Package: ggquickeda (via r-universe)

September 12, 2024

Title Quickly Explore Your Data Using 'ggplot2' and 'table1' Summary Tables

Version 0.3.1.9000

Description Quickly and easily perform exploratory data analysis by uploading your data as a 'csv' file. Start generating insights using 'ggplot2' plots and 'table1' tables with descriptive stats, all using an easy-to-use point and click 'Shiny' interface.

URL `https://github.com/smouksassi/ggquickeda`, `https://smouksassi.github.io/ggquickeda/`

BugReports `https://github.com/smouksassi/ggquickeda/issues`

Depends R (>= 4.1.0)

Imports colourpicker, dplyr, data.table, DT, Formula, GGally (>= 2.1.0), ggbeeswarm, ggh4x, ggplot2 (>= 3.4.0), ggpmisc, ggrepel (>= 0.7.0), ggpubr, ggstance, glue, gridExtra, Hmisc, markdown, methods, plotly, quantreg, rlang, scales, shiny (>= 1.0.4), shinyjs (>= 1.1), shinyjqui, stringr, survival, survminer, tidyr, table1 (>= 1.4.2), zoo, shinyFiles, RPostgres, forcats, ggridges, rms, tibble, patchwork (>= 1.2.0)

Suggests knitr, rmarkdown

License MIT + file LICENSE

SystemRequirements pandoc with https support

LazyData true

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

Roxygen list(markdown = TRUE)

Repository https://smouksassi.r-universe.dev

RemoteUrl https://github.com/smouksassi/ggquickeda

RemoteRef HEAD

RemoteSha e004748cd639f3110472bc7cb1bfef916626ebb5

Contents

`geom_km` *Add a Kaplan-Meier survival curve*

Description

Add a 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 `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.
`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()`.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, 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` understands the following aesthetics (required aesthetics are in bold):

- x The survival/censoring times. This is automatically mapped by `stat_km()`
- y The survival probability estimates. This is automatically mapped by `stat_km()` smallest level in sort order is assumed to be 0, with a warning.
- `alpha`
- `color`
- `linetype`
- `size`

See Also

The default stat for this geom is `stat_km()` see that documentation for more options to control the underlying statistical transformation.

Examples

```
library(ggplot2)
set.seed(123)
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_kmband` *Add confidence bands to a Kaplan-Meier survival curve*

Description

Add confidence bands to a Kaplan-Meier survival curve

Usage

```
geom_kmband(
  mapping = NULL,
  data = NULL,
  stat = "kmband",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  na.rm = 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.

`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()`.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, 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_kmband` understands the following aesthetics (required aesthetics are in bold):

- x The survival/censoring times. This is automatically mapped by `stat_kmband()`
- y The survival probability estimates. This is automatically mapped by `stat_kmband()` smallest level in sort order is assumed to be 0, with a warning
- `alpha`
- `color`
- `linetype`
- `linewidth`

See Also

The default stat for this geom is `stat_kmband()`. See that documentation for more options to control the underlying statistical transformation.

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), fill =factor(sex))) +
 geom_km() + geom_kmband()
```

---

geom_kmticks *Add tick marks to a Kaplan-Meier survival curve*

Description

Adds tickmarks at the times when there are censored observations but no events

Usage

```
geom_kmticks(
  mapping = NULL,
  data = NULL,
  stat = "kmticks",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  na.rm = 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.

`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()`.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, 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_kmticks` understands the following aesthetics (required aesthetics are in bold):

- `x` The survival/censoring times. This is automatically mapped by `stat_kmticks()`
- `y` The survival probability estimates. This is automatically mapped by `stat_kmticks()` smallest level in sort order is assumed to be 0, with a warning
- `alpha`
- `color`
- `linetype`
- `size`

See Also

The default stat for this geom is stat_kmticks see that documentation for more options to control the underlying statistical transformation.

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_kmticks(col="black")
```

`ggcontinuousexpdist` *Create a continuous exposure fit plot*

Description

Produces a logistic fit plot with a facettable exposures/quantiles/distributions in ggplot2

Usage

