

Package ‘ggtext’

August 28, 2026

Type Package

Title Improved Text Rendering Support for 'ggplot2'

Version 0.2.0

Description A 'ggplot2' extension that enables the rendering of complex formatted plot labels (titles, subtitles, facet labels, axis labels, etc.). Text boxes with automatic word wrap are also supported.

URL <https://wilkelab.org/ggtext/>

BugReports <https://github.com/wilkelab/ggtext/issues>

License GPL-2

Depends R (>= 3.5)

Imports ggplot2 (>= 3.3.0), grid, gridtext, rlang, scales

Suggests cowplot, dplyr, glue, knitr, rmarkdown, testthat, vdiff

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Claus O. Wilke [aut] (ORCID: <<https://orcid.org/0000-0002-7470-9261>>),
Brenton M. Wiernik [aut, cre] (ORCID:
<<https://orcid.org/0000-0001-9560-6336>>, Twitter: @bmwiernik)

Maintainer Brenton M. Wiernik <brenton@wiernik.org>

Repository CRAN

Date/Publication 2026-08-28 10:30:07 UTC

Contents

draw_key_richtext	2
element_markdown	3
element_textbox	4

geom_richtext	7
geom_textbox	11
ggtext	15
Index	17

draw_key_richtext	<i>Key glyph for rich text</i>
-------------------	--------------------------------

Description

This is a function for rendering the key in the rich text style when a geom needs to be displayed in a legend. It is designed to be provided to a layer’s key_glyph argument, either as a function or as key_glyph = "richtext".

Usage

```
draw_key_richtext(data, params, size)
```

Arguments

- data A single row data frame containing the scaled aesthetics to display in this key
- params A list of additional parameters supplied to the geom.
- size Width and height of key in mm.

Value

A richtext_grob that represents formatted text.

Examples

```
library(ggplot2)

ggplot(mtcars, aes(wt, mpg, label = rownames(mtcars))) +
  geom_richtext(aes(colour = factor(cyl)), key_glyph = "richtext")
```

element_markdown	<i>Theme element that enables markdown text.</i>
------------------	--

Description

Theme element that enables markdown text.

Usage

```
element_markdown(  
  family = NULL,  
  face = NULL,  
  size = NULL,  
  colour = NULL,  
  fill = NULL,  
  box.colour = NULL,  
  linetype = NULL,  
  linewidth = NULL,  
  hjust = NULL,  
  vjust = NULL,  
  halign = NULL,  
  valign = NULL,  
  angle = NULL,  
  lineheight = NULL,  
  margin = NULL,  
  padding = NULL,  
  r = NULL,  
  color = NULL,  
  box.color = NULL,  
  align_widths = NULL,  
  align_heights = NULL,  
  rotate_margins = NULL,  
  debug = FALSE,  
  inherit.blank = FALSE  
)
```

Arguments

family	Font family
face	Font face
size	Font size
colour, color	Text color
fill	Fill color of the enclosing box
box.colour, box.color	Line color of the enclosing box (if different from the text color)

linetype	Line type of the enclosing box (like lty in base R)
linewidth	Line width of the enclosing box (measured in mm, just like size in <code>ggplot2::element_line()</code>).
hjust	Horizontal justification
vjust	Vertical justification
halign	Horizontal justification
valign	Vertical justification
angle	Angle (in degrees)
lineheight	Line height
padding, margin	Padding and margins around the text box. See <code>gridtext::richtext_grob()</code> for details.
r	Unit value specifying the corner radius of the box
align_widths, align_heights	Should multiple elements be aligned by their widths or height? See <code>gridtext::richtext_grob()</code> for details.
rotate_margins	Should margins get rotated in frame with rotated text? If TRUE, the margins are applied relative to the text direction. If FALSE, the margins are applied relative to the plot direction, i.e., the top margin, for example, is always placed above the text label, regardless of the direction in which the text runs. The default is FALSE, which mimics the behavior of <code>element_text()</code> .
debug	Draw a debugging box around each label
inherit.blank	See <code>ggplot2::margin()</code> for details.

Value

A ggplot2 theme element that can be used inside a `ggplot2::theme()` call.

See Also

`gridtext::richtext_grob()`, `element_textbox()`, `geom_richtext()`

element_textbox	<i>Theme element that enables markdown text in a box.</i>
-----------------	---

Description

The theme elements `element_textbox()` and `element_textbox_simple()` enable Markdown text in a box, with word wrap. Both functions implement exactly the same functionality; they only differ in the default values for the various element values. `element_textbox()` sets all values that are not specified to NULL, as is the usual practice in ggplot2 themes. These missing values are usually completed by inheritance from parent theme elements. By contrast, `element_textbox_simple()` provides meaningful default values for many of the values that are not usually defined in ggplot2 themes. This makes it simpler to use a textbox element in the context of an existing theme.

