information on valid colour names in ggplot2 see https://ggplot2.tidyverse.org/articles/ggplot2-specs.html?q=colour#colour-and-fill

`angle = 0` - The angle of the image as a numerical value between 0° and 360°.

`hjust = 0.5` - The horizontal adjustment relative to the given x coordinate. Must be a numerical value between 0 and 1.

`vjust = 0.5` - The vertical adjustment relative to the given y coordinate. Must be a numerical value between 0 and 1.

`width = 1.0` - The desired width of the image in `npc` (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is `width = 0.075` (see below examples).

`height = 1.0` - The desired height of the image in `npc` (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is `height = 0.1` (see below examples).

Examples

```
library(mlbplotR)
library(ggplot2)

df <- data.frame(
  a = c(rep(1:3, 3), 1.5, 2.5),
  b = c(sort(rep(1:3, 3), decreasing = TRUE), 2.5, 2.5),
  player_id = c("665742",
                "677594",
                "592450",
                "608070",
                "694973",
                "518692",
                "0",
                "663728",
                "120074",
                "608369",
                "606466"),
  player_name = c("Juan Soto",
                  "Julio Rodríguez",
                  "Aaron Judge",
                  "José Ramírez",
                  "Paul Skenes",
                  "Freddie Freeman",
```

```
                    "Non Match",
                    "Cal Raleigh",
                    "David Ortiz",
                    "Corey Seager",
                    "Ketel Marte")
)

# set a custom fill colour for one player
df$colour <- ifelse(df$a == 2 & df$b == 2, NA, "b/w")

# scatterplot of the headshots
ggplot(df, aes(x = a, y = b)) +
  geom_mlb_headshots(aes(player_id = player_id), height = 0.2) +
  geom_label(aes(label = player_name), nudge_y = -0.35, alpha = 0.5) +
  coord_cartesian(xlim = c(0.75, 3.25), ylim = c(0.7, 3.25)) +
  theme_void()

# apply alpha as constant
ggplot(df, aes(x = a, y = b)) +
  geom_mlb_headshots(aes(player_id = player_id), height = 0.2, alpha = 0.5) +
  geom_label(aes(label = player_name), nudge_y = -0.35, alpha = 0.5) +
  coord_cartesian(xlim = c(0.75, 3.25), ylim = c(0.7, 3.25)) +
  theme_void()

# apply colour as an aesthetic and use the dot version with colored background
ggplot(df, aes(x = a, y = b)) +
  geom_mlb_dot_headshots(aes(player_id = player_id, colour = colour), height = 0.2) +
  geom_label(aes(label = player_name), nudge_y = -0.35, alpha = 0.5) +
  coord_cartesian(xlim = c(0.75, 3.25), ylim = c(0.7, 3.25)) +
  scale_colour_identity() +
  theme_void()
```

`geom_mlb_logos` *ggplot2 Layer for Visualizing MLB Team Logos*

Description

`geom_mlb_logos()`, `geom_mlb_scoreboard_logos()`, and `geom_mlb_dot_logos()` are used to plot MLB team and league logos instead of points in a ggplot. It requires x, y aesthetics as well as a valid MLB team abbreviation. The latter can be checked with `valid_team_names()` but is also cleaned before being plotted.

Usage

```
geom_mlb_logos(
  mapping = NULL,
  data = NULL,
  stat = "identity",
```

```
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)

geom_mlb_scoreboard_logos(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)

geom_mlb_dot_logos(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)
```

Arguments

`mapping` Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

`data` The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function`

can be created from a `formula` (e.g. `~ head(.x, 10)`).

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `ggplot2::layer()`. These are often aesthetics, used to set an aesthetic to a fixed value. See the below section "Aesthetics" for a full list of possible arguments.

`nudge_x, nudge_y`
Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with `position`.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

Value

A ggplot2 layer (`ggplot2::layer()`) that can be added to a plot created with `ggplot2::ggplot()`.

Aesthetics

`geom_mlb_logos()`, `geom_mlb_scoreboard_logos()`, and `geom_mlb_dot_logos()` understand the following aesthetics:

`x` - The x-coordinate. Required.

`y` - The y-coordinate. Required.

`team_abbr` - The team abbreviation. Need to use Savant's abbreviation. Required.

`alpha = NULL` - The alpha channel, i.e. transparency level, as a numerical value between 0 and 1.

`colour = NULL` - The image will be colourized with this colour. Use the special character `"b/w"` to set it to black and white. For more information on valid colour names in ggplot2 see https://ggplot2.tidyverse.org/articles/ggplot2-specs.html?q=colour#colour-and-fill

`angle = 0` - The angle of the image as a numerical value between 0° and 360°.

`hjust = 0.5` - The horizontal adjustment relative to the given x coordinate. Must be a numerical value between 0 and 1.

`vjust = 0.5` - The vertical adjustment relative to the given y coordinate. Must be a numerical value between 0 and 1.

`width = 1.0` - The desired width of the image in `npc` (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is `width = 0.075` (see below examples).

`height = 1.0` - The desired height of the image in `npc` (Normalised Parent Coordinates). The default value is set to 1.0 which is *big* but it is necessary because all used values are computed relative to the default. A typical size is `height = 0.1` (see below examples).