```
ggcontinuousexpdist(
  data = effICGI,
  response = "response",
  endpoint = "Endpoint",
  DOSE = "DOSE",
  color_fill = "DOSE",
  exposure_metrics = c("AUC", "CMAX"),
  exposure_metric_split = c("median", "tertile", "quartile", "none"),
  exposure_metric_soc_value = -99,
  exposure_metric_plac_value = 0,
  exposure_distribution = c("distributions", "lineranges", "none"),
  dose_plac_value = "Placebo",
  xlab = "Exposure Values",
  ylab = "Response",
  mean_obs_byexptile = TRUE,
  mean_text_size = 5,
  mean_obs_bydose = TRUE,
  mean_obs_bydose_plac = FALSE,
  N_text_size = 5,
  binlimits_text_size = 5,
  binlimits_ypos = -Inf,
  binlimits_color = "gray70",
  dist_position_scaler = 0.2,
  dist_offset = 0,
  dist_scale = 0.9,
  lineranges_ypos = -1,
  lineranges_dodge = 1,
  lineranges_doselabel = FALSE,
  proj_bydose = TRUE,
  yproj = TRUE,
  yproj_xpos = 0,
  yproj_dodge = 0.2,
  yaxis_position = c("left", "right"),
  facet_formula = NULL,
  theme_certara = TRUE,
  return_list = FALSE
)
```

Arguments

`data`	Data to use with multiple endpoints stacked into response (values), Endpoint(endpoint name)
`response`	name of the column holding the response values
`endpoint`	name of the column holding the name/key of the endpoint default to `Endpoint`
`DOSE`	name of the column holding the DOSE values default to `DOSE`
`color_fill`	name of the column to be used for color/fill default to DOSE column
`exposure_metrics`	name(s) of the column(s) to be stacked into `expname` `exptile` and split into `exposure_metric_split`
`exposure_metric_split`	one of "median", "tertile", "quartile", "none"
`exposure_metric_soc_value`	special exposure code for standard of care default -99
`exposure_metric_plac_value`	special exposure code for placebo default 0
`exposure_distribution`	one of distributions, lineranges or none
`dose_plac_value`	string identifying placebo in DOSE column

`xlab`	text to be used as x axis label
`ylab`	text to be used as y axis label
`mean_obs_byexptile`	observed probability by exptile `TRUE/FALSE`
`mean_text_size`	mean text size default to 5
`mean_obs_bydose`	observed mean by dose `TRUE/FALSE`
`mean_obs_bydose_plac`	observed probability by placebo dose `TRUE/FALSE`
`N_text_size`	N by exposure bin text size default to 5
`binlimits_text_size`	5 binlimits text size
`binlimits_ypos`	binlimits y position default to 0
`binlimits_color`	binlimits text color default to "gray70"
`dist_position_scaler`	space occupied by the distribution default to 0.2
`dist_offset`	offset where the distribution position starts default to 0
`dist_scale`	scaling parameter for ggridges default to 0.9
`lineranges_ypos`	where to put the lineranges -1
`lineranges_dodge`	lineranges vertical dodge value 1
`lineranges_doselabel`	`TRUE/FALSE`
`proj_bydose`	project the probabilities on logistic curve `TRUE/FALSE`
`yproj`	project the probabilities on y axis `TRUE/FALSE`
`yproj_xpos`	y projection x position 0
`yproj_dodge`	y projection dodge value 0.2
`yaxis_position`	where to put y axis "left" or "right"
`facet_formula`	facet formula to be use otherwise `endpoint ~ expname`
`theme_certara`	apply certara colors and format for strips and default colour/fill
`return_list`	What to return if True a list of the datasets and plot is returned instead of only the plot

Examples

```
# Example 1
library(ggplot2)
library(patchwork)
effICGI <- logistic_data |>
dplyr::filter(!is.na(ICGI7))|>
dplyr::filter(!is.na(AUC))
```

```
effICGI$DOSE <- factor(effICGI$DOSE,
                      levels=c("0", "600", "1200","1800","2400"),
                      labels=c("Placebo", "600 mg", "1200 mg","1800 mg","2400 mg"))
effICGI$STUDY <- factor(effICGI$STUDY)
effICGI <- tidyr::gather(effICGI,Endpoint,response,ICGI7,BRLS)
a <- ggcontinuousexpdist(data = effICGI |> dplyr::filter(Endpoint =="ICGI7"),
                 response = "response",
                 endpoint = "Endpoint",
                 exposure_metrics = c("AUC"),
                 exposure_metric_split = c("quartile"),
                 exposure_metric_soc_value = -99,
                 exposure_metric_plac_value = 0,
                 dist_position_scaler = 1, dist_offset = -1 ,
                 yproj_xpos =  -20 ,
                 yproj_dodge = 20 ,
                 exposure_distribution ="distributions")

b <- ggcontinuousexpdist(data = effICGI |> dplyr::filter(Endpoint =="BRLS"),
                 response = "response",
                 endpoint = "Endpoint",
                 exposure_metrics = c("AUC"),
                 exposure_metric_split = c("quartile"),
                 exposure_metric_soc_value = -99,
                 exposure_metric_plac_value = 0,
                 dist_position_scaler = 4.2, dist_offset = 5 ,
                 yproj_xpos =  -20 ,
                 yproj_dodge = 20 ,
                 exposure_distribution ="distributions",
                 return_list = FALSE)
(a / b)  +
plot_layout(guides = "collect") &
 theme(legend.position = "top")

#Example 2
effICGI$SEX <- as.factor(effICGI$SEX)
ggcontinuousexpdist(data = effICGI |>
  dplyr::filter(Endpoint =="ICGI7"),
                 response = "response",
                 endpoint = "Endpoint",
                 color_fill = "SEX",
                 exposure_metrics = c("AUC"),
                 exposure_metric_split = c("quartile"),
                 exposure_metric_soc_value = -99,
                 exposure_metric_plac_value = 0,
                 dist_position_scaler = 1, dist_offset = -1 ,
                 yproj_xpos =  -20 ,
                 yproj_dodge = 20 ,
                 exposure_distribution ="lineranges")
## Not run:
#Example 5

