Package 'pammtools'

May 9, 2026

Title Piece-Wise Exponential Additive Mixed Modeling Tools for Survival Analysis

Version 0.7.4

Date 2026-02-16

Description The Piece-wise exponential (Additive Mixed) Model (PAMM; Bender and others (2018) <doi:10.1177/1471082X17748083>) is a powerful model class for the analysis of survival (or time-to-event) data, based on Generalized Additive (Mixed) Models (GA(M)Ms).
It offers intuitive specification and robust estimation of complex survival models with stratified baseline hazards, random effects, time-varying effects, time-dependent covariates and cumulative effects (Bender and others (2019)), as well as support for left-truncated data as well as competing risks, recurrent events and multi-state settings.
pammtools provides tidy workflow for survival analysis with PAMMs, including data simulation, transformation and other functions for data preprocessing and model post-processing as well as visualization.

Depends R (>= 4.1.0)

Imports mgcv, survival (>= 2.39-5), checkmate, magrittr, rlang, tidyr (>= 1.0.0), ggplot2 (>= 3.2.2), dplyr (>= 1.0.0), purrr (>= 0.2.3), tibble, lazyeval, Formula, mvtnorm, pec, vctrs (>= 0.3.0), scam

Suggests testthat, mstate, broom

Config/Needs/website coxme, eha, etm, scam, msm, mvna, TBFmultinomial

License MIT + file LICENSE

LazyData true

URL https://adibender.github.io/pammtools/

BugReports https://github.com/adibender/pammtools/issues

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Andreas Bender [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5628-8611>),
Fabian Scheipl [aut] (ORCID: <https://orcid.org/0000-0001-8172-3603>),
Johannes Piller [aut] (ORCID: <https://orcid.org/0009-0008-3010-9556>),
Philipp Kopper [aut] (ORCID: <https://orcid.org/0000-0002-5037-7135>),
Lukas Burk [ctb] (ORCID: <https://orcid.org/0000-0001-7528-3795>)

Maintainer Andreas Bender <andreas.bender@stat.uni-muenchen.de>

Repository CRAN

Date/Publication 2026-02-16 13:40:02 UTC

Contents

`add_cif` *Add cumulative incidence function to data*

Description

Add cumulative incidence function to data

Usage

```
add_cif(newdata, object, ...)

## Default S3 method:
add_cif(
  newdata,
  object,
  ci = TRUE,
  overwrite = FALSE,
  alpha = 0.05,
  nsim = 500L,
  cause_var = "cause",
  time_var = NULL,
  ...
)
```

Arguments

`newdata`	A data frame or list containing the values of the model covariates at which predictions are required. If this is not provided then predictions corresponding to the original data are returned. If `newdata` is provided then it should contain all the variables needed for prediction: a warning is generated if not. See details for use with `link{linear.functional.terms}`.
`object`	a fitted `gam` object as produced by `gam()`.
`...`	Further arguments passed to `predict.gam` and `get_hazard`
`ci`	`logical`. Indicates if confidence intervals should be calculated. Defaults to `TRUE`.
`overwrite`	Should hazard columns be overwritten if already present in the data set? Defaults to `FALSE`. If `TRUE`, columns with names `c("hazard", "se", "lower", "upper")` will be overwritten.
`alpha`	The alpha level for confidence/credible intervals.
`nsim`	Number of simulations (draws from posterior of estimated coefficients) on which estimation of CIFs and their confidence/credible intervals will be based on.
`cause_var`	Character. Column name of the 'cause' variable.

time_var	Name of the variable used for the baseline hazard. If not given, defaults to "tend" for gam fits, else "interval". The latter is assumed to be a factor, the former numeric.

---

add_counterfactual_transitions *Add counterfactual observations for possible transitions*

Description

If data only contains one row per transition that took place, this function adds additional rows for each transition that was possible at that time (for each subject in the data).

Usage

```
add_counterfactual_transitions(
  data,
  from_to_pairs = list(),
  from_col = "from",
  to_col = "to",
  transition_col = "transition"
)
```

Arguments

data	Data set that only contains rows for transitions that took place.
from_to_pairs	A list with one element for each possible initial state. The values of each list element indicate possible transitions from that state. Will be calculated from the data if unspecified.
from_col	Name of the column that stores initial state.
to_col	Name of the column that stores end state.
transition_col	Name of the column that contains the transition identifier (factor variable).

---

add_hazard *Add predicted (cumulative) hazard to data set*

Description

Add (cumulative) hazard based on the provided data set and model. If ci=TRUE confidence intervals (CI) are also added. Their width can be controlled via the se_mult argument. The method by which the CI are calculated can be specified by ci_type. This is a wrapper around predict.gam. When reference is specified, the (log-)hazard ratio is calculated.

