Package ‘cifmodeling’

May 8, 2026

Title Visualization and Polytomous Modeling of Survival and Competing Risks

Version 0.9.8

Description A publication-ready toolkit for modern survival and competing risks analysis with a minimal, formula-based interface. Both nonparametric estimation and direct polytomous regression of cumulative incidence functions (CIFs) are supported. The main functions 'cifcurve()', 'cifplot()', and 'cifpanel()' estimate survival and CIF curves and produce high-quality graphics with risk tables, censoring and competing-risk marks, and multi-panel or inset layouts built on 'ggplot2' and 'ggsurvfit'. The modeling function 'polyreg()' performs direct polytomous regression for coherent joint modeling of all cause-specific CIFs to estimate risk ratios, odds ratios, or subdistribution hazard ratios at user-specified time points. All core functions adopt a formula-and-data syntax and return tidy and extensible outputs that integrate smoothly with 'modelsummary', 'broom', and the broader 'tidyverse' ecosystem. Key numerical routines are implemented in C++ via 'Rcpp'.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

RdMacros lifecycle

Config/Needs/website lifecycle

Depends R (>= 4.1.0)

Suggests survival, mets, modelsummary, gtsummary, knitr, rmarkdown, testthat (>= 3.0.0), pkgdown

Config/testthat/edition 3

LinkingTo Rcpp

Imports Rcpp, nleqslv, boot, ggsurvfit, ggplot2, patchwork, scales, generics, lifecycle

VignetteBuilder knitr

URL https://gestimation.github.io/cifmodeling/, https://github.com/gestimation/cifmodeling

BugReports https://github.com/gestimation/cifmodeling/issues

NeedsCompilation yes

Author Shiro Tanaka [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-6817-5235>),
Shigetaka Kobari [ctb],
Chisato Honda [ctb]

Maintainer Shiro Tanaka <gestimation@gmail.com>

Repository CRAN

Date/Publication 2026-01-10 09:20:02 UTC

Contents

cifcurve 2
cifpanel 5
cifplot 15
diabetes.complications 24
Event 25
extract_time_to_event 26
polyreg 28
polyreg-methods 35
prostate 37

Index **39**

`cifcurve` *Calculate the Kaplan-Meier estimator and the Aalen-Johansen estimator*

Description

Core estimation routine that computes a survfit-compatible object from a formula + data interface (`Event()` or `survival::Surv()` on the LHS, and a stratification variable on the RHS if necessary). The back-end C++ routine supports both weighted and stratified data. Use this when you want **numbers only** (e.g. estimates, SEs, CIs and influence functions) and will plot it yourself.

Usage

```
cifcurve(
  formula,
  data,
  weights = NULL,
  n.risk.type = "weighted",
  subset.condition = NULL,
  na.action = na.omit,
  outcome.type = c("survival", "competing-risk"),
  code.event1 = 1,
```

```
  code.event2 = 2,
  code.censoring = 0,
  error = NULL,
  conf.type = "arcsine-square root",
  conf.int = 0.95,
  report.influence.function = FALSE,
  report.survfit.std.err = FALSE,
  engine = "calculateAJ_Rcpp",
  prob.bound = 1e-07
)
```

Arguments

`formula` A model formula specifying the time-to-event outcome on the LHS (typically `Event(time, status)` or `survival::Surv(time, status)`) and, optionally, a stratification variable on the RHS. Unlike `cifplot()`, this function does not accept a fitted survfit object.

`data` A data frame containing variables in the formula.

`weights` Optional name of the weight variable in `data`. Weights must be nonnegative.

`n.risk.type` Character string; one of `"weighted"`, `"unweighted"`, or `"ess"`. Controls which risk set size is returned in `$n.risk` without affecting estimates or standard errors. (default `"weighted"`).

`subset.condition` Optional character string giving a logical condition to subset `data` (default `NULL`).

`na.action` A function specifying the action to take on missing values (default `na.omit`).

`outcome.type` Character string specifying the type of time-to-event outcome. One of `"survival"` (Kaplan-Meier) or `"competing-risk"` (Aalen-Johansen). If `NULL` (default), the function automatically infers the outcome type from the data: if the event variable has more than two unique levels, `"competing-risk"` is assumed; otherwise, `"survival"` is used. You can also use abbreviations such as `"S"` or `"C"`. Mixed or ambiguous inputs (e.g., `c("S", "C")`) trigger automatic detection based on the event coding.

`code.event1` Integer code of the event of interest (default `1`).

`code.event2` Integer code of the competing risk (default `2`).

`code.censoring` Integer code of censoring (default `0`).

`error` Character string specifying the method for SEs and CIs used internally. For `"survival"` without weights, choose one of `"greenwood"` (default), `"tsiatis"`, or `"if"`. For `"competing-risk"` without weights, choose one of `"delta"` (default), `"aalen"`, or `"if"`. SEs and CIs based on influence functions (`"if"`) is recommended for weighted analysis.

`conf.type` Character specifying the method of transformation for CIs used internally (default `arcsine-square root`).

`conf.int` Numeric two-sided level of CIs (default `0.95`).

`report.influence.function` Logical. When `TRUE` and `engine = "calculateAJ_Rcpp"`, the influence function is also computed and returned (default `FALSE`).

`report.survfit.std.err`
: Logical. If `TRUE`, report SE on the log-survival scale (survfit's convention). Otherwise SE is on the probability scale.

`engine`
: Character. One of `"auto"`, `"calculateKM"`, or `"calculateAJ_Rcpp"` (default `"calculateAJ_Rcpp"`).

`prob.bound`
: Numeric lower bound used to internally truncate probabilities away from 0 and 1 (default `1e-7`).

Details

Typical use cases:

- When `outcome.type = "survival"`, this is a thin wrapper around the KM estimator with the chosen variance / CI transformation.
- When `outcome.type = "competing-risk"`, this computes the AJ estimator of CIF for `code.event1`. The returned `$surv` is **1 - CIF**, i.e. in the format that ggsurvfit expects.
- Use `cifplot()` if you want to go straight to a figure; use `cifcurve()` if you only want the numbers.

Risk set display:

- Set `n.risk.type` to control whether `$n.risk` reflects weighted, unweighted, or Kish effective sample size (ESS) counts. This only affects the reported counts (e.g., for plotting or debugging) and leaves estimates and SEs unchanged.

Standard error and confidence intervals:

Argument	Description	Default
`error`	SE for KM: `"greenwood"`, `"tsiatis"`, `"if"`. For CIF: `"aalen"`, `"delta"`, `"if"`.	`"greenwood"`, `"delta`
`conf.type`	Transformation for CIs: `"plain"`, `"log"`, `"log-log"`, `"arcsin"`, `"logit"`, or `"none"`.	`"arcsin"`
`conf.int`	Two-sided CI level.	`0.95`

Value

A `"survfit"` object. For `outcome.type="survival"`, `$surv` is the survival function. For `outcome.type="competing-risk`
`$surv` equals `1 - CIF` for `code.event1`. SE and CIs are provided per `error`, `conf.type` and `conf.int`. This enables an independent use of standard methods for `survfit` such as:

- `summary()`: time-by-time estimates with SEs and CIs
- `plot()`: base R stepwise survival/CIF curves
- `mean()`: restricted mean survival estimates with CIs
- `quantile()`: quantile estimates with CIs

Note that `$n.risk`, `$n.event`, and `$n.censor` are rounded up to the nearest integer regardless of whether the data is weighted or not. Some methods (e.g. `residuals.survfit`) may not be supported.

Lifecycle

[Stable]

See Also

polyreg() for log-odds product modeling of CIFs; cifplot() for display of a CIF; cifpanel() for display of multiple CIFs; ggsurvfit::ggsurvfit, patchwork::patchwork and modelsummary::modelsummary for display helpers.

Examples

```
data(diabetes.complications)
output1 <- cifcurve(Event(t,epsilon) ~ fruitq,
                    data = diabetes.complications,
                    outcome.type="competing-risk")
cifplot(output1,
        outcome.type = "competing-risk",
        type.y = "risk",
        add.risktable = FALSE,
        label.y = "CIF of diabetic retinopathy",
        label.x = "Years from registration")
```

---

cifpanel *Arrange multiple survival and CIF plots in a panel display*

Description

cifpanel() is the panel-building counterpart of cifplot(). It takes one or more model formulas (or, alternatively, one formula and several event-coding specifications) and returns a multi-panel figure, typically as a patchwork-compatible object. Most display options (axis labels, marks, style, ggsave options) are shared with cifplot(), but per-panel legends and number-at-risk tables are suppressed to avoid duplicated display. Typical use cases are:

- Compare CIF (event 1) vs CIF (event 2) in a 1×2 layout.
- Compare survival/CIF curves across strata with a shared legend and matched axes.
- Display a plot with an enlarged y-axis inside a full-scale plot.