## End(Not run)
```

ggkmrisktable *Create a Kaplan-Meier plot with risk table*

Description

Produces a km plot with a facettable risk table in ggplot2

Usage

```
ggkmrisktable(
  data = lung_long,
  time = "time",
  status = "DV",
  endpoint = "Endpoint",
  groupvar1 = "Endpoint",
  groupvar2 = "expname",
  groupvar3 = "none",
  exposure_metrics = c("age", "ph.karno"),
  exposure_metric_split = c("median", "tertile", "quartile", "none"),
  exposure_metric_soc_value = -99,
  exposure_metric_plac_value = 0,
  show_exptile_values = FALSE,
  show_exptile_values_pos = c("left", "right"),
  show_exptile_values_textsize = 5,
  show_exptile_values_order = c("default", "reverse"),
  color_fill = "exptile",
  linetype = "exptile",
  xlab = "Time of follow_up",
  ylab = "Overall survival probability",
  nrisk_table_plot = TRUE,
  nrisk_table_variables = c("n.risk", "pct.risk", "n.event", "cum.n.event", "n.censor"),
  nrisk_table_breaktimeby = NULL,
  nrisk_table_textsize = 4,
  nrisk_position_scaler = 0.2,
  nrisk_position_dodge = 0.2,
  nrisk_offset = 0,
  nrisk_filterout0 = FALSE,
  km_logrank_pvalue = FALSE,
  km_logrank_pvalue_pos = c("left", "right"),
  km_logrank_pvalue_textsize = 5,
  km_trans = c("identity", "event", "cumhaz", "cloglog"),
  km_ticks = TRUE,
  km_band = TRUE,
  km_conf_int = 0.95,
  km_conf_type = c("log", "plain", "log", "log-log", "logit", "none"),
  km_conf_lower = c("usual", "peto", "modified"),
  km_median = c("none", "median", "medianci", "table"),
  km_median_table_pos = c("left", "right"),
  km_median_table_order = c("default", "reverse"),
  km_yaxis_position = c("left", "right"),
  facet_formula = NULL,
  facet_ncol = NULL,
  facet_strip_position = c("top", "top", "top", "top"),
  theme_certara = TRUE,
  return_list = FALSE
)
```

Arguments

`data`	Data to use with multiple endpoints stacked into time, status, endpoint name
`time`	name of the column holding the time to event information default to `time`
`status`	name of the column holding the event information default to `DV`
`endpoint`	name of the column holding the name/key of the endpoint default to `Endpoint`
`groupvar1`	name of the column to group by, default `Endpoint`
`groupvar2`	name of the column to group by in addition to groupvar1, default `expname`
`groupvar3`	name of the column to group by in addition to groupvar1 and groupvar2, default "none"
`exposure_metrics`	name(s) of the column(s) to be stacked into `expname` `exptile` and split into `exposure_metric_split`
`exposure_metric_split`	one of "median", "tertile", "quartile", "none"
`exposure_metric_soc_value`	special exposure code for standard of care default -99
`exposure_metric_plac_value`	special exposure code for placebo default 0
`show_exptile_values`	FALSE
`show_exptile_values_pos`	"left" or "right"
`show_exptile_values_textsize`	default to 5
`show_exptile_values_order`	the order of the entries "default" or "reverse"
`color_fill`	name of the column to be used for color/fill default to `exptile`
`linetype`	name of the column to be used for linetype default to `exptile`
`xlab`	text to be used as x axis label
`ylab`	text to be used as y axis label
`nrisk_table_plot`	TRUE
`nrisk_table_variables`	one or more from: "n.risk", "pct.risk", "n.event, "cum.n.event, "n.censor"

`nrisk_table_breaktimeby`	NULL
`nrisk_table_textsize`	4
`nrisk_position_scaler`	0.2
`nrisk_position_dodge`	0.2, negative values will reverse the order
`nrisk_offset`	0
`nrisk_filterout0`	FALSE
`km_logrank_pvalue`	FALSE
`km_logrank_pvalue_pos`	"left" or "right"
`km_logrank_pvalue_textsize`	pvalue text size default to 5
`km_trans`	one of "identity","event","cumhaz","cloglog"
`km_ticks`	TRUE
`km_band`	TRUE
`km_conf_int`	0.95
`km_conf_type`	default one of "log", "plain", "log-log", "logit", "none"
`km_conf_lower`	one of "usual", "peto", "modified"
`km_median`	add median survival information one of "none", "median", "medianci", "table"
`km_median_table_pos`	when table is chosen where to put it "left" or "right
`km_median_table_order`	when table is chosen the order of the entries "default" or "reverse"
`km_yaxis_position`	where to put y axis on "left" or "right
`facet_formula`	facet formula to be used otherwise ~ groupvar1 + groupvar2 + groupvar3
`facet_ncol`	NULL if not specified the automatic waiver will be used
`facet_strip_position`	position in sequence for the variable used in faceting default to c("top","top","top","top")
`theme_certara`	apply certara colors and format for strips and default colour/fill
`return_list`	What to return if True a list of the datasets and plot is returned instead of only the plot

Examples

