

# Package ‘survivalAnalysis’

April 24, 2021

**Type** Package

**Title** High-Level Interface for Survival Analysis and Associated Plots

**Version** 0.2.0

**Author** Marcel Wiesweg [aut, cre]

**Maintainer** Marcel Wiesweg <marcel.wiesweg@uk-essen.de>

**Description** A high-level interface to perform survival analysis, including Kaplan-Meier analysis and log-rank tests and Cox regression. Aims at providing a clear and elegant syntax, support for use in a pipeline, structured output and plotting. Builds upon the 'survminer' package for Kaplan-Meier plots and provides a customizable implementation for forest plots. Kaplan & Meier (1958) <doi:10.1080/01621459.1958.10501452> Cox (1972) <JSTOR:2985181> Peto & Peto (1972) <JSTOR:2344317>.

**License** GPL-3

**Encoding** UTF-8

**Depends** R (>= 3.3.0)

**Imports** grDevices, graphics, stats, utils, survival, rlang (>= 0.2.0), dplyr (>= 0.8.0), forcats, magrittr, purrr, stringr, tibble, tidyr, gridExtra, ggplot2 (>= 2.2.1), scales, survminer (> 0.4.0), cowplot, tidybits

**RoxygenNote** 7.1.1

**Suggests** knitr, rmarkdown, tidyverse

**VignetteBuilder** knitr

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2021-04-24 12:20:05 UTC

## R topics documented:

analyse\_multivariate . . . . . 2

analyse_survival . . . . .	4
cox_as_data_frame . . . . .	6
forest_plot . . . . .	7
forest_plot_grid . . . . .	11
format.SurvivalAnalysisMultivariateResult . . . . .	12
format.SurvivalAnalysisUnivariateResult . . . . .	12
ggsurvplot_to_gtable . . . . .	13
grid_layout . . . . .	14
identity_order . . . . .	14
kaplan_meier_grid . . . . .	15
kaplan_meier_plot . . . . .	16
multivariate_as_data_frame . . . . .	17
pluck_multivariate_analysis . . . . .	18
pluck_survival_analysis . . . . .	19
print.SurvivalAnalysisMultivariateResult . . . . .	20
print.SurvivalAnalysisUnivariateResult . . . . .	20
survival_data_frames . . . . .	21
survival_essentials . . . . .	22

## Index 24

---

analyse\_multivariate *Multivariate analysis (Cox Regression)*

---

### Description

Performs Cox regression on right-censored data using a multiple covariates.

### Usage

```
analyse_multivariate(
  data,
  time_status,
  covariates,
  strata = NULL,
  covariate_name_dict = NULL,
  covariate_label_dict = NULL,
  reference_level_dict = NULL,
  sort_frame_by = vars(HR)
)
```

```
analyze_multivariate(
  data,
  time_status,
  covariates,
  strata = NULL,
  covariate_name_dict = NULL,
  covariate_label_dict = NULL,
```

```

    reference_level_dict = NULL,
    sort_frame_by = vars(HR)
  )

```

### Arguments

<code>data</code>	A data frame containing the time/status information and, if used, the covariate.
<code>time_status</code>	A vector of length 2 giving the time and status fields. It is recommended to use <code>vars()</code> and symbolic column names or code that is tidily-evaluated on data. You can also pass a character vector with the column names or a numeric vector with column indices.
<code>covariates</code>	The covariates. Pass symbolic columns names or code that is tidily-evaluated on data. Column names or column indices are also possible. In any case, factors with appropriate labels will be generated which in all printouts. You can use <code>covariate_name_dict</code> and <code>covariate_label_dict</code> to rename these factors and their levels.
<code>strata</code>	Strata (optional). Same format as covariates. For each strata level (if multiple fields, unique combinations of levels) a separate baseline hazard is fit.
<code>covariate_name_dict</code>	A dictionary (named list or vector) of old->new covariate names
<code>covariate_label_dict</code>	A dictionary (named list or vector) of old->new covariate value level labels
<code>reference_level_dict</code>	For categorical variables, the Cox regression uses pseudo variables for each level relative to a reference category, resulting in n-1 variables for n levels of a categorical covariate. Hazard ratios will be relative to the reference level, which is defined as having hazard ratio 1.0. Per default, the reference level is the first factor level. You can specify a different level by passing a named vector: factor name -> value of reference level. Note that this is independent of <code>covariate_label_dict</code> , i.e. specify the factor level as it is in <code>data#</code>
<code>sort_frame_by</code>	A <code>vars()</code> list of one or more symbolic column names. The result contains a data frame of the cox regression results ( <code>cox_as_data_frame</code> ). This frame contains the variables "Lower_CI", "HR", "Upper_CI", "Inv_Lower_CI", "Inv_HR", "Inv_Upper_CI", "p". You can specify by which variables the frame should be sorted. Default: Hazard Ratio.

