Package 'visR'

March 15, 2024

Type Package

Title Clinical Graphs and Tables Adhering to Graphical Principles

Version 0.4.1

Description To enable fit-for-purpose, reusable clinical and medical research focused visualizations and tables with sensible defaults and based on graphical principles as described in: ``Vandemeulebroecke et al. (2018)" <doi:10.1002/pst.1912>, ``Vandemeulebroecke et al. (2019)" <doi:10.1002/psp4.12455>, and ``Morris et al. (2019)" <doi:10.1136/bmjopen-2019-030215>.

License MIT + file LICENSE

URL https://openpharma.github.io/visR/, https://github.com/openpharma/visR

BugReports https://github.com/openpharma/visR/issues

Depends R (>= 3.5)

Imports broom (>= 0.7.11), cowplot, dplyr (>= 1.0.0), DT, forcats, ggplot2, graphics, grDevices, grid, gridExtra, gt (>= 0.3.0), gtable, kableExtra, knitr, lifecycle, rlang (>= 1.0.0), stats, survival (>= 3.4-0), tibble, tidycmprsk (>= 0.1.1), tidyr (>= 1.0.0), utils

Suggests data.table, ggpubr, learnr, rmarkdown, spelling, testthat (>= 2.1.0), vdiffr

VignetteBuilder knitr

biocViews

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Mark Baillie [aut, cre, cph],
Diego Saldana [aut],
Charlotta Fruechtenicht [aut],
Marc Vandemeulebroecke [aut],
Thanos Siadimas [aut],
Pawel Kawski [aut],
Steven Haesendonckx [aut],
James Black [aut],
Pelagia Alexandra Papadopoulou [aut],
Tim Treis [aut],
Rebecca Albrecht [aut],
Ardalan Mirshani [ctb],
Daniel D. Sjoberg [aut] (<https://orcid.org/0000-0003-0862-2018>)

Maintainer Mark Baillie <bailliem@gmail.com>

Repository CRAN

Date/Publication 2024-03-15 21:50:02 UTC

R topics documented:

`add_annotation` *Add annotations to a visR object*

Description

Wrapper around ggplot2::annotation_custom for simplified annotation to ggplot2 plots. This function accepts a string, dataframe, data.table, tibble or customized objects of class `gtable` and places them on the specified location on the `ggplot`. The layout is fixed: bold column headers and plain body. Only the font size and type can be chosen. Both the initial plot as the individual annotation are stored as attribute `component` in the final object.

Usage

```
add_annotation(
  gg = NULL,
  label = NULL,
  base_family = "sans",
  base_size = 11,
  xmin = -Inf,
  xmax = Inf,
  ymin = -Inf,
  ymax = Inf
)
```

Arguments

`gg`	Object of class `ggplot`.
`label`	`String`, `dataframe`, `data.table`, `tibble` used to annotate the `ggplot`.
`base_family`	`character`. Base font family
`base_size`	`numeric`. Base font size in pt
`xmin`	x coordinates giving horizontal location of raster in which to fit annotation.
`xmax`	x coordinates giving horizontal location of raster in which to fit annotation.
`ymin`	y coordinates giving vertical location of raster in which to fit annotation.
`ymax`	y coordinates giving vertical location of raster in which to fit annotation.

Value

Object of class `ggplot` with added annotation with an object of class `gtable`.

See Also

tableGrob annotation_custom

Examples

```
## Estimate survival
surv_object <- visR::estimate_KM(data = adtte, strata = "TRTP")

## We want to annotate the survival KM plot with a simple string comment
visR::visr(surv_object) %>%
  visR::add_annotation(
    label = "My simple comment",
    base_family = "sans",
    base_size = 15,
    xmin = 110,
    xmax = 180,
    ymin = 0.80
  )

## Currently, care needs to be taken on the x-y values relative
## to the plot data area. Here we are plotting outside of the data area.
visR::visr(surv_object) %>%
  visR::add_annotation(
    label = "My simple comment",
    base_family = "sans",
    base_size = 15,
    xmin = 210,
    xmax = 380,
    ymin = 1.0
  )


## We may also want to annotate a KM plot with information
## from additional tests or estimates. This example we annotate
## with p-values contained in a tibble

## we calculate p-values for "Equality across strata"
lbl <- visR::get_pvalue(surv_object,
  statlist = c("test", "pvalue"),
  type = "All"
)

## display p-values
lbl

## Now annotate survival KM plot with the p-values
visR::visr(surv_object) %>%
  visR::add_annotation(
    label = lbl,
    base_family = "sans",
    base_size = 9,
    xmin = 100,
```

```
    xmax = 180,
    ymin = 0.80
  )
```

---

add_CI *Add confidence interval (CI) to visR object*

Description

Method to add pointwise confidence intervals to a an object created by visR through an S3 method. The method is set up to use the pipe %>%. There are two options to display CI's, a "ribbon" or as "step" lines.

No default method is available at the moment.

Usage

```
add_CI(gg, ...)

## S3 method for class 'ggsurvfit'
add_CI(gg, alpha = 0.1, style = "ribbon", linetype, ...)

## S3 method for class 'ggtidycuminc'
add_CI(gg, alpha = 0.1, style = "ribbon", linetype, ...)
```

Arguments

gg	A ggplot created with visR
...	other arguments passed on to the method to modify geom_ribbon
alpha	aesthetic of ggplot2 geom_ribbon. Default is 0.1.
style	aesthetic of ggplot2 geom_ribbon. Default is "ribbon". An alternative option is "step" that uses a line to display interval bounds.
linetype	aesthetic of ggplot2 geom_ribbon.

Value

Pointwise confidence interval overlayed on a visR ggplot

Examples

```
library(visR)

# Estimate KM curves by treatment group
survfit_object <- survival::survfit(data = adtte, survival::Surv(AVAL, 1 - CNSR) ~ TRTP)

## plot without confidence intervals (CI)
```

```
p <- visR::visr(survfit_object)
p

# add CI to plot with default settings
p %>% add_CI()

# change transparency of CI ribbon
p %>% add_CI(alpha = 0.9, style = "ribbon")

# plot CI as a step line instead of ribbon
p %>% add_CI(alpha = 0.1, style = "step")

# change linetype of CI
p %>% add_CI(style = "step", linetype = 1)
```

---

add_CNSR *Add censoring symbols to a visR object*

Description

Add censoring symbols to a visR ggplot through an S3 method. The S3 method is for adding censoring symbols to a visR ggplot. The method is set up to use the pipe %>%.

No default method is available at the moment.

