

Package: ggplot2.utils (via r-universe)

September 4, 2024

Type Package

Title Selected Utilities Extending 'ggplot2'

Version 0.3.2.9001

Date 2024-08-05

Description Selected utilities, in particular 'geoms' and 'stats' functions, extending the 'ggplot2' package. This package imports functions from 'EnvStats' <[doi:10.1007/978-1-4614-8456-1](https://doi.org/10.1007/978-1-4614-8456-1)> by Millard (2013), 'ggpp' <<https://CRAN.R-project.org/package=ggpp>> by Aphalo et al. (2023) and 'ggstats' <[doi:10.5281/zenodo.10183964](https://doi.org/10.5281/zenodo.10183964)> by Larmarange (2023), and then exports them. This package also contains modified code from 'ggquickedda' <<https://CRAN.R-project.org/package=ggquickedda>> by Mouksassi et al. (2023) for Kaplan-Meier lines and ticks additions to plots. All functions are tested to make sure that they work reliably.

License Apache License 2.0

URL <https://insightengineering.github.io/ggplot2.utils/>

BugReports <https://github.com/insightengineering/ggplot2.utils/issues>

Depends ggplot2 (>= 3.3.0), R (>= 3.6)

Imports checkmate, EnvStats, ggpp, ggstats, survival

Suggests dplyr, lifecycle, testthat (>= 3.0.0), tibble, vdiffr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Collate 'geom_km.R' 'geom_km_ticks.R' 'geom_table.R' 'ggproto.R' 'package.R' 'stat_km_compute.R' 'stat_km.R' 'stat_km_ticks.R' 'stat_n_text.R' 'stat_prop.R' 'table_themes.R' 'ttheme_set.R'

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

RemoteUrl <https://github.com/insightsengineering/ggplot2.utils>

RemoteRef HEAD

RemoteSha b8dff78375a7560dc412265e507acc4c12c165ca

Contents

ggplot2.utils-package	2
geom_km	3
geom_km_ticks	5
geom_table	8
stat_km	8
stat_km_ticks	11
stat_n_text	13
stat_prop	13
ttheme_gtdefault	14
ttheme_set	14
Index	15

ggplot2.utils-package ggplot2.utils *Package*

Description

ggplot2.utils provides simple access to utility functions extending ggplot2.

Details

Currently all of the functions are imported from other extension packages:

- ggpp: [geom_table\(\)](#) and associated functions.
- EnvStats: [stat_n_text\(\)](#) and associated functions.
- ggstats: [stat_prop\(\)](#) and associated functions.

Author(s)

Maintainer: Daniel Sabanés Bové <daniel.sabanes_bove@rconis.com>

Authors:

- Samer Mouksassi (wrote original Kaplan-Meier code)
- Michael Sachs (wrote original Kaplan-Meier code)

Other contributors:

- F. Hoffmann-La Roche AG [copyright holder, funder]

See Also

Useful links:

- <https://insightsengineering.github.io/ggplot2.utils/>
- Report bugs at <https://github.com/insightsengineering/ggplot2.utils/issues>

 geom_km

Add a Kaplan-Meier Survival Curve

Description

[Experimental] Adds the Kaplan-Meier survival curve.

Usage

```
geom_km(
  mapping = NULL,
  data = NULL,
  stat = "km",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  na.rm = 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 | 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 <code>ggplot()</code> .
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.
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 override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: <ul style="list-style-type: none"> • A <code>Stat</code> ggproto subclass, for example <code>StatCount</code>. |

	<ul style="list-style-type: none"> • 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	<p>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:</p> <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.
show.legend	<p>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.</p>
inherit.aes	<p>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. <code>borders()</code>.</p>
na.rm	<p>If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.</p>
...	<p>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.</p> <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 <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.

Aesthetics

geom_km() understands the following aesthetics (required aesthetics in bold):

- x: the survival/censoring times, automatically mapped by `stat_km()`.
- y: the survival probability estimates, automatically mapped by `stat_km()`.
- alpha
- color
- linetype
- linewidth

Author(s)

Inspired by geom_km written by Michael Sachs (in ggkm) and Samer Mouksassi (in ggquickeda). Here we directly use `ggplot2::geom_step()` instead of the more general `ggplot2::geom_path()`.

See Also

The default stat for this geom is `stat_km()`.

Examples