Usage

```
add_hazard(newdata, object, ...)

## Default S3 method:
add_hazard(
  newdata,
  object,
  reference = NULL,
  type = c("response", "link"),
  ci = TRUE,
  se_mult = 2,
  ci_type = c("default", "delta", "sim"),
  overwrite = FALSE,
  time_var = NULL,
  ...
)

add_cumu_hazard(
  newdata,
  object,
  ci = TRUE,
  se_mult = 2,
  overwrite = FALSE,
  time_var = NULL,
  interval_length = "intlen",
  ...
)
```

Arguments

newdata	A data frame or list containing the values of the model covariates at which predictions are required. If this is not provided then predictions corresponding to the original data are returned. If newdata is provided then it should contain all the variables needed for prediction: a warning is generated if not. See details for use with link{linear.functional.terms}.
object	a fitted gam object as produced by gam().
...	Further arguments passed to predict.gam and get_hazard
reference	A data frame with number of rows equal to nrow(newdata) or one, or a named list with (partial) covariate specifications. See examples.
type	Either "response" or "link". The former calculates hazard, the latter the log-hazard.
ci	logical. Indicates if confidence intervals should be calculated. Defaults to TRUE.
se_mult	Factor by which standard errors are multiplied for calculating the confidence intervals.

ci_type — The method by which standard errors/confidence intervals will be calculated. Default transforms the linear predictor at respective intervals. "delta" calculates CIs based on the standard error calculated by the Delta method. "sim" draws the property of interest from its posterior based on the normal distribution of the estimated coefficients. See here for details and empirical evaluation.

overwrite — Should hazard columns be overwritten if already present in the data set? Defaults to FALSE. If TRUE, columns with names c("hazard", "se", "lower", "upper") will be overwritten.

time_var — Name of the variable used for the baseline hazard. If not given, defaults to "tend" for gam fits, else "interval". The latter is assumed to be a factor, the former numeric.

interval_length — The variable in newdata containing the interval lengths. Can be either bare unquoted variable name or character. Defaults to "intlen".

See Also

predict.gam, add_surv_prob

Examples

```
ped <- tumor[1:50,] %>% as_ped(Surv(days, status)~ age)
pam <- mgcv::gam(ped_status ~ s(tend)+age, data = ped, family=poisson(), offset=offset)
ped_info(ped) %>% add_hazard(pam, type="link")
ped_info(ped) %>% add_hazard(pam, type = "response")
ped_info(ped) %>% add_cumu_hazard(pam)
```

---

add_surv_prob *Add survival probability estimates*

Description

Given suitable data (i.e. data with all columns used for estimation of the model), this functions adds a column surv_prob containing survival probabilities for the specified covariate and follow-up information (and CIs surv_lower, surv_upper if ci=TRUE).

Usage

```
add_surv_prob(
  newdata,
  object,
  ci = TRUE,
  se_mult = 2,
  overwrite = FALSE,
  time_var = NULL,
  interval_length = "intlen",
  ...
)
```

Arguments

newdata	A data frame or list containing the values of the model covariates at which predictions are required. If this is not provided then predictions corresponding to the original data are returned. If newdata is provided then it should contain all the variables needed for prediction: a warning is generated if not. See details for use with link{linear.functional.terms}.
object	a fitted gam object as produced by gam().
ci	logical. Indicates if confidence intervals should be calculated. Defaults to TRUE.
se_mult	Factor by which standard errors are multiplied for calculating the confidence intervals.
overwrite	Should hazard columns be overwritten if already present in the data set? Defaults to FALSE. If TRUE, columns with names c("hazard", "se", "lower", "upper") will be overwritten.
time_var	Name of the variable used for the baseline hazard. If not given, defaults to "tend" for gam fits, else "interval". The latter is assumed to be a factor, the former numeric.
interval_length	The variable in newdata containing the interval lengths. Can be either bare unquoted variable name or character. Defaults to "intlen".
...	Further arguments passed to predict.gam and get_hazard

See Also

predict.gam, add_surv_prob

Examples

```
ped <- tumor[1:50,] %>% as_ped(Surv(days, status)~ age)
pam <- mgcv::gam(ped_status ~ s(tend)+age, data=ped, family=poisson(), offset=offset)
ped_info(ped) %>% add_surv_prob(pam, ci=TRUE)
```

---

add_tdc *Add time-dependent covariate to a data set*

Description

Given a data set in standard format (with one row per subject/observation), this function adds a column with the specified exposure time points and a column with respective exposures, created from rng_fun. This function should usually only be used to create data sets passed to sim_pexp.

Usage

```
add_tdc(data, tz, rng_fun, ...)
```

Arguments

data	A data set with variables specified in formula.
tz	A numeric vector of exposure times (relative to the beginning of the follow-up time t)
rng_fun	A random number generating function that creates the time-dependent covariates at time points tz. First argument of the function should be n, the number of random numbers to generate. Within add_tdc, n will be set to length(tz).
...	Currently not used.

---

add_term *Embeds the data set with the specified (relative) term contribution*

Description

Adds the contribution of a specific term to the linear predictor to the data specified by newdata. Essentially a wrapper to predict.gam, with type="terms". Thus most arguments and their documentation below is from predict.gam.

Usage

```
add_term(newdata, object, term, reference = NULL, ci = TRUE, se_mult = 2, ...)
```

Arguments

newdata	A data frame or list containing the values of the model covariates at which predictions are required. If this is not provided then predictions corresponding to the original data are returned. If newdata is provided then it should contain all the variables needed for prediction: a warning is generated if not. See details for use with link{linear.functional.terms}.
object	a fitted gam object as produced by gam().
term	A character (vector) or regular expression indicating for which term(s) information should be extracted and added to data set.
reference	A data frame with number of rows equal to nrow(newdata) or one, or a named list with (partial) covariate specifications. See examples.
ci	logical. Indicates if confidence intervals should be calculated. Defaults to TRUE.
se_mult	The factor by which standard errors are multiplied to form confidence intervals.
...	Further arguments passed to predict.gam