### Details

This method builds upon the `survival` package and returns a comprehensive result object for survival analysis containing the `coxph` results. A `format/print` method is provided that prints the essential statistics.

### Value

An object of class "SurvivalAnalysisResult" and "SurvivalAnalysisMultivariateResult". You can use this result as a black box for further functions in this package, `format` or `print` it, retrieve information as a data frame via `multivariate_as_data_frame` or access individual pieces via `pluck_multivariate_analysis`

**See Also**[forest\\_plot](#)**Examples**

```
library(magrittr)
library(dplyr)
survival::colon %>%
  analyse_multivariate(vars(time, status),
                      vars(rx, sex, age, obstruct, perfor, nodes, differ, extent)) %>%
  print()
```

---

analyse_survival	<i>Univariate survival analysis</i>
------------------	-------------------------------------

---

**Description**

Performs survival analysis on right-censored data using a single covariate, or no covariate.

**Usage**

```
analyse_survival(
  data,
  time_status,
  by,
  by_label_map = NULL,
  by_order_vector = NULL,
  cox_reference_level = NULL,
  p_adjust_method = "none",
  plot_args = list()
)
```

```
analyze_survival(
  data,
  time_status,
  by,
  by_label_map = NULL,
  by_order_vector = NULL,
  cox_reference_level = NULL,
  p_adjust_method = "none",
  plot_args = list()
)
```

**Arguments**

**data** A data frame containing the time/status information and, if used, the covariate.

<code>time_status</code>	A vector of length 2 giving the time and status fields. It is recommended to use <code>vars()</code> and symbolic column names or code that is tidily-evaluated on data. You can also pass a character vector with the column names or a numeric vector with column indices.
<code>by</code>	The term by which survival curves will be separated. Pass <code>NULL</code> or omit to generate a single curve and only descriptive statistics. Pass symbolic columns names or code that is tidily-evaluated on data to generate more than one curve, and the appropriate statistics to compare the curves. A column name or column index is also possible. In any case, the parameter will be used to create a factor with appropriate labels. This factor will appear in all printouts and plots. You can use <code>by_label_map</code> and <code>by_order_vector</code> to rename and reorder this factor.
<code>by_label_map</code>	A dictionary (named list or vector) of old->new labels of the factor created using <code>by</code> . The factor will be renamed accordingly, and also reordered by the order of the vector.
<code>by_order_vector</code>	A vector of the labels of the factor created using <code>by</code> , after renaming them based on <code>by_label_map</code> (so specify the "new" level). The factor will be ordered according to the order of this vector. It need not contain all elements, only those found will be reorder at the top.
<code>cox_reference_level</code>	The result will include a univariate Cox regression. Use this parameter to specify the level of the factor generated using <code>by</code> that you want to use as the reference level (Hazard ratios will be relative to the reference level, which is defined as having hazard ratio 1.0) Note that the given string applies after all renaming has been done, so specify the "new" level.
<code>p_adjust_method</code>	If there are more than two levels in the <code>by</code> factor, the result will include the return value of <code>pairwise_survdiff</code> , which performs p adjustment. You can specify the desired method here. Note that other p values are not corrected, this is beyond the scope of this method.
<code>plot_args</code>	Named list of arguments that will be stored for later use in plotting methods, such as <code>kaplan_meier_plot</code> . There they will take precedence over arguments given to that method. This is useful when plotting multiple results with a set of default arguments, of which some such as title or axis scale differ per-plot.

## Details

This method builds upon the `survival` package and returns a comprehensive result object for survival analysis containing the `survfit`, `survdiff` and `coxph` results. A `format/print` method is provided that prints the essential statistics. Kaplan-Meier plots are readily generated using the `kaplan_meier_plot` or `kaplan_meier_grid` functions.

## Value

An object of class "SurvivalAnalysisResult" and "SurvivalAnalysisUnivariateResult". You can use this result as a black box for further functions in this package, `format` or `print` it, retrieve information as a data frame via `survival_data_frames` or access individual pieces via `pluck_survival_analysis`

**Examples**

```
library(magrittr)
library(dplyr)
survival::aml %>%
  analyse_survival(vars(time, status), x) %>%
  print
```

---

cox_as_data_frame	<i>Turns a coxph result to a data frame</i>
-------------------	---

---

**Description**

Extracts useful information from a coxph/summary.coxph into a data frame which is ready for printing or further analysis

**Usage**

```
cox_as_data_frame(
  coxphsummary,
  unmangle_dict = NULL,
  factor_id_sep = ":",
  sort_by = NULL
)
```

**Arguments**

coxphsummary	The summary.coxph or coxph result object
unmangle_dict	An unmangle dict of mangled column name -> readable column name (as created by analyse_multivariate)
factor_id_sep	The frame contains one column "factor.id" which is a composite of covariate name and, if categorical, the factor level (one line for each factor level except for the reference level)
sort_by	A vars() list of one or more symbolic column names. This frame contains the variables "Lower_CI", "HR", "Upper_CI", "Inv_Lower_CI", "Inv_HR", "Inv_Upper_CI", "p". You can choose to sort by any combination. Use desc() to sort a variable in descending order.

