Package 'pammtools'

August 25, 2026

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

Version 0.8.1

Date 2026-08-13

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, etm, xgboost

Config/Needs/website carData, coxme, eha, etm, scam, msm, mvna, rjags, brms, xgboost, TBFmultinomial

License MIT + file LICENSE

LazyData true

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

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

Encoding UTF-8

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.2

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 2026-08-25 11:20:09 UTC

Contents

add_cif . . . 3
add_counterfactual_transitions . . . 5
add_hazard . . . 6
add_inspections . . . 9
add_surv_prob . . . 10
add_tdc . . . 12
add_term . . . 13
add_trans_prob . . . 14
as.data.frame.crps . . . 16
daily . . . 16
geom_hazard . . . 17
geom_stepribbon . . . 20
get_cumu_coef . . . 22
get_cumu_eff . . . 23
get_intervals . . . 24
get_laglead . . . 25
get_plotinfo . . . 26
get_terms . . . 26
gg_fixed . . . 27
gg_laglead . . . 28
gg_partial . . . 29
gg_re . . . 30
gg_slice . . . 31
gg_smooth . . . 32
gg_state_occupation . . . 33
gg_tensor . . . 34
make_newdata . . . 34
nuclear . . . 36
pamm_ic_cr . . . 36
patient . . . 38
ped_info . . . 39
predictSurvProb.pamm . . . 39
print.pamm_ic . . . 40

seq_range 42
simdf_elra 43
sim_pexp 43
staph 45
tidy_re 46
tidy_smooth 46
tidy_smooth2d 47
tumor 48

Index **49**

`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,
  interval_length = "intlen",
  check_grouping = TRUE,
  ...
)

## S3 method for class 'pamm_ic'
add_cif(
  newdata,
  object,
  ci = TRUE,
  alpha = 0.05,
  nsim = 500L,
  cause_var = "cause",
  time_var = NULL,
  interval_length = "intlen",
  check_grouping = TRUE,
```

```
  ...
)
```

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	Significance level for pooled confidence intervals.
nsim	Total number of pooled posterior draws used for the interval.
cause_var	Character. Column name of the 'cause' variable.
time_var	Name of the variable used for the baseline hazard. Defaults to "tend".
interval_length	Character, defaults to "intlen". contains the interval length in newdata.
check_grouping	Logical. If TRUE (default), stop if newdata is not grouped so that the time variable is unique within each group, guarding against silently accumulating the cumulative incidence across distinct covariate profiles or causes. Note that check_grouping = FALSE only skips this profile-level safeguard; the independent check that newdata is grouped by cause (inside get_cif()) still applies.

Details

When computing cumulative incidence for multiple groups, the input data must be grouped via group_by() (by cause and any covariates) before calling this function. If newdata still contains several profiles per group (repeated time_var values within a group, typically a forgotten group_by()), the function now **stops with an error** rather than returning silently incorrect results, as the cumulative incidence would otherwise be accumulated across profiles rather than within each group.

The returned data contains one boundary row per group at time_var = 0 for plotting cumulative incidence from the time origin. On this row, cif = 0; if confidence intervals are requested, cif_lower = cif_upper = 0. If an interval-length column is present, it is set to 0 on the boundary row. add_cumu_hazard() adds an analogous boundary row (with cumu_hazard = 0) for continuous-time models (GAM/SCAM/PAMM), controllable via its boundary argument; interval-factor models (e.g. PEM via glm) keep the original prediction grid without a boundary row.

Examples

```
if (require("etm")) {
  data("fourD", package = "etm")
  ped_stacked <- fourD |>
    dplyr::select(-medication, -treated) |>
    as_ped(Surv(time, status) ~., id = "id") |>
    dplyr::mutate(cause = as.factor(cause))
  pam <- pamm(
    ped_status ~ s(tend, by = cause) + sex + sex:cause + age + age:cause,
    data = ped_stacked)
  ped_stacked |>
    make_newdata(tend = unique(tend), cause = unique(cause)) |>
    group_by(cause) |>
    add_cif(pam)
}
```

`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. In addition to models fit with `gam`/`bam` or `glm`, shape-constrained additive models fit with `scam` are supported (e.g., for monotone baseline hazards). For `scam` models all calculations (including delta-method and simulation based confidence intervals) are based on the re-parametrized coefficients and their covariance matrix, i.e., on the same normal approximation that underlies the standard errors reported by `scam` itself.

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,
  nsim = 100L,
  alpha = 0.05,
  ...
)

add_cumu_hazard(newdata, object, ...)

## Default S3 method:
add_cumu_hazard(
  newdata,
  object,
  ci = TRUE,
  se_mult = 2,
  overwrite = FALSE,
  time_var = NULL,
  interval_length = "intlen",
  boundary = TRUE,
  check_grouping = TRUE,
```

```
  ...
)

## S3 method for class 'pamm_ic'
add_hazard(
  newdata,
  object,
  ci = TRUE,
  alpha = 0.05,
  nsim = 500L,
  time_var = NULL,
  ...
)

## S3 method for class 'pamm_ic'
add_cumu_hazard(
  newdata,
  object,
  ci = TRUE,
  alpha = 0.05,
  nsim = 500L,
  time_var = NULL,
  interval_length = "intlen",
  check_grouping = TRUE,
  ...
)
```