Examples

```
library(ggplot2)
ped <- as_ped(tumor, Surv(days, status)~ age, cut = seq(0, 2000, by = 100))
pam <- mgcv::gam(ped_status ~ s(tend) + s(age), family = poisson(),
  offset = offset, data = ped)
#term contribution for sequence of ages
s_age <- ped %>% make_newdata(age = seq_range(age, 50)) %>%
  add_term(pam, term = "age")
ggplot(s_age, aes(x = age, y = fit)) + geom_line() +
  geom_ribbon(aes(ymin = ci_lower, ymax = ci_upper), alpha = .3)
# term contribution relative to mean age
s_age2 <- ped %>% make_newdata(age = seq_range(age, 50)) %>%
  add_term(pam, term = "age", reference = list(age = mean(.$age)))
ggplot(s_age2, aes(x = age, y = fit)) + geom_line() +
  geom_ribbon(aes(ymin = ci_lower, ymax = ci_upper), alpha = .3)
```

---

add_trans_prob *Add transition probabilities*

Description

add_trans_prob adds transition probabilities on the provided data set and model. Optionally, confidence intervals (CI) are added if ci=TRUE. The function builds on cumulative hazards cumu_hazard and mgcv::gam models.

Usage

```
add_trans_prob(
  newdata,
  object,
  overwrite = FALSE,
  ci = FALSE,
  alpha = 0.05,
  nsim = 100L,
  time_var = NULL,
  interval_length = "intlen",
  ...
)
```

Arguments

newdata	A data frame or list containing the values of the model covariates at which predictions are required. If this is not provided then predictions corresponding to the original data are returned. If newdata is provided then it should contain all the variables needed for prediction: a warning is generated if not. See details for use with linear.functional.terms.
object	A fitted gam object as produced by mgcv::gam

overwrite	Should transition probability columns be overwritten if already present in the data set? Defaults to FALSE. If TRUE, columns with names c("trans_prob", "trans_upper", "trans_lower") will be overwritten.
ci	Logical, defaults to TRUE. Decides if confidence intervals for transition probabilities are calculated.
alpha	Sets the confidence intervals' α level, Defaults to 0.05
nsim	Sets the number of iterations for simulated confidence intervals. Defaults to 100L
time_var	Name of the variable used for the baseline hazard. If not given, defaults to "tend" for gam fits, else "interval". The latter is assumed to be a factor, the former numeric.
interval_length	Character, defaults to "intlen". contains the interval length in newdata.
...	Further arguments passed to underlying methods.

Examples

```
data("prothr", package = "mstate")
prothr <- prothr |>
  mutate(transition = as.factor(paste0(from, "->", to))
  , treat = as.factor(treat)) |>
  filter(Tstart != Tstop, id <= 100) |> select(-trans)
ped <- as_ped(data= prothr, formula= Surv(Tstart, Tstop, status)~ .,
  transition = "transition", id= "id", timescale  = "calendar")
pam <- mgcv::bam(ped_status ~ s(tend, by=transition) + transition * treat,
  data = ped, family = poisson(), offset = offset,
  method = "fREML", discrete = TRUE)
ndf <- make_newdata(ped, tend  = unique(tend),
  treat  = unique(treat),
  transition = unique(transition)) |>
  group_by(treat, transition) |>  # important!
  arrange(treat, transition, tend) |>
  add_trans_prob(pam)
```

---

as.data.frame.crps *Transform crps object to data.frame*

Description

Aas.data.frame S3 method for objects of class crps.

Usage

```
## S3 method for class 'crps'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	An object of class crps. See crps.
row.names	NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
optional	logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional. Note that all of R's **base** package as.data.frame() methods use optional only for column names treatment, basically with the meaning of data.frame(*, check.names = !optional). See also the make.names argument of the matrix method.
...	additional arguments to be passed to or from methods.

---

daily *Time-dependent covariates of the patient data set.*

Description

This data set contains the time-dependent covariates (TDCs) for the patient data set. Note that nutrition was protocoled for at most 12 days after ICU admission. The data set includes:

CombinedID Unique patient identifier. Can be used to merge with patient data

Study_Day The calendar (!) day at which calories (or proteins) were administered

caloriesPercentage The percentage of target calories supplied to the patient by the ICU staff

proteinGproKG The amount of protein supplied to the patient by the ICU staff

Usage

```
daily
```

Format

An object of class tbl_df (inherits from tbl, data.frame) with 18797 rows and 4 columns.

---

geom_hazard *(Cumulative) (Step-) Hazard Plots.*

Description

geom_hazard is an extension of the geom_line, and is optimized for (cumulative) hazard plots. Essentially, it adds a (0,0) row to the data, if not already the case. Stolen from the RmcdrPlugin.KMggplot2 (slightly modified).

Usage

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

geom_stephazard(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  direction = "vh",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

geom_surv(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = 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 to 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.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

`...` 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.

`direction` direction of stairs: 'vh' for vertical then horizontal, 'hv' for horizontal then vertical, or 'mid' for step half-way between adjacent x-values.

See Also

`geom_line`, `geom_step`.

Examples

```
library(ggplot2)
library(pammtools)
ped <- tumor[10:50,] %>% as_ped(Surv(days, status)~1)
pam <- mgcv::gam(ped_status ~ s(tend), data=ped, family = poisson(), offset = offset)
ndf <- make_newdata(ped, tend = unique(tend)) %>% add_hazard(pam)
# piece-wise constant hazards
ggplot(ndf, aes(x = tend, y = hazard)) +
 geom_vline(xintercept = c(0, ndf$tend[c(1, (nrow(ndf)-2):nrow(ndf))]), lty = 3) +
 geom_hline(yintercept = c(ndf$hazard[1:3], ndf$hazard[nrow(ndf)]), lty = 3) +
 geom_stephazard() +
 geom_step(col=2) +
 geom_step(col=2, lty = 2, direction="vh")