**Value**

A tibble.

---

forest_plot	<i>Forest plots for survival analysis.</i>
-------------	--

---

### Description

Creates a forest plot from `SurvivalAnalysisResult` objects. Both univariate (`analyse_survival`) results, typically with `use_one_hot=TRUE`, and multivariate (`analyse_multivariate`) results are acceptable.

### Usage

```
forest_plot(  
  ...,  
  use_one_hot = FALSE,  
  factor_labeller = identity,  
  endpoint_labeller = identity,  
  orderer = identity_order,  
  categorizer = NULL,  
  relative_widths = c(1, 1, 1),  
  ggtheme = theme_bw(),  
  labels_displayed = c("endpoint", "factor"),  
  label_headers = c(endpoint = "Endpoint", factor = "Subgroup", n = "n"),  
  values_displayed = c("HR", "CI", "p"),  
  value_headers = c(HR = "HR", CI = "CI", p = "p", n = "n", subgroup_n = "n"),  
  HRsprintfFormat = "%.2f",  
  psprintfFormat = "%.3f",  
  p_less_than_cutoff = 0.001,  
  log_scale = TRUE,  
  HR_x_breaks = seq(0, 10),  
  HR_x_limits = NULL,  
  factor_id_sep = ":",  
  na_rm = TRUE,  
  title = NULL,  
  title_relative_height = 0.1,  
  title_label_args = list(),  
  base_papersize = dinA(4)  
)  
  
forest_plot.df(  
  .df,  
  factor_labeller = identity,  
  endpoint_labeller = identity,  
  orderer = identity_order,  
  categorizer = NULL,  
  relative_widths = c(1, 1, 1),  
  ggtheme = theme_bw(),  
  labels_displayed = c("endpoint", "factor"),
```

```

label_headers = c(endpoint = "Endpoint", factor = "Subgroup", n = "n"),
values_displayed = c("HR", "CI", "p"),
value_headers = c(HR = "HR", CI = "CI", p = "p", n = "n", subgroup_n = "n"),
HRsprintfFormat = "%.2f",
psprintfFormat = "%.3f",
p_lessthan_cutoff = 0.001,
log_scale = TRUE,
HR_x_breaks = seq(0, 10),
HR_x_limits = NULL,
factor_id_sep = ":",
na_rm = TRUE,
title = NULL,
title_relative_height = 0.1,
title_label_args = list(),
base_papersize = dinA(4)
)

```

## Arguments

- ... The `SurvivalAnalysisResult` objects. You can also pass one list of such objects, or use explicit splicing (`!!!` operator). If not `use_one_hot`, also a list of `coxph` objects, or a mix is acceptable.
- `use_one_hot` If not `use_one_hot` (default), will take univariate or multivariate results and plot hazard ratios against the reference level (as provided to the [analyse\\_survival](#) or [analyse\\_multivariate](#) function, or, per default, the first factor level), resulting in  $k-1$  values for  $k$  levels. If `use_one_hot == TRUE`, will only accept univariate results from [analyse\\_survival](#) and plot HRs of one factor level vs. remaining cohort, resulting in  $k$  values for  $k$  levels.
- `factor_labeller`, `endpoint_labeller`  
 Either
- A function which returns labels for the input: First argument, a vector of either (factor.ids) or (endpoints), resp. If the function takes ... or two arguments, as second argument a data frame with (at least) the columns `survivalResult`, `endpoint`, `factor.id`, `factor.name`, `factor.value`, `HR`, `Lower_CI`, `Upper_CI`, `p`, `n`, where `survivalResult` is the corresponding result object passed to `forest_plot`; Note the function must be vectorized, if you have a non-vectorized function taking single arguments, you may want to have a look at `purrr::map_chr` or `purrr::pmap_chr`.
  - a dictionaryish list, looks up by (endpoints) or (factor.ids). The `factor.id` value: For continuous factors, the factor name (column name in data frame); For categorical factors, factor name, `factor_id_sep`, and the factor level value. (note: If `use_one_hot = FALSE`, the HR is factor level value vs. cox reference given to `survival_analysis`; if `use_one_hot = TRUE`, the HR is the factor level value vs. remaining population)
- `orderer` A function which returns an integer ordering vector for the input:
- if the supplied function takes exactly one argument, a data frame with (at least) the columns `survivalResult`, `endpoint`, `factor.id`, `factor.name`, `fac-`



tor.value, HR, Lower\_CI, Upper\_CI, p, n, subgroup\_n where survivalResult is the corresponding result object passed to forest\_plot;