Examples

```
library(mlbplotR)
library(ggplot2)

team_abbr <- valid_team_names()
# remove conference logos from this example
team_abbr <- team_abbr[!team_abbr %in% c("NL", "AL", "MLB")]

df <- data.frame(
  a = rep(1:6, 5),
  b = sort(rep(1:5, 6), decreasing = TRUE),
  teams = team_abbr
)

# keep alpha == 1 for all teams including an "A"
matches <- grepl("A", team_abbr)
df$alpha <- ifelse(matches, 1, 0.2)
# also set a custom fill colour for the non "A" teams
df$colour <- ifelse(matches, NA, "gray")

# scatterplot of all logos
ggplot(df, aes(x = a, y = b)) +
  geom_mlb_logos(aes(team_abbr = teams), width = 0.075) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  theme_void()

# apply alpha via an aesthetic from inside the dataset `df`
# please note that you have to add scale_alpha_identity() to use the alpha
```

```
# values in your dataset!
ggplot(df, aes(x = a, y = b)) +
  geom_mlb_scoreboard_logos(aes(team_abbr = teams, alpha = alpha), width = 0.075) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  scale_alpha_identity() +
  theme_void()

# apply alpha and colour via an aesthetic from inside the dataset `df`
# please note that you have to add scale_alpha_identity() as well as
# scale_colour_identity() to use the alpha and colour values in your dataset!
ggplot(df, aes(x = a, y = b)) +
  geom_mlb_logos(aes(team_abbr = teams, alpha = alpha, colour = colour), width = 0.075) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  scale_alpha_identity() +
  scale_colour_identity() +
  theme_void()

# apply alpha as constant for all logos
ggplot(df, aes(x = a, y = b)) +
  geom_mlb_dot_logos(aes(team_abbr = teams), width = 0.075, alpha = 0.6) +
  geom_label(aes(label = teams), nudge_y = -0.35, alpha = 0.5) +
  theme_void()

# it's also possible to plot league logos
league <- data.frame(a = 1:3, b = 0, teams = c("AL", "NL", "MLB"))
ggplot(league, aes(x = a, y = b)) +
  geom_mlb_logos(aes(team_abbr = teams), width = 0.3) +
  geom_label(aes(label = teams), nudge_y = -0.4, alpha = 0.5) +
  coord_cartesian(xlim = c(0.5,3.5), ylim = c(-0.75,.75)) +
  theme_void()
```

`ggpreview` *Preview ggplot in Specified Dimensions*

Description

This function previews a ggplot in its actual dimensions in order to see how it will look when saved. It is also significantly faster than the default preview in RStudio for ggplots created using mlbplotR.

Usage

```
ggpreview(
  plot = ggplot2::last_plot(),
  width = NA,
  height = NA,
  asp = NULL,
  dpi = 300,
  device = "png",
```

```
  units = c("in", "cm", "mm", "px"),
  scale = 1,
  limitsize = TRUE,
  bg = NULL,
  ...
)
```

Arguments

plot	Plot to save, defaults to last plot displayed.
width, height	Plot size in units expressed by the units argument. If not supplied, uses the size of the current graphics device.
asp	The aspect ratio of the plot calculated as width / height. If this is a numeric value (and not NULL) the height of the plot will be recalculated to height = width / asp.
dpi	Plot resolution. Also accepts a string input: "retina" (320), "print" (300), or "screen" (72). Applies only to raster output types.
device	Device to use. Can either be a device function (e.g. png), or one of "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf" (windows only). If NULL (default), the device is guessed based on the filename extension.
units	One of the following units in which the width and height arguments are expressed: "in", "cm", "mm" or "px".
scale	Multiplicative scaling factor.
limitsize	When TRUE (the default), ggsave() will not save images larger than 50x50 inches, to prevent the common error of specifying dimensions in pixels.
bg	Background colour. If NULL, uses the plot.background fill value from the plot theme.
...	Other arguments passed on to the graphics device function, as specified by device.

Value

No return value, called for side effects.

Examples

```
library(mlbplotR)
library(ggplot2)

team_abbr <- valid_team_names()
# remove league logos from this example
team_abbr <- team_abbr[!team_abbr %in% c("AL", "NL", "MLB")]

df <- data.frame(
  random_value = runif(length(team_abbr), 0, 1),
  teams = team_abbr
)
```

```
# use logos for x-axis
# note that the plot is assigned to the object "p"
p <- ggplot(df, aes(x = teams, y = random_value)) +
  geom_col(aes(color = teams, fill = teams), width = 0.5) +
  scale_color_mlb(type = "secondary") +
  scale_fill_mlb(alpha = 0.4) +
  theme_minimal() +
  theme(axis.text.x = element_mlb_logo())

# preview p with defined width and aspect ratio (only available in RStudio)
if (rstudioapi::isAvailable()){
  ggpreview(p, width = 5, asp = 16/9)
}
```

`gt_merge_stack_team_color`

Merge and stack text from two columns in `gt` *and color one with team colors*

Description

The `gt_merge_stack_team_color()` function takes an existing `gt` table and merges column 1 and column 2, stacking column 1's text on top of column 2's. Top text is in all caps while the lower text is bigger, bolded, and colored by the team name in another column. This is a slightly modified version of `gtExtras::gt_merge_stack()` written by Tom Mock.

Usage

```
gt_merge_stack_team_color(
  gt_object,
  col1,
  col2,
  team_col,
  font_sizes = c(12, 14),
  font_weights = c("lighter", "bold"),
  font_variants = c("small-caps"),
  color = "black"
)
```

Arguments

`gt_object`	An existing gt table object of class `gt_tbl`
`col1`	The column to stack on top.
`col2`	The column to merge and place below with the text team color that corresponds to `team_col`.

team_col	The column of team abbreviations (cleaned with clean_team_abbrs()) that match valid_team_names() for the color of the bottom text.
font_sizes	the font size for the top and bottom text in px. Can be vector of length 1 or 2. Defaults to c(12, 14)
font_weights	the font weight of the top and bottom text. Can be vector of length 1 or 2. Defaults to c("lighter", "bold")
font_variants	the font variant of the top and bottom text. Can be vector of length 1 or 2. Defaults to "small-caps"
color	The color for the top text.