# comulative hazard
ndf <- ndf %>% add_cumu_hazard(pam)
ggplot(ndf, aes(x = tend, y = cumu_hazard)) +
 geom_hazard() +
 geom_line(col=2) # doesn't start at (0, 0)

# survival probability
ndf <- ndf %>% add_surv_prob(pam)
ggplot(ndf, aes(x = tend, y = surv_prob)) +
 geom_surv() +
 geom_line(col=2) # doesn't start at c(0,1)
```

`geom_stepribbon` *Step ribbon plots.*

Description

geom_stepribbon is an extension of the geom_ribbon, and is optimized for Kaplan-Meier plots with pointwise confidence intervals or a confidence band. The default direction-argument "hv" is appropriate for right-continuous step functions like the hazard rates etc returned by pammtools.

Usage

```
geom_stepribbon(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  direction = "hv",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = 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 to 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.

`direction` direction of stairs: 'vh' for vertical then horizontal, 'hv' for horizontal then vertical, or 'mid' for step half-way between adjacent x-values.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

`...` 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.

See Also

`geom_ribbon` `geom_stepribbon`

Examples

```
library(ggplot2)
huron <- data.frame(year = 1875:1972, level = as.vector(LakeHuron))
h <- ggplot(huron, aes(year))
h + geom_stepribbon(aes(ymin = level - 1, ymax = level + 1), fill = "grey70") +
    geom_step(aes(y = level))
h + geom_ribbon(aes(ymin = level - 1, ymax = level + 1), fill = "grey70") +
    geom_line(aes(y = level))
```

---

get_cumu_coef *Extract cumulative coefficients (cumulative hazard differences)*

Description

These functions are designed to extract (or mimic) the cumulative coefficients usually used in additive hazards models (Aalen model) to depict (time-varying) covariate effects. For PAMMs, these are the differences between the cumulative hazard rates where all covariates except one have the identical values. For a numeric covariate of interest, this calculates $\Lambda(t|x+1) - \Lambda(t|x)$. For non-numeric covariates the cumulative hazard of the reference level is subtracted from the cumulative hazards evaluated at all non reference levels. Standard errors are calculated using the delta method.

Usage

```
get_cumu_coef(model, data = NULL, terms, ...)

## S3 method for class 'gam'
get_cumu_coef(model, data, terms, ...)

## S3 method for class 'aalen'
get_cumu_coef(model, data = NULL, terms, ci = TRUE, ...)

## S3 method for class 'cox.aalen'
get_cumu_coef(model, data = NULL, terms, ci = TRUE, ...)
```

Arguments

model	Object from which to extract cumulative coefficients.
data	Additional data if necessary.
terms	A character vector of variables for which the cumulative coefficient should be calculated.
...	Further arguments passed to methods.
ci	Logical. Indicates if confidence intervals should be returned as well.

get_cumu_eff *Calculate (or plot) cumulative effect for all time-points of the follow-up*

Description

Calculate (or plot) cumulative effect for all time-points of the follow-up

Usage

```
get_cumu_eff(data, model, term, z1, z2 = NULL, se_mult = 2)

gg_cumu_eff(data, model, term, z1, z2 = NULL, se_mult = 2, ci = TRUE)
```

Arguments

data	Data used to fit the model.
model	A suitable model object which will be used to estimate the partial effect of term.
term	A character string indicating the model term for which partial effects should be plotted.
z1	The exposure profile for which to calculate the cumulative effect. Can be either a single number or a vector of same length as unique observation time points.
z2	If provided, calculated cumulative effect is for the difference between the two exposure profiles (g(z1,t)-g(z2,t)).
se_mult	Multiplicative factor used to calculate confidence intervals (e.g., lower = fit - 2*se).
ci	Logical. Indicates if confidence intervals for the term of interest should be calculated/plotted. Defaults to TRUE.

get_intervals *Information on intervals in which times fall*

Description

Information on intervals in which times fall

Usage

```
get_intervals(x, times, ...)

## Default S3 method:
get_intervals(x, times, left.open = TRUE, rightmost.closed = TRUE, ...)
```

Arguments

x	An object from which interval information can be obtained, see int_info.
times	A vector of times for which corresponding interval information should be returned.
...	Further arguments passed to findInterval.
left.open	logical; if true all the intervals are open at left and closed at right; in the formulas below, $\leq$ should be swapped with $<$ (and $>$ with $\geq$), and rightmost.closed means 'leftmost is closed'. This may be useful, e.g., in survival analysis computations.
rightmost.closed	logical; if true, the rightmost interval, vec[N-1] .. vec[N] is treated as *closed*, see below.

Value

A data.frame containing information on intervals in which values of times fall.

See Also

findInterval int_info

Examples

```
set.seed(111018)
brks <- c(0, 4.5, 5, 10, 30)
int_info(brks)
x <- runif (3, 0, 30)
x
get_intervals(brks, x)
```

---

get_laglead *Construct or extract data that represents a lag-lead window*

Description

Constructs lag-lead window data set from raw inputs or from data objects with suitable information stored in attributes, e.g., objects created by as_ped.