- or, if the function takes more than one argument, or its arguments include ..., the nine vectors (endpoint, factor.name, factor.value, HR, Lower\_CI, Upper\_CI, p, n, subgroup\_n): a vector of endpoints (as given to Surv(endpoint, ...) in coxph), a vector of factors (as given to the right hand side of the coxph formula), and numeric vectors of the HR, lower CI, upper CI, p-value
- You can create a function from ordered vectors via `orderer_function_from_sorted_vectors`, or call `order()` with one or more of these vectors.
- Alternatively, you can provide a quosure of code, or a right-hand side formula; it will be executed such that the above nine vectors are available as symbols.

Example:

- `orderer = quo(order(endpoint, HR))`
- equivalent to `orderer = ~order(endpoint, HR)`
- equivalent to `orderer = function(df) df %>% order(endpoint, HR)`
- equivalent to `orderer = function(df) { order(df$endpoint, df$HR) }`
- equivalent to `orderer = function(endpoint, factor.name, factor.value, HR, ...) order(endpoint, HR)`

categorizer	A function which returns one logical value if a breaking line should be inserted <code>_above_</code> the input: Same semantics as for <code>orderer</code> . !Please note!: The order of the data is not yet ordered as per your <code>orderer</code> ! If you do calculations depending on order, first order with your own <code>orderer</code> function. A proper implementation is easy using <a href="#">sequential_duplicates</a> , for example <code>categorizer=~!sequential_duplicates(endpoint, HR)</code>
relative_widths	relation of the width of the plots, labels, plot, values. Default is 1:1:1.
ggtheme	ggplot2 theme to use
labels_displayed	Combination of "endpoint", "factor", "n", determining what is shown on the left-hand table and in which order.
label_headers	Named vector with name=<allowed values of labels_displayed>, value=<your heading>.
values_displayed	Combination of "HR", "CI", "p", "subgroup_n", determining what is shown on the right-hand table and in which order. Note: subgroup_n is only applicable if <code>oneHot=TRUE</code> .
value_headers	Named vector with name=<allowed values of values_displayed>, value=<your heading>.
HRsprintfFormat, psprintfFormat	<code>sprintf()</code> format strings for hazard ratio and p value
p_less_than_cutoff	The lower limit below which p value will be displayed as "less than". If <code>p_less_than_cutoff == 0.001</code> , the a p value of 0.002 will be displayed as is, while 0.0005 will become "p < 0.001".

log_scale	Plot on log scale, which is quite common and gives symmetric length for the CI bars. Note that HRs of 0 (did not converge) will not be plotted in this case.
HR_x_breaks	Breaks of the x scale for plotting HR and CI
HR_x_limits	Limits of the x scale for plotting HR and CI. Default (HR_x_lim = NULL) depends on log_scale and existing limits. Pass NA to use the existing minimum and maximum values without interference. Pass a vector of size 2 to specify (min, max) manually
factor_id_sep	Allows you to customize the separator of the factor id, the documentation of factor_labeller.
na_rm	Only used in the multivariate case (use_one_hot = FALSE). Should null coefficients (NA/0/Inf) be removed?
title, title_relative_height, title_label_args	A title on top of the plot, taking a fraction of title_relative_height of the returned plot. The title is drawn using draw_label; you can specify any arguments to this function by giving title_label_args Per default, font attributes are taken from the "title" entry from the given ggtheme, and the label is drawn centered as per draw_label defaults.
base_papersize	numeric vector of length 2, c(width, height), unit inches. forest_plot will store a suggested "papersize" attribute in the return value, computed from base_papersize and the number of entries in the plot (in particular, the height will be adjusted) The attribute is read by save_pdf. It will also store a "forestplot_entries" attribute which you can use for your own calculations.
.df	Data frame containing the columns survivalResult, endpoint, factor.id, factor.name, factor.value giving the information that is to be presented in the forest plot
df	For the variant taking a data frame: A data frame which must contain (at least) the columns: endpoint, factor.id, factor.name, factor.value, HR, Lower_CI, Upper_CI, p, n, subgroup_n

## Details

The plot has a left column containing the labels (covariate name, levels for categorical variables, optionally subgroup size), the actual line plot in the middle column, and a right column to display the hazard ratios and their confidence intervals. A rich set of parameters allows full customizability to create publication-ready plots.