```
library(tidyr)
# Example 1
lung_long <-  survival::lung |>
 dplyr::mutate(status = ifelse(status==1,0,1)) |>
 tidyr::gather(Endpoint,DV,status) |>
 dplyr::filter(!is.na(ph.karno))|>
 dplyr::filter(!is.na(pat.karno))|>
 dplyr::filter(!is.na(ph.ecog))
lung_long$ph.ecog <- ifelse(lung_long$ph.ecog>1,2,lung_long$ph.ecog)
lung_long$ph.ecog <- as.factor(lung_long$ph.ecog )
lung_long$ph.ecog <- as.factor(lung_long$ph.ecog )
lung_long$facetdum <- "(all)"

ggkmrisktable(data = lung_long, time= "time", status ="DV",
             exposure_metrics =c("age","ph.karno"),
             exposure_metric_split = "tertile",
             color_fill = "exptile",
             linetype = "expname",
             groupvar1 = "Endpoint",
             groupvar2 = "exptile",
             xlab = "Time of follow_up",
             ylab ="Overall survival probability",
             nrisk_table_variables = c("n.risk","n.event"),
             km_median = "medianci",
             km_band = FALSE,
             nrisk_table_breaktimeby = 200,
             facet_ncol = 3)
#Example 2
ggkmrisktable(data = lung_long, time= "time", status ="DV",
             exposure_metrics =c("age","ph.karno"),
             exposure_metric_split = "quartile",
             color_fill = "exptile",
             linetype = "none",
             groupvar1 = "Endpoint",
             groupvar2 = "exptile",
             xlab = "Time of follow_up",
             ylab ="Overall survival probability",
             nrisk_table_variables = c("cum.n.event","pct.risk","n.censor"),
             km_median = "medianci",
             km_band = TRUE,
             km_trans = "event",
             show_exptile_values = TRUE,
             show_exptile_values_pos = "right",
             nrisk_table_breaktimeby = 200,
             facet_ncol = 3,
             facet_formula = ~expname)
## Not run:
#Example 3
ggkmrisktable(data = lung_long, time = "time", status = "DV",
             exposure_metrics =c("ph.karno","pat.karno"),
             exposure_metric_split = "median",
             color_fill = "exptile",
             linetype = "exptile",
             groupvar1 = "Endpoint",
             groupvar2 = "expname",
             xlab = "Time of follow_up",
             ylab ="Overall survival probability",
             nrisk_table_variables = c("n.event"),
             km_trans = "event",
             km_median = "table",
             km_median_table_pos = "right",
             km_logrank_pvalue = TRUE,
             km_band = TRUE,
             nrisk_table_breaktimeby = 200,
             facet_ncol = 3,
             facet_formula = ~expname)
#Example 4
ggkmrisktable(data=lung_long,
             exposure_metrics = c("ph.karno","age"),
             exposure_metric_split = "median",
             time = "time",
             status ="DV",
             color_fill = "ph.ecog",
             linetype = "ph.ecog",
             groupvar1 = "Endpoint",
             groupvar2 = "expname",
             groupvar3 = "exptile",
             nrisk_filterout0 = FALSE,
             nrisk_table_breaktimeby = 200,
             km_logrank_pvalue = TRUE,
             km_median = "table",
             km_median_table_pos = "left",
             facet_formula = ~expname+exptile)
#Example 5

ggkmrisktable(data=lung_long,
             exposure_metrics = c("ph.karno","age"),
             exposure_metric_split = "none",
              color_fill = "none",
             linetype = "none",
         nrisk_table_variables = c("n.risk", "pct.risk", "n.event", "cum.n.event", "n.censor"),
              km_median = "table",
              nrisk_position_scaler = 0.1
             )


## End(Not run)
```

`gglogisticexpdist` *Create a logistic fit plot*

Description

Produces a logistic fit plot with a facettable exposures/quantiles/distributions in ggplot2

Usage