Usage

```
get_laglead(x, ...)

## Default S3 method:
get_laglead(x, tz, ll_fun, ...)

## S3 method for class 'data.frame'
get_laglead(x, ...)
```

Arguments

x	Either a numeric vector of follow-up cut points or a suitable object.
...	Further arguments passed to methods.
tz	A vector of exposure times
ll_fun	Function that specifies how the lag-lead matrix should be constructed. First argument is the follow up time second argument is the time of exposure.

Examples

```
get_laglead(0:10, tz=-5:5, ll_fun=function(t, tz) { t >= tz + 2 & t <= tz + 2 + 3})
gg_laglead(0:10, tz=-5:5, ll_fun=function(t, tz) { t >= tz + 2 & t <= tz + 2 + 3})
```

---

get_plotinfo *Extract plot information for all special model terms*

Description

Given a mgcv gamObject, returns the information used for the default plots produced by plot.gam.

Usage

```
get_plotinfo(x, ...)
```

Arguments

x	a fitted gam object as produced by gam().
...	Further arguments passed to plot.gam

---

get_terms *Extract the partial effects of non-linear model terms*

Description

This function basically creates a new df from data for each term in terms, creating a range from minimum and maximum of the predict(fit, newdata=df, type="terms"). Terms are then stacked to a tidy data frame.

Usage

```
get_terms(data, fit, terms, ...)
```

Arguments

data	A data frame containing variables used to fit the model. Only first row will be used.
fit	A fitted object of class gam.
terms	A character vector (can be length one). Specifies the terms for which partial effects will be returned
...	Further arguments passed to seq_range.

Value

A tibble with 5 columns.

Examples

```
library(survival)
fit <- coxph(Surv(time, status) ~ pspline(karno) + pspline(age), data=veteran)
terms_df <- veteran %>% get_terms(fit, terms = c("karno", "age"))
head(terms_df)
tail(terms_df)
```

---

gg_fixed *Forrest plot of fixed coefficients*

Description

Given a model object, returns a data frame with columns variable, coef (coefficient), ci_lower (lower 95\ ci_upper (upper 95\

Usage

```
gg_fixed(x, intercept = FALSE, ...)
```

Arguments

x	A model object.
intercept	Logical, indicating whether intercept term should be included. Defaults to FALSE.
...	Currently not used.

See Also

tidy_fixed

Examples

```
g <- mgcv::gam(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species,
 data=iris)
gg_fixed(g, intercept=TRUE)
gg_fixed(g)
```

gg_laglead *Plot Lag-Lead windows*

Description

Given data defining a Lag-lead window, returns respective plot as a ggplot2 object.

Usage

```
gg_laglead(x, ...)

## Default S3 method:
gg_laglead(x, tz, ll_fun, ...)

## S3 method for class 'LL_df'
gg_laglead(
  x,
  high_col = "grey20",
  low_col = "whitesmoke",
  grid_col = "lightgrey",
  ...
)

## S3 method for class 'nested_fdf'
gg_laglead(x, ...)
```

Arguments

x	Either a numeric vector of follow-up cut points or a suitable object.
...	Further arguments passed to methods.
tz	A vector of exposure times
ll_fun	Function that specifies how the lag-lead matrix should be constructed. First argument is the follow up time second argument is the time of exposure.
high_col	Color used to highlight exposure times within the lag-lead window.
low_col	Color of exposure times outside the lag-lead window.
grid_col	Color of grid lines.

See Also

get_laglead

Examples

```
## Example 1: supply t, tz, ll_fun directly
 gg_laglead(1:10, tz=-5:5,
  ll_fun=function(t, tz) { t >= tz + 2 & t <= tz + 2 + 3})

## Example 2: extract information on t, tz, ll_from data with respective attributes
data("simdf_elra", package = "pammtools")
gg_laglead(simdf_elra)
```

`gg_partial` *Visualize effect estimates for specific covariate combinations*

Description

Depending on the plot function and input, creates either a 1-dimensional slices, bivariate surface or (1D) cumulative effect.

Usage

```
gg_partial(data, model, term, ..., reference = NULL, ci = TRUE)

gg_partial_ll(
  data,
  model,
  term,
  ...,
  reference = NULL,
  ci = FALSE,
  time_var = "tend"
)

get_partial_ll(
  data,
  model,
  term,
  ...,
  reference = NULL,
  ci = FALSE,
  time_var = "tend"
)
```

Arguments

`data`	Data used to fit the `model`.
`model`	A suitable model object which will be used to estimate the partial effect of `term`.
`term`	A character string indicating the model term for which partial effects should be plotted.

...	Covariate specifications (expressions) that will be evaluated by looking for variables in x. Must be of the form z = f(z) where z is a variable in the data set and f a known function that can be usefully applied to z. Note that this is also necessary for single value specifications (e.g. age = c(50)). For data in PED (piece-wise exponential data) format, one can also specify the time argument, but see "Details" an "Examples" below.
reference	If specified, should be a list with covariate value pairs, e.g. list(x1 = 1, x2=50). The calculated partial effect will be relative to an observation specified in reference.
ci	Logical. Indicates if confidence intervals for the term of interest should be calculated/plotted. Defaults to TRUE.
time_var	The name of the variable that was used in model to represent follow-up time.

