Package: TemporalHazard (via r-universe)

June 2, 2026

Title Temporal Parametric Hazard Modeling

Version 1.0.3.9000

URL `https://ehrlinger.github.io/temporal_hazard/`, `https://github.com/ehrlinger/temporal_hazard`

BugReports `https://github.com/ehrlinger/temporal_hazard/issues`

Description Provides native R implementations of the multiphase parametric hazard model of Blackstone, Naftel, and Turner (1986) <doi:10.1080/01621459.1986.10478314> with a focus on behavioral parity, transparent numerics, and reproducible validation against reference outputs from the original 'C'/'SAS' HAZARD program, originally developed at the University of Alabama at Birmingham (UAB). The 'SAS'/'C' code and this R package are currently developed and maintained at The Cleveland Clinic Foundation, and the R code was wholly developed at The Cleveland Clinic Foundation. The generalized temporal decomposition family extends to longitudinal mixed-effects settings (Rajeswaran et al. 2018 <doi:10.1177/0962280215623583>). The package is intentionally implemented in pure R first; performance-critical paths may later be accelerated with 'Rcpp' without changing the public interface.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

VignetteBuilder quarto

Depends R (>= 4.1.0)

Imports survival

Suggests covr, ggplot2, knitr, lintr, numDeriv, pkgdown, quarto, roxygen2, rmarkdown, scales, testthat (>= 3.0.0)

LazyData true

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 8.0.0

Repository https://ehrlinger.r-universe.dev

Date/Publication 2026-06-02 13:53:17 UTC

RemoteUrl https://github.com/ehrlinger/temporal_hazard

RemoteRef HEAD

RemoteSha 381b9a4142eb28b4985b7278701e82bd2cf7d1aa

Contents

`avc` *AVC: Atrioventricular Canal Repair*

Description

Survival data for 310 patients who underwent repair of atrioventricular septal defects (congenital heart disease) at the Cleveland Clinic between 1977 and 1993. Exhibits two identifiable hazard phases: an early post-operative risk and a constant late phase.

Usage

```
avc
```

Format

A data frame with 310 rows and 11 variables:

study Patient identifier

status NYHA functional class (1–4)

inc_surg Surgical grade of AV valve incompetence

opmos Date of operation (months since January 1967)

age Age at repair (months)

mal Malalignment indicator (0/1)

com_iv Interventricular communication indicator (0/1)

orifice Associated cardiac anomaly indicator (0/1)

dead Death indicator (1 = dead, 0 = censored)

int_dead Follow-up interval to death or last contact (months)

op_age Interaction term: opmos x age

Source

Blackstone, Naftel, and Turner (1986) doi:10.1080/01621459.1986.10478314. Cleveland Clinic Foundation.

See Also

`vignette("fitting-hazard-models")`, `vignette("prediction-visualization")`

Other datasets: `cabgkul`, `omc`, `tga`, `valves`

Examples

```
data(avc)
avc <- na.omit(avc)

# Kaplan-Meier survival
km <- survival::survfit(survival::Surv(int_dead, dead) ~ 1, data = avc)
plot(km, xlab = "Months after AVC repair", ylab = "Survival",
     main = "AVC: Kaplan-Meier survival estimate")


# Two-phase hazard fit (early CDF + constant -- what AVC supports)
fit <- hazard(
  survival::Surv(int_dead, dead) ~ 1, data = avc,
  dist = "multiphase",
  phases = list(
    early    = hzr_phase("cdf", t_half = 0.5, nu = 1, m = 1),
    constant = hzr_phase("constant")
  ),
  fit = TRUE, control = list(n_starts = 5, maxit = 1000)
)
summary(fit)
```

`cabgkul` *CABGKUL: Primary Isolated Coronary Artery Bypass Grafting (KU Leuven)*

Description

Survival data for 5,880 patients who underwent primary isolated CABG at KU Leuven, Belgium, between 1971 and July 1987. The simplest dataset structure (intercept-only, right-censored) with large sample size exercising all three temporal hazard phases.

Usage

```
cabgkul
```

Format

A data frame with 5880 rows and 2 variables:

int_dead Follow-up interval to death or last contact (months)

dead Death indicator (1 = dead, 0 = censored)

Source

KU Leuven cardiac surgery registry. Primary benchmark dataset for C binary parity testing.