Usage

```
add_CNSR(gg, ...)

## S3 method for class 'ggsurvfit'
add_CNSR(gg, shape = 3, size = 2, ...)

## S3 method for class 'ggtidycuminc'
add_CNSR(gg, shape = 3, size = 2, ...)
```

Arguments

gg	A ggplot created with visR
...	other arguments passed on to the method to modify geom_point
shape	aesthetic of ggplot2 geom_point. Default is 3.
size	aesthetic of ggplot2 geom_point. Default is 2.

Value

Censoring symbols overlayed on a visR ggplot

Examples

```
library(visR)

# Estimate KM curves by treatment group
survfit_object <- survival::survfit(data = adtte, survival::Surv(AVAL, 1 - CNSR) ~ TRTP)

## plot without confidence intervals
p <- visR::visr(survfit_object)
p

# add censoring to plot
p %>% visR::add_CNSR()

# change censor symbol shape
p %>% visR::add_CNSR(shape = 1)

# change size and shape
p %>% visR::add_CNSR(size = 4, shape = 2)
```

---

add_highlight *Highlight a specific strata*

Description

S3 method for highlighting a specific strata by lowering the opacity of all other strata.

Usage

```
add_highlight(gg, ...)

## S3 method for class 'ggsurvfit'
add_highlight(gg = NULL, strata = NULL, bg_alpha = 0.2, ...)
```

Arguments

gg	A ggplot created with visR
...	other arguments passed on to the method
strata	String representing the name and value of the strata to be highlighted as shown in the legend.
bg_alpha	A numerical value between 0 and 1 that is used to decrease the opacity off all strata not chosen to be highlighted in `strata`. The other strata's existing alpha values are multiplied by `bg_alpha` to decrease their opacity, highlighting the target strata. This works on both `colour` and `fill` properties, as for example present after applying `visR::add_CI()`.

Value

The input ggsurvfit object with adjusted alpha values

Examples

```
adtte %>%
  visR::estimate_KM(strata = "SEX") %>%
  visR::visr() %>%
  visR::add_CI(alpha = 0.4) %>%
  visR::add_highlight(strata = "M", bg_alpha = 0.2)

strata <- c("Placebo", "Xanomeline Low Dose")

adtte %>%
  visR::estimate_KM(strata = "TRTP") %>%
  visR::visr() %>%
  visR::add_CI(alpha = 0.4) %>%
  visR::add_highlight(strata = strata, bg_alpha = 0.2)
```

---

add_quantiles *Add quantile indicators to visR plot*

Description

Method to add quantile lines to a plot.

Usage

```
add_quantiles(gg, ...)

## S3 method for class 'ggsurvfit'
add_quantiles(
  gg,
  quantiles = 0.5,
  linetype = "dashed",
  linecolour = "grey50",
  alpha = 1,
  ...
)
```

Arguments

gg	A ggplot created with visR
...	other arguments passed on to the method to modify geom_line
quantiles	vector of quantiles to be displayed on the probability scale, default: 0.5

linetype	string indicating the linetype as described in the aesthetics of ggplot2 geom_line, default: dashed (also supports "mixed" -> horizontal lines are solid, vertical ones are dashed)
linecolour	string indicating the linetype as described in the aesthetics of ggplot2 geom_line, default: grey, (also supports "strata" -> horizontal lines are grey50, vertical ones are the same colour as the respective strata)
alpha	numeric value between 0 and 1 as described in the aesthetics of ggplot2 geom_line, default: 1

Value

Lines indicating the quantiles overlayed on a visR ggplot

Examples

```
library(visR)

adtte %>%
  estimate_KM("SEX") %>%
  visr() %>%
  add_quantiles()

adtte %>%
  estimate_KM("SEX") %>%
  visr() %>%
  add_quantiles(quantiles = c(0.25, 0.50))

adtte %>%
  estimate_KM("SEX") %>%
  visr() %>%
  add_quantiles(
    quantiles = c(0.25, 0.50),
    linetype = "solid",
    linecolour = "grey"
  )

adtte %>%
  estimate_KM("SEX") %>%
  visr() %>%
  add_quantiles(
    quantiles = c(0.25, 0.50),
    linetype = "mixed",
    linecolour = "strata"
  )
```

`add_risktable` *Add risk tables to visR plots through an S3 method*

Description

S3 method for adding risk tables to visR plots. The function has following workflow:

- The risktables are calculated using `get_risktable`
- The risktables are placed underneath visR plots using `plot_grid`
- Both the initial visR plot as the individual risktables are stored as attribute `component` in the final object to allow post-modification of the individual plots if desired

Usage

```
add_risktable(gg, ...)

## S3 method for class 'ggsurvfit'
add_risktable(
  gg,
  times = NULL,
  statlist = "n.risk",
  label = NULL,
  group = "strata",
  collapse = FALSE,
  rowgutter = 0.16,
  ...
)

## S3 method for class 'ggtidycuminc'
add_risktable(
  gg,
  times = NULL,
  statlist = "n.risk",
  label = NULL,
  group = "strata",
  collapse = FALSE,
  rowgutter = 0.16,
  ...
)
```

Arguments

`gg`	visR plot of class `ggsurvfit` or `ggtidycmprsk`
`...`	other arguments passed on to the method add_risktable
`times`	Numeric vector indicating the times at which the risk set, censored subjects, events are calculated.

statlist	Character vector indicating which summary data to present. Current choices are "n.risk" "n.event" "n.censor", "cum.event", "cum.censor". Default is "n.risk".
label	Character vector with labels for the statlist. Default matches "n.risk" with "At risk", "n.event" with "Events", "n.censor" with "Censored", "cum.event" with "Cum. Event", and "cum.censor" with "Cum. Censor".
group	String indicating the grouping variable for the risk tables. Current options are: • "strata": groups the risk tables per stratum. The `label` specifies the label within each risk table. The strata levels are used for the titles of the risk tables. This is the default • "statlist": groups the risk tables per statlist. The `label` specifies the title for each risk table. The strata levels are used for labeling within each risk table. Default is "strata".
collapse	Boolean, indicates whether to present the data overall. Default is FALSE.
rowgutter	A numeric relative value between 0 and 1 indicates the height used by the table versus the height used by the plot, as described in `cowplot::plot_grid(rel_heights=)`. The default is 0.16.

Value

Object of class `ggplot` with added risk table.