```
gglogisticexpdist(
    data = effICGI,
    response = "response",
    endpoint = "Endpoint",
    DOSE = "DOSE",
    color_fill = "DOSE",
    logistic_by_color_fill = FALSE,
    exposure_metrics = c("AUC", "CMAX"),
    exposure_metric_split = c("median", "tertile", "quartile", "none"),
    exposure_metric_soc_value = -99,
    exposure_metric_plac_value = 0,
    exposure_distribution = c("distributions", "lineranges", "none"),
    dose_plac_value = "Placebo",
    xlab = "Exposure Values",
    ylab = "Probability of Response",
    points_alpha = 0.2,
    points_show = TRUE,
    prob_obs_byexptile = TRUE,
    prob_obs_byexptile_group = "none",
    prob_text_size = 5,
    prob_obs_bydose = TRUE,
    prob_obs_bydose_plac = FALSE,
    Nresp_Ntot = TRUE,
    Nresp_Ntot_ypos = c("with percentages", "top"),
    Nresp_Ntot_sep = "/",
    binlimits_show = TRUE,
    binlimits_text_size = 5,
    binlimits_ypos = 0,
    binlimits_color = "gray70",
    dist_position_scaler = 0.2,
    dist_offset = 0,
    dist_scale = 0.9,
    lineranges_ypos = 0.2,
    lineranges_dodge = 0.15,
    lineranges_doselabel = FALSE,
    proj_bydose = TRUE,
    yproj = TRUE,
    yproj_xpos = 0,
    yproj_dodge = 0.2,
    yaxis_position = c("left", "right"),
    facet_formula = NULL,
    theme_certara = TRUE,
    return_list = FALSE
  )
```

Arguments

`data`	Data to use with multiple endpoints stacked into response (values 0/1), Endpoint(endpoint name)

`response`	name of the column holding the response values 0/1
`endpoint`	name of the column holding the name/key of the endpoint default to `Endpoint`
`DOSE`	name of the column holding the DOSE values default to `DOSE`
`color_fill`	name of the column to be used for color/fill default to DOSE column
`logistic_by_color_fill`	logistic fit split by color ? default `FALSE`
`exposure_metrics`	name(s) of the column(s) to be stacked into `expname` `exptile` and split into `exposure_metric_split`
`exposure_metric_split`	one of "median", "tertile", "quartile", "none"
`exposure_metric_soc_value`	special exposure code for standard of care default -99
`exposure_metric_plac_value`	special exposure code for placebo default 0
`exposure_distribution`	one of distributions, lineranges or none
`dose_plac_value`	string identifying placebo in DOSE column
`xlab`	text to be used as x axis label
`ylab`	text to be used as y axis label
`points_alpha`	alpha transparency for points
`points_show`	show the 0/1 observations `TRUE/FALSE`
`prob_obs_byexptile`	observed probability by exptile `TRUE/FALSE`
`prob_obs_byexptile_group`	additional grouping for exptile probabilities default `none`
`prob_text_size`	probability text size default to 5
`prob_obs_bydose`	observed probability by dose `TRUE/FALSE`
`prob_obs_bydose_plac`	observed probability by placebo dose `TRUE/FALSE`
`Nresp_Ntot`	show N responders/Ntotal ? `TRUE/FALSE`
`Nresp_Ntot_ypos`	y position for N responders/Ntotal two text elements the first for by exptile and the second for by dose/color options include `with percentages` `top` `bottom`
`Nresp_Ntot_sep`	character string to separat N responders/ Ntotal default `/`
`binlimits_show`	show the binlimits vertical lines `TRUE/FALSE`
`binlimits_text_size`	5 binlimits text size
`binlimits_ypos`	binlimits y position default to 0
`binlimits_color`	binlimits text color default to "gray70"

`dist_position_scaler`	space occupied by the distribution default to 0.2
`dist_offset`	offset where the distribution position starts default to 0
`dist_scale`	scaling parameter for ggridges default to 0.9
`lineranges_ypos`	where to put the lineranges -1
`lineranges_dodge`	lineranges vertical dodge value 1
`lineranges_doselabel`	`TRUE/FALSE`
`proj_bydose`	project the probabilities on logistic curve `TRUE/FALSE`
`yproj`	project the probabilities on y axis `TRUE/FALSE`
`yproj_xpos`	y projection x position 0
`yproj_dodge`	y projection dodge value 0.2
`yaxis_position`	where to put y axis "left" or "right"
`facet_formula`	facet formula to be use otherwise `endpoint ~ expname`
`theme_certara`	apply certara colors and format for strips and default colour/fill
`return_list`	What to return if True a list of the datasets and plot is returned instead of only the plot

Examples

```
# Example 1
library(ggplot2)
effICGI <- logistic_data |>
dplyr::filter(!is.na(ICGI))|>
dplyr::filter(!is.na(AUC))
effICGI$DOSE <- factor(effICGI$DOSE,
                       levels=c("0", "600", "1200","1800","2400"),
                       labels=c("Placebo", "600 mg", "1200 mg","1800 mg","2400 mg"))
effICGI$STUDY <- factor(effICGI$STUDY)
effICGI$ICGI2 <- effICGI$ICGI
effICGI <- tidyr::gather(effICGI,Endpoint,response,ICGI,ICGI2)
gglogisticexpdist(data = effICGI |>
                  dplyr::filter(Endpoint=="ICGI"),
                  response = "response",
                  endpoint = "Endpoint",
                  exposure_metrics = c("AUC"),
                  exposure_metric_split = c("quartile"),
                  exposure_metric_soc_value = -99,
                  exposure_metric_plac_value = 0,
                  exposure_distribution ="distributions",
                  yproj_xpos = -15,
                  yproj_dodge = 10,
                  dist_position_scaler = 0.1,
                  dist_offset = -0.1,
                  Nresp_Ntot_ypos = c("with percentages","bottom"),
                  prob_obs_bydose_plac = FALSE,
                  prob_obs_byexptile_group = "none",
                  binlimits_ypos = -0.08,
                  points_alpha= 1)

 # Example 2
 gglogisticexpdist(data = effICGI |>
                  dplyr::filter(Endpoint=="ICGI"),
                  response = "response",
                  endpoint = "Endpoint",
                  exposure_metrics = c("CMAX"),
                  exposure_metric_split = c("tertile"),
                  exposure_metric_soc_value = -99,
                  exposure_metric_plac_value = 0,
                  exposure_distribution ="lineranges",
                  lineranges_ypos = -0.2,
                  lineranges_dodge = 0.2,
                  prob_obs_bydose = TRUE,
                  yproj_xpos = -1,
                  yproj_dodge = 2,
                  dist_position_scaler = 0.1)