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. For `ci_type = "sim"`, interval bounds are empirical quantiles (type 6, see `quantile`) of `nsim` posterior draws (default `nsim = 100L`, passed via `...`). Type-6 quantiles avoid the systematic inward bias that the `quantile` default (type 7) exhibits for small `nsim`, but at the default `nsim = 100` the bounds are estimated from few tail draws and thus noisy; increase `nsim` (e.g., to 500 or more) for more stable interval bounds. Very small `nsim` (`nsim < 2 / alpha - 1`, i.e., below 39 for `alpha = 0.05`) cannot achieve the nominal level at all.

`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. Defaults to `"tend"`.

`nsim` Total number of pooled posterior draws used for the interval.

`alpha` Significance level for pooled confidence intervals (a $(1 - \alpha)$ interval).

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

`boundary` Logical. If `TRUE` (default), a boundary row at `time = 0` with cumulative hazard `0` is prepended (per group), so that cumulative hazards start at the natural origin (consistent with `add_surv_prob`, `add_cif` and `add_trans_prob`).

`check_grouping` Logical. If `TRUE` (default), the function checks that `newdata` is grouped so that the time variable is unique within each group and `stop`s otherwise, guarding against silently accumulating cumulative quantities across distinct covariate profiles (a forgotten `group_by()`). Set to `FALSE` to skip the check (e.g.\ for internal calls on already-validated data).

Details

When computing cumulative hazards or survival probabilities across groups, the input data must be grouped via `group_by()` prior to calling `add_cumu_hazard()` or `add_surv_prob()`, so that the cumulative quantity is accumulated within each covariate profile rather than across the whole dataset. If `newdata` still contains several profiles per group (i.e.\ repeated `time_var` values within a group, typically a forgotten `group_by()`), the functions now **stop with an error** rather than returning silently incorrect results. Set `check_grouping = FALSE` to skip this safeguard. Note the check detects the *common* mis-grouping cases – in particular any grid built with `make_newdata`, where profiles share time values – but cannot catch hand-built grids that stack profiles with disjoint time grids, as these are indistinguishable from a single profile with time-varying covariates. See the workflow vignette for a worked example.

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_inspections` *Turn exact event times into interval-censored observations*

Description

Convenience helper to manufacture interval-censored (panel) data from exact simulated survival times (e.g.\ the output of `sim_pexp`), for coverage studies and examples. Each subject is "inspected" at a sequence of times; the true event time is then only known to lie between the last clean and the first positive inspection. The exact time is retained (by default in column `true_time`) so that coverage can be scored against the truth.

Usage

```
add_inspections(
  data,
  time_var = "time",
  status_var = "status",
  mechanism = c("random", "fixed", "mixed"),
  rate = 1,
  schedule = NULL,
  max_time = NULL,
  terminal_exam = TRUE,
  keep_truth = TRUE,
  L = "L",
  R = "R"
)
```

Arguments

`data`	A data frame with one row per subject containing an exact event time and a status indicator (as produced by `sim_pexp`).
`time_var`, `status_var`	Names of the (exact) event-time and status columns. `status_var` may be missing, in which case all rows are treated as events.
`mechanism`	Inspection mechanism: `"random"` (default) draws inter-inspection gaps from an `Exp(rate)` distribution; `"fixed"` uses the common grid given in `schedule`; `"mixed"` jitters the fixed grid by a random offset per subject.
`rate`	Inspection rate for `mechanism = "random"` / `"mixed"` (expected gap $1/\mathrm{rate}$).
`schedule`	Numeric vector of inspection times for `mechanism = "fixed"`/`"mixed"`.
`max_time`	Inspection horizon. Defaults to `max(data[[time_var]])`.