Value

An object of class gt_tbl.

Examples

```
library(gt)
library(mlbplotR)

gt_merge_example <- mlbplotR::load_mlb_teams() %>%
  dplyr::slice(1:5) %>%
  dplyr::select(team_abbr, team_location, team_mascot) %>%
  gt::gt() %>%
  gt_merge_stack_team_color(col1 = "team_location",
                            col2 = "team_mascot",
                            team_col = "team_abbr")
```

---

gt_milb *Add MiLB team logos into rows of a* gt *table*

Description

The gt_fmt_milb_logo and gt_fmt_milb_dot_logo functions take an existing gt_tbl object and converts MiLB team names into team logos. This is a wrapper around gtExtras::gt_image_rows() written by Tom Mock, which is a wrapper around gt::text_transform() + gt::web_image()/ gt::local_image() with the necessary boilerplate already applied.

Usage

```
gt_fmt_milb_logo(gt_object, columns, height = 30, locations = NULL)

gt_fmt_milb_dot_logo(gt_object, columns, height = 30, locations = NULL)
```

Arguments

gt_object	An existing gt table object of class gt_tbl
columns	The columns wherein changes to cell data colors should occur. Argument has no effect if locations is not NULL.
height	The absolute height (px) of the image in the table cell
locations	If NULL (the default), the function will render logos in argument columns. Otherwise, the cell or set of cells to be associated with the team name transformation. Only the gt::cells_body(), gt::cells_stub(), gt::cells_column_labels(), and gt::cells_row_groups() helper functions can be used here. We can enclose several of these calls within a list() if we wish to make the transformation happen at different locations.

Value

An object of class gt_tbl.

Examples

```
library(gt)
library(mlbplotR)
gt_milb_example <- mlbplotR::load_milb_teams() %>%
  dplyr::filter(parent_org_name == "Texas Rangers") %>%
  dplyr::mutate(dot = team_name) %>%
  dplyr::select(team_name, dot, team_location, team_mascot) %>%
  gt::gt() %>%
  gt_fmt_milb_logo(columns = "team_name") %>%
  gt_fmt_milb_dot_logo(columns = "dot")
```

---

gt_mlb *Add MLB team logos into rows of a* gt *table*

Description

The gt_fmt_mlb_logo, gt_fmt_mlb_scoreboard_logo, and gt_fmt_mlb_dot_logo functions take an existing gt_tbl object and converts MLB team names from valid_team_names() into team logos. This is a wrapper around gtExtras::gt_image_rows() written by Tom Mock, which is a wrapper around gt::text_transform() + gt::web_image()/ gt::local_image() with the necessary boilerplate already applied.

Usage

```
gt_fmt_mlb_logo(gt_object, columns, height = 30, locations = NULL)

gt_fmt_mlb_scoreboard_logo(gt_object, columns, height = 30, locations = NULL)

gt_fmt_mlb_dot_logo(gt_object, columns, height = 30, locations = NULL)
```

Arguments

gt_object	An existing gt table object of class gt_tbl
columns	The columns wherein changes to cell data colors should occur. Argument has no effect if locations is not NULL.
height	The absolute height (px) of the image in the table cell
locations	If NULL (the default), the function will render logos in argument columns. Otherwise, the cell or set of cells to be associated with the team name transformation. Only the gt::cells_body(), gt::cells_stub(), gt::cells_column_labels(), and gt::cells_row_groups() helper functions can be used here. We can enclose several of these calls within a list() if we wish to make the transformation happen at different locations.

Value

An object of class gt_tbl.

Examples

```
library(gt)
library(mlbplotR)

df <- data.frame(team = valid_team_names()[1:5],
                 logo = valid_team_names()[1:5],
                 scoreboard_logo = valid_team_names()[1:5],
                 dot_logo = valid_team_names()[1:5])

gt_logo_example <- df %>%
 gt::gt() %>%
 gt_fmt_mlb_logo(columns = "logo") %>%
 gt_fmt_mlb_scoreboard_logo(columns = "scoreboard_logo") %>%
 gt_fmt_mlb_dot_logo(columns = "dot_logo")
```

---

gt_mlb_column_labels *Replace Team Abbreviations/Player IDs With Images In Column Labels*

Description

gt_mlb_column_labels takes in a value of a team abbreviation or player id and converts the designated column to the corresponding image.

Usage

```
gt_mlb_column_labels(
  value,
 type = c("mlb_logo", "scoreboard_logo", "dot_logo", "headshot", "dot_headshot"),
  height = 30,
  na_headshot_to_logo = TRUE
)
```

Arguments

value	What team abbreviation/player id should be replaced with an image?
type	What type of image is replacing the value?
height	The absolute height (px) of the image
na_headshot_to_logo	should NA/non player id matches return the MLB logo instead of a grayed out blank headshot? Ignored unless value is equal to "headshot". Defaults to TRUE