Usage

```
element_textbox(  
  family = NULL,  
  face = NULL,  
  size = NULL,  
  colour = NULL,  
  fill = NULL,  
  box.colour = NULL,  
  linetype = NULL,  
  linewidth = NULL,  
  hjust = NULL,  
  vjust = NULL,  
  halign = NULL,  
  valign = NULL,  
  lineheight = NULL,  
  margin = NULL,  
  padding = NULL,  
  width = NULL,  
  height = NULL,  
  minwidth = NULL,  
  maxwidth = NULL,  
  minheight = NULL,  
  maxheight = NULL,  
  r = NULL,  
  orientation = NULL,  
  color = NULL,  
  box.color = NULL,  
  debug = FALSE,  
  inherit.blank = FALSE  
)  
  
element_textbox_simple(  
  family = NULL,  
  face = NULL,  
  size = NULL,  
  colour = NULL,  
  fill = NA,  
  box.colour = NULL,  
  linetype = 0,  
  linewidth = 0.5,  
  hjust = 0.5,  
  vjust = 0.5,  
  halign = 0,  
  valign = 1,  
  lineheight = 1.2,  
  margin = ggplot2::margin(0, 0, 0, 0),  
  padding = ggplot2::margin(0, 0, 0, 0),  
  width = grid::unit(1, "npc"),
```

```

height = NULL,
minwidth = NULL,
maxwidth = NULL,
minheight = NULL,
maxheight = NULL,
r = grid::unit(0, "pt"),
orientation = "upright",
color = NULL,
box.color = NULL,
debug = FALSE,
inherit.blank = FALSE
)

```

Arguments

family	Font family
face	Font face
size	Font size (in pt)
colour, color	Text color
fill	Fill color of the enclosing box
box.colour, box.color	Line color of the enclosing box (if different from the text color)
linetype	Line type of the enclosing box (like lty in base R)
linewidth	Line width of the enclosing box (measured in mm, just like size in ggplot2::element_line()).
hjust	Horizontal justification
vjust	Vertical justification
halign	Horizontal justification
valign	Vertical justification
lineheight	Line height, in multiples of the font size
padding, margin	Padding and margins around the text box. See gridtext::textbox_grob() for details.
width, height	Unit objects specifying the width and height of the textbox, as in gridtext::textbox_grob() .
minwidth, minheight, maxwidth, maxheight	Min and max values for width and height. Set to NULL to impose neither a minimum nor a maximum.
r	Unit value specifying the corner radius of the box
orientation	Orientation of the text box. See gridtext::textbox_grob() for details.
debug	Not implemented.
inherit.blank	See ggplot2::margin() for details.

Value

A ggplot2 theme element that can be used inside a [ggplot2::theme\(\)](#) call.

See Also

[gridtext::textbox_grob\(\)](#), [element_markdown\(\)](#), [geom_textbox\(\)](#)

Examples

```
library(ggplot2)

ggplot(mtcars, aes(displacement, mpg)) +
  geom_point() +
  labs(
    title = "<b><span style = 'font-size:13pt'>Fuel economy vs. engine displacement</span></b><br>
    Lorem ipsum *dolor sit amet,* consectetur adipiscing elit, **sed do eiusmod tempor
    incididunt** ut labore et dolore magna aliqua. Ut enim ad minim veniam,
    <span style = 'color:red;'>quis nostrud exercitation ullamco</span> laboris nisi ut
    aliquip ex ea commodo consequat.",
    x = "displacement (in<sup>3</sup>)",
    y = "Miles per gallon (mpg)"
  ) +
  theme(
    plot.title.position = "plot",
    plot.title = element_textbox_simple(
      size = 10,
      padding = margin(5.5, 5.5, 5.5, 5.5),
      margin = margin(0, 0, 5.5, 0),
      fill = "cornsilk"
    ),
    axis.title.x = element_textbox_simple(
      width = NULL,
      padding = margin(4, 4, 4, 4),
      margin = margin(4, 0, 0, 0),
      linetype = 1,
      r = grid::unit(8, "pt"),
      fill = "azure1"
    ),
    axis.title.y = element_textbox_simple(
      hjust = 0,
      orientation = "left-rotated",
      minwidth = unit(1, "in"),
      maxwidth = unit(2, "in"),
      padding = margin(4, 4, 2, 4),
      margin = margin(0, 0, 2, 0),
      fill = "lightsteelblue1"
    )
  )
)
```

Description

This geom draws text labels similar to `ggplot2::geom_label()`, but formatted using basic mark-down/html. Parameter and aesthetic names follow the conventions of `ggplot2::geom_label()`, and therefore the appearance of the frame around the label is controlled with `label.colour`, `label.padding`, `label.margin`, `label.size`, `label.r`, even though the same parameters are called `box.colour`, `box.padding`, `box.margin`, `box.size`, and `box.r` in `geom_textbox()`. Most styling parameters can be used as aesthetics and can be applied separately to each text label drawn. The exception is styling parameters that are specified as grid units (e.g., `label.padding` or `label.r`), which can only be specified for all text labels at once. See examples for details.