---

gg_re *Plot Normal QQ plots for random effects*

Description

Plot Normal QQ plots for random effects

Usage

```
gg_re(x, ...)
```

Arguments

x	a fitted gam object as produced by gam().
...	Further arguments passed to plot.gam

See Also

tidy_re

Examples

```
library(pammtools)
data("patient")
ped <- patient %>%
 dplyr::slice(1:100) %>%
 as_ped(Surv(Survdays, PatientDied)~ ApacheIIScore + CombinedicuID, id="CombinedID")
pam <- mgcv::gam(ped_status ~ s(tend) + ApacheIIScore + s(CombinedicuID, bs="re"),
 data=ped, family=poisson(), offset=offset)
gg_re(pam)
plot(pam, select = 2)
```

gg_slice *Plot 1D (smooth) effects*

Description

Flexible, high-level plotting function for (non-linear) effects conditional on further covariate specifications and potentially relative to a comparison specification.

Usage

```
gg_slice(data, model, term, ..., reference = NULL, ci = TRUE)
```

Arguments

data	Data used to fit the model.
model	A suitable model object which will be used to estimate the partial effect of term.
term	A character string indicating the model term for which partial effects should be plotted.
...	Covariate specifications (expressions) that will be evaluated by looking for variables in x. Must be of the form z = f(z) where z is a variable in the data set and f a known function that can be usefully applied to z. Note that this is also necessary for single value specifications (e.g. age = c(50)). For data in PED (piece-wise exponential data) format, one can also specify the time argument, but see "Details" an "Examples" below.
reference	If specified, should be a list with covariate value pairs, e.g. list(x1 = 1, x2=50). The calculated partial effect will be relative to an observation specified in reference.
ci	Logical. Indicates if confidence intervals for the term of interest should be calculated/plotted. Defaults to TRUE.

Examples

```
ped <- tumor[1:200, ] %>% as_ped(Surv(days, status) ~ . )
model <- mgcv::gam(ped_status~s(tend) + s(age, by = complications), data=ped,
  family = poisson(), offset=offset)
make_newdata(ped, age = seq_range(age, 20), complications = levels(complications))
gg_slice(ped, model, "age", age=seq_range(age, 20), complications=levels(complications))
gg_slice(ped, model, "age", age=seq_range(age, 20), complications=levels(complications),
 ci = FALSE)
gg_slice(ped, model, "age", age=seq_range(age, 20), complications=levels(complications),
  reference=list(age = 50))
```

gg_smooth *Plot smooth 1d terms of gam objects*

Description

Given a gam model this convenience function returns a plot of all smooth terms contained in the model. If more than one smooth is present, the different smooth are faceted.

Usage

```
gg_smooth(x, ...)

## Default S3 method:
gg_smooth(x, fit, ...)
```

Arguments

x	A data frame or object of class ped.
...	Further arguments passed to get_terms
fit	A model object.

Value

A ggplot object.

See Also

get_terms

Examples

```
g1 <- mgcv::gam(Sepal.Length ~ s(Sepal.Width) + s(Petal.Length), data=iris)
gg_smooth(iris, g1, terms=c("Sepal.Width", "Petal.Length"))
```

gg_tensor *Plot tensor product effects*

Description

Given a gam model this convenience function returns a ggplot2 object depicting 2d smooth terms specified in the model as heat/contour plots. If more than one 2d smooth term is present individual terms are faceted.

Usage

```
gg_tensor(x, ci = FALSE, ...)
```

Arguments

x	a fitted gam object as produced by gam().
ci	A logical value indicating whether confidence intervals should be calculated and returned. Defaults to TRUE.
...	Further arguments passed to plot.gam

See Also

tidy_smooth2d

Examples

```
g <- mgcv::gam(Sepal.Length ~ te(Sepal.Width, Petal.Length), data=iris)
gg_tensor(g)
gg_tensor(g, ci=TRUE)
gg_tensor(update(g, .~. + te(Petal.Width, Petal.Length)))
```

---

make_newdata *Construct a data frame suitable for prediction*

Description

This functions provides a flexible interface to create a data set that can be plugged in as newdata argument to a suitable predict function (or similar). The function is particularly useful in combination with one of the add_* functions, e.g., add_term, add_hazard, etc.

Usage

```
make_newdata(x, ...)

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

## S3 method for class 'ped'
make_newdata(x, ...)

## S3 method for class 'fped'
make_newdata(x, ...)
```

Arguments

x	A data frame (or object that inherits from data.frame).
...	Covariate specifications (expressions) that will be evaluated by looking for variables in x. Must be of the form z = f(z) where z is a variable in the data set and f a known function that can be usefully applied to z. Note that this is also necessary for single value specifications (e.g. age = c(50)). For data in PED (piece-wise exponential data) format, one can also specify the time argument, but see "Details" an "Examples" below.

Details

Depending on the type of variables in x, mean or modus values will be used for variables not specified in ellipsis (see also sample_info). If x is an object that inherits from class ped, useful data set completion will be attempted depending on variables specified in ellipsis. This is especially useful, when creating a data set with different time points, e.g. to calculate survival probabilities over time (add_surv_prob) or to calculate a time-varying covariate effects (add_term). To do so, the time variable has to be specified in ..., e.g., tend = seq_range(tend, 20). The problem with this specification is that not all values produced by seq_range(tend, 20) will be actual values of tend used at the stage of estimation (and in general, it will often be tedious to specify exact tend values). make_newdata therefore finds the correct interval and sets tend to the respective interval endpoint. For example, if the intervals of the PED object are $(0, 1], (1, 2]$ then tend = 1.5 will be set to 2 and the remaining time-varying information (e.g. offset) completed accordingly. See examples below.