Usage

```
cifpanel(
  plots = NULL,
  formula = NULL,
  formulas = NULL,
  data = NULL,
  weights = NULL,
  subset.condition = NULL,
  na.action = na.omit,
  outcome.type = NULL,
  code.event1 = 1,
  code.event2 = 2,
```

```
code.censoring = 0,
code.events = NULL,
error = NULL,
conf.type = NULL,
conf.int = NULL,
n.risk.type = c("weighted", "unweighted", "ess"),
type.y = NULL,
label.x = NULL,
label.y = NULL,
label.strata = NULL,
order.strata = NULL,
level.strata = NULL,
limits.x = NULL,
limits.y = NULL,
breaks.x = NULL,
breaks.y = NULL,
add.conf = NULL,
add.risktable = NULL,
add.estimate.table = NULL,
symbol.risk.table = NULL,
font.size.risk.table = NULL,
add.censor.mark = NULL,
shape.censor.mark = NULL,
size.censor.mark = NULL,
add.competing.risk.mark = NULL,
competing.risk.time = NULL,
shape.competing.risk.mark = NULL,
size.competing.risk.mark = NULL,
add.intercurrent.event.mark = NULL,
intercurrent.event.time = NULL,
shape.intercurrent.event.mark = NULL,
size.intercurrent.event.mark = NULL,
add.quantile = NULL,
level.quantile = NULL,
rows.columns.panel = c(1, 1),
inset.panel = FALSE,
title.panel = NULL,
subtitle.panel = NULL,
caption.panel = NULL,
tag.panel = NULL,
title.plot = NULL,
style = "classic",
palette = NULL,
linewidth = 0.8,
linetype = FALSE,
font.family = "sans",
font.size = 8,
legend.position = "top",
```

```
  legend.collect = TRUE,
  inset.left = 0.6,
  inset.bottom = 0.05,
  inset.right = 0.98,
  inset.top = 0.45,
  inset.align.to = c("panel", "plot", "full"),
  inset.legend.position = NULL,
  print.panel = FALSE,
  filename.ggsave = NULL,
  width.ggsave = NULL,
  height.ggsave = NULL,
  dpi.ggsave = 300,
  survfit.info = NULL,
  axis.info = NULL,
  visual.info = NULL,
  panel.info = NULL,
  style.info = NULL,
  inset.info = NULL,
  print.info = NULL,
  ggsave.info = NULL,
  engine = "cifplot",
  ...
)
```

Arguments

`plots` Optional list of existing ggplot objects to be arranged into a panel. When plots is supplied, no new models are fitted; the plots are used as-is.

`formula` A model formula specifying the time-to-event outcome on the left-hand side (typically `Event(time, status)` or `Surv(time, status)`) and, optionally, a stratification variable on the right-hand side. Unlike `cifplot()`, this function does not accept a fitted `survfit` object.

`formulas` Optional list of formulas. When given, each formula defines **one panel**. This is the most common way to create "one variable per plot" panels.

`data` A data frame containing variables in the formula.

`weights` Optional name of the weight variable in `data`. Weights must be nonnegative.

`subset.condition` Optional character string giving a logical condition to subset `data` (default `NULL`).

`na.action` A function specifying the action to take on missing values (default `na.omit`).

`outcome.type` Character string specifying the type of time-to-event outcome. One of `"survival"` (Kaplan-Meier) or `"competing-risk"` (Aalen-Johansen). If `NULL` (default), the function automatically infers the outcome type from the data: if the event variable has more than two unique levels, `"competing-risk"` is assumed; otherwise, `"survival"` is used. You can also use abbreviations such as `"S"` or `"C"`. Mixed or ambiguous inputs (e.g., `c("S", "C")`) trigger automatic detection based on the event coding.

`code.event1` Integer code of the event of interest (default `1`).

`code.event2` Integer code of the competing risk (default `2`).

`code.censoring` Integer code of censoring (default `0`).

`code.events` Optional specification of event/censoring codes. For single-panel calls, supply a numeric vector. For competing-risk outcomes, use `c(event1, event2, censoring)`. For survival outcomes, a length-2 or length-3 vector is allowed: `c(event, censoring)` or `c(event, *, censoring)`, where any middle element is ignored. When supplied, this argument overrides `code.event1`, `code.event2`, and `code.censoring` for the purpose of estimation. For panel displays (e.g. `cifpanel()` or when `panel.per.event = TRUE` or `panel.censoring = TRUE`), `code.events` may also be a list of such numeric vectors, one per panel.

`error` Character string specifying the method for SEs and CIs used internally. For `"survival"` without weights, choose one of `"greenwood"` (default), `"tsiatis"`, or `"if"`. For `"competing-risk"` without weights, choose one of `"delta"` (default), `"aalen"`, or `"if"`. SEs and CIs based on influence functions (`"if"`) is recommended for weighted analysis.

`conf.type` Character specifying the method of transformation for CIs used internally (default `arcsine-square root`).

`conf.int` Numeric two-sided level of CIs (default `0.95`).

`n.risk.type` Character string; one of `"weighted"`, `"unweighted"`, or `"ess"`. Controls which risk set size is returned in `$n.risk` without affecting estimates or standard errors. (default `"weighted"`).

`type.y` Character string specifying the y-scale. For survival/CIF curves, `"surv"` implies survival probabilities and `"risk"` implies CIF (1-survival in simple survival settings). Specify `"cumhaz"` to plot cumulative hazard or `"cloglog"` to generate a complementary log-log plot. If `NULL`, a default is chosen from `outcome.type` or the survfit object.

`label.x` Character x-axis label (default `"Time"`).

`label.y` Character y-axis label (default is chosen automatically from `outcome.type` and `type.y`, e.g. "Survival", "Cumulative incidence" or "Cumulative hazard").

`label.strata` Character vector or named character vector specifying labels for strata. Names (if present) must match the (re-ordered) underlying strata levels. **Note:** when any of the panel modes is active (`panel.per.variable = TRUE`, `panel.per.event = TRUE`, `panel.censoring = TRUE`, or `panel.mode = "auto"` and it actually dispatches to a panel), strata labels are suppressed to avoid duplicated legends across sub-plots.

`order.strata` Optional character vector specifying the display order of strata in the legend/number-at-risk table. Specify the levels of strata. Levels not listed are dropped.

`level.strata` Optional character vector giving the full set of expected strata levels. When provided, both `order.strata` and `label.strata` are validated against it before application.

`limits.x` Numeric length-2 vector specifying x-axis limits. If `NULL`, it is set from the fitted object (typically `c(0, max(time))`).

`limits.y` Numeric length-2 vector specifying y-axis limits. If `NULL`, it is set to `c(0, 1)` for probability-type outcomes.

`breaks.x` Numeric vector of x-axis breaks (default `NULL`).

`breaks.y` Numeric vector of y-axis breaks (default `NULL`).

`add.conf` Logical; if `TRUE`, adds a CI ribbon (via `ggsurvfit::add_confidence_interval()`). Default `TRUE`.

`add.risktable` Logical; if `TRUE`, adds a numbers-at-risk table under the plot. Default `TRUE`. **Note:** when a panel mode is active, tables are suppressed.

`add.estimate.table` Logical; if `TRUE`, adds a table of estimates and CIs. Default `FALSE`. **Note:** when a panel mode is active, tables are suppressed.

`symbol.risk.table` Character specifying the symbol used in the risk table to denote strata: `"square"`, `"circle"`, or `"triangle"` (default `"square"`).

`font.size.risk.table` Numeric font size for texts in risk / estimate tables (default `3`).

`add.censor.mark` Logical; if `TRUE`, draws censoring marks on each curve (via `ggsurvfit::add_censor_mark()`). Default `TRUE`.

`shape.censor.mark` Integer point shape used for censoring marks (default `3`).

`size.censor.mark` Numeric point size used for censoring marks (default `2`).

`add.competing.risk.mark` Logical; if `TRUE`, draws time marks for the competing event (event 2). If no times are supplied via `competing.risk.time`, the function tries to extract them automatically from the data. Default `FALSE`.

`competing.risk.time` A **named list** of numeric vectors. Each name must correspond to a strata label, and its numeric vector gives the times at which the competing event occurred in that stratum. Typically left as `list()` and filled internally.

`shape.competing.risk.mark` Integer point shape for competing-risk marks (default `16`).

`size.competing.risk.mark` Numeric point size for competing-risk marks (default `2`).

`add.intercurrent.event.mark` Logical; if `TRUE`, overlays user-specified intercurrent-event times per stratum. Default `FALSE`.

`intercurrent.event.time` A **named list** of numeric vectors for intercurrent events (names must match strata labels).

`shape.intercurrent.event.mark` Integer point shape for intercurrent-event marks (default `1`).

`size.intercurrent.event.mark` Numeric point size for intercurrent-event marks (default `2`).

`add.quantile` Logical; if `TRUE`, adds a quantile reference line (via `ggsurvfit::add_quantile()`). Default `FALSE`.

`level.quantile` Numeric quantile level to be shown (default `0.5` for the median).

`rows.columns.panel`
Optional integer vector `c(nrow, ncol)` controlling the layout of the panel returned by the panel modes. If `NULL`, an automatic layout is determined from the number of subplots.

`inset.panel` Logical. If `FALSE` (default), all panels are arranged in a regular grid using `patchwork::wrap_plots()` and `plot_layout()`. If `TRUE`, the function switches to "inset mode": the **first** plot becomes the main plot and the **second** plot (only the second) is drawn on top of it as an inset. Additional plots beyond the second are ignored in inset mode. Use grid mode to display more than two panels (`inset.panel = FALSE`).

`title.panel`, `subtitle.panel`, `caption.panel`
Character annotations applied to the **whole** panel layout (not to individual plots). These are passed to `patchwork::plot_annotation()` and are useful for creating figure-like outputs (title + subfigures + caption).

`tag.panel` Passed to `patchwork::plot_annotation()` to auto-label individual panels (e.g. `"A"`, `"B"`, `"C"`). Typical values are `"A"`, `"1"`, or `"a"`. See `?patchwork::plot_annotation`.

`title.plot` Character vector of titles for **each panel** in the order they are drawn. Length-1 values are recycled to all panels. In inset mode, the first element refers to the main plot and the second (if present) to the inset.

`style` Character choosing the base plot style: `"classic"`, `"bold"`, `"framed"`, `"grid"`, `"gray"` or `"ggsurvfit"` (default `"classic"`). Abbreviations such as `"C"`, `"B"`, `"F"`, or `"G"` are also accepted.