See Also

plot_grid

Examples

```
## Display 2 risk tables, 1 per statlist
adtte %>%
  visR::estimate_KM(strata = "TRTP") %>%
  visR::visr() %>%
  visR::add_risktable(
    label = c("Subjects at Risk", "Censored"),
    statlist = c("n.risk", "n.censor", "n.event"),
    group = "statlist"
  )

## Display overall risk table at selected times
adtte %>%
  visR::estimate_KM(strata = "TRTP") %>%
  visR::visr() %>%
  visR::add_risktable(
    label = c("Subjects at Risk", "Censored"),
    statlist = c("n.risk", "n.censor"),
    collapse = TRUE,
    times = c(0, 20, 40, 60)
  )
```

```
## Add risk set as specified times
adtte %>%
  visR::estimate_KM(strata = "TRTP") %>%
  visR::visr() %>%
  visR::add_risktable(times = c(0, 20, 40, 100, 111, 200))
```

---

adtte *adtte - CDISC ADaM compliant time to event data set*

Description

ADTTE data copied from the 2013 CDISC Pilot

Usage

```
adtte
```

Format

A data frame with 254 rows and 26 variables:

STUDYID Study Identifier

SITEID Study Site Identifier

USUBJID Unique Subject Identifier

AGE Age

AGEGR1 Pooled Age Group 1

AGEGR1N Pooled Age Group 1 (N)

RACE Race

RACEN Race (N)

SEX Sex

TRTSDT Date of First Exposure to Treatment

TRTEDT Date of Last Exposure to Treatment

TRTDUR Duration of treatment (days)

TRTP Planned Treatment

TRTA Actual Treatment

TRTAN Actual Treatment (N)

PARAM Parameter Description

PARAMCD Parameter Code

AVAL Analysis Value

STARTDT Time to Event Origin Date for Subject

ADT Analysis Date

CNSR Censor

EVNTDESC Event or Censoring Description

SRCDOM Source Domain

SRCVAR Source Variable

SRCSEQ Source Sequence Number

SAFFL Safety Population Flag

Source

CDISC SDTM/ADAM Pilot Project. https://github.com/phuse-org/phuse-scripts/tree/master/data

Examples

```
data("adtte")
```

`align_plots` *Align multiple ggplot graphs, taking into account the legend*

Description

This function aligns multiple `ggplot` graphs by making them the same width by taking into account the legend width.

Usage

```
align_plots(pltlist)
```

Arguments

`pltlist`	A list of plots

Value

List of `ggplot` with equal width.

References

https://stackoverflow.com/questions/26159495

Examples

```
## create 2 graphs
p1 <- ggplot2::ggplot(adtte, ggplot2::aes(x = as.numeric(AGE), fill = "Age")) +
  ggplot2::geom_histogram(bins = 15)

p2 <- ggplot2::ggplot(adtte, ggplot2::aes(x = as.numeric(AGE))) +
  ggplot2::geom_histogram(bins = 15)

## default alignment does not take into account legend size
cowplot::plot_grid(
  plotlist = list(p1, p2),
  align = "none",
  nrow = 2
)

## align_plots() takes into account legend width
cowplot::plot_grid(
  plotlist = visR::align_plots(pltlist = list(p1, p2)),
  align = "none",
  nrow = 2
)
```

---

apply_attrition *Apply list of inclusion/exclusion criteria to a patient-level dataframe*

Description

[Questioning] Apply list of inclusion/exclusion criteria to a patient-level dataframe

Usage

```
apply_attrition(data, criteria_conditions)
```

Arguments

data	data.frame. Data set to be filtered
criteria_conditions	character dplyr-filter compatible conditions of the filtering criteria. These conditions will be applied to filter the input data set and obtain final analysis data set

Value

Filtered data frame

Examples

```
adtte_filtered <- visR::apply_attrition(adtte,
  criteria_conditions = c(
    "TRTP=='Placebo'", "AGE>=75",
    "RACE=='WHITE'", "SITEID==709"
  )
)
```

---

apply_theme *Applies a theme to a ggplot object.*

Description

[Experimental] Takes in the styling options defined through `visR::define_theme` and applies them to a plot.

Usage

```
apply_theme(gg, visR_theme_dict = NULL)
```

Arguments

gg	object of class ggplot
visR_theme_dict	nested list containing possible font options

Value

object of class ggplot

Examples

```
library(visR)

theme <- visR::define_theme(
  strata = list(
    "SEX" = list(
      "F" = "red",
      "M" = "blue"
    ),
    "TRTA" = list(
      "Placebo" = "cyan",
      "Xanomeline High Dose" = "purple",
      "Xanomeline Low Dose" = "brown"
    )
  ),
  fontsizes = list(
```

```
    "axis" = 12,
    "ticks" = 10,
    "legend_title" = 10,
    "legend_text" = 8
  ),
  fontfamily = "Helvetica",
  grid = FALSE,
  bg = "transparent",
  legend_position = "top"
)


gg <- adtte %>%
  visR::estimate_KM(strata = "SEX") %>%
  visR::visr() %>%
  visR::apply_theme(theme)
gg
```

`brca_cohort` *Cancer survival data*

Description

Creation script in data-raw

Usage

```
brca_cohort
```

Format

An object of class `data.frame` with 1098 rows and 10 columns.

`define_theme` *Provides a simple wrapper for themes*

Description

[Experimental] This function collects several lists if they are present. If absent, reasonable defaults are used. When strata are not defined in the theme, they default to grey50 and will not be presented in the legend.

Usage

```
define_theme(
  strata = NULL,
  fontsizes = NULL,
  fontfamily = "Helvetica",
  grid = FALSE,
  bg = "transparent",
  legend_position = NULL
)
```

Arguments

strata	named list containing the different strata and name:colour value pairs
fontsizes	named list containing the font sizes for different options
fontfamily	string with the name of a supported font
grid	boolean that specifies whether the major and minor grid should be drawn. The drawing of major and minor gridlines can be manipulated separately by using a boolean indicator in a named `list` with elements `major` and `minor`.
bg	string defining the colour for the background of the plot
legend_position	string defining the legend position. Valid options are NULL, 'top' 'bottom' 'right' 'left'

Value

Nested list with styling preferences for a ggplot object

Examples

```
theme <- visR::define_theme(
  strata = list("SEX" = list(
    "F" = "red",
    "M" = "blue"
  )),
  fontsizes = list(
    "axis" = 12,
    "ticks" = 10,
    "legend_title" = 10,
    "legend_text" = 8
  ),
  fontfamily = "Helvetica",
  grid = list(
    "major" = FALSE,
    "minor" = FALSE
  ),
  bg = "transparent",
  legend_position = "top"
)
```

`estimate_cuminc` *Competing Events Cumulative Incidence*

Description

Function creates a cumulative incidence object using the `tidycmprsk::cuminc()` function.