terminal_exam	Logical; if TRUE (default), every subject is additionally examined at max_time (an end-of-study examination), so events before max_time always have a finite upper bound and only subjects event-free at max_time are right-censored. If FALSE, there is no closing examination: events after a subject's last inspection are right-censored at that inspection, and *subjects that exit event-free* (status == 0) are likewise right-censored at their last inspection before exit (not at their exact exit time). Both conventions yield coarsening-at-random data; mixing them (exact exit times for survivors but open intervals for undetected events) would make the right-censoring informative and bias every interval-censoring likelihood.
keep_truth	Logical; keep the exact event time in true_time.
L, R	Names of the created lower/upper bound columns.

Value

data augmented with interval bounds in columns L and R (and true_time). Use Surv(L, R, type = "interval2") on the result.

See Also

pamm_ic, sim_pexp

Examples

```
set.seed(1)
df <- data.frame(x = runif(100, -1, 1))
sdf <- sim_pexp(~ -2 + 0.4 * x, df, cut = seq(0, 10, by = 0.5))
icd <- add_inspections(sdf, rate = 1)
fit <- pamm_ic(Surv(L, R, type = "interval2") ~ x, icd, m = 5)
```

---

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, ...)

## Default S3 method:
add_surv_prob(
  newdata,
```

```
  object,
  ci = TRUE,
  se_mult = 2,
  overwrite = FALSE,
  time_var = NULL,
  interval_length = "intlen",
  boundary = TRUE,
  check_grouping = TRUE,
  ...
)

## S3 method for class 'pamm_ic'
add_surv_prob(
  newdata,
  object,
  ci = TRUE,
  alpha = 0.05,
  nsim = 500L,
  time_var = NULL,
  interval_length = "intlen",
  check_grouping = TRUE,
  ...
)
```

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.
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. Defaults to "tend".
interval_length	The variable in newdata containing the interval lengths. Can be either bare unquoted variable name or character. Defaults to "intlen".
boundary	Logical. If TRUE (default), a boundary row at time = 0 with cumulative hazard 0 is prepended (per group), so that cumulative hazards start at the natural origin (consistent with add_surv_prob, add_cif and add_trans_prob).

check_grouping	Logical. If TRUE (default), the function checks that newdata is grouped so that the time variable is unique within each group and stops otherwise, guarding against silently accumulating cumulative quantities across distinct covariate profiles (a forgotten group_by()). Set to FALSE to skip the check (e.g.\ for internal calls on already-validated data).
alpha	Significance level for pooled confidence intervals.
nsim	Total number of pooled posterior draws used for the interval.

Details

When computing cumulative hazards or survival probabilities across groups, the input data must be grouped via group_by() prior to calling add_cumu_hazard() or add_surv_prob(), so that the cumulative quantity is accumulated within each covariate profile rather than across the whole dataset. If newdata still contains several profiles per group (i.e.\ repeated time_var values within a group, typically a forgotten group_by()), the functions now **stop with an error** rather than returning silently incorrect results. Set check_grouping = FALSE to skip this safeguard. Note the check detects the *common* mis-grouping cases – in particular any grid built with make_newdata, where profiles share time values – but cannot catch hand-built grids that stack profiles with disjoint time grids, as these are indistinguishable from a single profile with time-varying covariates. See the workflow vignette for a worked example.

The returned data contains one boundary row per group at time_var = 0 for plotting cumulative quantities from the time origin. On this row, surv_prob = 1; if confidence intervals are requested, surv_lower = surv_upper = 1. If an interval-length column is present, it is set to 0 on the boundary row.

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. Shape-constrained additive models fit with scam are supported as well.

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 = "tend",
  interval_length = "intlen",
  transition = "transition",
  check_grouping = TRUE,
  ...
)
```

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. Interval bounds are empirical type-6 quantiles of the nsim draws; larger values of nsim yield more stable interval bounds.
time_var	Name of the variable used for the baseline hazard. Defaults to "tend".
interval_length	Character, defaults to "intlen". contains the interval length in newdata.
transition	Character, defaults to "transition". contains the transition labels in newdata.
check_grouping	Logical. If TRUE (default), the function checks that newdata is grouped so that the time variable is unique within each group once transition is part of the grouping, and stops otherwise, guarding against silently accumulating transition probabilities across distinct covariate profiles (a forgotten group_by()). Set to FALSE to skip the check. As for the other cumulative add_* functions, hand-built grids stacking profiles with disjoint time grids cannot be detected (see add_cumu_hazard).
...	Further arguments passed to underlying methods.