`palette` Optional character vector specifying the color palette to use across strata.

`linewidth` Optional numeric specifying the line width of curve (default `0.8`).

`linetype` Optional logical using different line types of curve (default `FALSE`).

`font.family` Character specifying the font family: `"sans"`, `"serif"`, or `"mono"` (default `"sans"`).

`font.size` Integer specifying the base font size (default `12`).

`legend.position`
Character specifying the legend position: `"top"`, `"right"`, `"bottom"`, `"left"`, or `"none"` (default `"top"`).

`legend.collect` Logical; if `TRUE`, try to collect a single legend for all panels (passed to **patchwork**). Default `TRUE`.

`inset.left`, `inset.bottom`, `inset.right`, `inset.top`
Numeric values in the range `[0, 1]` that define the inset box as fractions of the reference area. `inset.left` / `inset.right` control the horizontal position, `inset.bottom` / `inset.top` control the vertical position. Values are interpreted as "from the left/bottom" of the reference. For example, `inset.left = 0.4`, `inset.right = 1.0` draws the inset over the right 60% of the reference area.

`inset.align.to` Character string specifying the coordinate system for the inset box. One of `"panel"` (default; the box is placed relative to the panel area, i.e. the plotting region excluding outer titles/margins), `"plot"` (relative to the entire plot area, including axes and titles of the main plot), or `"full"` (relative to the full patchwork canvas). This argument is passed to `patchwork::inset_element()`.

`inset.legend.position`
: Optional legend position **for the inset plot only**. If `NULL` (default), the inset plot keeps whatever legend position was defined for it (often this means a legend will also be inset). Set, for example, `"none"` to hide the legend inside the inset, while still showing the main plot's legend.

`print.panel`
: Logical. When `TRUE`, panel displays created internally are printed automatically in interactive sessions; otherwise they are returned invisibly for further modification (default `FALSE`).

`filename.ggsave`
: Character; if non-`NULL`, save the plot to this file.

`width.ggsave`
: Numeric width passed to `ggplot2::ggsave()` (default `6`).

`height.ggsave`
: Numeric height passed to `ggplot2::ggsave()` (default `6`).

`dpi.ggsave`
: Numeric DPI passed to `ggplot2::ggsave()` (default `300`).

`survfit.info`, `axis.info`, `visual.info`, `panel.info`, `style.info`, `print.info`, `ggsave.info`, `inset.info`
: Internal lists used for programmatic control. Not intended for direct user input.

`engine`
: Character string specifying the internal rendering engine used to build each panel. Currently intended for internal use; default is `"cifplot"`.

`...`
: Additional arguments forwarded to the internal `cifplot_single()` calls that build each panel. Use this to pass low-level options such as `competing.risk.time`, `intercurrent.event.time`, or styling overrides.

Details

Overview:

`cifpanel()` composes multiple survival/CIF plots into a single figure. For each panel, it estimates curves via `cifcurve()` and renders them with `cifplot()`. You can supply a single `formula` reused across panels or a list in `formulas` (one per panel). When both are provided, `formulas` wins.

Outcome type & event coding:

- Use `outcome.type` to set per-panel estimator (`"survival"`=KM, `"competing-risk"`=AJ).
- Alternatively, pass `code.events` per panel to infer the type:
  - length 2 = survival: `c(event1, censor)`
  - length 3 = competing-risk: `c(event1, event2, censor)`
- If `code.events` is `NULL`, `code.event1`, `code.event2`, `code.censoring` are combined into `code.events = list(c(code.event1, code.event2, code.censoring))` with `NA` values dropped.
- If `outcome.type` is `NULL`, the function infers each panel from its `code.events[[i]]` length. When both are given, `outcome.type` takes precedence.
- Control risk-set displays via `n.risk.type`, which is recycled per panel and forwarded to `cifcurve()` to decide which risk set size populates `$n.risk` (e.g., weighted vs. unweighted counts).

Panel-wise vs shared arguments:

Panel layout is specified by length-2 vector `rows.columns.panel`. This function can also automatically determine the panel count in the following order: (1) if `plots` is supplied, its length defines the number of plots, (2) else if `formulas` is supplied, its length defines the number of plots, (3) else if `code.events` is supplied, its length defines the number of plots together with formula, and (4) otherwise `rows.columns.panel=c(1,1)`.

Many arguments accept a **scalar** (recycled to all panels) or a **list/vector** (one entry per panel). Precedence: **panel-wise explicit values > shared scalar > internal defaults**. Length-1 inputs are recycled.

Grid vs inset composition:

- **Grid mode** (`inset.panel = FALSE`, default): plots are arranged with `patchwork::wrap_plots()` and `plot_layout()`. If `legend.collect = TRUE`, legends are collected across panels where possible.
- **Inset mode** (`inset.panel = TRUE`): the **second** plot is overlaid into the **first** using `patchwork::inset_element()`. Only the first two plots are used; extra plots are ignored. Control the inset box with `inset.left`, `inset.bottom`, `inset.right`, `inset.top`, and its reference frame via `inset.align.to` (`"panel"`, `"plot"`, or `"full"`).

Advanced panel controls (forwarded to `cifplot()`):

The following arguments allow **per-panel** control by supplying vectors/lists, or **shared** control by supplying scalars. They are forwarded to `cifplot()`.

- `formula` or `formulas`: one formula or a list of formulas; each entry creates a panel.
- `data`, `outcome.type`, `code.events`, `type.y`: recycled across panels unless a list is supplied for per-panel control.
- `rows.columns.panel`: specification of grid layout by c(rows, cols).
- `inset.panel`: inset layout.
- `title.panel`, `subtitle.panel`, `caption.panel`, `title.plot`: overall titles and captions.
- `tag.panel`: panel tag style (e.g., "A", "a", "1").
- `label.x`, `label.y`, `limits.x`, `limits.y`, `breaks.x`, `breaks.y`: shared axis control unless a list is supplied for per-panel control.

Scale & labels:

Argument	Meaning	Default
`type.y`	`"risk"` (CIF y-axis) or `NULL` (survival).	inferred
`label.x`, `label.y`	Axis labels per panel.	auto
`label.strata`	Legend labels per panel.	from data
`limits.x`, `limits.y`	Axis limits `c(min, max)`.	auto
`breaks.x`, `breaks.y`	Axis breaks (forwarded to `breaks.x`/`breaks.y`).	auto

Plot layers (toggles):

Argument	Effect	Default
`add.conf`	CI ribbon.	`TRUE`
`add.censor.mark`	Censor marks.	`TRUE`

add.competing.risk.mark	Marks for event2 at supplied times.	FALSE
add.intercurrent.event.mark	User-specified intercurrent marks.	FALSE
add.quantile	Quantile reference line(s).	FALSE

(Time marks inputs such as competing.risk.time / intercurrent.event.time *can be given via ... if needed; names must match strata labels.)*

Legend & annotations:

- legend.position: "top", "right", "bottom", "left", or "none" (applies to all panels).
- Grid mode: legend.collect = TRUE attempts a shared legend.
- Panel annotations: title.panel, subtitle.panel, caption.panel.
- Tagging: tag.panel is passed to patchwork::plot_annotation().
- In inset mode, title.plot = c(title_base, title_inset) labels the two plots.

Export (optional):

If filename.ggsave is non-NULL, the composed panel is saved with ggsave() using width.ggsave, height.ggsave, and dpi.ggsave. Otherwise, the function returns objects without saving.

Notes

- Mixed panel types are supported (e.g., AJ in panel 1; KM in panel 2).
- If formulas is shorter than the grid capacity, empty slots are ignored.
- When supplying vectors/lists per panel, their lengths must match the number of panels; length-1 inputs are recycled; otherwise an error is thrown.
- For CIF displays, set type.y = "risk". For survival scale, use type.y = NULL or = "surv". For ADaM-style data, use code.events=c(0,1) or code.event1 = 0, code.censoring = 1.
- Additional graphical options (e.g., theme) can be added post-hoc to each element of list.plot or to the composed patchwork.

Value

A "cifpanel" object (returned invisibly), which is a list with at least the following elements:

- list.plot: a list of ggplot objects, one per panel
- patchwork: a patchwork object representing the composed panel
- plot: reserved for backwards compatibility (always NULL)
- metadata fields mirroring those in cifplot() (such as information on the fitted curves and display settings)

When print.panel = TRUE, the patchwork object is printed in interactive sessions in addition to being returned.

Lifecycle

[Experimental]

See Also

polyreg() for log-odds product modeling of CIFs; cifcurve() for KM/AJ estimators; cifplot() for display of a CIF; ggsurvfit::ggsurvfit, patchwork::patchwork and modelsummary::modelsummary for display helpers.