See Also

`vignette("fitting-hazard-models")`

Other datasets: `avc`, `omc`, `tga`, `valves`

Examples

```
data(cabgkul)

# Kaplan-Meier survival
km <- survival::survfit(survival::Surv(int_dead, dead) ~ 1, data = cabgkul)
plot(km, xlab = "Months after CABG", ylab = "Survival",
     main = "CABGKUL: Kaplan-Meier survival (n = 5,880)")


# Single-phase Weibull fit with parametric overlay
fit <- hazard(survival::Surv(int_dead, dead) ~ 1, data = cabgkul,
              dist = "weibull", theta = c(mu = 0.10, nu = 1.0), fit = TRUE)
t_grid <- seq(0.01, max(cabgkul$int_dead) * 0.9, length.out = 200)
surv   <- predict(fit, newdata = data.frame(time = t_grid),
                  type = "survival")
plot(km, xlab = "Months after CABG", ylab = "Survival",
     main = "CABGKUL: Weibull vs. Kaplan-Meier")
lines(t_grid, surv, col = "blue", lwd = 2)
legend("bottomleft", c("KM", "Weibull"), col = c("black", "blue"),
       lty = 1, lwd = c(1, 2))
```

`coef.hazard` *Extract coefficients from hazard model*

Description

Extract coefficients from hazard model

Usage

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

Arguments

`object`	A hazard object.
`...`	Unused; for S3 compatibility.

Value

A named numeric vector of fitted parameter estimates, or `NULL` if the model has not been fitted (`fit = FALSE`).

Examples

```
fit <- hazard(time = rexp(30, 0.5), status = rep(1L, 30),
              theta = c(0.3, 1.0), dist = "weibull", fit = TRUE)
coef(fit)
```

`hazard` *Build and optionally fit a hazard model*

Description

Creates a `hazard` object and optionally fits it via maximum likelihood. This mirrors the argument-oriented workflow of the legacy HAZARD C/SAS implementation: supply starting values in `theta` and the function will optimize to produce fitted estimates.

Usage

```
hazard(
  formula = NULL,
  data = NULL,
  time = NULL,
  status = NULL,
  time_lower = NULL,
  time_upper = NULL,
  x = NULL,
  time_windows = NULL,
  theta = NULL,
  dist = "weibull",
  phases = NULL,
  fit = FALSE,
  weights = NULL,
  control = list(),
  ...
)
```

Arguments

`formula`	Optional formula of the form `Surv(time, status) ~ predictors`. When provided, overrides direct time/status/x arguments and extracts from data. Example: `hazard(Surv(time, status) ~ x1 + x2, data = df, dist = "weibull", fit = TRUE)`.
`data`	Optional data frame containing variables referenced in formula.
`time`	Numeric follow-up time vector.
`status`	Numeric or logical event indicator vector.
`time_lower`	Optional numeric lower bound vector for censoring intervals. Used when `status == 2` (interval-censored); defaults to `time` if NULL.

`time_upper`	Optional numeric upper bound vector for censoring intervals. Used when `status %in% c(-1, 2)`; defaults to `time` if NULL.
`x`	Optional design matrix (or data frame coercible to matrix).
`time_windows`	Optional numeric vector of strictly positive cut points for piecewise time-varying coefficients. When provided, each predictor column in `x` is expanded into one column per time window so each window gets its own coefficient.
`theta`	Optional numeric coefficient vector (starting values for optimization).
`dist`	Character baseline distribution label (default "weibull"). Use `"multiphase"` for N-phase additive hazard models (requires `phases`).
`phases`	Optional named list of `hzr_phase()` objects specifying the phases for a multiphase model (`dist = "multiphase"`). See Examples.
`fit`	Logical; if TRUE, fit the model via maximum likelihood (default FALSE).
`weights`	Optional numeric vector of observation weights (non-negative). Each observation's log-likelihood contribution is multiplied by its weight. Use for severity-weighted repeated events. Default `NULL` (unit weights). Implements the SAS `WEIGHT` statement.
`control`	Named list of control options (see Details).
`...`	Additional named arguments retained for parity with legacy calling conventions.