Details

When computing transition probabilities for multiple groups, the input data must be grouped via group_by() before calling this function. If newdata still contains several covariate profiles per (group, transition) – i.e.\ repeated time_var values within a group once transition is added to the grouping, typically a forgotten group_by() – the function now **stops with an error** rather than returning silently incorrect results, as the transition probability would otherwise be accumulated across profiles rather than within each group.

The returned data contains one boundary row per group and transition at time_var = 0 for plotting transition probabilities from the time origin. On this row, trans_prob = 0; if confidence intervals are requested, trans_lower = trans_upper = 0. If an interval-length column is present, it is set to 0 on the boundary row.

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!
  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,
  time_var = "tend",
  interval_length = "intlen",
  ...
```

```
)

## S3 method for class 'scam'
get_cumu_coef(
  model,
  data,
  terms,
  time_var = "tend",
  interval_length = "intlen",
  ...
)

## 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.
time_var	Name of the evaluation time variable in data. Defaults to "tend".
interval_length	Name of the interval-length variable in data. Defaults to "intlen".
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 (or a scam object), returns the information used for the default plots produced by plot.gam (plot.scam, respectively).

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 univariate smooth model terms*

Description

Creates, for each requested univariate smooth, a sequence over the range of the smooth's numeric covariate, evaluates the term-wise contribution via predict(fit, newdata = ., type = "terms") and stacks the results into a tidy data frame.

Usage

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

Arguments

data	A data frame containing variables used to fit the model. The first row is used as the basis for all covariates other than the one being varied (their values are irrelevant for the term-wise contribution).
fit	A fitted object of class gam.
terms	A character vector (can be length one) specifying the terms for which partial effects will be returned. If NULL (the default) all univariate smooth terms in the model are used.
...	Further arguments controlling extraction, passed on per term, e.g. n (number of evaluation points) and conf_level.

Details

For gam fits the requested terms are matched against the model's smooths (see get_smooth_terms): a bare variable name (e.g. "tend") selects *every* univariate smooth over that variable – the main effect s(tend) as well as any s(tend, by = ...) or factor-smooth interaction – while an exact smooth label (e.g. "s(tend)") selects a single smooth. Names that do not match any smooth (for example parametric factor main effects) are skipped with a warning; use gg_fixed for those. For factor-indexed smooths one curve per factor level is returned, identified by the level column.

For models without mgcv smooth metadata (e.g. coxph) terms must be supplied and is matched against the columns of predict(type = "terms").

Value

A tibble with columns term, x, level, eff, se, ci_lower and ci_upper.

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 its univariate smooth terms. If `terms` is not specified, all univariate smooths are plotted; otherwise only the requested ones (see `get_terms` for how terms are matched). Different smooths are faceted. Smooths that are indexed by a factor – a factor by-variable or a factor-smooth interaction (`bs = "fs"/"sz"`) – are drawn in a single facet with one coloured/filled curve per factor level.

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` (e.g. `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"))
# all univariate smooths (terms omitted)
gg_smooth(iris, g1)
# factor-by smooth: one coloured curve per Species
g2 <- mgcv::gam(Sepal.Length ~ s(Sepal.Width, by = Species), data = iris)
gg_smooth(iris, g2, terms = "Sepal.Width")
```

gg_state_occupation *Plot state occupation probabilities*

Description

Creates a stacked area plot of state occupation probabilities over time, computed from transition probability matrices stored as an attribute of the input data. Optionally facets by a grouping variable.

Usage

```
gg_state_occupation(
  newdata,
  init_state,
  group_var = NULL,
  time_var = "tend",
  ncol = NULL
)
```

Arguments

newdata	A data frame with an attribute matrix containing a data frame with a column trans_prob_matrix. Each element of trans_prob_matrix should be a 3-dimensional array of dimensions n_states x n_states x n_timepoints.
init_state	A numeric vector specifying the initial state distribution. Should sum to 1 and have length equal to the number of states. For example, c(0, 1, 0, 0) places all subjects in state 2 at baseline.
group_var	A character string giving the name of the column in newdata to facet by (e.g., "treat"). If NULL (default), no faceting is applied.
time_var	A character string giving the name of the time variable in newdata. Defaults to "tend".
ncol	An integer specifying the number of columns in the facet wrap. If NULL (default), defaults to the number of unique groups.

Value

A ggplot object showing stacked-area state occupation probabilities over time, optionally faceted by group_var.

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.

The returned data frame contains `tend`, `id`, the user-supplied covariates (and `cause`/`transition` for competing risks / multi-state models). Internal PED columns `tstart`, `intlen`, `interval`, `offset`, and `ped_status` are dropped. Downstream `add_*` functions reconstruct `intlen` on demand via `reconstruct_intlen()` when needed. 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))
```