Usage

```
geom_richtext(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  label.padding = unit(c(0.25, 0.25, 0.25, 0.25), "lines"),
  label.margin = unit(c(0, 0, 0, 0), "lines"),
  label.r = unit(0.15, "lines"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

- | | |
|---------|---|
| mapping | Set of aesthetic mappings created by <code>aes()</code> . 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 | <p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p> |
| stat | <p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A <code>Stat</code> gproto subclass, for example <code>StatCount</code>. |

	• A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, 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 <code>position</code> argument accepts the following: • 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 <code>layer()</code>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> 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. • 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 <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> 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.
nudge_x	Shifts text horizontally, as in geom_text() .
nudge_y	Shifts text vertically, as in geom_text() .
label.padding	Amount of padding around label. Defaults to 0.25 lines.
label.margin	Unit vector of length four specifying the margin outside the text label.
label.r	Radius of rounded corners. Defaults to 0.15 lines.
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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. annotation_borders() .

Value

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

Aesthetics

`geom_richtext()` understands the following aesthetics (required aesthetics are in bold; select aesthetics are annotated):

- x
- y
- label
- alpha
- angle
- colour Default color of label text and label outline.
- family
- fontface
- fill Default fill color of label background.
- group
- hjust
- label.colour Color of label outline. Overrides colour.
- label.size Width of label outline.
- lineheight
- size Default font size of label text.
- text.colour Color of label text. Overrides colour.
- vjust

See Also

[geom_textbox\(\)](#), [element_markdown\(\)](#)

Examples

```

library(ggplot2)

df <- data.frame(
  label = c(
    "Some text in bold",
    "Linebreaks<br>Linebreaks<br>Linebreaks",
    "x<sup>2</sup> + 5x + c<sub>i</sub>",
    "Some <span style='color:blue'>blue text in bold</span><br>And italics text.<br>
      And some <span style='font-size:18pt; color:black'>large</span> text."
  ),
  x = c(.2, .1, .5, .9),
  y = c(.8, .4, .1, .5),
  hjust = c(0.5, 0, 0, 1),
  vjust = c(0.5, 1, 0, 0.5),
  angle = c(0, 0, 45, -45),
  color = c("black", "blue", "black", "red"),
  fill = c("cornsilk", "white", "lightblue1", "white")
)

ggplot(df) +
  aes(
    x, y, label = label, angle = angle, color = color, fill = fill,
    hjust = hjust, vjust = vjust
  ) +
  geom_richtext() +
  geom_point(color = "black", size = 2) +
  scale_color_identity() +
  scale_fill_identity() +
  xlim(0, 1) + ylim(0, 1)

# labels without frame or background are also possible
ggplot(df) +
  aes(
    x, y, label = label, angle = angle, color = color,
    hjust = hjust, vjust = vjust
  ) +
  geom_richtext(
    fill = NA, label.color = NA, # remove background and outline
    label.padding = grid::unit(rep(0, 4), "pt") # remove padding
  ) +
  geom_point(color = "black", size = 2) +
  scale_color_identity() +
  xlim(0, 1) + ylim(0, 1)

```

Description

Draw boxes of defined width and height containing word-wrapped text. Multiple boxes can be drawn at once. Most styling parameters can be used as aesthetics and can be applied separately to each text box drawn. The exception is styling parameters that are specified as grid units (e.g., `box.padding` or `box.r`), which can only be specified for all text boxes at once. See examples for details.

Usage

```
geom_textbox(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  nudge_x = 0,
  nudge_y = 0,
  box.padding = unit(c(5.5, 5.5, 5.5, 5.5), "pt"),
  box.margin = unit(c(0, 0, 0, 0), "pt"),
  box.r = unit(5.5, "pt"),
  width = unit(2, "inch"),
  minwidth = NULL,
  maxwidth = NULL,
  height = NULL,
  minheight = NULL,
  maxheight = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

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	The data to be displayed in this layer. There are three options: If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot(). A <code>data.frame</code>, 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 <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).
stat	The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to over-

ride 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 `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.

`nudge_x, nudge_y`

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

`box.padding`

Unit vector of length four specifying the padding inside the text box.