```
library(ggplot2)
sex <- rbinom(250, 1, .5)
df <- data.frame(
  time = exp(rnorm(250, mean = sex)),
  status = rbinom(250, 1, .75),
  sex = sex
)
ggplot(df, aes(time = time, status = status, color = factor(sex))) +
  geom_km()
```

geom_km_ticks

Add Tick Marks to a Kaplan-Meier Survival Curve

Description

[Experimental] Adds tickmarks at the times when there are censored observations but no events.

Usage

```
geom_km_ticks(
  mapping = NULL,
  data = NULL,
  stat = "km_ticks",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
```

```

    na.rm = 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	<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 ggplot().</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 fortify() 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 Stat ggproto 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	<p>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:</p> <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.
show.legend	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.
inherit.aes	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. borders() .
na.rm	If <code>FALSE</code> , the default, missing values are removed with a warning. If <code>TRUE</code> , missing values are silently removed.

... 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.

Aesthetics

`geom_km_ticks()` understands the following aesthetics (required aesthetics in bold):

- **x**: the survival/censoring times, automatically mapped by `stat_km_ticks()`.
- **y**: the survival probability estimates, automatically mapped by `stat_km_ticks()`.
- **alpha**
- **color**
- **shape**
- **size**
- **stroke**
- **fill**

Author(s)

Michael Sachs (in `ggkm`), Samer Mouksassi (in `ggquickeda`).

See Also

The default `stat` for this geom is `stat_km_ticks()`.

Examples

```
library(ggplot2)
sex <- rbinom(250, 1, .5)
df <- data.frame(
  time = exp(rnorm(250, mean = sex)),
  status = rbinom(250, 1, .75),
  sex = sex
)
ggplot(df, aes(time = time, status = status, color = factor(sex), group = factor(sex))) +
  geom_km() +
  geom_km_ticks(col = "black")
```

geom_table

Inset tables

Description

[Experimental]

See [ggpp::geom_table\(\)](#) for details.

Value

A plot layer instance.

stat_km

Adds a Kaplan-Meier Estimate of Survival Statistic

Description

[Experimental] This stat is for computing the Kaplan-Meier survival estimate for right-censored data. It requires the aesthetic mapping `time` for the observation times and `status` which indicates the event status, either 0 for alive and 1 for dead, or 1 for alive and 2 for dead.

Usage

```
stat_km(
  mapping = NULL,
  data = NULL,
  geom = "km",
  position = "identity",
  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	<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 ggplot().</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 fortify() 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>
geom	<p>The geometric object to use to display the data for this layer. When using a <code>stat_*()</code> function to construct a layer, the <code>geom</code> argument can be used to override the default coupling between stats and geoms. The <code>geom</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A <code>Geom</code> ggproto subclass, for example <code>GeomPoint</code>. • A string naming the geom. To give the geom as a string, strip the function name of the <code>geom_</code> prefix. For example, to use <code>geom_point()</code>, give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	<p>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:</p> <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.
show.legend	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.
inherit.aes	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. borders() .
...	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 <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, `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.

Value

A data.frame with columns:

- `time`: time in data.
- `survival`: survival estimate at time.

Note

Logical status is not supported.

Author(s)

Michael Sachs (in ggkm), Samer Mouksassi (in ggquickeda).

Examples

```
library(ggplot2)
sex <- rbinom(250, 1, .5)
df <- data.frame(
  time = exp(rnorm(250, mean = sex)),
  status = rbinom(250, 1, .75),
  sex = sex
)
ggplot(df, aes(time = time, status = status, color = factor(sex))) +
  stat_km()
```

stat_km_ticks	<i>Adds Tick Marks to a Kaplan-Meier Estimate of Survival Statistic</i>
---------------	---

Description

[Experimental] This stat is for computing the location of the tick marks for the Kaplan-Meier survival estimate for right-censored data. It requires the aesthetic mapping `time` for the observation times and `status` which indicates the event status, either 0 for alive and 1 for dead, or 1 for alive and 2 for dead.

Usage

```
stat_km_ticks(
  mapping = NULL,
  data = NULL,
  geom = "km_ticks",
  position = "identity",
  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	<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 ggplot().</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 fortify() 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>
geom	<p>The geometric object to use to display the data for this layer. When using a <code>stat_*()</code> function to construct a layer, the <code>geom</code> argument can be used to override the default coupling between stats and geoms. The <code>geom</code> argument accepts the following:</p> <ul style="list-style-type: none"> • A <code>Geom</code> ggproto subclass, for example <code>GeomPoint</code>. • A string naming the geom. To give the geom as a string, strip the function name of the <code>geom_</code> prefix. For example, to use <code>geom_point()</code>, give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.

position	<p>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:</p> <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.
show.legend	<p>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.</p>
inherit.aes	<p>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. <code>borders()</code>.</p>
...	<p>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 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.</p> <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 <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.