Examples

```
data(diabetes.complications)
output1 <- cifpanel(
  title.panel = "A comparison of cumulative incidence of competing events",
  rows.columns.panel = c(1,2),
  formula = Event(t, epsilon) ~ fruitq,
  data = diabetes.complications,
  outcome.type = "competing-risk",
  code.events = list(c(1,2,0), c(2,1,0)),
  label.y = c("Diabetic retinopathy", "Macrovascular complications"),
  label.x = "Years from registration",
  subtitle.panel = "Stratified by fruit intake",
  caption.panel  = "Data: diabetes.complications",
  title.plot = c("Diabetic retinopathy", "Macrovascular complications"),
  legend.position = "bottom",
  legend.collect=TRUE
)
print(output1)

output2 <- cifplot(Event(t,epsilon) ~ fruitq,
                   data = diabetes.complications,
                   outcome.type="competing-risk",
                   code.event1=2,
                   code.event2=1,
                   add.conf = FALSE,
                   add.risktable = FALSE,
                   label.y="CIF of macrovascular complications",
                   label.x="Years from registration")
output3 <- cifplot(Event(t,epsilon) ~ fruitq,
                   data = diabetes.complications,
                   outcome.type="competing-risk",
                   code.event1=2,
                   code.event2=1,
                   add.conf = FALSE,
                   add.risktable = FALSE,
                   label.y="",
                   label.x="",
                   limits.y=c(0,0.15))
output4 <- list(a = output2$plot, b = output3$plot)
output5 <- cifpanel(plots = output4,
         inset.panel = TRUE,
         inset.left = 0.40, inset.bottom = 0.45,
         inset.right = 1.00, inset.top = 0.95,
         inset.align.to = "plot",
         inset.legend.position = "none",
         legend.position = "bottom")
```

```
print(output5)
```

`cifplot` *Generate a survival/CIF curve with marks that represent censoring, competing risks and intercurrent events*

Description

This function generates a survival or CIF curve from a unified formula–data interface or from an existing survfit object. When a formula is supplied, the LHS is typically `Event()` or `survival::Surv()`, and the RHS specifies an optional stratification variable. In addition to the curves themselves, `cifplot()` can add numbers-at-risk tables, tables of point estimates and CIs, censoring marks, competing-risk marks, and intercurrent-event marks.

For usual single-panel mode, the function returns an object whose `plot` component is a regular ggplot object that can be further modified (compatible with + and %+%). For more complex multi-panel displays, `cifplot()` can internally call `cifpanel()` via several "panel modes" (per event, per variable, or censoring-focused).

Usage

```
cifplot(
  formula_or_fit,
  data = NULL,
  weights = NULL,
  subset.condition = NULL,
  na.action = na.omit,
  outcome.type = c("competing-risk", "survival"),
  code.event1 = 1,
  code.event2 = 2,
  code.censoring = 0,
  code.events = NULL,
  error = NULL,
  conf.type = "arcsine-square root",
  conf.int = 0.95,
  n.risk.type = c("weighted", "unweighted", "ess"),
  type.y = NULL,
  label.x = "Time",
  label.y = NULL,
  label.strata = NULL,
  level.strata = NULL,
  order.strata = NULL,
  limits.x = NULL,
  limits.y = NULL,
  breaks.x = NULL,
  breaks.y = NULL,
  use.coord.cartesian = FALSE,
```

```
  add.conf = TRUE,
  add.risktable = TRUE,
  add.estimate.table = FALSE,
  symbol.risk.table = "square",
  font.size.risk.table = 3,
  add.censor.mark = TRUE,
  shape.censor.mark = 3,
  size.censor.mark = 2,
  add.competing.risk.mark = FALSE,
  competing.risk.time = list(),
  shape.competing.risk.mark = 16,
  size.competing.risk.mark = 2,
  add.intercurrent.event.mark = FALSE,
  intercurrent.event.time = list(),
  shape.intercurrent.event.mark = 1,
  size.intercurrent.event.mark = 2,
  add.quantile = FALSE,
  level.quantile = 0.5,
  panel.per.event = FALSE,
  panel.censoring = FALSE,
  panel.per.variable = FALSE,
  panel.mode = "auto",
  rows.columns.panel = NULL,
  style = "classic",
  palette = NULL,
  linewidth = 0.8,
  linetype = FALSE,
  font.family = "sans",
  font.size = 12,
  legend.position = "top",
  print.panel = FALSE,
  filename.ggsave = NULL,
  width.ggsave = 6,
  height.ggsave = 6,
  dpi.ggsave = 300,
  survfit.info = NULL,
  axis.info = NULL,
  visual.info = NULL,
  panel.info = NULL,
  style.info = NULL,
  inset.info = NULL,
  print.info = NULL,
  ggsave.info = NULL,
  ...
)
```

Arguments

- `formula_or_fit` Either a model formula or a survfit object. When a formula is supplied, the LHS must be `Event(time, status)` or `Surv(time, status)`. The RHS specifies an optional stratification variable.
- `data` A data frame containing variables in the formula.
- `weights` Optional name of the weight variable in `data`. Weights must be nonnegative.
- `subset.condition` Optional character string giving a logical condition to subset `data` (default `NULL`).
- `na.action` A function specifying the action to take on missing values (default `na.omit`).
- `outcome.type` Character string specifying the type of time-to-event outcome. One of `"survival"` (Kaplan-Meier) or `"competing-risk"` (Aalen-Johansen). If `NULL` (default), the function automatically infers the outcome type from the data: if the event variable has more than two unique levels, `"competing-risk"` is assumed; otherwise, `"survival"` is used. You can also use abbreviations such as `"S"` or `"C"`. Mixed or ambiguous inputs (e.g., `c("S", "C")`) trigger automatic detection based on the event coding.
- `code.event1` Integer code of the event of interest (default `1`).
- `code.event2` Integer code of the competing risk (default `2`).
- `code.censoring` Integer code of censoring (default `0`).
- `code.events` Optional specification of event/censoring codes. For single-panel calls, supply a numeric vector. For competing-risk outcomes, use `c(event1, event2, censoring)`. For survival outcomes, a length-2 or length-3 vector is allowed: `c(event, censoring)` or `c(event, *, censoring)`, where any middle element is ignored. When supplied, this argument overrides `code.event1`, `code.event2`, and `code.censoring` for the purpose of estimation. For panel displays (e.g. `cifpanel()` or when `panel.per.event = TRUE` or `panel.censoring = TRUE`), `code.events` may also be a list of such numeric vectors, one per panel.
- `error` Character string specifying the method for SEs and CIs used internally. For `"survival"` without weights, choose one of `"greenwood"` (default), `"tsiatis"`, or `"if"`. For `"competing-risk"` without weights, choose one of `"delta"` (default), `"aalen"`, or `"if"`. SEs and CIs based on influence functions (`"if"`) is recommended for weighted analysis.
- `conf.type` Character specifying the method of transformation for CIs used internally (default `arcsine-square root`).
- `conf.int` Numeric two-sided level of CIs (default `0.95`).
- `n.risk.type` Character string; one of `"weighted"`, `"unweighted"`, or `"ess"`. Controls which risk set size is returned in `$n.risk` without affecting estimates or standard errors. (default `"weighted"`).
- `type.y` Character string specifying the y-scale. For survival/CIF curves, `"surv"` implies survival probabilities and `"risk"` implies CIF (1-survival in simple survival settings). Specify `"cumhaz"` to plot cumulative hazard or `"cloglog"` to generate a complementary log-log plot. If `NULL`, a default is chosen from `outcome.type` or the survfit object.
- `label.x` Character x-axis label (default `"Time"`).

`label.y` Character y-axis label (default is chosen automatically from `outcome.type` and `type.y`, e.g. "Survival", "Cumulative incidence" or "Cumulative hazard").

`label.strata` Character vector or named character vector specifying labels for strata. Names (if present) must match the (re-ordered) underlying strata levels. **Note:** when any of the panel modes is active (`panel.per.variable = TRUE`, `panel.per.event = TRUE`, `panel.censoring = TRUE`, or `panel.mode = "auto"` and it actually dispatches to a panel), strata labels are suppressed to avoid duplicated legends across sub-plots.

`level.strata` Optional character vector giving the full set of expected strata levels. When provided, both `order.strata` and `label.strata` are validated against it before application.

`order.strata` Optional character vector specifying the display order of strata in the legend/number-at-risk table. Specify the levels of strata. Levels not listed are dropped.

`limits.x` Numeric length-2 vector specifying x-axis limits. If `NULL`, it is set from the fitted object (typically `c(0, max(time))`).

`limits.y` Numeric length-2 vector specifying y-axis limits. If `NULL`, it is set to `c(0, 1)` for probability-type outcomes.

`breaks.x` Numeric vector of x-axis breaks (default `NULL`).

`breaks.y` Numeric vector of y-axis breaks (default `NULL`).

`use.coord.cartesian` Logical; if `TRUE`, uses `ggplot2::coord_cartesian()` for zooming instead of changing the scale limits (default `FALSE`).

`add.conf` Logical; if `TRUE`, adds a CI ribbon (via `ggsurvfit::add_confidence_interval()`). Default `TRUE`.

`add.risktable` Logical; if `TRUE`, adds a numbers-at-risk table under the plot. Default `TRUE`. **Note:** when a panel mode is active, tables are suppressed.

`add.estimate.table` Logical; if `TRUE`, adds a table of estimates and CIs. Default `FALSE`. **Note:** when a panel mode is active, tables are suppressed.

`symbol.risk.table` Character specifying the symbol used in the risk table to denote strata: `"square"`, `"circle"`, or `"triangle"` (default `"square"`).

`font.size.risk.table` Numeric font size for texts in risk / estimate tables (default `3`).

`add.censor.mark` Logical; if `TRUE`, draws censoring marks on each curve (via `ggsurvfit::add_censor_mark()`). Default `TRUE`.

`shape.censor.mark` Integer point shape used for censoring marks (default `3`).

`size.censor.mark` Numeric point size used for censoring marks (default `2`).

`add.competing.risk.mark` Logical; if `TRUE`, draws time marks for the competing event (event 2). If no times are supplied via `competing.risk.time`, the function tries to extract them automatically from the data. Default `FALSE`.

`competing.risk.time`
: A **named list** of numeric vectors. Each name must correspond to a strata label, and its numeric vector gives the times at which the competing event occurred in that stratum. Typically left as `list()` and filled internally.

`shape.competing.risk.mark`
: Integer point shape for competing-risk marks (default `16`).

`size.competing.risk.mark`
: Numeric point size for competing-risk marks (default `2`).

`add.intercurrent.event.mark`
: Logical; if `TRUE`, overlays user-specified intercurrent-event times per stratum. Default `FALSE`.