Usage

```
estimate_cuminc(
  data = NULL,
  strata = NULL,
  CNSR = "CNSR",
  AVAL = "AVAL",
  conf.int = 0.95,
  ...
)
```

Arguments

`data`	A data frame. The dataset is expected to have one record per subject per analysis parameter. Rows with missing observations included in the analysis are removed.
`AVAL, CNSR, strata`	These arguments are used to construct a formula to be passed to `tidycmprsk::cuminc(formula=)`. • `AVAL` Analysis value for Time-to-Event analysis. Default is `"AVAL"`, as per CDISC ADaM guiding principles. • `CNSR` Column name indicating the outcome and censoring statuses. Column must be a factor and the first level indicates censoring, the next level is the outcome of interest, and the remaining levels are the competing events. Default is `"CNSR"` • `strata` Character vector, representing the strata for Time-to-Event analysis. When NULL, an overall analysis is performed. Default is `NULL`.
`conf.int`	Confidence internal level. Default is 0.95. Parameter is passed to `tidycmprsk::cuminc(conf.level=)`
`...`	Additional argument passed to `tidycmprsk::cuminc()`

Value

A cumulative incidence object as explained at https://mskcc-epi-bio.github.io/tidycmprsk/reference/cuminc.html

Examples

```
cuminc <-
  visR::estimate_cuminc(
    data = tidycmprsk::trial,
    strata = "trt",
    CNSR = "death_cr",
    AVAL = "ttdeath"
  )
cuminc

cuminc %>%
  visR::visr() %>%
  visR::add_CI() %>%
  visR::add_risktable(statlist = c("n.risk", "cum.event"))
```

`estimate_KM` *Wrapper for Kaplan-Meier Time-to-Event analysis*

Description

[Deprecated]

This function is a wrapper around `survival::survfit.formula()` to perform a Kaplan-Meier analysis, assuming right-censored data. The result is an object of class `survfit` which can be used in downstream functions and methods that rely on the `survfit` class.

The function can leverage the conventions and controlled vocabulary from CDISC ADaM ADTTE data model, and also works with standard, non-CDISC datasets through the `formula` argument.

Usage

```
estimate_KM(
  data = NULL,
  strata = NULL,
  CNSR = "CNSR",
  AVAL = "AVAL",
  formula = NULL,
  ...
)
```

Arguments

`data`	A data frame. The dataset is expected to have one record per subject per analysis parameter. Rows with missing observations included in the analysis are removed.
`AVAL, CNSR, strata`	These arguments are used to construct a formula to be passed to `survival::survfit(formula=Surv(AVA 1-CNSR)~strata)`. These arguments' default values follow the naming conventions in CDISC.

- `AVAL` Analysis value for Time-to-Event analysis. Default is `"AVAL"`, as per CDISC ADaM guiding principles.
- `CNSR` Censor for Time-to-Event analysis. Default is `"CNSR"`, as per CDISC ADaM guiding principles. It is expected that CNSR = 1 for censoring and CNSR = 0 for the event of interest.
- `strata` Character vector, representing the strata for Time-to-Event analysis. When NULL, an overall analysis is performed. Default is `NULL`.

`formula` **[Experimental]** formula with `Surv()` on RHS and stratifying variables on the LHS. Use `~1` on the LHS for unstratified estimates. This argument will be passed to `survival::survfit(formula=)`. When this argument is used, arguments AVAL, CNSR, and strata are ignored.

`...` additional arguments passed on to the ellipsis of the call `survival::survfit.formula(...)`. Use `?survival::survfit.formula` and `?survival::survfitCI` for more information.

Value

survfit object ready for downstream processing in estimation or visualization functions and methods.

Estimation of 'survfit' object

The `estimate_KM()` function utilizes the defaults in `survival::survfit()`:

- The Kaplan Meier estimate is estimated directly (stype = 1).
- The cumulative hazard is estimated using the Nelson-Aalen estimator (ctype = 1): H.tilde = cumsum(x$n.event/x$n.risk). The MLE (H.hat(t) = -log(S.hat(t))) can't be requested.
- A two-sided pointwise 0.95 confidence interval is estimated using a log transformation (conf.type = "log").

When strata are present, the returned survfit object is supplemented with the a named list of the stratum and associated label. To support full traceability, the data set name is captured in the named list and the call is captured within its corresponding environment.

PARAM/PARAMCD and CDISC

If the data frame includes columns PARAM/PARAMCD (part of the CDISC format), the function expects the data has been filtered on the parameter of interest.

References

https://github.com/therneau/survival

See Also

`survfit.formula` `survfitCI`

Examples

```
## No stratification
visR::estimate_KM(data = adtte)

## Stratified Kaplan-Meier analysis by `TRTP`
visR::estimate_KM(data = adtte, strata = "TRTP")

## Stratified Kaplan-Meier analysis by `TRTP` and `SEX`
visR::estimate_KM(data = adtte, strata = c("TRTP", "SEX"))

## Stratification with one level
visR::estimate_KM(data = adtte, strata = "PARAMCD")

## Analysis on subset of adtte
visR::estimate_KM(data = adtte[adtte$SEX == "F", ])

## Modify the default analysis by using the ellipsis
visR::estimate_KM(
  data = adtte, strata = NULL,
  type = "kaplan-meier", conf.int = FALSE, timefix = TRUE
)

## Example working with non CDISC data
head(survival::veteran[c("time", "status", "trt")])

# Using non-CDSIC data
visR::estimate_KM(data = survival::veteran, formula = Surv(time, status) ~ trt)
```

---

get_attrition *Generate cohort attrition table*

Description

[Questioning] This is an experimental function that may be developed over time.

This function calculates the subjects counts excluded and included for each step of the cohort selection process.

Usage

```
get_attrition(data, criteria_descriptions, criteria_conditions,
subject_column_name)
```

Arguments

data	Dataframe. It is used as the input data to count the subjects that meets the criteria of interest

`criteria_descriptions`
: `character` It contains the descriptions of the inclusion/exclusion criteria. Each element of the vector corresponds to the description of each criterion.

`criteria_conditions`
: `character` It contains the corresponding conditions of the criteria. These conditions will be used in the table to compute the counts of the subjects.

`subject_column_name`
: `character` The column name of the table that contains the subject id.

Details

criteria_descriptions and criteria_conditions need to be of same length

Value

The counts and percentages of the remaining and excluded subjects for each step of the cohort selection in a table format.