## Value

A ggplot2 plot object

## Functions

- `forest_plot.df`: Creates a forest plot from the given data frame

## See Also

[forest\\_plot\\_grid](#)

## Examples

```
library(magrittr)
library(dplyr)
survival::colon %>%
  analyse_multivariate(vars(time, status),
                      vars(rx, sex, age, obstruct, perfor, nodes, differ, extent)) %>%
  forest_plot()
```

---

forest_plot_grid	<i>Create a grid of forest plots</i>
------------------	--------------------------------------

---

## Description

Makes use of the stored layout information in a [forest\\_plot](#) plot to create grids of plots.

## Usage

```
forest_plot_grid(
  ...,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  plot_grid_args = list()
)
```

## Arguments

...	Pass individual plots returned by <a href="#">forest_plot</a> , or lists of such plots (bare lists will be spliced).
nrow, ncol	Specify the grid (one is sufficient, uses auto layout if both are null)
byrow	If the plots are given in by-row, or by-column (byrow=FALSE) order
plot_grid_args	Additional arguments to the <a href="#">plot_grid</a> function which is used to create the grid.

## Value

Return value of [plot\\_grid](#)

---

```
format.SurvivalAnalysisMultivariateResult
Formats a SurvivalAnalysisMultivariateResult for printing
```

---

**Description**

Formats a SurvivalAnalysisMultivariateResult for printing

**Usage**

```
## S3 method for class 'SurvivalAnalysisMultivariateResult'
format(x, ..., p_precision = 3, hr_precision = 2, p_less_than_cutoff = 0.001)
```

**Arguments**

```
x          The result generated by analyse\_multivariate
...        Further arguments passed from other methods.
p_precision, hr_precision
           Precision with which to print floating point values
p_less_than_cutoff
           Cut-off for small p values. Values smaller than this will be displayed like "<..."
```

**Value**

A formatted string, ready for output with cat()

---

```
format.SurvivalAnalysisUnivariateResult
Formats a SurvivalAnalysisUnivariateResult for printing
```

---

**Description**

Formats a SurvivalAnalysisUnivariateResult for printing

**Usage**

```
## S3 method for class 'SurvivalAnalysisUnivariateResult'
format(
  x,
  ...,
  label = NULL,
  p_precision = 3,
  hr_precision = 2,
  p_less_than_cutoff = 0.001,
  time_precision = 1,
```

```

include_end_separator = FALSE,
timespan_unit = c("days", "months", "years")
)

```

### Arguments

`x` The result generated by [analyse\\_survival](#)

`...` Further arguments passed from other methods.

`label` A label describing the result

`p_precision`, `hr_precision`, `time_precision`  
Precision with which to print floating point values

`p_less_than_cutoff`  
Cut-off for small p values. Values smaller than this will be displayed like "<..."

`include_end_separator`  
Append "\n—\n"? Comes handy if printing multiple results following each other

`timespan_unit` Unit for time spans: "days", "months" or "years".

### Value

A formatted string, ready for output with `cat()`

---

`ggsurvplot_to_gtable` *Build a gtable representation from a ggsurvplot object*

---

### Description

Build a gtable representation from a ggsurvplot object

### Usage

```

ggsurvplot_to_gtable(
  ggsurv_obj,
  surv.plot.height = NULL,
  risk.table.height = NULL,
  ncensor.plot.height = NULL
)

```

### Arguments

`ggsurv_obj` The ggsurvplot object

`surv.plot.height`, `risk.table.height`, `ncensor.plot.height`  
Layout parameters, see [arrange\\_ggsurvplots](#)

### Value

A gtable object

---

grid_layout	<i>Grid layouting</i>
-------------	-----------------------

---

**Description**

Creates a grid layout nrow x ncol for n items.

**Usage**

```
grid_layout(n, rows = NULL, cols = NULL)
```

**Arguments**

n	Number of items in grid
rows, cols	Pass one of rows or cols, or none, in which case auto layout is used.

**Value**

A numeric vector of length 2: rows, cols

**Examples**

```
grid_layout(24, cols=4)
grid_layout(24)
grid_layout(24, rows=2)
```

---

identity_order	<i>Ordering function: identity order</i>
----------------	--

---

**Description**

This can be used in a place where a function with a signature like `order` is required. It simply retains the original order.

**Usage**

```
identity_order(x, ...)
```

**Arguments**

x	a vector
...	Effectively ignored

**Value**

An integer vector

---

kaplan\_meier\_grid      *A grid of kaplan meier plots*

---

## Description

A grid of kaplan meier plots

## Usage

```
kaplan_meier_grid(
  ...,
  nrow = NULL,
  ncol = NULL,
  layout_matrix = NULL,
  byrow = TRUE,
  mapped_plot_args = list(),
  paperwidth = NULL,
  paperheight = NULL,
  size_per_plot = dinAWidth(5),
  title = NA,
  surv.plot.height = NULL,
  risk.table.height = NULL,
  ncensor.plot.height = NULL,
  p_lessthan_cutoff = 0.001
)
```