Value

HTML tag for image

Examples

```
library(gt)
library(mlbplotR)

df <- data.frame(BAL = 1,
                 TEX = 1,
                 LAD = 1,
                 "Mike_Trout" = 1,
                 "Shohei_Ohtani" = 1
                 )

gt_column_example <- df %>%
  gt::gt() %>%
  gt::cols_label(BAL = gt_mlb_column_labels("BAL", "mlb_logo"),
                 TEX = gt_mlb_column_labels("TEX", "scoreboard_logo"),
                 LAD = gt_mlb_column_labels("LAD", "dot_logo"),
                 "Mike_Trout" = gt_mlb_column_labels(545361, "dot_headshot"),
                 "Shohei_Ohtani" = gt_mlb_column_labels(660271, "headshot"))
```

---

gt_mlb_headshots *Render Player Headshots in 'gt' Tables*

Description

gt_fmt_mlb_headshot, gt_fmt_mlb_dot_headshot, and gt_fmt_milb_dot_headshot take an existing gt_tbl object and converts player ids into headshots. This is a wrapper around gtExtras::gt_image_rows() written by Tom Mock, which is a wrapper around gt::text_transform() + gt::web_image()/ gt::local_image() with the necessary boilerplate already applied.

Usage

```
gt_fmt_mlb_headshot(gt_object, columns, height = 30, locations = NULL)

gt_fmt_mlb_dot_headshot(gt_object, columns, height = 30, locations = NULL)

gt_fmt_milb_dot_headshot(gt_object, columns, height = 30, locations = NULL)
```

Arguments

gt_object	An existing gt table object of class gt_tbl
columns	The columns wherein changes to cell data colors should occur. Has no effect if locations is not NULL
height	The absolute height (px) of the image in the table cell
locations	If NULL (the default), the function will render logos in argument columns. Otherwise, the cell or set of cells to be associated with the team name transformation. Only the gt::cells_body(), gt::cells_stub(), gt::cells_column_labels(), and gt::cells_row_groups() helper functions can be used here. We can enclose several of these calls within a list() if we wish to make the transformation happen at different locations.

Value

An object of class gt_tbl.

Examples

```
library(gt)
library(mlbplotR)
gt_headshot_example <- data.frame(
  player_name = c(
    "Nathan Eovaldi", "Marcus Semien", "Corey Seager", "Jacob deGrom", "Chris Martin"
  ),
  savant_id1 = c(543135, 543760, 608369, 594798, 455119),
  savant_id2 = c(543135, 543760, 608369, 594798, 455119)
  ) %>%
  gt::gt() %>%
  gt_fmt_mlb_headshot(columns = "savant_id1") %>%
  gt_fmt_mlb_dot_headshot(columns = "savant_id2")
```

---

load_headshots *Load Player ID Map With Headshot Info*

Description

[Deprecated] load_headshots() has been deprecated due to lack of support. Please use baseballr::chadwick_player_lu

Usage

```
load_headshots()
```

Value

A tibble of player names and ids from various sources.

Examples

```
# load_headshots()
```

---

load_milb_teams *Load MiLB Team Colors, and Logos*

Description

Loads team information and logos - useful for plots!

Usage

```
load_milb_teams()
```

Value

A tibble of team-level abbreviations, image URLs, and league info for Minor League Baseball Teams.

See Also

Issues with this data should be filed here: https://github.com/camdenk/mlbplotR

Examples

```
load_milb_teams()
```

load_mlb_teams *Load MLB Team Colors, and Logos*

Description

Loads team colors, and logos - useful for plots!

Usage

```
load_mlb_teams()
```

Value

A tibble of team-level abbreviations, image URLs, and hex color codes.

See Also

Issues with this data should be filed here: https://github.com/camdenk/mlbplotR

Examples

```
load_mlb_teams()
```

mlb_player_tiers *Create MLB Player Tiers*

Description

This function sets up a ggplot to visualize MLB player tiers. Adapted from nflplotR

Usage

```
mlb_player_tiers(
  data,
  title = "MLB Player Tiers",
  subtitle = "Created with the #mlbplotR Tiermaker",
  caption = NULL,
 tier_desc = c(`1` = "MVP Candidates", `2` = "Very Good", `3` = "Medium", `4` = "Bad",
    `5` = "Negative WAR", `6` = "", `7` = ""),
  presort = FALSE,
  alpha = 1,
  width = 0.1,
  no_line_below_tier = NULL,
  devel = FALSE,
```

```
  background_color = "#1e1e1e",
  line_color = "#e0e0e0",
  title_color = "white",
  subtitle_color = "#8e8e93",
  caption_color = subtitle_color,
  tier_label_color = title_color,
  headshot_type = "main",
  na_headshot_to_logo = TRUE
)
```

Arguments

`data` A data frame that has to include the variables `tier_no` (the number of the tier starting from the top tier no. 1) and `player_id` (the player's MLBAM/Savant ID). If data includes the variable `tier_rank`, these ranks will be used within each tier. Otherwise, if `presort = FALSE`, the function will assume that data is already sorted and if `presort = TRUE`, teams will be sorted alphabetically within tiers.

`title` The title of the plot. If `NULL`, it will be omitted.

`subtitle` The subtitle of the plot. If `NULL`, it will be omitted.

`caption` The caption of the plot. If `NULL`, it will be omitted.

`tier_desc` A named vector consisting of the tier descriptions. The names must equal the tier numbers from `tier_no`

`presort` If `FALSE` (the default) the function assumes that the teams are already sorted within the tiers. Will otherwise sort alphabetically.

`alpha` The alpha channel of the logos, i.e. transparency level, as a numerical value between 0 and 1. Defaults to 1

`width` The desired width of the logo in `npc` (Normalised Parent Coordinates). A typical size is 0.1.

`no_line_below_tier` Vector of tier numbers. The function won't draw tier separation lines below these tiers. This is intended to be used for tiers that shall be combined (see examples).