 ## Not run:
 #' # Example 3
 library(ggh4x)
 gglogisticexpdist(data = effICGI |>
                  dplyr::filter(Endpoint=="ICGI"),
                  response = "response",
                  endpoint = "Endpoint",
                  DOSE = "DOSE",
                  exposure_metrics = c("AUC"),
                  exposure_metric_split = c("quartile"),
                  exposure_distribution ="distributions",
                  exposure_metric_soc_value = -99,
                  exposure_metric_plac_value = 0,
                  dist_position_scaler = 0.15)+
  facet_grid2(Endpoint~expname+DOSE2,scales="free",
  margins = "DOSE2",strip = strip_nested())
 # Example 4
 effICGI$SEX <- as.factor(effICGI$SEX)
 gglogisticexpdist(data = effICGI  |>
                  dplyr::filter(Endpoint=="ICGI"),
                  response = "response",
                  endpoint = "Endpoint",
                  DOSE = "DOSE",
                  color_fill = "SEX",
                  exposure_metrics = c("AUC"),
                  exposure_metric_split = c("quartile"),
                  exposure_distribution ="distributions",
                  exposure_metric_soc_value = -99,
                  exposure_metric_plac_value = 0,
                  lineranges_ypos = -0.2,
                  yproj_xpos = -10,
                  yproj_dodge = 20,
                  prob_text_size = 6,
                  binlimits_text_size = 6,
                  Nresp_Ntot = TRUE,
                  dist_position_scaler = 0.15)+
                  ggplot2::scale_x_continuous(breaks = seq(0,350,50),
                  expand = ggplot2::expansion(add= c(0,0),mult=c(0,0)))+
                  ggplot2::coord_cartesian(xlim = c(-30,355))+
               ggplot2::facet_grid(Endpoint~expname+color_fill2, margins ="color_fill2" )

 #Example 4b
   effICGI$SEX <- as.factor(effICGI$SEX)
  gglogisticexpdist(data = effICGI |>
   dplyr::filter(Endpoint =="ICGI"),
                  response = "response",
                  endpoint = "Endpoint",
                  color_fill = "SEX",
                  exposure_metrics = c("AUC"),
                  exposure_metric_split = c("quartile"),
                  exposure_metric_soc_value = -99,
                  exposure_metric_plac_value = 0,
                  dist_position_scaler = 1, dist_offset = -1 ,
                  yproj_xpos =  -20 ,
                  yproj_dodge = 20 ,
                  exposure_distribution ="lineranges")

 #Example 5
 gglogisticexpdist(data = effICGI |> dplyr::filter(Endpoint=="ICGI"),
                   response = "response",
                   endpoint = "Endpoint",
                   DOSE = "DOSE",
                   exposure_metrics = c("AUC"),
                   exposure_metric_split = c("quartile"),
                   exposure_distribution ="distributions",
                   exposure_metric_soc_value = -99,
                   exposure_metric_plac_value = 0,
                   dist_position_scaler = 0.15)+
                  facet_grid(Endpoint~expname+exptile,scales="free",
                  margins = "exptile")
 #Example 6
 a <- gglogisticexpdist(data = effICGI, #
                   response = "response",
                   endpoint = "Endpoint",
                   DOSE = "DOSE",yproj_dodge = 36,
                   exposure_metrics = c("AUC"),
                   exposure_metric_split = c("quartile"),
                   exposure_distribution ="lineranges",
                   exposure_metric_soc_value = -99,
                   exposure_metric_plac_value = 0) +
   facet_grid(Endpoint~expname,switch = "both")
 b <-   gglogisticexpdist(data = effICGI, #
                     response = "response",
                     endpoint = "Endpoint",
                     DOSE = "DOSE",yproj_dodge = 2,
                     exposure_metrics = c("CMAX"),
                     exposure_metric_split = c("quartile"),
                     exposure_distribution ="lineranges",
                     exposure_metric_soc_value = -99,
                     exposure_metric_plac_value = 0,
                     yaxis_position = "right")+
   facet_grid(Endpoint~expname,switch = "x")+
   theme(strip.text.y.right = element_blank(),
         strip.background.y = element_blank())
 library(patchwork)
 (a | b ) +
   plot_layout(guides = "collect", axes = "collect_x")&
   theme(legend.position = "top")