## Arguments

- ...
- One or many `SurvivalAnalysisResult` objects as returned by [analyse\\_survival](#) and arguments that will be passed to `ggsurvplot`. Bare lists will be spliced. If using lists, the same argument may be contained in multiple lists; in this case, the last occurrence is used, i.e. you can first pass a list with default arguments, and then override some of them. In addition to all arguments supported by [ggsurvplot](#), these arguments and shortcuts can be used additionally:
- `break.time.by`: `breakByYear`, `breakByHalfYear`, `breakByQuarterYear`, `breakByMonth` (numeric value only in `ggsurvplot`)
  - `xscale`: `scaleByYear`, `scaleByMonth` (numeric value only in `ggsurvplot`)
  - `hazard.ratio` (logical): display hazard ratios in addition to p value, complementing `pval=T`
  - `xlab`: `{.OS,.PFS,.TTF,.DFS}.{years,months,days}`
  - `table.layout`: `clean`, displays risk table only with color code and number, no grid, axes or labels. (do not forget `risk.table=TRUE` to see something)
  - `papersize`: numeric vector of length 2, `c(width, height)`, unit inches. `kaplan_meier_plot` will store a "papersize" attribute with this value which is read by [save\\_pdf](#)

nrow, ncol	Determines the layout by giving nrow and/or ncol, if missing, uses an auto layout.
layout_matrix	Optionally specify a layout matrix, which is passed to <code>gridExtra::marrangeGrob</code>
byrow	If no layout_matrix is specified and there are multiple rows: How should the plots be laid out? The order of the plots can be by-row (default) or by-col (set byrow=FALSE).
mapped_plot_args	Optionally, if given n objects to plot, a named list of vectors of size n. The name is an argument name passed to <code>ggsurvplot</code> . The elements of the vector will be mapped 1:1 to each object. This allows to perform batch plotting where only few arguments differ (e.g. titles A, B, C...) between the plots. Please note that only objects that need plotting (survival_analysis results) are considered, not those that are already plotted ( <code>kaplan_meier_plot</code> results)
paperwidth, paperheight, size_per_plot	You can specify the size per plot, or the full paper width and height. <code>size_per_plot</code> may be a number (width == height) or two-dimensional, width and height. The resulting paper size will be stored as a <code>papersize</code> attribute that is e.g. read by <code>tidytidbits::save_pdf</code>
title, surv.plot.height, risk.table.height, ncensor.plot.height	Passed to <code>arrange_ggsurvplots</code>
p_less_than_cutoff	The lower limit below which p value will be displayed as "less than". If <code>p_less_than_cutoff</code> == 0.001, the a p value of 0.002 will be displayed as is, while 0.0005 will become "p < 0.001".

**Value**

An object of class `arrangelist`, which can be printed or saved to pdf with `ggsave()`.

---

`kaplan_meier_plot`      *Kaplan Meier plots from survival results.*

---

**Description**

Uses `ggsurvplot` from the `survminer` package to create publication-ready plots.

**Usage**

```
kaplan_meier_plot(..., mapped_plot_args = list(), p_less_than_cutoff = 0.001)
```

**Arguments**

...      One or many `SurvivalAnalysisResult` objects as returned by `analyse_survival` and arguments that will be passed to `ggsurvplot`. Bare lists will be spliced. If using lists, the same argument may be contained in multiple lists; in this case, the last occurrence is used, i.e. you can first pass a list with default arguments, and then override some of them. In addition to all arguments supported by `ggsurvplot`, these arguments and shortcuts can be used additionally:



- `break.time.by`: `breakByYear`, `breakByHalfYear`, `breakByQuarterYear`, `breakByMonth` (numeric value only in `ggsurvplot`)
- `xscale`: `scaleByYear`, `scaleByMonth` (numeric value only in `ggsurvplot`)
- `hazard.ratio` (logical): display hazard ratios in addition to p value, complementing `pval=T`
- `xlab`: `{.OS,.PFS,.TTF,.DFS}.{years,months,days}`
- `table.layout`: `clean`, displays risk table only with color code and number, no grid, axes or labels. (do not forget `risk.table=TRUE` to see something)
- `papersize`: numeric vector of length 2, `c(width, height)`, unit inches. `kaplan_meier_plot` will store a "papersize" attribute with this value which is read by `save_pdf`

**mapped\_plot\_args**

Optionally, if given `n` objects to plot, a named list of vectors of size `n`. The name is an argument names passed to `ggsurvplot`. The elements of the vector will be mapped 1:1 to each object. This allows to perform batch plotting where only few arguments differ (e.g. titles A, B, C...) between the plots.

**p\_lessthan\_cutoff**

The lower limit below which p value will be displayed as "less than". If `p_lessthan_cutoff == 0.001`, the a p value of 0.002 will be displayed as is, while 0.0005 will become "p < 0.001".