`devel` Determines if headshots shall be rendered. If `FALSE` (the default), headshots will be rendered on each run. If `TRUE` the player ids will be plotted instead of the logos. This is much faster and helps with the plot development.

`background_color` Background color for the plot. Defaults to "#1e1e1e"

`line_color` Line color for the plot. Defaults to "#e0e0e0"

`title_color` Text color for the title. Defaults to "white"

`subtitle_color` Text color the the subtitle. Defaults to "#8e8e93"

`caption_color` Text color the the caption. Defaults to be equal to the subtitle

`tier_label_color` Text color for the tier labels. Defaults to be equal to the title

headshot_type "main" or "dot" headshots? Defaults to "main"

na_headshot_to_logo
Should NA/non-matches return the MLB logo instead of a grayed out blank headshot? Defaults to TRUE

Value

A plot object created with ggplot2::ggplot().

Examples

```
library(ggplot2)
library(dplyr, warn.conflicts = FALSE)
player_ids <- load_headshots() %>%
  head(35) %>%
  pull(savant_id)

# Build the player tiers data frame
# This is completely random!
df <- data.frame(
  tier_no = sample(1:5, length(player_ids), replace = TRUE),
  player_id = player_ids
) %>%
  dplyr::group_by(tier_no) %>%
  dplyr::mutate(tier_rank = sample(1:n(), n()))

# Plot player tiers
mlb_player_tiers(df)

# Create a combined tier which is useful for tiers with lots of players that
# should be split up in two or more rows. This is done by setting an empty
# string for the tier 5 description and removing the tier separation line
# below tier number 4.
# This example also shows how to turn off the subtitle and add a caption
mlb_player_tiers(df,
                 subtitle = NULL,
                 caption = "This is the caption",
                 tier_desc = c("1" = "MVP Candidates",
                               "2" = "Very Good",
                               "3" = "Medium",
                               "4" = "A Combined Tier",
                               "5" = ""),
                 no_line_below_tier = 4)

# For the development of the tiers, it can be useful to turn off image
# rendering as this can take quite a long time. By setting `devel = TRUE`, the
# headshots are replaced by player ids which is much faster
mlb_player_tiers(df,
                 tier_desc = c("1" = "MVP Candidates",
                               "2" = "Very Good",
                               "3" = "",
                               "4" = "A Combined Tier",
```

```
                    "5" = ""),
no_line_below_tier = c(2, 4),
devel = TRUE)
```

`mlb_team_factor` *Create Ordered MLB Team Factor*

Description

Convert a vector of MLB team abbreviations to an ordered factor by division and team name. Intended to be used for faceted plots where team logos are used in strip texts.

Usage

```
mlb_team_factor(teams)
```

Arguments

`teams`	A vector of MLB team abbreviations that should be included in `valid_team_names()`. The function tries to clean team names internally by calling `clean_team_abbrs()`.

Value

Object of class `"factor"`

Examples

```
# unsorted vector including NFL team abbreviations
teams <- c("ATL", "WSH", "MIA", "BAL", "NYY", "BOS", "PHI", "NYM", "TB", "TOR")

# defaults to sort by division and nick name in ascending order
mlb_team_factor(teams)


######### HOW TO USE IN PRACTICE #########

library(ggplot2)
library(magrittr)
# load some sample data from the ggplot2 package
plot_data <- mpg
# add a new column by randomly sampling the above defined teams vector
plot_data$team <- sample(teams, nrow(mpg), replace = TRUE)

# Now we plot the data and facet by team abbreviation. ggplot automatically
# converts the team names to a factor and sorts it alphabetically
ggplot(plot_data, aes(displ, hwy)) +
  geom_point() +
  facet_wrap(~team, ncol = 5) +
  theme_minimal()
```

```
# We'll change the order of facets by making another team name column and
# converting it to an ordered factor. Again, this defaults to sort by division,
# league, and location in ascending order.
plot_data$ordered_team <- sample(teams, nrow(mpg), replace = TRUE) %>%
  mlb_team_factor()

# Let's check how the facets are ordered now.
ggplot(plot_data, aes(displ, hwy)) +
  geom_point() +
  facet_wrap(~ordered_team, ncol = 5) +
  theme_minimal()

# The facet order looks weird because the defaults is meant to be used with
# MLB team logos. So let's use the actual logos and look at the result.
ggplot(plot_data, aes(displ, hwy)) +
  geom_point() +
  facet_wrap(~ordered_team, ncol = 5) +
  theme_minimal() +
  theme(strip.text = element_mlb_logo(size = .85))
```

`mlb_team_tiers` *Create MLB Team Tiers*

Description

This function sets up a ggplot to visualize MLB team tiers. Adapted from `nflplotR`

Usage

```
mlb_team_tiers(
  data,
  title = "MLB Team Tiers",
  subtitle = "Created with the #mlbplotR Tiermaker",
  caption = NULL,
 tier_desc = c(`1` = "World Series", `2` = "Very Good", `3` = "Medium", `4` = "Bad", `5`
    = "Tankathon", `6` = "", `7` = ""),
  presort = FALSE,
  alpha = 1,
  width = 0.075,
  no_line_below_tier = NULL,
  devel = FALSE,
  background_color = "#1e1e1e",
  line_color = "#e0e0e0",
  title_color = "white",
  subtitle_color = "#8e8e93",
  caption_color = subtitle_color,
```

```
  tier_label_color = title_color,
  logo_type = "main"
)
```