Examples

```
visR::get_attrition(adtte,
  criteria_descriptions =
    c(
      "1. Placebo Group", "2. Be 75 years of age or older.",
      "3. White", "4. Site 709"
    ),
  criteria_conditions = c(
    "TRTP=='Placebo'", "AGE>=75",
    "RACE=='WHITE'", "SITEID==709"
  ),
  subject_column_name = "USUBJID"
)
```

`get_COX_HR` *Summarize Hazard Ratio from a survival object using S3 method*

Description

S3 method for extracting information regarding Hazard Ratios. The function allows the survival object's formula to be updated. No default method is available at the moment.

Usage

```
get_COX_HR(x, ...)

## S3 method for class 'survfit'
get_COX_HR(x, update_formula = NULL, ...)
```

Arguments

x	An object of class survfit
...	other arguments passed on to the method survival::coxph
update_formula	Template which specifies how to update the formula of the survfit object update.formula

Value

A tidied object of class coxph containing Hazard Ratios

See Also

coxph update.formula

Examples

```
## treatment effect
survfit_object_trt <- visR::estimate_KM(data = adtte, strata = c("TRTP"))
visR::get_COX_HR(survfit_object_trt)

## treatment and gender effect
survfit_object_trt_sex <- visR::estimate_KM(data = adtte, strata = c("TRTP", "SEX"))
visR::get_COX_HR(survfit_object_trt_sex)

## update formula of KM estimates by treatment to include "SEX" for HR estimation
visR::get_COX_HR(survfit_object_trt, update_formula = ". ~ . + SEX")

## update formula of KM estimates by treatment to include "AGE" for
## HR estimation with ties considered via the efron method
visR::get_COX_HR(survfit_object_trt,
  update_formula = ". ~ . + survival::strata(AGE)", ties = "efron"
)
```

---

get_pvalue *Summarize the test for equality across strata from a survival object using S3 method*

Description

Wrapper around survival::survdiff that tests the null hypothesis of equality across strata.

Usage

```
get_pvalue(
  survfit_object,
  ptype = "All",
  rho = NULL,
  statlist = c("test", "Chisq", "df", "pvalue"),
  ...
)
```

Arguments

survfit_object	An object of class survfit
ptype	Character vector containing the type of p-value desired. Current options are "Log-Rank" "Wilcoxon" "Tarone-Ware" "Custom" "All". "Custom" allows the user to specify the weights on the Kaplan-Meier estimates using the argument rho. The default is "All" displaying all types possible. When rho is specified in context of "All", also a custom p-value is displayed.
rho	a scalar parameter that controls the type of test.
statlist	Character vector containing the desired information to be displayed. The order of the arguments determines the order in which they are displayed in the final result. Default is the test name ("test"), Chi-squared test statistic ("Chisq"), degrees of freedom ("df") and p-value ("pvalue").
...	other arguments passed on to the method

Value

A data frame with summary measures for the Test of Equality Across Strata

See Also

survdiff

Examples

```
## general examples
survfit_object <- visR::estimate_KM(data = adtte, strata = "TRTP")
visR::get_pvalue(survfit_object)
visR::get_pvalue(survfit_object, ptype = "All")

## examples to obtain specific tests
visR::get_pvalue(survfit_object, ptype = "Log-Rank")
visR::get_pvalue(survfit_object, ptype = "Wilcoxon")
visR::get_pvalue(survfit_object, ptype = "Tarone-Ware")

## Custom example - obtain Harrington and Fleming test
visR::get_pvalue(survfit_object, ptype = "Custom", rho = 1)

## Get specific information and statistics
visR::get_pvalue(survfit_object, ptype = "Log-Rank", statlist = c("test", "Chisq", "df", "pvalue"))
visR::get_pvalue(survfit_object, ptype = "Wilcoxon", statlist = c("pvalue"))
```

get_quantile *Wrapper around quantile methods*

Description

S3 method for extracting quantiles. No default method is available at the moment.

Usage

```
get_quantile(x, ...)

## S3 method for class 'survfit'
get_quantile(
  x,
  ...,
  probs = c(0.25, 0.5, 0.75),
  conf.int = TRUE,
  tolerance = sqrt(.Machine$double.eps)
)
```

Arguments

x	An object of class survfit
...	other arguments passed on to the method
probs	probabilities Default = c(0.25,0.50,0.75)
conf.int	should lower and upper confidence limits be returned?
tolerance	tolerance for checking that the survival curve exactly equals one of the quantiles

Value

A data frame with quantiles of the object

See Also

quantile.survfit

Examples

```
## Kaplan-Meier estimates
survfit_object <- visR::estimate_KM(data = adtte, strata = c("TRTP"))

## visR quantiles
visR::get_quantile(survfit_object)

## survival quantiles
quantile(survfit_object)
```

`get_risktable` *Obtain risk tables for tables and plots*

Description

Create a risk table from an object using an S3 method. Currently, no default method is defined.

Usage

```
get_risktable(x, ...)

## S3 method for class 'survfit'
get_risktable(
  x,
  times = NULL,
  statlist = "n.risk",
  label = NULL,
  group = c("strata", "statlist"),
  collapse = FALSE,
  ...
)

## S3 method for class 'tidycuminc'
get_risktable(
  x,
  times = pretty(x$tidy$time, 10),
  statlist = "n.risk",
  label = NULL,
  group = c("strata", "statlist"),
  collapse = FALSE,
  ...
)
```

Arguments

`x`	an object of class `survfit` or `tidycuminc`
`...`	other arguments passed on to the method
`times`	Numeric vector indicating the times at which the risk set, censored subjects, events are calculated.
`statlist`	Character vector indicating which summary data to present. Current choices are "n.risk" "n.event" "n.censor", "cum.event", "cum.censor". Default is "n.risk".
`label`	Character vector with labels for the statlist. Default matches "n.risk" with "At risk", "n.event" with "Events", "n.censor" with "Censored", "cum.event" with "Cum. Event", and "cum.censor" with "Cum. Censor".
`group`	String indicating the grouping variable for the risk tables. Current options are:

- "strata": groups the risk tables per stratum. The `label` specifies the label within each risk table. The strata levels are used for the titles of the risk tables. This is the default
- "statlist": groups the risk tables per statlist. The `label` specifies the title for each risk table. The strata levels are used for labeling within each risk table.

Default is "strata".

`collapse` Boolean, indicates whether to present the data overall. Default is FALSE.

Value

return list of attributes the form the risk table i.e. number of patients at risk per strata

See Also

`summary.survfit`

`get_summary` *Summarize the descriptive statistics across strata from a survival object using S3 method*

Description

S3 method for extracting descriptive statistics across strata. No default method is available at the moment.