**Value**

If given one result to plot, one `ggsurvplot` object; if given more than one result, a list of `ggsurvplot` objects.

**Examples**

```
library(magrittr)
library(dplyr)
survival::aml %>%
  analyse_survival(vars(time, status), x) %>%
  kaplan_meier_plot(break.time.by="breakByMonth",
                   xlab=".OS.months",
                   risk.table=TRUE,
                   ggtheme=ggplot2::theme_bw(10))
```

---

**multivariate\_as\_data\_frame**

*Turns a multivariate analysis result to a data frame*

---

**Description**

Extracts useful information into a data frame which is ready for printing or further analysis

**Usage**

```
multivariate_as_data_frame(result, factor_id_sep = ":", sort_by = NULL)
```

**Arguments**

result	An object of class "SurvivalAnalysisMultivariateResult" as returned by <a href="#">analyse_multivariate</a>
factor_id_sep	The frame contains one column "factor.id" which is a composite of covariate name and, if categorical, the factor level (one line for each factor level except for the reference level)
sort_by	A vars() list of one or more symbolic column names. This frame contains the variables "Lower_CI", "HR", "Upper_CI", "Inv_Lower_CI", "Inv_HR", "Inv_Upper_CI", "p". You can choose to sort by any combination. Use desc() to sort a variable in descending order.

**Value**

A tibble.

---

pluck\_multivariate\_analysis

*Access individual components of multivariate survival analysis*

---

**Description**

Allows access to the [analyse\\_multivariate](#) result object.

**Usage**

```
pluck_multivariate_analysis(result, term)
```

**Arguments**

result	An object of class SurvivalAnalysisMultivariateResult as returned by <a href="#">analyse_multivariate</a>
term	The item to be retrieved: <ul style="list-style-type: none"> <li>• "coxph" containing the result of the <a href="#">coxph</a> function</li> <li>• "summary" containing the result of the <a href="#">summary</a> of the "coxph" result</li> <li>• "summary_data_frame" containing summary as a data frame (see <a href="#">multivariate_as_data_frame</a>)</li> <li>• "p" A vector of p values for the covariates, equivalent to the "p" column of "summary_data_frame"</li> <li>• "overall" A named list with human-readable labels giving information about the overall fit, including the three flavors of p values contained in "summary"</li> </ul>

**Value**

object as specified by term, or NULL if not contained in result

## Examples

```
library(magrittr)
library(dplyr)
survival::colon %>%
  analyse_multivariate(vars(time, status),
                      vars(rx, sex, age, obstruct, perfor, nodes, differ, extent)) %>%
  pluck_multivariate_analysis("p")
print
```

---

pluck\_survival\_analysis

*Access individual components of univariate survival analysis*

---

## Description

Allows access to the [analyse\\_survival](#) result object.

## Usage

```
pluck_survival_analysis(result, term)
```

## Arguments

result	An object of class <code>SurvivalAnalysisUnivariateResult</code> as returned by <a href="#">analyse_survival</a>
term	The item to be retrieved: <ul style="list-style-type: none"><li>• "survfit" containing the result of the <a href="#">survfit</a> function</li><li>• "survdiff" containing the result of the <a href="#">survdiff</a> function</li><li>• "survfit_overall" containing the result of the <a href="#">survfit</a> function without terms, i.e. the full group not comparing subgroups</li><li>• "coxph" containing the result of the <a href="#">coxph</a> function</li><li>• "p" The log-rank p value (if by provided at least two strata)</li></ul>

## Value

object as specified by term, or NULL if not contained in result

## Examples

```
library(magrittr)
library(dplyr)
survival::aml %>%
  analyse_survival(vars(time, status), x) %>%
  pluck_survival_analysis("p") %>%
  print
```

---

```
print.SurvivalAnalysisMultivariateResult
    Print the essentials of a SurvivalAnalySurvivalAnalysisMultivari-
    ateResult
```

---

**Description**

Print the essentials of a SurvivalAnalySurvivalAnalysisMultivariateResult

**Usage**

```
## S3 method for class 'SurvivalAnalysisMultivariateResult'
print(x, ..., p_precision = 3, hr_precision = 2, p_less_than_cutoff = 0.001)
```

**Arguments**

<code>x</code>	The result generated by <a href="#">analyse_multivariate</a>
<code>...</code>	Further arguments passed from other methods.
<code>p_precision</code>	Precision with which to print floating point values
<code>hr_precision</code>	Precision with which to print floating point values
<code>p_less_than_cutoff</code>	Cut-off for small p values. Values smaller than this will be displayed like "<..."