Arguments

- `data` — A data frame that has to include the variables `tier_no` (the number of the tier starting from the top tier no. 1) and `team_abbr` (the team abbreviation). `team_abbr` should be one of `valid_team_names()` and the function tries to clean team names internally by calling `clean_team_abbrs()`. If data includes the variable `tier_rank`, these ranks will be used within each tier. Otherwise, if `presort = FALSE`, the function will assume that data is already sorted and if `presort = TRUE`, teams will be sorted alphabetically within tiers.
- `title` — The title of the plot. If `NULL`, it will be omitted.
- `subtitle` — The subtitle of the plot. If `NULL`, it will be omitted.
- `caption` — The caption of the plot. If `NULL`, it will be omitted.
- `tier_desc` — A named vector consisting of the tier descriptions. The names must equal the tier numbers from `tier_no`
- `presort` — If `FALSE` (the default) the function assumes that the teams are already sorted within the tiers. Will otherwise sort alphabetically.
- `alpha` — The alpha channel of the logos, i.e. transparency level, as a numerical value between 0 and 1. Defaults to 1
- `width` — The desired width of the logo in `npc` (Normalised Parent Coordinates). A typical size is 0.075.
- `no_line_below_tier` — Vector of tier numbers. The function won't draw tier separation lines below these tiers. This is intended to be used for tiers that shall be combined (see examples).
- `devel` — Determines if logos shall be rendered. If `FALSE` (the default), logos will be rendered on each run. If `TRUE` the team abbreviations will be plotted instead of the logos. This is much faster and helps with the plot development.
- `background_color` — Background color for the plot. Defaults to "#1e1e1e"
- `line_color` — Line color for the plot. Defaults to "#e0e0e0"
- `title_color` — Text color for the title. Defaults to "white"
- `subtitle_color` — Text color the the subtitle. Defaults to "#8e8e93"
- `caption_color` — Text color the the caption. Defaults to be equal to the subtitle
- `tier_label_color` — Text color for the tier labels. Defaults to be equal to the title
- `logo_type` — What logo should be used for each team ("main", "scoreboard", or "dot")? Defaults to "main"

Value

A plot object created with `ggplot2::ggplot()`.

Examples

```
library(ggplot2)
library(dplyr, warn.conflicts = FALSE)
teams <- valid_team_names()
# remove conference logos from this example
teams <- teams[!teams %in% c("AL", "NL", "MLB")]

# Build the team tiers data frame
# This is completely random!
df <- data.frame(
  tier_no = sample(1:5, length(teams), replace = TRUE),
  team_abbr = teams
) %>%
  dplyr::group_by(tier_no) %>%
  dplyr::mutate(tier_rank = sample(1:n(), n()))

# Plot team tiers
mlb_team_tiers(df)

# Create a combined tier which is useful for tiers with lots of teams that
# should be split up in two or more rows. This is done by setting an empty
# string for the tier 5 description and removing the tier separation line
# below tier number 4.
# This example also shows how to turn off the subtitle and add a caption
mlb_team_tiers(df,
               subtitle = NULL,
               caption = "This is the caption",
               tier_desc = c("1" = "World Series",
                             "2" = "Very Good",
                             "3" = "Medium",
                             "4" = "A Combined Tier",
                             "5" = ""),
               no_line_below_tier = 4)

# For the development of the tiers, it can be useful to turn off logo image
# rendering as this can take quite a long time. By setting `devel = TRUE`, the
# logo images are replaced by team abbreviations which is much faster
mlb_team_tiers(df,
               tier_desc = c("1" = "World Series",
                             "2" = "Very Good",
                             "3" = "",
                             "4" = "A Combined Tier",
                             "5" = ""),
               no_line_below_tier = c(2, 4),
               devel = TRUE)
```

`scale_axes_mlb` *Axis Scales for MLB Team Logos*

Description

[Superseded] `scale_x_mlb()` and `scale_y_mlb()` have been superceded in favor of `element_*_logo()` functions These functions map MLB team names to their team logos and make them available as axis labels

Usage

```
scale_x_mlb(
  ...,
  expand = ggplot2::waiver(),
  guide = ggplot2::waiver(),
  position = "bottom",
  size = 12
)

scale_y_mlb(
  ...,
  expand = ggplot2::waiver(),
  guide = ggplot2::waiver(),
  position = "left",
  size = 12
)
```

Arguments

`...` Arguments passed on to `discrete_scale`

`palette` A palette function that when called with a single integer argument (the number of levels in the scale) returns the values that they should take (e.g., `scales::pal_hue()`).

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output. Also accepts rlang lambda function notation.

`limits` One of:

- `NULL` to use the default scale values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`na.translate` Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.

`na.value` If `na.translate = TRUE`, what aesthetic value should the missing values be displayed as? Does not apply to position scales where `NA` is always placed at the far right.

`aesthetics` The names of the aesthetics that this scale works with.

`labels` One of:

- `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.

`call` The `call` used to construct the scale for reporting messages.

`super` The super class to use for the constructed scale

`expand` For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the `expand` argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`position` For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.

`size` The logo size in pixels. It is applied as height for an x-scale and as width for an y-scale.

Details

The scale translates MLB team abbreviations into raw image html and places the html as axis labels. Because of the way ggplots are constructed, it is necessary to adjust the `ggplot2::theme()` after calling this scale. This can be done by calling `theme_x_mlb()` or `theme_y_mlb()` or alternatively by manually changing the relevant `axis.text` to `ggtext::element_markdown()`. However, this will only work if an underlying dependency, "gridtext", is installed with a newer version than 0.1.4

Value

A discrete ggplot2 scale created with `ggplot2::scale_x_discrete()` or `ggplot2::scale_y_discrete()`.