`intercurrent.event.time`
: A **named list** of numeric vectors for intercurrent events (names must match strata labels).

`shape.intercurrent.event.mark`
: Integer point shape for intercurrent-event marks (default `1`).

`size.intercurrent.event.mark`
: Numeric point size for intercurrent-event marks (default `2`).

`add.quantile`
: Logical; if `TRUE`, adds a quantile reference line (via `ggsurvfit::add_quantile()`). Default `FALSE`.

`level.quantile`
: Numeric quantile level to be shown (default `0.5` for the median).

`panel.per.event`
: Logical. **Explicit panel mode.** If `TRUE` and `outcome.type == "competing-risk"`, `cifplot()` internally calls `cifpanel()` to display two event-specific CIFs side-by-side (event 1 and event 2) using reversed `code.events`. Ignored for non-competing-risk outcomes.

`panel.censoring`
: Logical. **Explicit panel mode.** If `TRUE` and `outcome.type == "survival"`, `cifplot()` internally calls `cifpanel()` to display KM curves for `(event, censor)` and `(censor, event)` so that censoring patterns can be inspected.

`panel.per.variable`
: Logical. **Explicit panel mode.** If `TRUE` and the RHS of the formula has multiple covariates (e.g. `~ a + b + c`), the function produces a panel where each variable in the RHS is used once as the stratification factor.

`panel.mode`
: Character specifying **Automatic panel mode.** If `"auto"` and none of `panel.per.variable`, `panel.per.event`, `panel.censoring` has been set to `TRUE`, the function chooses a suitable panel mode automatically: (i) if the formula RHS has 2+ variables, it behaves like `panel.per.variable = TRUE`; (ii) otherwise, if `outcome.type == "competing-risk"`, it behaves like `panel.per.event = TRUE`; (iii) otherwise, if `outcome.type == "survival"`, it behaves like `panel.censoring = TRUE`. If a panel mode is explicitly specified, `panel.mode` is ignored.

`rows.columns.panel`
: Optional integer vector `c(nrow, ncol)` controlling the layout of the panel returned by the panel modes. If `NULL`, an automatic layout is determined from the number of subplots.

`style`	Character choosing the base plot style: `"classic"`, `"bold"`, `"framed"`, `"grid"`, `"gray"` or `"ggsurvfit"` (default `"classic"`). Abbreviations such as `"C"`, `"B"`, `"F"`, or `"G"` are also accepted.
`palette`	Optional character vector specifying the color palette to use across strata.
`linewidth`	Optional numeric specifying the line width of curve (default `0.8`).
`linetype`	Optional logical using different line types of curve (default `FALSE`).
`font.family`	Character specifying the font family: `"sans"`, `"serif"`, or `"mono"` (default `"sans"`).
`font.size`	Integer specifying the base font size (default `12`).
`legend.position`	Character specifying the legend position: `"top"`, `"right"`, `"bottom"`, `"left"`, or `"none"` (default `"top"`).
`print.panel`	Logical. When `TRUE`, panel displays created internally are printed automatically in interactive sessions; otherwise they are returned invisibly for further modification (default `FALSE`).
`filename.ggsave`	Character; if non-`NULL`, save the plot to this file.
`width.ggsave`	Numeric width passed to `ggplot2::ggsave()` (default `6`).
`height.ggsave`	Numeric height passed to `ggplot2::ggsave()` (default `6`).
`dpi.ggsave`	Numeric DPI passed to `ggplot2::ggsave()` (default `300`).
`survfit.info`, `axis.info`, `visual.info`, `panel.info`, `style.info`, `inset.info`, `print.info`, `ggsave.info`	Internal lists used for programmatic control. Not intended for direct user input.
`...`	Additional arguments passed to internal helper functions.

Details

Typical use cases:

- Draw one survival/CIF curve set by exposure groups (e.g., treatment vs control).
- Call `cifpanel()` with a simplified code to create a panel displaying plots of multiple stratified survival/CIF curves or CIF curves for each event type.
- Add CIs and censor/competing-risk/intercurrent-event marks.
- Add number-at-risk table to display the number at risk or the estimated survival probabilities or CIFs and CIs at each point in time.

Key arguments shared with cifcurve():

- **Outcome type and estimator**
  - `outcome.type = "survival"`: Kaplan-Meier estimator
  - `outcome.type = "competing-risk"`: Aalen-Johansen estimator
- **Confidence intervals**
  - `conf.int` sets the two-sided level (default 0.95)
  - `conf.type` chooses the transformation (`"arcsine-square root"`, `"plain"`, `"log"`, `"log-log"`, `"logit"`, or `"none"`)

  - `error` chooses the estimator for SE (`"greenwood"`, `"tsiatis"` or `"if"` for survival curves and `"delta"`, `"aalen"` or `"if"` for CIFs)
- **Risk sets used in tables**
  - `n.risk.type` controls whether `$n.risk` reflects weighted, unweighted, or effective sample size counts when building risk tables (forwarded to `cifcurve()`); when a fitted `survfit` object is supplied, existing risk sets are used as-is.

Key arguments for cifplot():

- **Data visualization**
  - `add.conf` adds CIs on the ggplot2-based plot
  - `add.competing.risk.mark` and `add.intercurrent.event.mark` adds symbols to describe competing risks or intercurrent events in addition to conventional censoring marks with `add.censor.mark`
  - `add.risktable` adds numbers at risk
  - `add.estimate.table` adds time-by-time estimates and CIs
  - `add.quantile` adds a reference line at a chosen quantile level
- **Plot customization**
  - `type.y` chooses y-axis (`"surv"` for survival and `"risk"` for 1-survival/CIF)
  - `limits.x`, `limits.y`, `breaks.x`, `breaks.y`: numeric vectors for axis control
  - `style` specifies the appearance of plot (`"classic"`, `"bold"`, `"framed"`, `"grid"`, `"gray"` or `"ggsurvfit"`)
  - `palette` specifies color of each curve (e.g. `palette=c("blue1", "cyan3", "navy", "deepskyblue3")`))
- **Panel display**
  - `panel.per.variable` produces multiple survival/CIF curves per stratification variable specified in the formula
  - `panel.per.event` produces CIF curves for each event type
  - `panel.censoring` produces the Kaplan–Meier curves for (event, censor) and (censor, event) so that censoring patterns can be inspected
  - `panel.mode` uses automatic panel mode

When `panel.per.event = TRUE`, two panels are created with `code.events = list(c(e1, e2, c), c(e2, e1, c))`, where `code.events = c(e1, e2, c)` is the input coding for event1, event2, and censoring. Common legend is collected by default (`legend.collect = TRUE`).

Numeric stratification variables are normalized automatically. Columns with fewer than nine distinct numeric values are coerced to factors; columns with nine or more distinct numeric values are split at the median into "Below median" and "Above median" strata.

Advanced control not required for typical use:

The arguments below fine-tune internal estimation and figure appearance. **Most users do not need to change these defaults.**

Graphical layers:

Argument	Description	Default
`add.conf`	Add confidence interval ribbon.	`TRUE`
`add.risktable`	Add numbers-at-risk table below the plot.	`TRUE`
`add.estimate.table`	Add estimates and confidence intervals table.	`FALSE`

`symbol.risk.table`	Symbol for strata in risk / estimate tables	`"square"`
`font.size.risk.table`	Font size for texts in risk / estimate tables	`3`
`add.censor.mark`	Add censoring marks.	`TRUE`
`add.competing.risk.mark`	Add marks for event2 of "competing-risk" outcome.	`FALSE`
`add.intercurrent.event.mark`	Add intercurrent event marks at user-specified times.	`FALSE`
`add.quantile`	Add quantile reference lines.	`FALSE`
`level.quantile`	Quantile level for `add.quantile`.	`0.5`

Time for marks:

Argument	Description
`competing.risk.time`	**Named list** of numeric vectors that contains times of competing risks. Names must match strat
`intercurrent.event.time`	**Named list** of numeric vectors that contains times of intercurrent events. Names must match st

Appearance of marks:

Argument	Applies to	Default
`shape.censor.mark`	Censoring marks	`3` (cross)
`size.censor.mark`	Censoring marks	`2`
`shape.competing.risk.mark`	Competing-risk marks	`16` (filled circle)
`size.competing.risk.mark`	Competing-risk marks	`2`
`shape.intercurrent.event.mark`	Intercurrent marks	`1` (circle)
`size.intercurrent.event.mark`	Intercurrent marks	`2`

Panel display:

Argument	Description
`panel.per.variable`	One panel per stratification variable
`panel.per.event`	For `"competing-risk"`, show CIFs of event 1 and event 2
`panel.censoring`	For survival, show (event, censor) vs (censor, event)
`panel.mode` with 2+ stratification variables	Behave like `panel.per.variable`
`panel.mode` with outcome.type = "competing-risk"	Behave like `panel.per.event`
`panel.mode` with outcome.type = "survival"	Behave like `panel.censoring`

Axes and legend:

Argument	Description	Default
`limits.x`, `limits.y`	Axis limits (`c(min, max)`)	Auto
`breaks.x`, `breaks.y`	Tick breaks for x and y axes	Auto
`use.coord.cartesian`	For zooming use `coord_cartesian()`	`FALSE`
`legend.position`	`"top"`, `"right"`, `"bottom"`, `"left"`, `"none"`	`"top"`

Export:

Argument	Description	Default
`filename.ggsave`	If non-`NULL`, save the plot using `ggsave()`	`NULL`
`width.ggsave`	Size passed to `ggsave()`	`6`
`height.ggsave`	Size passed to `ggsave()`	`6`
`dpi.ggsave`	DPI passed to `ggsave()`	`300`

Notes

- For CIF displays, set `type.y = "risk"`. For survival scale, use `type.y = NULL` or `= "surv"`. For a cumulative hazard plot, use `type.y = "cumhaz"`. To generate a log-log plot, use `type.y = "cloglog"`.
- Event coding can be controlled via `code.event1`, `code.event2`, `code.censoring`. For ADaM-style data, use `code.event1 = 0`, `code.censoring = 1`.
- Per-stratum time lists should have names identical to plotted strata labels.