<code>box.margin</code>	Unit vector of length four specifying the margin outside the text box.
<code>box.r</code>	Unit vector of length one specifying the radius of the box.
<code>width, height</code>	Unit values specifying the width and height of the text box (including margins!). If <code>height = NULL</code> (the default), the height is chosen automatically to accommodate all the text.
<code>minwidth, maxwidth, minheight, maxheight</code>	Unit values specifying the minimum and maximum values for width and height, respectively. If set to <code>NULL</code> , are not enforced.
<code>na.rm</code>	If <code>FALSE</code> , the default, missing values are removed with a warning. If <code>TRUE</code> , missing values are silently removed.
<code>show.legend</code>	logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use <code>TRUE</code> . If <code>NA</code> , all levels are shown in legend, but unobserved levels are omitted.
<code>inherit.aes</code>	If <code>FALSE</code> , 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. <code>annotation_borders()</code> .

Value

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

Aesthetics

`geom_textbox()` understands the following aesthetics (required aesthetics are in bold; select aesthetics are annotated):

- `x`
- `y`
- `label`
- `alpha`
- `box.colour` Color of box outline. Overrides `colour`.
- `box.size` Width of box outline.
- `colour` Default color of box text and box outline.
- `family`
- `fontface`
- `fill` Default fill color of box background.
- `group`
- `halign` Horizontal alignment of text inside box.
- `hjust` Horizontal alignment of box.
- `lineheight`
- `orientation` One of "upright", "left-rotated", "right-rotated", "inverted".

- size Default font size of box text.
- text.colour Color of box text. Overrides colour.
- valign Vertical alignment of text inside box.
- vjust Vertical alignment of box.

See Also

[geom_richtext\(\)](#), [element_textbox\(\)](#)

Examples

```
library(ggplot2)

df <- data.frame(
  label = rep("Lorem ipsum dolor **sit amet,** consectetur adipiscing elit,
    sed do *eiusmod tempor incididunt* ut labore et dolore magna
    aliqua.", 2),
  x = c(0, .6),
  y = c(1, .6),
  hjust = c(0, 0),
  vjust = c(1, 0),
  orientation = c("upright", "right-rotated"),
  color = c("black", "blue"),
  fill = c("cornsilk", "white")
)

ggplot(df) +
  aes(
    x, y, label = label, color = color, fill = fill,
    hjust = hjust, vjust = vjust,
    orientation = orientation
  ) +
  geom_textbox(width = unit(0.4, "npc")) +
  geom_point(color = "black", size = 2) +
  scale_discrete_identity(aesthetics = c("color", "fill", "orientation")) +
  xlim(0, 1) + ylim(0, 1)
```

Description

The ggtext package implements both geoms ([geom_richtext\(\)](#), [geom_textbox\(\)](#)) and theme elements ([element_markdown\(\)](#), [element_textbox\(\)](#)) for improved text rendering with ggplot2.

Author(s)

Maintainer: Brenton M. Wiernik <brenton@wiernik.org> ([ORCID](#)) (Twitter: @bmwiernik)

Authors:

- Brenton M. Wiernik <brenton@wiernik.org> ([ORCID](#)) (Twitter: @bmwiernik)
- Claus O. Wilke <wilke@austin.utexas.edu> ([ORCID](#))

See Also

Useful links:

- <https://wilkelab.org/ggtext/>
- Report bugs at <https://github.com/wilkelab/ggtext/issues>

Index

`aes()`, [8](#), [12](#)
`annotation_borders()`, [10](#), [14](#)

`draw_key_richtext`, [2](#)

`element_markdown`, [3](#)
`element_markdown()`, [7](#), [10](#), [15](#)
`element_textbox`, [4](#)
`element_textbox()`, [4](#), [15](#)
`element_textbox_simple`
 (`element_textbox`), [4](#)

`fortify()`, [8](#), [12](#)

`geom_richtext`, [7](#)
`geom_richtext()`, [4](#), [15](#)
`geom_text()`, [9](#)
`geom_textbox`, [11](#)
`geom_textbox()`, [7](#), [8](#), [10](#), [15](#)
`GeomRichText` (`geom_richtext`), [7](#)
`GeomRichText` (`geom_richtext`), [7](#)
`GeomTextBox` (`geom_richtext`), [7](#)
`ggplot()`, [8](#), [12](#)
`ggplot2::element_line()`, [4](#), [6](#)
`ggplot2::geom_label()`, [8](#)
`ggplot2::ggplot()`, [10](#), [14](#)
`ggplot2::margin()`, [4](#), [6](#)
`ggplot2::theme()`, [4](#), [6](#)
`ggtext`, [15](#)
`ggtext-package` (`ggtext`), [15](#)
`gridtext::richtext_grob()`, [4](#)
`gridtext::textbox_grob()`, [6](#), [7](#)

`key glyphs`, [9](#), [13](#)

`layer position`, [9](#), [13](#)
`layer stat`, [9](#), [13](#)
`layer()`, [9](#), [13](#)

`richtext_grob`, [2](#)