Examples

```
library(mlbplotR)
library(ggplot2)

team_abbr <- valid_team_names()
```

```
# remove league logos from this example
team_abbr <- team_abbr[!team_abbr %in% c("AL", "NL", "MLB")]

df <- data.frame(
  random_value = runif(length(team_abbr), 0, 1),
  teams = team_abbr
)

if (utils::packageVersion("gridtext") > "0.1.4"){
  # use logos for x-axis
  ggplot(df, aes(x = teams, y = random_value)) +
    geom_col(aes(color = teams, fill = teams), width = 0.5) +
    scale_color_mlb(type = "secondary") +
    scale_fill_mlb(alpha = 0.4) +
    scale_x_mlb() +
    theme_minimal() +
    # theme_*_mlb requires gridtext version > 0.1.4
    theme_x_mlb()

  # use logos for y-axis
  ggplot(df, aes(y = teams, x = random_value)) +
    geom_col(aes(color = teams, fill = teams), width = 0.5) +
    scale_color_mlb(type = "secondary") +
    scale_fill_mlb(alpha = 0.4) +
    scale_y_mlb() +
    theme_minimal() +
    # theme_*_mlb requires gridtext version > 0.1.4
    theme_y_mlb()
}
```

---

scale_mlb *Scales for MLB Team Colors*

Description

These functions map MLB team names to their team colors in color and fill aesthetics

Usage

```
scale_color_mlb(
  type = c("primary", "secondary"),
  values = NULL,
  ...,
  aesthetics = "colour",
  breaks = ggplot2::waiver(),
  na.value = "grey50",
  guide = NULL,
  alpha = NA
```

```
)

scale_colour_mlb(
  type = c("primary", "secondary"),
  values = NULL,
  ...,
  aesthetics = "colour",
  breaks = ggplot2::waiver(),
  na.value = "grey50",
  guide = NULL,
  alpha = NA
)

scale_fill_mlb(
  type = c("primary", "secondary"),
  values = NULL,
  ...,
  aesthetics = "fill",
  breaks = ggplot2::waiver(),
  na.value = "grey50",
  guide = NULL,
  alpha = NA
)
```

Arguments

`type` One of `"primary"` or `"secondary"` to decide which colortype to use.

`values` If `NULL` (the default) use the internal team color vectors. Otherwise a set of aesthetic values to map data values to. The values will be matched in order (usually alphabetical) with the limits of the scale, or with `breaks` if provided. If this is a named vector, then the values will be matched based on the names instead. Data values that don't match will be given `na.value`.

`...` Arguments passed on to `discrete_scale`

`limits` One of:

- `NULL` to use the default scale values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`na.translate` Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.

name The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

labels One of:

- `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.

guide A function used to create a guide or its name. See `guides()` for more information.

call The `call` used to construct the scale for reporting messages.

super The super class to use for the constructed scale

`aesthetics` Character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with. This can be useful, for example, to apply colour settings to the `colour` and `fill` aesthetics at the same time, via `aesthetics = c("colour", "fill")`.

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output

`na.value` The aesthetic value to use for missing (`NA`) values

`guide` A function used to create a guide or its name. If `NULL` (the default) no guide will be plotted for this scale. See `ggplot2::guides()` for more information.

`alpha` Factor to modify color transparency via a call to `scales::alpha()`. If `NA` (the default) no transparency will be applied. Can also be a vector of alphas. All alpha levels must be in range `[0,1]`.

Value

A discrete ggplot2 scale created with `ggplot2::scale_color_manual()` or `ggplot2::scale_fill_manual()`.

Examples

```
library(mlbplotR)
library(ggplot2)

team_abbr <- valid_team_names()
# remove league logos from this example
team_abbr <- team_abbr[!team_abbr %in% c("AL", "NL", "MLB")]

df <- data.frame(
  random_value = runif(length(team_abbr), 0, 1),
```

```
  teams = team_abbr
)
ggplot(df, aes(x = teams, y = random_value)) +
  geom_col(aes(color = teams, fill = teams), width = 0.5) +
  scale_color_mlb(type = "secondary") +
  scale_fill_mlb(alpha = 0.4) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

---

theme_mlb *Theme for MLB Team Logos*

Description

[Superseded] theme_x_mlb() and theme_y_mlb() have been superceded in favor of element_*_logo() functions These functions are convenience wrappers around a theme call that activates markdown in x-axis and y-axis labels using ggtext::element_markdown().

Usage

```
theme_x_mlb()

theme_y_mlb()
```

Details

These functions are a wrapper around the function calls ggplot2::theme(axis.text.x = ggtext::element_markdown()) as well as ggplot2::theme(axis.text.y = ggtext::element_markdown()). They are made to be used in conjunction with scale_x_mlb() and scale_y_mlb() respectively.

Value

A ggplot2 theme created with ggplot2::theme().

See Also

theme_x_mlb(), theme_y_mlb()

Examples

```
library(mlbplotR)
library(ggplot2)

team_abbr <- valid_team_names()
# remove conference logos from this example
team_abbr <- team_abbr[!team_abbr %in% c("AL", "NL", "MLB")]

df <- data.frame(
```

```
  random_value = runif(length(team_abbr), 0, 1),
  teams = team_abbr
)
if (utils::packageVersion("gridtext") > "0.1.4"){
  ggplot(df, aes(x = teams, y = random_value)) +
    geom_col(aes(color = teams, fill = teams), width = 0.5) +
    scale_color_mlb(type = "secondary") +
    scale_fill_mlb(alpha = 0.4) +
    scale_x_mlb() +
    theme_minimal() +
    # theme_*_mlb requires gridtext version > 0.1.4
    theme_x_mlb()
}
```

`valid_team_names` *Output Valid MLB Team Abbreviations*

Description

Output Valid MLB Team Abbreviations

Usage