Usage

```
get_summary(x, ...)

## S3 method for class 'survfit'
get_summary(
  x,
  statlist = c("strata", "records", "events", "median", "LCL", "UCL", "CI"),
  ...
)
```

Arguments

`x`	An object of class `survfit`
`...`	other arguments passed on to the method
`statlist`	Character vector containing the desired information to be displayed. The order of the arguments determines the order in which they are displayed in the final result. Default is the strata ("strata"), number of subjects ("records"), number of events ("events"), the median survival time ("median"), the Confidence Interval ("CI"), the Lower Confidence Limit ("UCL") and the Upper Confidence Limit ("UCL").

Value

list of summary statistics from survfit object

A data frame with summary measures from a `survfit` object

Examples

```
survfit_object <- survival::survfit(data = adtte, survival::Surv(AVAL, 1 - CNSR) ~ TRTP)
get_summary(survfit_object)
```

`get_tableone` *Calculate summary statistics*

Description

[Questioning] S3 method for creating a table of summary statistics. The summary statistics can be used for presentation in tables such as table one or baseline and demography tables.

The summary statistics estimated are conditional on the variable type: continuous, binary, categorical, etc.

By default the following summary stats are calculated:

- Numeric variables: mean, min, 25th-percentile, median, 75th-percentile, maximum, standard deviation
- Factor variables: proportion of each factor level in the overall dataset
- Default: number of unique values and number of missing values

Usage

```
get_tableone(
  data,
  strata = NULL,
  overall = TRUE,
  summary_function = summarize_short
)

## Default S3 method:
get_tableone(
  data,
  strata = NULL,
  overall = TRUE,
  summary_function = summarize_short
)
```

Arguments

data	The dataset to summarize as dataframe or tibble
strata	Stratifying/Grouping variable name(s) as character vector. If NULL, only overall results are returned
overall	If TRUE, the summary statistics for the overall dataset are also calculated
summary_function	A function defining summary statistics for numeric and categorical values

Details

It is possible to provide your own summary function. Please have a loot at summary for inspiration.

Value

object of class tableone. That is a list of data specified summaries for all input variables.

Note

All columns in the table will be summarized. If only some columns shall be used, please select only those variables prior to creating the summary table by using dplyr::select()

Examples

```
# Example using the ovarian data set

survival::ovarian %>%
  dplyr::select(-fustat) %>%
  dplyr::mutate(
    age_group = factor(
      dplyr::case_when(
        age <= 50 ~ "<= 50 years",
        age <= 60 ~ "<= 60 years",
        age <= 70 ~ "<= 70 years",
        TRUE ~ "> 70 years"
      )
    ),
    rx = factor(rx),
    ecog.ps = factor(ecog.ps)
  ) %>%
  dplyr::select(age, age_group, everything()) %>%
  visR::get_tableone()

# Examples using ADaM data

# display patients in an analysis set
adtte %>%
  dplyr::filter(SAFFL == "Y") %>%
  dplyr::select(TRTA) %>%
  visR::get_tableone()
```

```
## display overall summaries for demog
adtte %>%
  dplyr::filter(SAFFL == "Y") %>%
  dplyr::select(AGE, AGEGR1, SEX, RACE) %>%
  visR::get_tableone()

## By actual treatment
adtte %>%
  dplyr::filter(SAFFL == "Y") %>%
  dplyr::select(AGE, AGEGR1, SEX, RACE, TRTA) %>%
  visR::get_tableone(strata = "TRTA")

## By actual treatment, without overall
adtte %>%
  dplyr::filter(SAFFL == "Y") %>%
  dplyr::select(AGE, AGEGR1, SEX, EVNTDESC, TRTA) %>%
  visR::get_tableone(strata = "TRTA", overall = FALSE)
```

`legendopts` *Translates options for legend into a list that can be passed to ggplot2*

Description

This function takes the legend position and orientation, defined by the user and puts them into a list for ggplot2.

Usage

```
legendopts(legend_position = "right", legend_orientation = NULL)
```

Arguments

`legend_position`
: Default = "right".

`legend_orientation`
: Default = NULL.

Value

List of legend options for ggplot2.

render *Render a data.frame, risktable, or tableone object as a table*

Description

[Questioning] Render a previously created data.frame, tibble or tableone object to html, rtf or latex

Usage

```
render(
  data,
  title = "",
  datasource,
  footnote = "",
  output_format = "html",
  engine = "gt",
  download_format = c("copy", "csv", "excel")
)
```

Arguments

data	Input data.frame or tibble to visualize
title	Specify the title as a text string to be displayed in the rendered table. Default is no title.
datasource	String specifying the data source underlying the data set. Default is no title.
footnote	String specifying additional information to be displayed as a footnote alongside the data source and specifications of statistical tests.
output_format	Type of output that is returned, can be "html" or "latex". Default is "html".
engine	If "html" is selected as output_format, one can chose between using kable, gt and DT as engine to render the output table. Default is "gt".
download_format	Options formats generated for downloading the data. Default is a list "c('copy', 'csv', 'excel')".

Value

A table data structure with possible interactive functionality depending on the choice of the engine.

stat_stepribbon *Step ribbon statistic*

Description

[Experimental] Provides stair-step values for ribbon plots, often using in conjunction with `ggplot2::geom_step()`. The step ribbon can be added with `stat_stepribbon()` or identically with `ggplot2::geom_ribbon(stat = "stepribbon")`

Usage

```
stat_stepribbon(
  mapping = NULL,
  data = NULL,
  geom = "ribbon",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  direction = "hv",
  ...
)

StatStepribbon
```

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`	which geom to use; defaults to `"ribbon"`
`position`	Position adjustment, either as a string naming the adjustment (e.g. `"jitter"` to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
direction	hv for horizontal-vertical steps, vh for vertical-horizontal steps
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

Format

An object of class StatStepRibbon (inherits from Stat, ggproto, gg) of length 3.