Details

Control parameters:

- `maxit`: Maximum iterations (default 1000)
- `reltol`: Relative parameter change tolerance (default 1e-5)
- `abstol`: Absolute gradient norm tolerance (default 1e-6)
- `method`: Optimization method: "bfgs" or "nm" (default "bfgs")
- `condition`: Condition number control (default 14)
- `nocov`, `nocor`: Suppress covariance/correlation output (legacy; no-op in M2)

Censoring status coding:

- 1: Exact event at time
- 0: Right-censored at time
- -1: Left-censored with upper bound at time_upper \(or time\)
- 2: Interval-censored in the interval \(time_lower, time_upper\)

Time-varying coefficients:

- If `time_windows` is supplied, predictors are expanded to piecewise window interactions so each window has its own coefficient vector.
- This is implemented as design-matrix expansion, so the existing likelihood engines remain unchanged.

Value

An object of class `hazard`, a named list with components: `call` (the matched call), `spec` (model specification: `dist`, `control`, `time_windows`, `phases`), `data` (input data: `time`, `status`, `x`, `weights`, etc.), `fit` (optimisation results: `theta`, `objective`, `converged`, `se`, `vcov`, `counts`, `message`; all `NULL` when `fit = FALSE`), and `engine` (implementation tag, `"native-r-m2"`).

References

Blackstone EH, Naftel DC, Turner ME Jr. The decomposition of time-varying hazard into phases, each incorporating a separate stream of concomitant information. *J Am Stat Assoc.* 1986;81(395):615–624. doi:10.1080/01621459.1986.10478314

Rajeswaran J, Blackstone EH, Ehrlinger J, Li L, Ishwaran H, Parides MK. Probability of atrial fibrillation after ablation: Using a parametric nonlinear temporal decomposition mixed effects model. *Stat Methods Med Res.* 2018;27(1):126–141. doi:10.1177/0962280215623583

See Also

`predict.hazard()` for survival/cumulative-hazard predictions, `summary.hazard()` for model summaries, `hzr_phase()` for specifying multiphase temporal shapes.

Vignettes with worked examples: `vignette("fitting-hazard-models")` — single-phase through multiphase fitting, `vignette("prediction-visualization")` — prediction types and decomposed hazard plots, `vignette("inference-diagnostics")` — bootstrap CIs and model diagnostics.