 #Example 7
 effICGI <- logistic_data |>
 dplyr::filter(!is.na(ICGI))|>
 dplyr::filter(!is.na(AUC))
 effICGI$DOSE <- factor(effICGI$DOSE,
                       levels=c("0", "600", "1200","1800","2400"),
                       labels=c("Placebo", "600 mg", "1200 mg","1800 mg","2400 mg"))
 effICGI$STUDY <- factor(effICGI$STUDY)
 effICGI$ICGI2 <- ifelse(effICGI$ICGI7 < 4,1,0)
 effICGI$ICGI3 <- ifelse(effICGI$ICGI7 < 5,1,0)

 effICGI <- tidyr::gather(effICGI,Endpoint,response,ICGI,ICGI2,ICGI3)
 effICGI$endpointcol2 <- effICGI$Endpoint
 gglogisticexpdist(data = effICGI,
                   response = "response",
                  endpoint = "Endpoint",
                   exposure_metrics = c("AUC"),
                   exposure_metric_split = c("median"),
                   exposure_metric_soc_value = -99,
                   exposure_metric_plac_value = 0,
                   color_fill = "endpointcol2",
                   prob_obs_byexptile = FALSE,
                   logistic_by_color_fill = TRUE,
                   Nresp_Ntot = TRUE,
                   exposure_distribution ="distributions",
                   lineranges_doselabel = TRUE,
                   prob_obs_bydose = TRUE,
                   proj_bydose = FALSE,
                   yproj = FALSE,
                   dist_position_scaler = 0.1,
                   dist_offset = -0.1)+
   facet_grid(expname~.,scales="free_x")

 ## End(Not run)
```

`logistic_data` *Simulated Exposure Response Data*

Description

A dataset containing data suitable for logistic regression

Usage

```
logistic_data
```

Format

A data frame with 600 rows and 10 variables

STUDY Study identifier

ID Subject Identifier

DOSE Dose, in mg

GBDS Dose, in alternative salt

SEX Sex of the subject

AGE age of the subject, in years

WT weight of the subject, in kg

RACE Race of the subject

CRCL Creatinine clearance

BRLS RLS score

PRLS RLS score

AUC Area under the curve exposure

CMAX Maximun concentration exposure

ICGI response 0/1

ICGI7 response 1 to 7

Source

inspired from a real data submission

Examples

```
logistic_data
```

`run_ggquickeda` *Run the ggquickeda application*

Description

Run the `ggquickeda` application.

Usage

```
run_ggquickeda(data = NULL, ...)
```

Arguments

`data`	The initial data.frame to load into the application.
`...`	Additional arguments for bookmarking

Examples

```
if (interactive()) {
  run_ggquickeda()
}
```

`sample_data` *Simulated Pharmacokinetic Concentration Data*

Description

A dataset containing concentration-time data with the given dose and some subject characteristics to help in the app exploration.

Usage

```
sample_data
```

Format

A data frame with 600 rows and 10 variables

ID Subject Identifier, an integer from 1 to 150

Time Time of dose given or drug sample measured, in hours

Amt dose given at the corresponding Time, in milligrams

Conc drug concentrations in the plasma sample, in mg/L

Age age of the subject, in years

Weight weight of the subject, in kg

Gender Sex of the subject, a factor with Female and Male levels

Race Race of the subject, a factor with Asian, Black, Caucasian, Hispanic and Other levels

Dose dose group of the subject, in milligrams

AGECAT age category of the subject, a variable cutting Age into two values 0/1

Source

"sd_oral_richpk" from 'PKPDmisc' R package with an additional AGECAT variable

Examples

```
sample_data
```

`stat_km` *Adds a Kaplan Meier Estimate of Survival*

Description

Adds a Kaplan Meier Estimate of Survival

Usage

```
stat_km(
  mapping = NULL,
  data = NULL,
  geom = "km",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  trans = scales::identity_trans(),
  firstx = 0,
  firsty = 1,
  type = "kaplan-meier",
  start.time = 0,
  ...
)
```

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)`).
`geom`	The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following: • A `Geom` ggproto subclass, for example `GeomPoint`. • A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`. • For more information and other ways to specify the geom, see the layer geom 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.
`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()`.
`trans`	Transformation to apply to the survival probabilities. Defaults to "identity". Other options include "event", "cumhaz", "cloglog", or define your own using trans_new.
`firstx, firsty`	the starting point for the survival curves. By default, the plot program obeys tradition by having the plot start at `(0,1)`.
`type`	an older argument that combined stype and ctype, now deprecated. Legal values were "kaplan-meier" which is equivalent to stype=1, ctype=1, "fleming-harrington" which is equivalent to stype=2, ctype=1, and "fh2" which is equivalent to stype=2, ctype=2.

`start.time`	numeric value specifying a time to start calculating survival information. The resulting curve is the survival conditional on surviving to start.time.
`...`	Other arguments passed to survfit.formula

Details

This stat is for computing the confidence intervals for the Kaplan-Meier survival estimate for right-censored data. It requires the aesthetic mapping `x` for the observation times and `status` which indicates the event status, 0=alive, 1=dead or 1/2 (2=death). Logical status is not supported.

Value

a data.frame with additional columns:

`x`	x in data
`y`	Kaplan-Meier Survival Estimate at x

Aesthetics

`stat_km` understands the following aesthetics (required aesthetics are in bold):

- `time` The survival times
- `status` The censoring indicator, see Surv for more information.
- `alpha`
- `color`
- `linetype`
- `size`

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()

## Examples illustrating the options passed to survfit.formula

p1 <- ggplot(df, aes(time = time, status = status))
p1 + stat_km()
p1 + stat_km(trans = "cumhaz")
# for cloglog plots also log transform the time axis
p1 + stat_km(trans = "cloglog") + scale_x_log10()
p1 + stat_km(type = "fleming-harrington")
p1 + stat_km(start.time = 5)
```