Value

a ggplot

References

https://groups.google.com/forum/?fromgroups=#!topic/ggplot2/9cFWHaH1CPs

Examples

```
# using ggplot2::geom_ribbon()
survival::survfit(survival::Surv(time, status) ~ 1, data = survival::lung) %>%
  survival::survfit0() %>%
  broom::tidy() %>%
 ggplot2::ggplot(ggplot2::aes(x = time, y = estimate, ymin = conf.low, ymax = conf.high)) +
  ggplot2::geom_step() +
  ggplot2::geom_ribbon(stat = "stepribbon", alpha = 0.2)

# using stat_stepribbon() with the same result
survival::survfit(survival::Surv(time, status) ~ 1, data = survival::lung) %>%
  survival::survfit0() %>%
  broom::tidy() %>%
 ggplot2::ggplot(ggplot2::aes(x = time, y = estimate, ymin = conf.low, ymax = conf.high)) +
  ggplot2::geom_step() +
  visR::stat_stepribbon(alpha = 0.2)
```

---

summarize_long *Calculate summary statistics from a vector*

Description

[Questioning] Calculates several summary statistics. The summary statistics depend on the vector class

Usage

```
summarize_long(x)

## S3 method for class 'factor'
summarize_long(x)

## S3 method for class 'integer'
summarize_long(x)

## S3 method for class 'numeric'
summarize_long(x)

## Default S3 method:
summarize_long(x)
```

Arguments

x	an object

Value

A summarized version of the input.

`summarize_short` *Create abbreviated variable summary for table1*

Description

[Questioning] This function creates summaries combines multiple summary measures in a single formatted string. Create variable summary for numeric variables. Calculates mean (standard deviation), median (IQR), min-max range and N/% missing elements for a numeric vector.

Create variable summary for integer variables Calculates mean (standard deviation), median (IQR), min-max range and N/% missing elements for a integer vector.

Usage

```
summarize_short(x)

## S3 method for class 'factor'
summarize_short(x)

## S3 method for class 'numeric'
summarize_short(x)

## S3 method for class 'integer'
summarize_short(x)
```

```
## Default S3 method:
summarize_short(x)
```

Arguments

x	a vector to be summarized

Value

A summarized less detailed version of the input.

Surv_CNSR *Create a Survival Object from CDISC Data*

Description

[Experimental]

The aim of `Surv_CNSR()` is to map the inconsistency in convention between the survival package and CDISC ADaM ADTTE data model.

The function creates a survival object (e.g. `survival::Surv()`) that uses CDISC ADaM ADTTE coding conventions and converts the arguments to the status/event variable convention used in the survival package.

The `AVAL` and `CNSR` arguments are passed to `survival::Surv(time = AVAL, event = 1 - CNSR, type = "right", origin = 0)`.

Usage

```
Surv_CNSR(AVAL, CNSR)
```

Arguments

AVAL	The follow-up time. The follow-up time is assumed to originate from zero. When no argument is passed, the default value is a column/vector named `AVAL`.
CNSR	The censoring indicator where `1=censored` and `0=death/event`. When no argument is passed, the default value is a column/vector named `CNSR`.

Value

Object of class 'Surv'

Details

The `Surv_CNSR()` function creates a survival object utilizing the expected data structure in the CDISC ADaM ADTTE data model, mapping the CDISC ADaM ADTTE coding conventions with the expected status/event variable convention used in the survival package—specifically, the coding convention used for the status/event indicator. The survival package expects the status/event indicator in the following format: `0=alive`, `1=dead`. Other accepted choices are `TRUE/FALSE` (`TRUE` = `death`) or `1/2` (`2=death`). A final but risky option is to omit the indicator variable, in which case all subjects are assumed to have an event.

The CDISC ADaM ADTTE data model adopts a different coding convention for the event/status indicator. Using this convention, the event/status variable is named `'CNSR'` and uses the following coding: `censor` = `1`, `status/event` = `0`.

See Also

`survival::Surv()`, `estimate_KM()`

Examples

```
# Use the `Surv_CNSR()` function with visR functions
adtte %>%
  visR::estimate_KM(formula = visR::Surv_CNSR() ~ SEX)

# Use the `Surv_CNSR()` function with functions from other packages as well
survival::survfit(visR::Surv_CNSR() ~ SEX, data = adtte)
survival::survreg(visR::Surv_CNSR() ~ SEX + AGE, data = adtte) %>%
  broom::tidy()
```

---

tableone *Display a summary Table (i.e. table one)*

Description

[Questioning] Wrapper function to produce a summary table (i.e. Table One). Create and render a summary table for a dataset. A typical example of a summary table are "table one", the first table in an applied medical research manuscript.

Calculate summary statistics and present them in a formatted table

Usage

```
tableone(
  data,
  title,
  datasource,
  footnote = "",
  strata = NULL,
  overall = TRUE,
  summary_function = summarize_short,
```

```
  ...
)
```

Arguments

data	The dataframe or tibble to visualize
title	Table title to include in the rendered table. Input is a text string.
datasource	String specifying the datasource underlying the data set
footnote	Table footnote to include in the rendered table. Input is a text string.
strata	Character vector with column names to use for stratification in the summary table. Default: NULL , which indicates no stratification.
overall	If TRUE, the summary statistics for the overall dataset are also calculated
summary_function	A function defining summary statistics for numeric and categorical values Pre-implemented functions are summarize_long and summarize_short
...	Pass options to render_table

Value

A table-like data structure, possibly interactive depending on the choice of the engine

Example Output

Examples

```
# metadata for table
t1_title <- "Cohort Summary"
t1_ds <- "ADaM Interim Dataset for Time-to-Event Analysis"
t1_fn <- "My table one footnote"


## table by treatment - without overall and render with GT
tbl_gt <-
  adtte %>%
  dplyr::filter(SAFFL == "Y") %>%
  dplyr::select(AGE, AGEGR1, SEX, EVNTDESC, TRTA) %>%
  visR::tableone(
    strata = "TRTA",
    overall = FALSE,
    title = t1_title,
    datasource = t1_ds,
    footnote = t1_fn,
    engine = "gt"
  )

## table by treatment - without overall and render with DT
```

```
tbl_DT <-
  adtte %>%
  dplyr::filter(SAFFL == "Y") %>%
  dplyr::select(AGE, AGEGR1, SEX, EVNTDESC, TRTA) %>%
  visR::tableone(
    strata = "TRTA",
    overall = FALSE,
    title = t1_title,
    datasource = t1_ds,
    footnote = t1_fn,
    engine = "DT"
  )

## table by treatment - without overall and render with kable
tbl_kable_html <-
  adtte %>%
  dplyr::filter(SAFFL == "Y") %>%
  dplyr::select(AGE, AGEGR1, SEX, EVNTDESC, TRTA) %>%
  visR::tableone(
    strata = "TRTA",
    overall = FALSE,
    title = t1_title,
    datasource = t1_ds,
    footnote = t1_fn,
    engine = "kable"
  )

## table by treatment - without overall and render with kable as
## a latex table format rather than html
tbl_kable_latex <-
  adtte %>%
  dplyr::filter(SAFFL == "Y") %>%
  dplyr::select(AGE, AGEGR1, SEX, EVNTDESC, TRTA) %>%
  visR::tableone(
    strata = "TRTA",
    overall = FALSE,
    title = t1_title,
    datasource = t1_ds,
    footnote = t1_fn,
    output_format = "latex",
    engine = "kable"
  )
```

`the_lhs` *Find the "lhs" in the pipeline*

Description

This function finds the left-hand sided symbol in a magrittr pipe and returns it as a character.