Value

A `"cifplot"` object (a list) with at least the following elements:

- `plot`: a ggplot object containing the main plot
- `patchwork`: reserved for compatibility with panel displays (typically `NULL` for single-panel plots)
- `survfit.info`, `axis.info`, `visual.info`, `panel.info`, `style.info`, `inset.info`, `print.info`, `ggsave.info`: internal lists storing the fitted curves and display settings
- `version`: a character string giving the cifmodeling version used
- `call`: the original function call

When a panel mode is active and `print.panel = TRUE`, the panel is also printed in interactive sessions.

Lifecycle

[Stable]

See Also

`polyreg()` for log-odds product modeling of CIFs; `cifcurve()` for KM/AJ estimators; `cifpanel()` for display of multiple CIFs; ggsurvfit::ggsurvfit, patchwork::patchwork and modelsummary::modelsummary for display helpers.

Examples

```
data(diabetes.complications)
cifplot(Event(t,epsilon) ~ fruitq,
        data = diabetes.complications,
        outcome.type="competing-risk",
        add.risktable = FALSE,
        label.y='CIF of diabetic retinopathy',
        label.x='Years from registration')
```

`diabetes.complications` *Data from a cohort study of patients with type 2 diabetes*

Description

Anonymized data from a cohort study of patients with type 2 diabetes followed for ocular and macro-vascular complications.

Usage

```
data(diabetes.complications)
```

Format

A data frame with 978 observations and 19 variables:

t Follow-up time in years.

epsilon Event type indicator (0 = censored, 1 = diabetic retinopathy, 2 = macro-vascular complication).

fruit Fruit intake (g/day).

fruitq Quartile of fruit intake.

fruitq1 Binary indicator for low fruit intake.

strata Stratum used for inverse probability of censoring weights.

age Age at baseline (years).

sex Sex coded as 0 = woman, 1 = man.

bmi Body mass index at baseline.

hba1c Hemoglobin A1c (%).

diabetes_duration Duration of diabetes (years).

drug_oha Indicator for oral hypoglycemic agent use.

drug_insulin Indicator for insulin use.

sbp Systolic blood pressure (mmHg).

ldl Low-density lipoprotein cholesterol (mg/dL).

hdl High-density lipoprotein cholesterol (mg/dL).

tg Triglycerides (mg/dL).

current_smoker Indicator for current smoking status.

alcohol_drinker Indicator for current alcohol drinking.

ltpa Leisure-time physical activity (METs).

Details

The variables include follow-up time, cause-specific event indicators, exposure indicators for fruit intake, censoring strata, and a set of covariates used in the package vignettes.

Source

Anonymized data supplied with the package for documentation and demonstration purposes.

Examples

```
data(diabetes.complications)
str(diabetes.complications)
```

`Event` *Create a survival or competing-risks response*

Description

A lightweight response constructor used in `cifcurve()` and `polyreg()` to pass survival and competing-risks data via a model formula.

Usage

```
Event(time, event, allowed = getOption("cifmodeling.allowed", c(0, 1, 2)))
```

Arguments

`time`	Numeric vector of follow-up times (non-negative).
`event`	Integer (0=censor, 1,2,...) or a character/factor vector whose levels are numeric codes "0","1","2",... for competing events.
`allowed`	Numeric vector of acceptable event codes.

Value

An object of class `"Event"` (a 2-column matrix) with columns `time`, `event`.

Lifecycle

[Stable]

See Also

`polyreg()` for log-odds product modeling of CIFs; `cifcurve()` for KM/AJ estimators; `cifplot()` for display of a CIF; `cifpanel()` for display of multiple CIFs; ggsurvfit::ggsurvfit, patchwork::patchwork and modelsummary::modelsummary for display helpers.

Examples

```
## event: 0=censor, 1=primary, 2=competing
data(diabetes.complications)
output <- polyreg(
  nuisance.model = Event(t, epsilon) ~ +1,
  exposure = "fruitq1",
  data = diabetes.complications,
  effect.measure1 = "RR",
  effect.measure2 = "RR",
  time.point = 8,
  outcome.type = "competing-risk"
)
```

---

extract_time_to_event *Extract per-stratum event times from a formula and data*

Description

Creates a list of event times that can be passed to downstream visualization or analysis functions such as `competing.risk.time` or `intercurrent.event.time` in `cifplot()` and `cifpanel()`. Event types are specified by event 1, event 2, censoring, or user-specified codes.

Usage

```
extract_time_to_event(
  formula,
  data,
  subset.condition = NULL,
  na.action = na.omit,
  which.event = c("event2", "event1", "censor", "censoring", "user_specified"),
  code.event1 = 1,
  code.event2 = 2,
  code.censoring = 0,
  code.user.specified = NULL,
  read.unique.time = TRUE,
  drop.empty = TRUE
)
```

Arguments

`formula`	A model formula specifying the outcome and (optionally) `strata()`.
`data`	A data frame containing variables in `formula`.
`subset.condition`	Optional expression (as a character string) defining a subset of `data` to analyse. Defaults to `NULL`.
`na.action`	Function to handle missing values (default: `na.omit` in **stats**).

`which.event` One of `"event1"`, `"event2"`, `"censor"`, `"censoring"`, or `"user_specified"`, indicating which event type to extract times for.

`code.event1`, `code.event2`, `code.censoring`
Integer codes representing the event and censoring categories. Defaults are `1`, `2`, and `0`, respectively.

`code.user.specified`
When `which.event = "user_specified"`, the integer event code to extract (e.g., 3 for an intercurrent event).

`read.unique.time`
Logical if `TRUE`, only unique and sorted time points are returned for each stratum.

`drop.empty` Logical if `TRUE` (default), strata with no events are dropped from the returned list. Set to `FALSE` to retain empty strata as `numeric(0)` vectors (useful for diagnostics or consistent list length).

Details

This function is typically used internally by plotting and model functions, but can also be called directly to inspect the per-stratum event-time structure of a data frame.

Value

A named list of numeric vectors, where each element corresponds to a stratum and contains the event times of the selected type.

Lifecycle

[Stable]

See Also

`polyreg()` for log-odds product modeling of CIFs; `cifcurve()` for KM/AJ estimators; `cifplot()` for display of a CIF; `cifpanel()` for display of multiple CIFs; ggsurvfit::ggsurvfit, patchwork::patchwork and modelsummary::modelsummary for display helpers.

Examples

```
data(diabetes.complications)
output <- extract_time_to_event(Event(t,epsilon) ~ fruitq,
                                data = diabetes.complications,
                                which.event = "event2")
cifplot(Event(t,epsilon) ~ fruitq,
        data = diabetes.complications,
        outcome.type="competing-risk",
        add.conf=FALSE,
        add.risktable=FALSE,
        add.censor.mark=FALSE,
        add.competing.risk.mark=TRUE,
        competing.risk.time=output,
        label.y="CIF of diabetic retinopathy",
```

```
          label.x="Years from registration")
```

---

polyreg — *Fit coherent regression models of CIFs using polytomous log odds products*

Description

`polyreg()` fits regression models of CIFs, targeting familiar effect measures (risk ratios, odds ratios and subdistribution hazard ratios). Modeling the nuisance structure using polytomous log odds products ensures that the sum of cause-specific CIFs does not exceed one, and enables coherent modelling of the multiplicative effects.

This function follows a familiar formula–data workflow: the outcome and covariates other than the exposure are specified through a formula in `nuisance.model` (with `Event()` or `survival::Surv()` on the LHS), and the exposure of interest is given by a separate variable name in `exposure`. The fitted object contains tidy summaries of exposure effects (point estimates, SEs, CIs, and p-values) and can be summarised with `summary.polyreg()` or formatted with external tools such as `modelsummary::modelsummary()`.

Usage

```
polyreg(
  nuisance.model,
  exposure,
  strata = NULL,
  data,
  subset.condition = NULL,
  na.action = na.omit,
  code.event1 = 1,
  code.event2 = 2,
  code.censoring = 0,
  code.exposure.ref = 0,
  effect.measure1 = "RR",
  effect.measure2 = "RR",
  time.point = NULL,
  outcome.type = "competing-risk",
  conf.int = 0.95,
  report.nuisance.parameter = FALSE,
  report.optim.convergence = FALSE,
  report.sandwich.conf = TRUE,
  report.boot.conf = NULL,
  boot.bca = FALSE,
  boot.multiplier = "rademacher",
  boot.replications = 200,
  boot.seed = 46,
  nleqslv.method = "Newton",
  optim.parameter1 = 1e-06,
```

```
  optim.parameter2 = 1e-06,
  optim.parameter3 = 100,
  optim.parameter4 = 50,
  optim.parameter5 = 50,
  optim.parameter6 = 50,
  optim.parameter7 = 1e-10,
  optim.parameter8 = 1e-06,
  optim.parameter9 = 1e-06,
  optim.parameter10 = 40,
  optim.parameter11 = 0.025,
  optim.parameter12 = 2,
  optim.parameter13 = 0.5,
  data.initial.values = NULL,
  normalize.covariate = TRUE,
  terminate.time.point = TRUE,
  prob.bound = 1e-07
)
```

Arguments

`nuisance.model` A `formula` describing the outcome and nuisance covariates, excluding the exposure of interest. The LHS must be `Event(time, status)` or `survival::Surv(time, status)`.