`stat_kmband` *Adds confidence bands to a Kaplan Meier Estimate of Survival*

Description

Adds confidence bands to a Kaplan Meier Estimate of Survival

Usage

```
stat_kmband(
  mapping = NULL,
  data = NULL,
  geom = "kmband",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  trans = "identity",
  firstx = 0,
  firsty = 1,
  type = "kaplan-meier",
  error = "greenwood",
  conf.type = "log",
  conf.lower = "usual",
  start.time = 0,
  conf.int = 0.95,
  ...
)
```

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)`).
`geom`	The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following:

	• A `Geom` ggproto subclass, for example `GeomPoint`. • A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`. • For more information and other ways to specify the geom, see the layer geom 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.
`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()`.
`trans`	Transformation to apply to the survival probabilities. Defaults to "identity". Other options include "event", "cumhaz", "cloglog", or define your own using `scales::trans_new()`.
`firstx`, `firsty`	the starting point for the survival curves. By default, the plot program obeys tradition by having the plot start at (0,1).
`type`	an older argument that combined stype and ctype, now deprecated. Legal values were "kaplan-meier" which is equivalent to stype=1, ctype=1, "fleming-harrington" which is equivalent to stype=2, ctype=1, and "fh2" which is equivalent to stype=2, ctype=2.
`error`	either the string "greenwood" for the Greenwood formula or "tsiatis" for the Tsiatis formula, (only the first character is necessary). The default is "greenwood".
`conf.type`	One of "none", "plain", "log" (the default), "log-log" or "logit".
`conf.lower`	a character string to specify modified lower limits to the curve, the upper limit remains unchanged. Possible values are "usual" (unmodified), "peto", and "modified". The modified lower limit is based on an "effective n" argument. The confidence bands will agree with the usual calculation at each death time, but unlike the usual bands the confidence interval becomes wider at each censored observation. The extra width is obtained by multiplying the usual variance by a factor m/n, where n is the number currently at risk and m is the number at risk at the last death time. (The bands thus agree with the un-modified bands at each death time.) This is especially useful for survival curves with a long flat tail. The Peto lower limit is based on the same "effective n" argument as the modified limit, but also replaces the usual Greenwood variance term with a simple approximation. It is known to be conservative.

`start.time`	numeric value specifying a time to start calculating survival information. The resulting curve is the survival conditional on surviving to start.time.
`conf.int`	the level for a two-sided confidence interval on the survival curve(s). Default is 0.95.
`...`	Other arguments passed to survfit.formula

Details

This stat is for computing the confidence intervals for the Kaplan-Meier survival estimate for right-censored data. It requires the aesthetic mapping `x` for the observation times and `status` which indicates the event status, 0=alive, 1=dead or 1/2 (2=death). Logical status is not supported.

Value

a data.frame with additional columns:

`x`	x in data
`ymin`	Lower confidence limit of KM curve
`ymax`	Upper confidence limit of KM curve

Aesthetics

`stat_kmband` understands the following aesthetics (required aesthetics are in bold):

- `time` The survival times
- `status` The censoring indicator, see Surv for more information.
- `alpha`
- `color`
- `linetype`
- `linewidth`

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()

## Examples illustrating the options passed to survfit.formula

p1 <- ggplot(df, aes(time = time, status = status))
p1 + stat_km() + stat_kmband(conf.int = .99)
p1 + stat_kmband(error = "greenwood",fill="red",alpha=0.2) +
 stat_kmband(error = "tsiatis",fill="blue",alpha=0.2)+ stat_km()
p1 + stat_km() + stat_kmband(conf.type = "log-log")+ stat_kmband(conf.type = "log")
```

`stat_kmticks` *Adds tick marks to a Kaplan Meier Estimate of Survival*

Description

Adds tick marks to a Kaplan Meier Estimate of Survival

Usage

```
stat_kmticks(
  mapping = NULL,
  data = NULL,
  geom = "kmticks",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  trans,
  ...
)
```

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)`).
`geom`	The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following: • A `Geom` ggproto subclass, for example `GeomPoint`. • A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`. • For more information and other ways to specify the geom, see the layer geom 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.
`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()`.
`trans`	Transformation to apply to the survival probabilities. Defaults to "identity". Other options include "event", "cumhaz", "cloglog", or define your own using trans_new.
`...`	Other arguments passed to survfit.formula

Details

This stat is for computing the tick marks for a Kaplan-Meier survival estimate for right-censored data. The tick marks will appear at each censoring time which is also not a death time, which is the default for plot.survfit. It requires the aesthetic mapping `x` for the observation times and `status` which indicates the event status, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death).

Value

a data.frame with additional columns:

`x`	x in data
`y`	Kaplan-Meier Survival Estimate at x

Aesthetics

`stat_kmticks` understands the following aesthetics (required aesthetics are in bold):

- `time` The survival times
- `status` The censoring indicator, see Surv for more information.
- `alpha`
- `color`
- `linetype`
- `size`

See Also

stat_km; stat_kmband

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_kmticks()
```

Index