```
valid_team_names(remove_league_info = FALSE)
```

Arguments

`remove_league_info`
: Should "AL", "NL", and "MLB" be removed from the returned vector? Defaults to `FALSE`.

Value

A vector of type `"character"`.

Examples

```
# List valid team abbreviations excluding duplicates
valid_team_names()
valid_team_names(TRUE)
```

Index

aes(), *8*, *11*, *15*, *18*, *21*, *25*

borders(), *9*, *12*, *19*, *22*, *26*

clean_team_abbrs, 2
clean_team_abbrs(), *40*, *42*

discrete_scale, *44*, *47*

element, 3
element_milb_dot_headshot *(element)*, 3
element_milb_dot_logo *(element)*, 3
element_milb_light_cap_logo *(element)*, 3
element_milb_logo *(element)*, 3
element_mlb_dark_cap_logo *(element)*, 3
element_mlb_dot_headshot *(element)*, 3
element_mlb_dot_logo *(element)*, 3
element_mlb_headshot *(element)*, 3
element_mlb_light_cap_logo *(element)*, 3
element_mlb_logo *(element)*, 3
element_mlb_scoreboard_logo *(element)*, 3
element_path *(element)*, 3
expansion(), *45*

fortify(), *8*, *12*, *15*, *18*, *21*, *25*

geom_cap_logos, 7
geom_from_path, 11
geom_from_path(), *6*
geom_lines, 14
geom_mean_lines *(geom_lines)*, 14
geom_median_lines *(geom_lines)*, 14
geom_milb_dot_headshots *(geom_mlb_headshots)*, 20
geom_milb_dot_logos *(geom_milb_logos)*, 17
geom_milb_light_cap_logos *(geom_milb_logos)*, 17
geom_milb_logos, 17
geom_mlb_dark_cap_logos *(geom_cap_logos)*, 7
geom_mlb_dot_headshots *(geom_mlb_headshots)*, 20
geom_mlb_dot_logos *(geom_mlb_logos)*, 24
geom_mlb_headshots, 20
geom_mlb_headshots(), *6*
geom_mlb_light_cap_logos *(geom_cap_logos)*, 7
geom_mlb_logos, 24
geom_mlb_logos(), *6*
geom_mlb_scoreboard_logos *(geom_mlb_logos)*, 24
ggplot(), *8*, *12*, *15*, *18*, *21*, *25*
ggplot2::geom_hline(), *16*
ggplot2::geom_vline(), *16*
ggplot2::ggplot(), *9*, *13*, *16*, *19*, *22*, *26*, *39*, *42*
ggplot2::guides(), *48*
ggplot2::layer(), *9*, *12*, *13*, *16*, *18*, *19*, *22*, *26*
ggplot2::scale_color_manual(), *48*
ggplot2::scale_fill_manual(), *48*
ggplot2::scale_x_discrete(), *45*
ggplot2::scale_y_discrete(), *45*
ggplot2::theme, *3*
ggplot2::theme(), *45*, *49*
ggpreview, 28
ggtext::element_markdown(), *45*, *49*
gt::cells_body(), *32*, *33*, *35*
gt::cells_column_labels(), *32*, *33*, *35*
gt::cells_row_groups(), *32*, *33*, *35*
gt::cells_stub(), *32*, *33*, *35*
gt_fmt_milb_dot_headshot *(gt_mlb_headshots)*, 34
gt_fmt_milb_dot_logo *(gt_milb)*, 31
gt_fmt_milb_logo *(gt_milb)*, 31
gt_fmt_mlb_dot_headshot *(gt_mlb_headshots)*, 34
gt_fmt_mlb_dot_logo *(gt_mlb)*, 32
gt_fmt_mlb_headshot *(gt_mlb_headshots)*,

34
gt_fmt_mlb_logo *(gt_mlb)*, 32
gt_fmt_mlb_scoreboard_logo *(gt_mlb)*, 32
gt_merge_stack_team_color, 30
gt_milb, 31
gt_mlb, 32
gt_mlb_column_labels, 33
gt_mlb_headshots, 34
guides(), *45*, *48*

key glyphs, *15*

lambda, *44, 45*, *47, 48*
layer position, *9*, *12*, *18*, *22*, *26*
layer stat, *8*, *12*, *18*, *22*, *26*
layer(), *15*
load_headshots, 35
load_milb_teams, 36
load_mlb_teams, 37

magick::image_read(), *13*
mlb_player_tiers, 37
mlb_team_factor, 40
mlb_team_tiers, 41

png, *29*

scale_axes_mlb, 43
scale_color_mlb *(scale_mlb)*, 46
scale_colour_mlb *(scale_mlb)*, 46
scale_fill_mlb *(scale_mlb)*, 46
scale_mlb, 46
scale_x_mlb *(scale_axes_mlb)*, 43
scale_x_mlb(), *49*
scale_y_mlb *(scale_axes_mlb)*, 43
scale_y_mlb(), *49*
scales::alpha(), *48*
scales::pal_hue(), *44*

theme_mlb, 49
theme_x_mlb *(theme_mlb)*, 49
theme_x_mlb(), *45*, *49*
theme_y_mlb *(theme_mlb)*, 49
theme_y_mlb(), *45*, *49*

valid_team_names, 50
valid_team_names(), *7*, *24*, *40*, *42*