`nuclear` *Time until nuclear power plant construction in different regions.*

Description

This dataset originates from IAEA and contains 730 power. The data contains the following variables:

months Construction time

status Event indicator (0 = censored, 1 = construction finished).

region Continent, Africa/Asia, America, Europe, Soviet Union and Warsaw Pact

Usage

```
nuclear
```

Format

An object of class `data.frame` with 724 rows and 3 columns.

`pamm_ic_cr` *Fit a competing-risks PAMM to interval-censored data via multiple imputation*

Description

Competing-risks extension of `pamm_ic`. The event time is drawn from the all-cause conditional hazard within $(L, R]$ and a cause is assigned: observed causes are retained (with the time drawn so that it follows the cause-specific conditional density, via rejection), unknown causes are sampled with probability proportional to the cause-specific hazards at the imputed time (see `impute_ic_cr`). Each completed data set is transformed with `as_ped_cr` (cause-specific hazards) and re-fit. Cf. Delord & Genin (2016) for MI of interval-censored competing-risks data.

Usage

```
pamm_ic_cr(
  formula,
  data,
  cause,
  model_formula = NULL,
  cut = NULL,
  max_time = NULL,
  m = 10L,
  iter = 1L,
  censor_code = 0L,
  id = "id",
  engine = "gam",
  ...
)
```

Arguments

formula	A two-sided formula whose left-hand side is an interval-censored response `Surv(L, R, type = "interval2")` and whose right-hand side lists the covariates to retain (as in as_ped).
data	A data frame in standard (one row per subject) format.
cause	Name of the column in `data` giving the observed cause for events (any factor/character coding). Rows with the censoring code are treated as right-censored; `NA` marks an event with unknown cause.
model_formula	Optional model formula passed to pamm (e.g.\ `ped_status ~ s(tend) + x`). If `NULL`, a default `ped_status ~ s(tend) + <covariates>` formula is constructed.
cut	Optional fixed vector of interval cut-points shared across all imputations. If `NULL`, the finite interval endpoints are used.
max_time	Optional cap on the cut-points.
m	Number of imputations (default 10).
iter	Number of impute-refit iterations per imputation chain (default `1` = classic one-step MI: all `m` imputations are drawn from the single initialiser fit). For `iter = k > 1`, each chain alternates imputation and re-fitting on its own completed data set `k` times, so later imputations are drawn from fits whose dependence on the midpoint initialiser is progressively attenuated – a sequential ("chained") MI scheme that progressively removes initialiser bias under sparse inspection, at roughly `iter`-fold fitting cost. Simulation evidence (see the package's interval-censoring benchmark): with inspection gaps that are small relative to the time scale, `iter = 1` is unbiased; with wide gaps (mean gap of order 1/3 of the follow-up), early-time survival estimates from `iter = 1` are biased upward and `iter = 3` removes most of that bias (`iter = 5` essentially all of it), with bias shrinking roughly geometrically in `iter`. Caveat: with flexible time-varying effect terms and small samples, iterating can occasionally amplify a weakly identified imputation chain into divergent estimates with very wide intervals (without `mgcv` warnings) – inspect pooled smooth effects for plausibility when iterating such models.

censor_code	Value of cause that encodes censoring (default 0).
id	Name of the subject identifier column.
engine	Estimation engine passed to pamm ("gam" or "bam").
...	Further arguments passed to pamm / mgcv.

Value

An object of class pamm_ic with type = "cr"; fits are cause-specific (stacked ped_cr) pamm objects and cause_levels records the competing causes.

See Also

pamm_ic, add_cif

---

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.