`exposure` A character string giving the name of the categorical exposure variable in `data`.

`strata` Optional character string with the name of the stratification variable used to adjust for dependent censoring (default `NULL`).

`data` A data frame containing the outcome, exposure and nuisance covariates referenced by `nuisance.model`.

`subset.condition` Optional character string giving a logical condition to subset `data` (default `NULL`).

`na.action` A function specifying the action to take on missing values (default `na.omit`).

`code.event1` Integer code of the event of interest (default `1`).

`code.event2` Integer code of the competing event (default `2`).

`code.censoring` Integer code of censoring (default `0`).

`code.exposure.ref` Integer code identifying the reference exposure category (default `0`).

`effect.measure1` Character string specifying the effect measure for the primary event. Supported values are `"RR"`, `"OR"` and `"SHR"`.

`effect.measure2` Character string specifying the effect measure for the competing event. Supported values are `"RR"`, `"OR"` and `"SHR"`.

`time.point` Numeric time point at which the exposure effect is evaluated for time-point models. Required for `"competing-risk"` and `"survival"` outcomes.

`outcome.type` Character string selecting the outcome type. Valid values are `"competing-risk"`, `"survival"`, `"binomial"`, `"proportional-survival"`, and `"proportional-competing-risk"`. The default is `"competing-risk"`. If explicitly set to `NULL`, `polyreg()` attempts to infer the outcome type from the data: if the event variable has more than two distinct levels, `"competing-risk"` is assumed; otherwise, `"survival"` is assumed. Abbreviations such as `"S"` or `"C"` are accepted; mixed or ambiguous inputs trigger automatic detection from the event coding in `data`.

`conf.int` Numeric two-sided level of CIs (default `0.95`).

`report.nuisance.parameter`
Logical; if `TRUE`, the returned object includes estimates of the nuisance model parameters (default `FALSE`).

`report.optim.convergence`
Logical; if `TRUE`, optimization convergence summaries are returned (default `FALSE`).

`report.sandwich.conf`
Logical or `NULL`. When `TRUE`, CIs based on sandwich variance are computed. When `FALSE`, they are omitted (default `TRUE`). This CI is default for time-point models (`"outcome.type=competing-risk"`, `"survival"` or `"binomial"`) and is not available otherwise.

`report.boot.conf`
Logical or `NULL`. When `TRUE`, bootstrap CIs are computed. When `FALSE`, they are omitted. If `NULL`, the function chooses based on `outcome.type` (default `NULL`). This CI is default for proportional models (`outcome.type="proportional-competing-risk"` or `"proportional-survival"`).

`boot.bca` Logical indicating the bootstrap CI method. Use `TRUE` for bias-corrected and accelerated intervals or `FALSE` for the normal approximation (default `FALSE`).

`boot.multiplier`
Character string specifying the wild bootstrap weight distribution. One of `"rademacher"`, `"mammen"` or `"gaussian"` (default `"rademacher"`).

`boot.replications`
Integer giving the number of bootstrap replications (default `200`).

`boot.seed` Numeric seed used for resampling of bootstrap.

`nleqslv.method` Character string specifying the solver used in **nleqslv()**. Available choices are `"Broyden"` and `"Newton"`.

`optim.parameter1`
Numeric tolerance for convergence of the outer loop (default `1e-6`).

`optim.parameter2`
Numeric tolerance for convergence of the inner loop (default `1e-6`).

`optim.parameter3`
Numeric constraint on the absolute value of parameters (default `100`).

`optim.parameter4`
Integer maximum number of outer loop iterations (default `50`).

`optim.parameter5`
Integer maximum number of `nleqslv` iterations per outer iteration (default `50`).

`optim.parameter6`
: Integer maximum number of iterations for the Levenberg-Marquardt routine (default `50`).

`optim.parameter7`
: Numeric convergence tolerance for the Levenberg-Marquardt routine (default `1e-10`).

`optim.parameter8`
: Numeric tolerance for updating the Hessian in the Levenberg-Marquardt routine (default `1e-6`).

`optim.parameter9`
: Numeric starting value for the Levenberg-Marquardt damping parameter lambda (default `1e-6`).

`optim.parameter10`
: Numeric upper bound for lambda in the Levenberg-Marquardt routine (default `40`).

`optim.parameter11`
: Numeric lower bound for lambda in the Levenberg-Marquardt routine (default `0.025`).

`optim.parameter12`
: Numeric multiplicative increment applied to lambda when the Levenberg-Marquardt step is successful (default `2`).

`optim.parameter13`
: Numeric multiplicative decrement applied to lambda when the Levenberg-Marquardt step is unsuccessful (default `0.5`).

`data.initial.values`
: Optional data frame providing starting values for the optimization (default `NULL`).

`normalize.covariate`
: Logical indicating whether covariates should be centered and scaled prior to optimization (default `TRUE`).

`terminate.time.point`
: Logical indicating whether time points that contribute estimation are terminated by min of max follow-up times of each exposure level (default `TRUE`).

`prob.bound`
: Numeric lower bound used to internally truncate probabilities away from 0 and 1 (default `1e-5`).

Details

Overview:

`polyreg()` implements **log odds product modeling** for CIFs at user-specified time points, focusing on multiplicative effects of a categorical exposure, or constant effects over time like Cox regression and Fine-Gray models. It estimates multiplicative effects such as **risk ratios**, **odds ratios**, or **subdistribution hazard ratios**, while ensuring that the probabilities across competing events sum to one. This is achieved through **reparameterization using polytomous log odds products**, which fits so-called effect-measure models and nuisance models on multiple competing events simultaneously. Additionally, `polyreg()` supports direct binomial regression for survival outcomes and the Richardson model for binomial outcomes, both of which use log odds products.

Key arguments:

- `nuisance.model`: a formula with `Event()` or `survivai::Surv()` describing the outcome and nuisance covariates, excluding the exposure of interest.
- `exposure`: name of the categorical exposure variable
- `effect.measure1` and `effect.measure2`: the effect measures for event1 and event2 (`"RR"`, `"OR"` or `"SHR"`).
- `outcome.type`: type of the outcome variable (`"competing-risk"`, `"survival"`, `"binomial"`, `"proportional-survival"` or `"proportional-competing-risk"`).
- `time.point`: time point(s) at which the exposure effect is evaluated. Required for `"competing-risk"` and `"survival"` outcomes.
- `strata`: name of the stratification variable used for IPCW adjustment for dependent censoring.

Outcome type and event status coding:

The `outcome.type` argument must be set to:

- Effects on cumulative incidence probabilities at a specific time: `"competing-risk"`.
- Effects on a risk at a specific time: `"survival"`.
- Common effects on cumulative incidence probabilities over time: `"proportional-competing-risk"`.
- Common effects on a risk over time: `"proportional-survival"`.
- Effects on a risk of a binomial outcome: `"binomial"`.

Setting	Codes	Meaning
competing-risk	`code.event1`, `code.event2`, `code.censoring`	event of interest / compe
competing-risk (default)	`code.event1 = 1`, `code.event2 = 2`, `code.censoring = 0`	event of interest / compe
survival	`code.event1`, `code.censoring`	event / censoring
survival (default)	`code.event1 = 1`, `code.censoring = 0`	event / censoring
survival (ADaM-ADTTE)	`code.event1 = 0`, `code.censoring = 1`	set to match ADaM con
proportional-survival	`code.event1`, `code.censoring`	event / censoring
proportional-survival (default)	`code.event1 = 1`, `code.censoring = 0`	event / censoring
proportional-survival (ADaM)	`code.event1 = 0`, `code.censoring = 1`	set to match ADaM con
proportional-competing-risk	`code.event1`, `code.event2`, `code.censoring`	event of interest / compe
proportional-competing-risk (default)	`code.event1 = 1`, `code.event2 = 2`, `code.censoring = 0`	event of interest / compe

Effect measures for categorical exposure:

Choose the effect scale for event 1 and (optionally) event 2:

Argument	Applies to	Choices	Default
`effect.measure1`	event of interest	`"RR"`, `"OR"`, `"SHR"`	`"RR"`
`effect.measure2`	competing event	`"RR"`, `"OR"`, `"SHR"`	`"RR"`

- `RR`: risk ratio at `time.point` or common over time.
- `OR`: odds ratio at `time.point` or common over time.
- `SHR`: subdistribution hazard ratio or common over time.

Inference and intervals (advanced):

Argument	Meaning	Default
`conf.int`	Wald-type CI level	`0.95`
`report.sandwich.conf`	Sandwich variance CIs	`TRUE`
`report.boot.conf`	Bootstrap CIs (used by `"proportional-*"` types)	`NULL`
`boot.bca`	Use BCa intervals (else normal approximation)	`FALSE`
`boot.multiplier`	Method for wild bootstrap	`"rademacher"`
`boot.replications`	Bootstrap replications	`200`
`boot.seed`	Seed for resampling	`46`

Optimization & solver controls (advanced):

`polyreg()` solves estimating equations with optional inner routines.

