

Package: ggvenn (via r-universe)

September 16, 2024

Title Draw Venn Diagram by 'ggplot2'

Version 0.1.16

Author Linlin Yan [aut, cre] (<<https://orcid.org/0000-0002-4990-6239>>)

Maintainer Linlin Yan <yanlinlin82@gmail.com>

Description An easy-to-use way to draw pretty venn diagram by 'ggplot2'.

URL <https://yanlinlin82.github.io/ggvenn/>,
<https://github.com/yanlinlin82/ggvenn>

BugReports <https://github.com/yanlinlin82/ggvenn/issues>

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Depends ggplot2

Suggests testthat (>= 3.0.0)

Imports dplyr, grid, scales, lifecycle

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Repository <https://yanlinlin82.r-universe.dev>

RemoteUrl <https://github.com/yanlinlin82/ggvenn>

RemoteRef HEAD

RemoteSha 0d6914a6134b2cae60051cbc1dc9850c2a2077e4

Contents

data_frame_to_list	2
geom_venn	2
ggvenn	5
list_to_data_frame	8

Index	9
--------------	----------

`data_frame_to_list` *Utility function for data type conversion.*

Description

Utility function for data type conversion.

Usage

```
data_frame_to_list(x)
```

Arguments

`x` A data.frame with logical columns representing sets.

Value

A list of sets.

Examples

```
d <- dplyr::tibble(name = 1:6,
  A = c(rep(TRUE, 5), FALSE),
  B = rep(c(FALSE, TRUE), each = 3))
print(d)
data_frame_to_list(d)
```

`geom_venn` *Plot venn diagram as a ggplot layer object. It supports only data frame as input.*

Description

Plot venn diagram as a ggplot layer object. It supports only data frame as input.

Usage

```
geom_venn(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  set_names = NULL,
  show_set_totals = "none",
  show_stats = "cp",
```

```

show_percentage = deprecated(),
digits = 1,
label_sep = ", ",
count_column = NULL,
show_outside = c("auto", "none", "always"),
auto_scale = FALSE,
fill_color = c("blue", "yellow", "green", "red"),
fill_alpha = 0.5,
stroke_color = "black",
stroke_alpha = 1,
stroke_size = 1,
stroke_linetype = "solid",
set_name_color = "black",
set_name_size = 6,
text_color = "black",
text_size = 4
)

```

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (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	A <code>data.frame</code> or a list as input data.
stat	The statistical transformation to use on the data for this layer, as a string.
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: <ul style="list-style-type: none"> • The result of calling a position function, such as <code>position_jitter()</code>. 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 <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, 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 layer() 's <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored. <ul style="list-style-type: none"> • 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, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. 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.

<code>set_names</code>	Set names, use column names if omitted.
<code>show_set_totals</code>	Show total count (c) and/or percentage (p) for each set. Pass a string like "cp" to show both. Any other string like "none" to hide both.
<code>show_stats</code>	Show count (c) and/or percentage (p) for each set. Pass a string like "cp" to show both.
<code>show_percentage</code>	Show percentage for each set. Deprecated, use <code>show_stats</code> instead.
<code>digits</code>	The desired number of digits after the decimal point
<code>label_sep</code>	separator character for displaying elements.
<code>count_column</code>	Specify column for element repeat count.
<code>show_outside</code>	Show outside elements (not belongs to any set).
<code>auto_scale</code>	Allow automatically resizing circles according to element counts.
<code>fill_color</code>	Filling colors in circles.
<code>fill_alpha</code>	Transparency for filling circles.
<code>stroke_color</code>	Stroke color for drawing circles.
<code>stroke_alpha</code>	Transparency for drawing circles.
<code>stroke_size</code>	Stroke size for drawing circles.
<code>stroke_linetype</code>	Line type for drawing circles.
<code>set_name_color</code>	Text color for set names.
<code>set_name_size</code>	Text size for set names.
<code>text_color</code>	Text color for intersect contents.
<code>text_size</code>	Text size for intersect contents.

Value

The ggplot object to print or save to file.

See Also

`ggvenn`

Examples

```

library(ggvenn)

# use data.frame as input
d <- dplyr::tibble(value = c(1, 2, 3, 5, 6, 7, 8, 9),
  `Set 1` = c(TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, FALSE, TRUE),
  `Set 2` = c(TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, TRUE),
  `Set 3` = c(TRUE, TRUE, FALSE, FALSE, FALSE, FALSE, TRUE, TRUE),
  `Set 4` = c(FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, FALSE, FALSE))

# ggplot gramma
ggplot(d) +
  geom_venn(aes(A = `Set 1`, B = `Set 2`)) +
  coord_fixed() +
  theme_void()
ggplot(d) +
  geom_venn(aes(A = `Set 1`, B = `Set 2`, C = `Set 3`)) +
  coord_fixed() +
  theme_void()
ggplot(d) +
  geom_venn(aes(A = `Set 1`, B = `Set 2`, C = `Set 3`, D = `Set 4`)) +
  coord_fixed() +
  theme_void()

# set fill color
ggplot(d) +
  geom_venn(aes(A = `Set 1`, B = `Set 2`), fill_color = c("red", "blue")) +
  coord_fixed() +
  theme_void()

# hide percentage
ggplot(d) +
  geom_venn(aes(A = `Set 1`, B = `Set 2`), show_stats = 'c') +
  coord_fixed() +
  theme_void()

# change precision of percentages
ggplot(d) +
  geom_venn(aes(A = `Set 1`, B = `Set 2`), digits = 2) +
  coord_fixed() +
  theme_void()

# show elements instead of count/percentage
ggplot(d) +
  geom_venn(aes(A = `Set 1`, B = `Set 2`, C = `Set 3`, D = `Set 4`, label = value)) +
  coord_fixed() +
  theme_void()