---

print.pamm_ic *Fit a PAMM to interval-censored data via multiple imputation*

Description

Fits a piecewise exponential additive (mixed) model to interval-censored time-to-event data using a multiple-imputation (MI) and re-fit strategy: exact event times are repeatedly drawn from the model-based conditional distribution $p(T \mid L < T \leq R, x, \theta)$ (see impute_ic_times), with θ drawn from the imputation model's asymptotic posterior before each imputation ("proper" MI – this is what makes the pooled intervals calibrated), each completed data set is transformed to PED format with the standard (right-censored) pipeline and re-fit, and the resulting fits are pooled for inference with the existing add_* family (see add_surv_prob and the pamm_ic methods).

Usage

```
## S3 method for class 'pamm_ic'
print(x, ...)

## S3 method for class 'pamm_ic'
summary(object, ...)

## S3 method for class 'summary.pamm_ic'
print(x, ...)

pamm_ic(
  formula,
  data,
  model_formula = NULL,
  cut = NULL,
  max_time = NULL,
  m = 10L,
  iter = 1L,
  init = c("midpoint", "uniform"),
  id = "id",
  engine = "gam",
  ...
)
```

Arguments

x, object	A pamm_ic object.
...	Further arguments passed to pamm / mgcv.
formula	A two-sided formula whose left-hand side is an interval-censored response Surv(L, R, type = "interval2") and whose right-hand side lists the covariates to retain (as in as_ped).
data	A data frame in standard (one row per subject) format.
model_formula	Optional model formula passed to pamm (e.g.\ ped_status ~ s(tend) + x). If NULL, a default ped_status ~ s(tend) + <covariates> formula is constructed.
cut	Optional fixed vector of interval cut-points shared across all imputations. If NULL, the finite interval endpoints are used.
max_time	Optional cap on the cut-points.
m	Number of imputations (default 10).
iter	Number of impute-refit iterations per imputation chain (default 1 = classic one-step MI: all m imputations are drawn from the single initialiser fit). For iter = k > 1, each chain alternates imputation and re-fitting on its own completed data set k times, so later imputations are drawn from fits whose dependence on the midpoint initialiser is progressively attenuated – a sequential ("chained") MI scheme that progressively removes initialiser bias under sparse inspection, at roughly iter-fold fitting cost. Simulation evidence (see the package's interval-censoring benchmark): with inspection gaps that are small relative to the time scale, iter = 1 is unbiased; with wide gaps (mean gap of order 1/3 of the follow-up), early-time survival estimates from iter = 1 are biased upward and iter = 3 removes most of that bias (iter = 5 essentially all of it), with bias shrinking roughly geometrically in iter. Caveat: with flexible time-varying effect terms and small samples, iterating can occasionally amplify a weakly identified imputation chain into divergent estimates with very wide intervals (without mgcv warnings) – inspect pooled smooth effects for plausibility when iterating such models.
init	Initialiser for the first fit: "midpoint" (default) or "uniform" imputation within each interval.
id	Name of the subject identifier column.
engine	Estimation engine passed to pamm ("gam" or "bam").

Details

An imputed event time is an exact event time, so once imputation has produced it, the entire downstream pipeline (split_data -> pamm -> add_*) is reused unchanged. The interval cut-points are resolved once and shared across all imputations, but mgcv's smooth bases and centering constraints can still differ across completed data sets. Pooled predictions therefore evaluate each fitted imputation model with its own design matrix; object$pooled is a summary container, not a gam-like model for direct predict() or plot() calls.

Value

An object of class `pamm_ic`: a list with

`fits` the `m` imputation fits, each *slimmed* (via `strip_pamm_fit`) to drop per-observation slots so memory does not scale with the number of imputations; they still support `coef`, `vcov` and `predict(type = "lpmatrix")`, which is all the pooled `add_*` methods need.

`pooled` a pooled summary container with Rubin-pooled parametric coefficients and covariance, pooled parametric/smooth tables with median-p values (`$p.table`, `$s.table`), parametric coefficient FMI diagnostics (`$fmi.table`) and smooth-term FMI five-number summaries over the training grid (`$smooth.fmi`).

`init_fit` the (slimmed) initialiser/imputation model.

`unstable_chains` indices of imputation chains flagged as numerically unstable (extreme coefficients or coefficient SEs on the log-hazard scale; also raised as a `warning`). Degenerate chains can arise – silently, without `mgcv` warnings – when iterating flexible time-varying models on small samples.

others the parsed bounds `ic`, the shared `cut`, and metadata.

`print`/`summary` report the pooled summary; `add_*` compute pooled quantities of interest from `fits`.