Value

A data.frame with columns:

- time: time in data.
- survival: survival estimate at time.
- n.risk: number of patients at risk.

- n.censor: number of patients censored.
- n.event: number of patients with event.

Note

Logical status is not supported.

Author(s)

Michael Sachs (in ggkm), Samer Mouksassi (in ggquickeda).

Examples

```
library(ggplot2)
sex <- rbinom(250, 1, .5)
df <- data.frame(
  time = exp(rnorm(250, mean = sex)),
  status = rbinom(250, 1, .75),
  sex = sex
)
ggplot(df, aes(time = time, status = status, color = factor(sex))) +
  stat_km() +
  stat_km_ticks()
```

stat_n_text

Add Text Indicating the Sample Size to a ggplot2 Plot

Description

[Experimental]

See [EnvStats::stat_n_text\(\)](#) for details.

Value

A plot layer including the sample size text.

stat_prop

Compute Proportions According to Custom Denominator

Description

[Experimental]

See [ggstats::stat_prop\(\)](#) for details.

Value

A plot layer containing the custom proportions.

ttheme_gtdefault	<i>Table themes</i>
------------------	---------------------

Description**[Experimental]**See `ggpp::ttheme_gtdefault()` for details.**Value**A list object that can be used as `ttheme` in the construction of tables with functions from package 'gridExtra'.

ttheme_set	<i>Set default table theme</i>
------------	--------------------------------

Description**[Experimental]**See `ggpp::ttheme_set()` for details.**Value**

A named list with the previous value of the option.

Note

When testing this function, we found that in contrast to the original documentation, the theme is not fixed when the plot object is constructed. Instead, the option setting affects the rendering of ready built plot objects.

Index

`aes()`, [3](#), [6](#), [9](#), [11](#)

`borders()`, [4](#), [6](#), [9](#), [12](#)

`EnvStats::stat_n_text()`, [13](#)

`fortify()`, [3](#), [6](#), [9](#), [11](#)

`geom_km`, [3](#)

`geom_km_ticks`, [5](#)

`geom_table`, [8](#)

`geom_table()`, [2](#)

`geom_table_npc` (`geom_table`), [8](#)

`ggplot()`, [3](#), [6](#), [9](#), [11](#)

`ggplot2.utils` (`ggplot2.utils-package`), [2](#)

`ggplot2.utils-package`, [2](#)

`ggplot2::geom_path()`, [5](#)

`ggplot2::geom_step()`, [5](#)

`ggpp::geom_table()`, [8](#)

`ggpp::ttheme_gtdefault()`, [14](#)

`ggpp::ttheme_set()`, [14](#)

`ggstats::stat_prop()`, [13](#)

key glyphs, [4](#), [7](#), [10](#), [12](#)

layer geom, [9](#), [11](#)

layer position, [4](#), [6](#), [9](#), [12](#)

layer stat, [4](#), [6](#)

`layer()`, [4](#), [7](#), [9](#), [10](#), [12](#)

`stat_km`, [8](#)

`stat_km()`, [5](#)

`stat_km_ticks`, [11](#)

`stat_km_ticks()`, [7](#)

`stat_n_text`, [13](#)

`stat_n_text()`, [2](#)

`stat_prop`, [13](#)

`stat_prop()`, [2](#)

`StatProp` (`stat_prop`), [13](#)

`ttheme_gtbw` (`ttheme_gtdefault`), [14](#)

`ttheme_gtdark` (`ttheme_gtdefault`), [14](#)

`ttheme_gtdefault`, [14](#)

`ttheme_gtlight` (`ttheme_gtdefault`), [14](#)

`ttheme_gtminimal` (`ttheme_gtdefault`), [14](#)

`ttheme_gtplain` (`ttheme_gtdefault`), [14](#)

`ttheme_gtsimple` (`ttheme_gtdefault`), [14](#)

`ttheme_gtstripes` (`ttheme_gtdefault`), [14](#)

`ttheme_set`, [14](#)