Examples

```
# General functionality
tumor %>% make_newdata()
tumor %>% make_newdata(age=c(50))
tumor %>% make_newdata(days=seq_range(days, 3), age=c(50, 55))
tumor %>% make_newdata(days=seq_range(days, 3), status=unique(status), age=c(50, 55))
# mean/modus values of unspecified variables are calculated over whole data
tumor %>% make_newdata(sex=unique(sex))
tumor %>% group_by(sex) %>% make_newdata()

# Examples for PED data
ped <- tumor %>% slice(1:3) %>% as_ped(Surv(days, status)~., cut = c(0, 500, 1000))
ped %>% make_newdata(age=c(50, 55))

# if time information is specified, other time variables will be specified
# accordingly and offset calculated correctly
ped %>% make_newdata(tend = c(1000), age = c(50, 55))
ped %>% make_newdata(tend = unique(tend))
ped %>% group_by(sex) %>% make_newdata(tend = unique(tend))

# tend is set to the end point of respective interval:
ped <- tumor %>% as_ped(Surv(days, status)~.)
seq_range(ped$tend, 3)
make_newdata(ped, tend = seq_range(tend, 3))
```

---

patient *Survival data of critically ill ICU patients*

Description

A data set containing the survival time (or hospital release time) among other covariates. The full data is available here. The following variables are provided:

Year The year of ICU Admission

CombinedicuID Intensive Care Unit (ICU) ID

CombinedID Patient identificator

Survdays Survival time of patients. Here it is assumed that patients survive until t=30 if released from hospital.

PatientDied Status indicator; 1=death, 0=censoring

survhosp Survival time in hospital. Here it is assumed that patients are censored at time of hospital release (potentially informative)

Gender Male or female

Age The patients age at Admission

AdmCatID Admission category: medical, surgical elective or surgical emergency

ApacheIIScore The patient's Apache II Score at Admission

BMI Patient's Body Mass Index

DiagID2 Diagnosis at admission in 9 categories

Usage

```
patient
```

Format

An object of class `data.frame` with 2000 rows and 12 columns.

`ped_info` *Extract interval information and median/modus values for covariates*

Description

Given an object of class ped, returns data frame with one row for each interval containing interval information, mean values for numerical variables and modus for non-numeric variables in the data set.

Usage

```
ped_info(ped)

## S3 method for class 'ped'
ped_info(ped)
```

Arguments

ped	An object of class ped as returned by as_ped.

Value

A data frame with one row for each unique interval in ped.

See Also

int_info, sample_info

Examples

```
ped <- tumor[1:4,] %>% as_ped(Surv(days, status)~ sex + age)
ped_info(ped)
```

---

predictSurvProb.pamm *S3 method for pamm objects for compatibility with package pec*

Description

S3 method for pamm objects for compatibility with package pec

Usage

```
## S3 method for class 'pamm'
predictSurvProb(object, newdata, times, ...)
```

Arguments

object	A fitted model from which to extract predicted survival probabilities
newdata	A data frame containing predictor variable combinations for which to compute predicted survival probabilities.
times	A vector of times in the range of the response variable, e.g. times when the response is a survival object, at which to return the survival probabilities.
...	Additional arguments that are passed on to the current method.

---

seq_range *Generate a sequence over the range of a vector*

Description

Stolen from here

Usage

```
seq_range(x, n, by, trim = NULL, expand = NULL, pretty = FALSE)
```

Arguments

x	A numeric vector
n, by	Specify the output sequence either by supplying the length of the sequence with n, or the spacing between value with by. Specifying both is an error. I recommend that you name these arguments in order to make it clear to the reader.
trim	Optionally, trim values off the tails. `trim / 2 * length(x)` values are removed from each tail.
expand	Optionally, expand the range by `expand * (1 + range(x)` (computed after trimming).
pretty	If TRUE, will generate a pretty sequence. If n is supplied, this will use `pretty()` instead of `seq()`. If by is supplied, it will round the first value to a multiple of by.

Examples

```
x <- rcauchy(100)
seq_range(x, n = 10)
seq_range(x, n = 10, trim = 0.1)
seq_range(x, by = 1, trim = 0.1)

# Make pretty sequences
y <- runif (100)
seq_range(y, n = 10)
seq_range(y, n = 10, pretty = TRUE)
seq_range(y, n = 10, expand = 0.5, pretty = TRUE)

seq_range(y, by = 0.1)
seq_range(y, by = 0.1, pretty = TRUE)
```

---

simdf_elra *Simulated data with cumulative effects*

Description

This is data simulated using the `sim_pexp` function. It contains two time-constant and two time-dependent covariates (observed on different exposure time grids). The code used for simulation is contained in the examples of `?sim_pexp`.

Usage

```
simdf_elra
```

Format

An object of class `nested_fdf` (inherits from `sim_df`, `tbl_df`, `tbl`, `data.frame`) with 250 rows and 9 columns.

sim_pexp *Simulate survival times from the piece-wise exponential distribution*

Description

Simulate survival times from the piece-wise exponential distribution

Usage

```
sim_pexp(formula, data, cut)
```

Arguments

formula	An extended formula that specifies the linear predictor. If you want to include a smooth baseline or time-varying effects, use `t` within your formula as if it was a covariate in the data, although it is not and should not be included in the `data` provided to `sim_pexp`. See examples below.
data	A data set with variables specified in `formula`.
cut	A sequence of time-points starting with 0.