```

Description

Plot venn diagram as an independent function. It supports both data frame and list as input.

Usage

```
ggvenn(
  data,
  columns = NULL,
  show_elements = FALSE,
  show_set_totals = "none",
  show_stats = "cp",
  show_percentage = lifecycle::deprecated(),
  digits = 1,
  fill_color = c("blue", "yellow", "green", "red"),
  fill_alpha = 0.5,
  stroke_color = "black",
  stroke_alpha = 1,
  stroke_size = 1,
  stroke_linetype = "solid",
  set_name_color = "black",
  set_name_size = 6,
  text_color = "black",
  text_size = 4,
  label_sep = ", ",
  count_column = NULL,
  show_outside = c("auto", "none", "always"),
  auto_scale = FALSE,
  comma_sep = FALSE,
  padding = 0.2
)
```

Arguments

<code>data</code>	A data.frame or a list as input data.
<code>columns</code>	A character vector use as index to select columns/elements.
<code>show_elements</code>	Show set elements instead of count/percentage.
<code>show_set_totals</code>	Show total count (c) and/or percentage (p) for each set. Pass a string like "cp" to show both. Any other string like "none" to hide both.
<code>show_stats</code>	Show count (c) and/or percentage (p) for each set.
<code>show_percentage</code>	Show percentage for each set. Deprecated, use <code>show_stats</code> instead. Pass a string like "cp" to show both. Any other string like "none" to hide both.
<code>digits</code>	The desired number of digits after the decimal point
<code>fill_color</code>	Filling colors in circles.
<code>fill_alpha</code>	Transparency for filling circles.

stroke_color	Stroke color for drawing circles.
stroke_alpha	Transparency for drawing circles.
stroke_size	Stroke size for drawing circles.
stroke_linetype	Line type for drawing circles.
set_name_color	Text color for set names.
set_name_size	Text size for set names.
text_color	Text color for intersect contents.
text_size	Text size for intersect contents.
label_sep	Separator character for displaying elements.
count_column	Specify column for element repeat count.
show_outside	Show outside elements (not belongs to any set).
auto_scale	Allow automatically resizing circles according to element counts.
comma_sep	Whether to use comma as separator for displaying numbers.
padding	Padding for the plot. Change this to allow longer labels to be displayed.

Value

The ggplot object to print or save to file.

See Also

geom_venn

Examples

```
library(ggvenn)

# use list as input
a <- list(`Set 1` = c(1, 3, 5, 7),
          `Set 2` = c(1, 5, 9),
          `Set 3` = c(1, 2, 8),
          `Set 4` = c(6, 7))
ggvenn(a, c("Set 1", "Set 2"))
ggvenn(a, c("Set 1", "Set 2", "Set 3"))
ggvenn(a)

# use data.frame as input
d <- dplyr::tibble(value = c(1, 2, 3, 5, 6, 7, 8, 9),
                  `Set 1` = c(TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, FALSE, TRUE),
                  `Set 2` = c(TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, TRUE),
                  `Set 3` = c(TRUE, TRUE, FALSE, FALSE, FALSE, FALSE, TRUE, TRUE),
                  `Set 4` = c(FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, FALSE, FALSE))
ggvenn(d, c("Set 1", "Set 2"))
ggvenn(d, c("Set 1", "Set 2", "Set 3"))
ggvenn(d)
```

```
# set fill color
ggvenn(d, c("Set 1", "Set 2"), fill_color = c("red", "blue"))

# hide percentage
ggvenn(d, c("Set 1", "Set 2"), show_stats = 'c')

# change precision of percentages
ggvenn(d, c("Set 1", "Set 2"), digits = 2)

# show elements instead of count/percentage
ggvenn(a, show_elements = TRUE)
ggvenn(d, show_elements = "value")
```

list_to_data_frame *Utility function for data type conversion.*

Description

Utility function for data type conversion.

Usage

```
list_to_data_frame(x)
```

Arguments

x A list of sets.

Value

A data.frame with logical columns representing sets.

Examples

```
a <- list(A = 1:5, B = 4:6)
print(a)
list_to_data_frame(a)
```


Index

`aes()`, 3

`data_frame_to_list`, 2

`geom_venn`, 2

`ggvenn`, 5

key glyphs, 4

layer position, 3

`layer()`, 3, 4

`list_to_data_frame`, 8