Argument	Role	Default
`nleqslv.method`	Root solver	`"Newton"`
`optim.parameter1`, `optim.parameter2`	Outer / inner convergence tolerances	`1e-6`, `1e-6`
`optim.parameter3`	Parameter absolute bound	`100`
`optim.parameter4`	Max outer iterations	`50`
`optim.parameter5`	Max `nleqslv` iterations per outer	`50`
`optim.parameter6:13`	Levenberg–Marquardt controls (iterations, tolerances, lambda)	see defaults

Data handling and stability:

Argument	Meaning	Default
`subset.condition`	Expression (as character) to subset `data`	`NULL`
`na.action`	NA handling function	`stats::na.omit`
`normalize.covariate`	Center/scale nuisance covariates	`TRUE`
`terminate.time.point`	Truncate support by exposure-wise follow-up maxima	`TRUE`
`prob.bound`	Truncate probabilities away from 0/1 (numerical guard)	`1e-5`
`data.initial.values`	Optional starting values data frame	`NULL`

Downstream use:

`polyreg()` returns an object of class `"polyreg"` that contains regression coefficients (`coef`), variance-covariance matrix (`vcov`) and a list of event-wise *tidy* and *glance* tables (`summary`). Users should typically access results via the S3 methods:

- `coef()` — extract regression coefficients.
- `vcov()` — extract the variance–covariance matrix (sandwich or bootstrap, depending on `outcome.type` and the `report.*` arguments).
- `nobs()` — number of observations used in the fit.
- `summary()` — print an event-wise, modelsummary-like table of estimates, CIs and p-values, and return the underlying list of tidy/glance tables invisibly.

For backward compatibility, components named `coefficient` and `cov` may also be present and mirror `coef` and `vcov`, respectively. The `summary` component can be passed to external functions such as `modelsummary()` for further formatting, if desired.

Reproducibility and conventions:

- If convergence warnings appear, relax/tighten tolerances or cap the parameter bound (`optim.parameter1–3`) and inspect the output with `report.optim.convergence = TRUE`.
- If necessary, modify other `optim.parameter`, provide user-specified initial values, or reduce the number of nuisance parameters (e.g., provide a small set of time points contributing to estimation when using `"proportional-survival"` or `"proportional-competing-risk"`).
- Set `boot.seed` for reproducible bootstrap results.
- Match CDISC ADaM conventions via `code.event1 = 0`, `code.censoring = 1` (and, if applicable, `code.event2` for competing events).

Value

A list of class `"polyreg"` containing the fitted exposure effects and supporting results. Key components and methods include:

- `coef`: regression coefficients on the chosen effect-measure scale
- `vcov`: variance–covariance matrix of the regression coefficients
- `diagnostic.statistics`: a data frame with inverse probability weights, influence function contributions, and predicted potential outcomes
- `summary`: event-wise tidy/glance summaries used by `summary.polyreg()` or `modelsummary::modelsummary()`
- additional elements storing convergence information and internal tuning parameters.

Standard S3 methods are available: `coef.polyreg()`, `vcov.polyreg()`, `nobs.polyreg()`, and `summary.polyreg()`.

Lifecycle

[Experimental]

See Also

`cifcurve()` for KM/AJ estimators; `cifplot()` for display of a CIF; `cifpanel()` for display of multiple CIFs; ggsurvfit::ggsurvfit, patchwork::patchwork and modelsummary::modelsummary for display helpers.

Examples

```
data(diabetes.complications)
output <- polyreg(
  nuisance.model = Event(t, epsilon) ~ +1,
  exposure = "fruitq1",
  data = diabetes.complications,
  effect.measure1 = "RR",
  effect.measure2 = "RR",
  time.point = 8,
  outcome.type = "competing-risk"
)

coef(output)
```

```
vcov(output)
nobs(output)
summary(output)
```

---

polyreg-methods *Methods for polyreg objects*

Description

S3 methods to extract coefficients, variance-covariance matrix, sample size, formatted summaries, and tidy/glance/augment from objects returned by `polyreg()`.

Usage

```
## S3 method for class 'polyreg'
coef(object, ...)

## S3 method for class 'polyreg'
vcov(object, type = c("default", "sandwich", "bootstrap"), ...)

## S3 method for class 'polyreg'
nobs(object, ...)

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

## S3 method for class 'summary.polyreg'
print(x, digits = 3, ...)

effect_label.polyreg(
  x,
  event = c("event1", "event2"),
  add.time.point = TRUE,
  add.outcome = TRUE,
  add.exposure.levels = TRUE,
  add.conf = TRUE,
  add.p = TRUE,
  value.time = NULL,
  unit.time = NULL,
  digits = 2,
  p_digits = 2,
  p_cut = 0.05,
  ...
)

## S3 method for class 'polyreg'
```

```
tidy(x, event = c("event1", "event2", "both"), ...)

## S3 method for class 'polyreg'
glance(x, event = c("event1", "event2"), ...)

## S3 method for class 'polyreg'
augment(x, ...)
```

Arguments

object	A polyreg object returned by polyreg().
...	Further arguments passed to or from methods.
type	Character string; one of "default", "sandwich", or "bootstrap". When "default", the function chooses between sandwich and bootstrap variance based on the original polyreg() settings, using outcome.type, report.sandwich.conf, and report.boot.conf. (Used only by vcov.polyreg().)
x	Object to be printed or summarised. Typically a "summary.polyreg" object for print.summary.polyreg(), or a "polyreg" object for tidy.polyreg(), glance.polyreg(), augment.polyreg(), and effect_label.polyreg().
digits	Number of digits to print for parameter estimates or effect measures. Used by print.summary.polyreg() and effect_label.polyreg().
event	Character string indicating which event to extract. For effect_label.polyreg() and glance.polyreg() this is one of "event1" or "event2". For tidy.polyreg() it can also be "both" to return rows for all events.
add.time.point	Logical; if TRUE, effect_label.polyreg() appends the time point to the label (e.g., "at 5 years").
add.outcome	Logical; if TRUE, effect_label.polyreg() appends the outcome/event description (e.g., "of event 1").
add.exposure.levels	Logical; if TRUE, effect_label.polyreg() includes the exposure level in the label (e.g., treatment group).
add.conf	Logical; if TRUE, effect_label.polyreg() includes a confidence interval in the label.
add.p	Logical; if TRUE, effect_label.polyreg() includes a p-value or thresholded p-value (e.g. $p < 0.05$).
value.time	Optional numeric value overriding the time point stored in the "polyreg" object when constructing labels in effect_label.polyreg().
unit.time	Optional character string giving the time unit to display in labels constructed by effect_label.polyreg(), such as "years" or "months".
p_digits	Integer; number of digits used to format p-values in effect_label.polyreg().
p_cut	Numeric threshold used by effect_label.polyreg() to decide between printing p < p_cut and an exact p-value.

Value

- `coef.polyreg()` returns a numeric vector of regression coefficients.
- `vcov.polyreg()` returns a variance-covariance matrix.
- `nobs.polyreg()` returns the number of observations.
- `summary.polyreg()` returns a list of tidy and glance summaries by event.
- `print.summary.polyreg()` is called for its side effect of printing a formatted, modelsummary-like table to the console and returns `x` invisibly.
- `tidy.polyreg()` returns a data frame of tidy coefficients by event.
- `glance.polyreg()` returns a data frame of model-level summaries by event.
- `augment.polyreg()` returns an augmented data frame containing diagnostics, weights, and predicted CIFs.

See Also

`polyreg()` for log odds product modeling of CIFs

`prostate` *Data from a prostate cancer trial in Byer & Green (1980)*

Description

Anonymized data from a randomized clinical trial of prostate cancer published in Byer & Green (1980).

Usage

```
data(prostate)
```

Format

A data frame with 502 observations and 16 variables, including:

dtime Follow-up time in days.

status Event status ("alive", "dead - prostatic ca", "dead - other ca", "dead - heart or vascular", "dead - cerebrovascular").

rx Treatment assignment to diethylstilbestrol (DES) or a placebo.

age Age at baseline (years).

wt Weight in pounds.

pf Performance status.

hx History of cardiovascular disease.

sbp Systolic blood pressure.

dbp Diastolic blood pressure.

ekg Electrocardiogram category.

hg Hemoglobin level.

sz Size of the primary tumor.

sg Stage/grade of disease.

ap Serum acid phosphatase.

bm Bone metastases indicator.

stage Clinical stage.

sdate Start date.

patno Patient number.

Details

The dataset records follow-up for cause of death together with treatment assignment and baseline characteristics. It is used in the package documentation to illustrate stratified cumulative incidence analyses.

Source

Byer, D. P. & Green, S. B. (1980), 'Prognostic variables for survival in a randomized comparison of treatments for prostatic cancer', Bulletin du Cancer 67, 477-488

Examples

```
data(prostate)
head(prostate)
```

Index

∗ **datasets**
    diabetes.complications, 24
    prostate, 37

augment.polyreg *(*polyreg-methods*)*, 35

cifcurve, 2
cifcurve(), *4*, *11*, *14*, *21*, *23*, *25*, *27*, *34*
cifpanel, 5
cifpanel(), *5*, *15*, *19*, *23*, *25–27*, *34*
cifplot, 15
cifplot(), *3–5*, *7*, *13–15*, *19*, *25–27*, *34*
coef.polyreg *(*polyreg-methods*)*, 35

diabetes.complications, 24

effect_label.polyreg *(*polyreg-methods*)*, 35
Event, 25
extract_time_to_event, 26

ggsurvfit::ggsurvfit, *5*, *14*, *23*, *25*, *27*, *34*
glance.polyreg *(*polyreg-methods*)*, 35

modelsummary::modelsummary, *5*, *14*, *23*, *25*, *27*, *34*

nobs.polyreg *(*polyreg-methods*)*, 35

patchwork::patchwork, *5*, *14*, *23*, *25*, *27*, *34*
polyreg, 28
polyreg(), *5*, *14*, *23*, *25*, *27*, *37*
polyreg-methods, 35
print.summary.polyreg *(*polyreg-methods*)*, 35
prostate, 37

summary.polyreg *(*polyreg-methods*)*, 35

tidy.polyreg *(*polyreg-methods*)*, 35

vcov.polyreg *(*polyreg-methods*)*, 35