Examples

```
library(survival)
library(dplyr)
library(pammtools)

# set number of observations/subjects
n <- 250
# create data set with variables which will affect the hazard rate.
df <- cbind.data.frame(x1 = runif (n, -3, 3), x2 = runif (n, 0, 6)) %>%
 as_tibble()
# the formula which specifies how covariates affet the hazard rate
f0 <- function(t) {
 dgamma(t, 8, 2) *6
}
form <- ~ -3.5 + f0(t) -0.5*x1 + sqrt(x2)
set.seed(24032018)
sim_df <- sim_pexp(form, df, 1:10)
head(sim_df)
plot(survfit(Surv(time, status)~1, data = sim_df ))

# for control, estimate with Cox PH
mod <- coxph(Surv(time, status) ~ x1 + pspline(x2), data=sim_df)
coef(mod)[1]
layout(matrix(1:2, nrow=1))
termplot(mod, se = TRUE)

# and using PAMs
layout(1)
```

```
ped <- sim_df %>% as_ped(Surv(time, status)~., max_time=10)
library(mgcv)
pam <- gam(ped_status ~ s(tend) + x1 + s(x2), data=ped, family=poisson, offset=offset)
coef(pam)[2]
plot(pam, page=1)

## Not run:
# Example 2: Functional covariates/cumulative coefficients
# function to generate one exposure profile, tz is a vector of time points
# at which TDC z was observed
rng_z = function(nz) {
  as.numeric(arima.sim(n = nz, list(ar = c(.8, -.6))))
}
# two different exposure times  for two different exposures
tz1 <- 1:10
tz2 <- -5:5
# generate exposures and add to data set
df <- df %>%
  add_tdc(tz1, rng_z) %>%
  add_tdc(tz2, rng_z)
df

# define tri-variate function of time, exposure time and exposure z
ft <- function(t, tmax) {
  -1*cos(t/tmax*pi)
}
fdnorm <- function(x) (dnorm(x,1.5,2)+1.5*dnorm(x,7.5,1))
wpeak2 <- function(lag) 15*dnorm(lag,8,10)
wdnorm <- function(lag) 5*(dnorm(lag,4,6)+dnorm(lag,25,4))
f_xyz1 <- function(t, tz, z) {
  ft(t, tmax=10) * 0.8*fdnorm(z)* wpeak2(t - tz)
}
f_xyz2 <- function(t, tz, z) {
  wdnorm(t-tz) * z
}

# define lag-lead window function
ll_fun <- function(t, tz) {t >= tz}
ll_fun2 <- function(t, tz) {t - 2 >= tz}
# simulate data with cumulative effect
sim_df <- sim_pexp(
  formula = ~ -3.5 + f0(t) -0.5*x1 + sqrt(x2)|
     fcumu(t, tz1, z.tz1, f_xyz=f_xyz1, ll_fun=ll_fun) +
     fcumu(t, tz2, z.tz2, f_xyz=f_xyz2, ll_fun=ll_fun2),
  data = df,
  cut = 0:10)

## End(Not run)
```

---

staph *Time until staphylococcus aureaus infection in children, with possible recurrence*

Description

This dataset originates from the Drakenstein child health study. The data contains the following variables:

id Randomly generated unique child ID

t.start The time at which the child enters the risk set for the k-th event

t.stop Time of k-th infection or censoring.

enum Event number. Maximum of 6.

hiv

Usage

```
staph
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 374 rows and 6 columns.

`tidy_re` *Extract random effects in tidy data format.*

Description

Extract random effects in tidy data format.

Usage

```
tidy_re(x, keep = c("fit", "main", "xlab", "ylab"), ...)
```

Arguments

`x`	a fitted `gam` object as produced by `gam()`.
`keep`	A vector of variables to keep.
`...`	Further arguments passed to `plot.gam`

See Also

`qqline`

tidy_smooth *Extract 1d smooth objects in tidy data format.*

Description

Extract 1d smooth objects in tidy data format.

Usage

```
tidy_smooth(x, keep = c("x", "fit", "se", "xlab", "ylab"), ci = TRUE, ...)
```

Arguments

x	a fitted gam object as produced by gam().
keep	A vector of variables to keep.
ci	A logical value indicating whether confidence intervals should be calculated and returned. Defaults to TRUE.
...	Further arguments passed to plot.gam

tidy_smooth2d *Extract 2d smooth objects in tidy format.*

Description

Extract 2d smooth objects in tidy format.

Usage

```
tidy_smooth2d(
  x,
  keep = c("x", "y", "fit", "se", "xlab", "ylab", "main"),
  ci = FALSE,
  ...
)
```

Arguments

x	a fitted gam object as produced by gam().
keep	A vector of variables to keep.
ci	A logical value indicating whether confidence intervals should be calculated and returned. Defaults to TRUE.
...	Further arguments passed to plot.gam

`tumor` *Stomach area tumor data*

Description

Information on patients treated for a cancer disease located in the stomach area. The data set includes:

days Time from operation until death in days.

status Event indicator (0 = censored, 1 = death).

age The subject's age.

sex The subject's sex (male/female).

charlson_score Charlson comorbidity score, 1-6.

transfusion Has subject received transfusions (no/yes).

complications Did major complications occur during operation (no/yes).

metastases Did the tumor develop metastases? (no/yes).

resection Was the operation accompanied by a major resection (no/yes).

Usage

```
tumor
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 776 rows and 9 columns.

Index