Usage

```
the_lhs()
```

Value

Left-hand sided symbol as string in the magrittr pipe.

References

https://github.com/tidyverse/magrittr/issues/115#issuecomment-173894787

Examples

```
blah <- function(x) the_lhs()
adtte %>%
  blah()
```

---

tidyme *Extended tidy cleaning of selected objects using S3 method*

Description

S3 method for extended tidying of selected model outputs. Note that the visR method retains the original nomenclature of the objects, and adds the one of broom::tidy to ensure compatibility with tidy workflows. The default method relies on `broom::tidy` to return a tidied object

Usage

```
tidyme(x, ...)

## Default S3 method:
tidyme(x, ...)

## S3 method for class 'survfit'
tidyme(x, ...)
```

Arguments

x	An S3 object
...	other arguments passed on to the method

Value

Data frame containing all list elements of the S3 object as columns. The column 'strata' is a factor to ensure that the strata are sorted in agreement with the order in the `survfit` object

See Also

tidy

Examples

```
## Extended tidying for a survfit object
surv_object <- visR::estimate_KM(data = adtte, strata = "TRTA")
tidied <- visR::tidyme(surv_object)

## Tidyme for non-included classes
data <- cars
lm_object <- stats::lm(data = cars, speed ~ dist)
lm_tidied <- visR::tidyme(lm_object)
lm_tidied
```

---

visr *Plot a supported S3 object*

Description

S3 method for creating plots directly from objects using ggplot2, similar to the base R plot() function.

[Deprecated] Methods visr.survfit() and visr.tidycuminc() have been deprecated in favor of ggsurvfit::ggsurvfit() and ggsurvfit::ggcuminc(), respectively.

[Questioning] visr.attrition() function to draw a Consort flow diagram chart is currently being questioned.

Usage

```
visr(x, ...)

## Default S3 method:
visr(x, ...)

## S3 method for class 'survfit'
visr(
  x = NULL,
  x_label = NULL,
  y_label = NULL,
  x_units = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  fun = "surv",
  legend_position = "right",
  ...
```

```
)

## S3 method for class 'attrition'
visr(
  x,
  description_column_name = "Criteria",
  value_column_name = "Remaining N",
  complement_column_name = "",
  box_width = 50,
  font_size = 12,
  fill = "white",
  border = "black",
  ...
)

## S3 method for class 'tidycuminc'
visr(
  x = NULL,
  x_label = "Time",
  y_label = "Cumulative Incidence",
  x_units = NULL,
  x_ticks = pretty(x$tidy$time, 10),
  y_ticks = pretty(c(0, 1), 5),
  legend_position = "right",
  ...
)
```

Arguments

x	Object of class attritiontable
...	other arguments passed on to the method
x_label	character Label for the x-axis. When not specified, the function will look for "PARAM" or "PARAMCD" information in the original data set (CDISC standards). If no "PARAM"/"PARAMCD" information is available, the default x-axis label is "Time".
y_label	character Label for the y-axis. When not specified, the default will do a proposal, depending on the fun argument.
x_units	Unit to be added to the x_label (x_label (x_unit)). Default is NULL.
x_ticks	Ticks for the x-axis. When not specified, the default will do a proposal.
y_ticks	Ticks for the y-axis. When not specified, the default will do a proposal based on the fun argument.
fun	Function that represents the scale of the estimate. The current options are:

surv	is the survival probability. This is the default
log	is log of the survival probability
event	is the failure probability
cloglog	is log(-log(survival probability))

pct	is survival as a percentage
logpct	is log survival as a percentage
cumhaz	is the cumulative hazard

legend_position
: Specifies the legend position in the plot. Character values allowed are "top" "left" "bottom" "right". Numeric coordinates are also allowed. Default is "right".

description_column_name
: character Name of the column containing the inclusion descriptions

value_column_name
: character Name of the column containing the remaining sample counts

complement_column_name
: character Optional: Name of the column containing the exclusion descriptions

box_width
: character The box width for each box in the flow chart

font_size
: character The fontsize in pt

fill
: The color (string or hexcode) to use to fill the boxes in the flowchart

border
: The color (string or hexcode) to use for the borders of the boxes in the flowchart

Value

Object of class ggplot and ggsurvplot for survfit objects.

See Also

ggplot

Examples

```
# fit KM
km_fit <- survival::survfit(survival::Surv(AVAL, 1 - CNSR) ~ TRTP, data = adtte)

# plot curves using survival plot function
plot(km_fit)

# plot same curves using visR::visr plotting function
visR::visr(km_fit)

# estimate KM using visR wrapper
survfit_object <- visR::estimate_KM(data = adtte, strata = "TRTP")

# Plot survival probability
visR::visr(survfit_object, fun = "surv")

# Plot survival percentage
visR::visr(survfit_object, fun = "pct")

# Plot cumulative hazard
```

```
visR::visr(survfit_object, fun = "cloglog")


## Create attrition
attrition <- visR::get_attrition(adtte,
  criteria_descriptions = c(
    "1. Not in Placebo Group",
    "2. Be 75 years of age or older.",
    "3. White",
    "4. Female"
  ),
  criteria_conditions = c(
    "TRTP != 'Placebo'",
    "AGE >= 75",
    "RACE=='WHITE'",
    "SEX=='F'"
  ),
  subject_column_name = "USUBJID"
)

## Draw a CONSORT attrition chart without specifying extra text for the complement
attrition %>%
  visr("Criteria", "Remaining N")

## Add detailed complement descriptions to the "exclusion" part of the CONSORT diagram
# Step 1. Add new column to attrition dataframe
attrition$Complement <- c(
  "NA",
  "Placebo Group",
  "Younger than 75 years",
  "Non-White",
  "Male"
)

# Step 2. Define the name of the column in the call to the plotting function
attrition %>%
  visr("Criteria", "Remaining N", "Complement")

## Styling the CONSORT flowchart
# Change the fill and outline of the boxes in the flowchart
attrition %>%
  visr("Criteria", "Remaining N", "Complement", fill = "lightblue", border = "grey")

## Adjust the font size in the boxes
attrition %>%
  visr("Criteria", "Remaining N", font_size = 10)
```

Index