See Also

`impute_ic_times`, `add_surv_prob`, `strip_pamm_fit`

`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. Covariates enter the (numeric) linear predictor directly, so `factor`/`character` covariates must be encoded explicitly, e.g.\ as an indicator (`trt == "1"`); using a factor in arithmetic (`b * trt`) is an error rather than a silent coercion.
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,
  conf_level = 0.95,
  ...
)
```

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.
conf_level	Numeric scalar in (0, 1). Confidence level used for the returned confidence intervals when ci = TRUE. Defaults to 0.95.
...	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,
  conf_level = 0.95,
  ...
)
```

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.
conf_level	Numeric scalar in (0, 1). Confidence level used for the returned confidence intervals when ci = TRUE. Defaults to 0.95.
...	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

∗ **datasets**
daily, 16
geom_hazard, 17
geom_stepribbon, 20
nuclear, 36
patient, 38
simdf_elra, 43
staph, 45
tumor, 48

add_cif, 3, *8*, *11*, *38*
add_counterfactual_transitions, 5
add_cumu_hazard, *15*
add_cumu_hazard *(*add_hazard*)*, 6
add_hazard, 6, *34*
add_inspections, 9
add_surv_prob, *8*, 10, *11, 12*, *35*, *40*, *42*
add_tdc, 12
add_term, 13, *34, 35*
add_trans_prob, *8*, *11*, 14
aes(), *18*, *20*
annotation_borders(), *19*, *21*
as.data.frame.crps, 16
as_ped, *25*, *37*, *39*, *41*
as_ped_cr, *36*

bam, *6*

coxph, *27*
crps, *16*

daily, 16
data.frame, *16*

findInterval, *24, 25*
fortify(), *18*, *20*

gam, *6*, *26, 27*
gamObject, *26*
geom_hazard, 17
geom_line, *19*
geom_ribbon, *22*
geom_step, *19*
geom_stephazard *(*geom_hazard*)*, 17
geom_stepribbon, 20
geom_surv *(*geom_hazard*)*, 17
GeomHazard *(*geom_hazard*)*, 17
GeomStepHazard *(*geom_hazard*)*, 17
GeomStepribbon *(*geom_stepribbon*)*, 20
GeomSurv *(*geom_hazard*)*, 17
get_cumu_coef, 22
get_cumu_eff, 23
get_hazard, *4*, *7*, *11*
get_intervals, 24
get_laglead, 25
get_partial_ll *(*gg_partial*)*, 29
get_plotinfo, 26
get_smooth_terms, *27*
get_terms, 26, *32*
gg_cumu_eff *(*get_cumu_eff*)*, 23
gg_fixed, *27*, 27
gg_laglead, 28
gg_partial, 29
gg_partial_ll *(*gg_partial*)*, 29
gg_re, 30
gg_slice, 31
gg_smooth, 32
gg_state_occupation, 33
gg_tensor, 34
ggplot, *32*
ggplot(), *18*, *20*
glm, *6*

impute_ic_cr, *36*
impute_ic_times, *40*, *42*
int_info, *24, 25*, *39*

key glyphs, *19*, *22*

layer position, *18*, *21*
layer stat, *18*, *21*

layer(), *19*, *21*, *22*
linear.functional.terms, *14*

make.names, *16*
make_newdata, *8*, *12*, 34

nuclear, 36

pamm, *37*, *38*, *41*
pamm_ic, *10*, *36*, *38*
pamm_ic *(*print.pamm_ic*)*, 40
pamm_ic_cr, 36
patient, *16*, 38
ped_info, 39
plot.gam, *26*, *30*, *34*, *46*, *47*
plot.scam, *26*
predict.gam, *4*, *6–8*, *11–13*
predictSurvProb.pamm, 39
pretty, *42*
print.pamm_ic, 40
print.summary.pamm_ic *(*print.pamm_ic*)*, 40

qqline, *46*
quantile, *8*

sample_info, *35*, *39*
scam, *6*, *13*, *26*
seq, *42*
seq_range, 42
sim_pexp, *9*, *10*, *12*, *43*, 43
simdf_elra, 43
split_data, *41*
staph, 45
stop, *8*, *12*, *15*
strip_pamm_fit, *42*
summary.pamm_ic *(*print.pamm_ic*)*, 40

tidy_fixed, *27*
tidy_re, *30*, 46
tidy_smooth, 46
tidy_smooth2d, *34*, 47
tumor, 48