**Value**

The formatted string, invisibly. #' @export

---

```
print.SurvivalAnalysisUnivariateResult
    Print the essentials of a SurvivalAnalysisUnivariateResult
```

---

**Description**

Print the essentials of a SurvivalAnalysisUnivariateResult

**Usage**

```
## S3 method for class 'SurvivalAnalysisUnivariateResult'
print(
  x,
  ...,
  label = NULL,
  p_precision = 3,
  hr_precision = 2,
```

```

    time_precision = 1,
    include_end_separator = FALSE,
    timespan_unit = c("days", "months", "years")
  )

```

### Arguments

x	The result generated by <a href="#">analyse_survival</a>
...	Further arguments passed from other methods.
label	A label describing the result
p_precision	Precision with which to print floating point values
hr_precision	Precision with which to print floating point values
time_precision	Precision with which to print floating point values
include_end_separator	Append "\n—\n"? Comes handy if printing multiple results following each other
timespan_unit	Unit for time spans: "days", "months" or "years".

### Value

The formatted string, invisibly.

---

survival\_data\_frames *Extract results from univariate survival analysis structured as data frames*

---

### Description

Extract results from univariate survival analysis structured as data frames

### Usage

```

survival_data_frames(
  result,
  format_numbers = TRUE,
  p_precision = 3,
  hr_precision = 2,
  p_less_than_cutoff = 0.001,
  time_precision = 1,
  timespan_unit = c("days", "months", "years")
)

```

**Arguments**

- `result`            The result generated by `analyse_survival`
- `format_numbers`   If true, all numbers will be formatted for printing according to the following options and will be returned as strings
- `p_precision`, `hr_precision`, `time_precision`  
Precision with which to print floating point values
- `p_less_than_cutoff`  
Cut-off for small p values. Values smaller than this will be displayed like "<..."
- `timespan_unit`    Unit for time spans: "days", "months" or "years".

**Value**

A named list list of data frame objects:

- `cohortMetadata`: information about the full cohort
- if there are strata (analysis performed "by" a covariate):
  - `strataMetadata`: information about each stratum
  - `hazardRatios`: hazard ratios for combinations of strata
  - only if there are more than two strata:
    - \* `pairwisePValues`: Matrix of pairwise (uncorrected) p values

---

`survival_essentials`    *Convenience formatting and printing of result*

---

**Description**

Takes the given result, formats and prints it

**Usage**

```
survival_essentials(
  result,
  label = NULL,
  p_precision = 3,
  hr_precision = 2,
  time_precision = 1,
  include_end_separator = TRUE,
  timespan_unit = "days",
  print = TRUE
)
```

**Arguments**

result	The result generated by <a href="#">analyse_survival</a>
label	Optional label to include
p_precision	Precision with which to print floating point values
hr_precision	Precision with which to print floating point values
time_precision	Precision with which to print floating point values
include_end_separator	Append "\n—\n"? Comes handy if printing multiple results following each other
timespan_unit	Unit for time spans: "days", "months" or "years".
print	Print string to console

**Value**

The formatted string, invisibly. Ready for output with cat or saving to a file.

# Index

analyse\_multivariate, [2](#), [7](#), [8](#), [12](#), [18](#), [20](#)  
analyse\_survival, [4](#), [7](#), [8](#), [13](#), [15](#), [16](#), [19](#),  
[21–23](#)  
analyze\_multivariate  
    (analyse\_multivariate), [2](#)  
analyze\_survival (analyse\_survival), [4](#)  
arrange\_ggsurvplots, [13](#), [16](#)

cox\_as\_data\_frame, [3](#), [6](#)  
coxph, [18](#), [19](#)

forest\_plot, [4](#), [7](#), [11](#)  
forest\_plot\_grid, [10](#), [11](#)  
format, [3](#), [5](#)  
format.SurvivalAnalysisMultivariateResult,  
[12](#)  
format.SurvivalAnalysisUnivariateResult,  
[12](#)

ggsurvplot, [15](#), [16](#)  
ggsurvplot\_to\_gtable, [13](#)  
grid\_layout, [14](#)

identity\_order, [14](#)

kaplan\_meier\_grid, [5](#), [15](#)  
kaplan\_meier\_plot, [5](#), [16](#)

marrangeGrob, [16](#)  
multivariate\_as\_data\_frame, [3](#), [17](#), [18](#)

order, [14](#)

plot\_grid, [11](#)  
pluck\_multivariate\_analysis, [3](#), [18](#)  
pluck\_survival\_analysis, [5](#), [19](#)  
print, [3](#), [5](#)  
print.SurvivalAnalysisMultivariateResult,  
[20](#)  
print.SurvivalAnalysisUnivariateResult,  
[20](#)

save\_pdf, [15–17](#)  
sequential\_duplicates, [9](#)  
summary, [18](#)  
survdiff, [19](#)  
survfit, [19](#)  
survival\_data\_frames, [5](#), [21](#)  
survival\_essentials, [22](#)