Examples

```
# -- Univariable Weibull -----------------------------------------------
set.seed(1)
time   <- rexp(50, rate = 0.3)
status <- sample(0:1, 50, replace = TRUE, prob = c(0.3, 0.7))
fit <- hazard(time = time, status = status,
              theta = c(0.3, 1.0), dist = "weibull", fit = TRUE)
summary(fit)

# -- Formula interface with covariates --------------------------------
set.seed(1001)
n   <- 180
dat <- data.frame(
  time   = rexp(n, rate = 0.35) + 0.05,
  status = rbinom(n, size = 1, prob = 0.6),
  age    = rnorm(n, mean = 62, sd = 11),
  nyha   = sample(1:4, n, replace = TRUE),
  shock  = rbinom(n, size = 1, prob = 0.18)
)

fit2 <- hazard(
  survival::Surv(time, status) ~ age + nyha + shock,
  data    = dat,
  theta   = c(mu = 0.25, nu = 1.10, beta1 = 0, beta2 = 0, beta3 = 0),
  dist    = "weibull",
```

```
  fit     = TRUE,
  control = list(maxit = 300)
)
summary(fit2)


# -- Parametric survival with Kaplan-Meier overlay -----------------
if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)

  # Parametric curve on a fine grid at median covariate profile
  t_grid   <- seq(0.05, max(dat$time), length.out = 80)
  curve_df <- data.frame(
    time = t_grid, age = median(dat$age), nyha = 2, shock = 0
  )
  curve_df$survival <- predict(fit2, newdata = curve_df,
                               type = "survival") * 100

  # Kaplan-Meier empirical overlay
  km    <- survival::survfit(survival::Surv(time, status) ~ 1, data = dat)
  km_df <- data.frame(time = km$time, survival = km$surv * 100)

  ggplot() +
    geom_step(data = km_df, aes(time, survival, colour = "Kaplan-Meier")) +
    geom_line(data = curve_df, aes(time, survival,
                                   colour = "Parametric (Weibull)")) +
    scale_colour_manual(
      values = c("Parametric (Weibull)" = "#0072B2",
                 "Kaplan-Meier"         = "#D55E00")
    ) +
    scale_y_continuous(limits = c(0, 100)) +
    labs(x = "Months after surgery", y = "Freedom from death (%)",
         colour = NULL) +
    theme_minimal() +
    theme(legend.position = "bottom")
}



# -- Multiphase model (two phases) ---------------------------------
fit_mp <- hazard(
  survival::Surv(time, status) ~ 1,
  data   = dat,
  dist   = "multiphase",
  phases = list(
    early = hzr_phase("cdf", t_half = 0.5, nu = 2, m = 0,
                      fixed = "shapes"),
    late  = hzr_phase("cdf", t_half = 5,   nu = 1, m = 0,
                      fixed = "shapes")
  ),
  fit     = TRUE,
  control = list(n_starts = 5, maxit = 1000)
)
```

```
summary(fit_mp)

# -- Per-phase decomposed cumulative hazard ------------------------
if (requireNamespace("ggplot2", quietly = TRUE)) {
  t_grid <- seq(0.01, max(dat$time), length.out = 100)
  decomp <- predict(fit_mp, newdata = data.frame(time = t_grid),
                    type = "cumulative_hazard", decompose = TRUE)

  df_long <- data.frame(
    time = rep(decomp$time, 3),
    cumhaz = c(decomp$total, decomp$early, decomp$late),
    component = rep(c("Total", "Early (cdf)", "Late (cdf)"),
                    each = nrow(decomp))
  )
  df_long$component <- factor(df_long$component,
    levels = c("Total", "Early (cdf)", "Late (cdf)"))

  ggplot2::ggplot(df_long,
    ggplot2::aes(x = time, y = cumhaz, colour = component,
                 linewidth = component)) +
    ggplot2::geom_line() +
    ggplot2::scale_colour_manual(values = c(
      "Total" = "black", "Early (cdf)" = "#0072B2",
      "Late (cdf)" = "#D55E00"
    )) +
    ggplot2::scale_linewidth_manual(values = c(
      "Total" = 1.2, "Early (cdf)" = 0.6, "Late (cdf)" = 0.6
    )) +
    ggplot2::labs(
      x = "Time", y = "Cumulative hazard H(t)",
      colour = NULL, linewidth = NULL,
      title = "Multiphase decomposition: early + late"
    ) +
    ggplot2::theme_minimal() +
    ggplot2::theme(legend.position = "bottom")
}
```

`hzr_argument_mapping` *Legacy HAZARD to TemporalHazard argument mapping*

Description

Returns a formal mapping table that defines how legacy SAS HAZARD/C-style inputs map to `hazard(...)` arguments in this package.

Usage

```
hzr_argument_mapping(include_planned = TRUE)
```

Arguments

`include_planned`	Logical; if `FALSE`, only rows marked as implemented are returned.

Value

A data frame with one row per mapping rule.

Examples

```
hzr_argument_mapping()
hzr_argument_mapping(include_planned = FALSE)
```

`hzr_bootstrap` *Bootstrap resampling for hazard model coefficients*

Description

Resample data with replacement, refit the hazard model on each replicate, and accumulate coefficient distributions. Returns a tidy data frame of per-replicate estimates with summary statistics. This is the R equivalent of the SAS `bootstrap.hazard.sas` macro.

Usage

```
hzr_bootstrap(
  object,
  n_boot = 200L,
  fraction = 1,
  seed = NULL,
  verbose = FALSE
)

## S3 method for class 'hzr_bootstrap'
print(x, digits = 4, ...)
```

Arguments

`object`	A fitted `hazard` object (with `fit` = `TRUE`).
`n_boot`	Integer: number of bootstrap replicates (default 200).
`fraction`	Numeric in (0, 1]: fraction of data to sample per replicate (default 1.0 for full bootstrap; < 1 for bagging).
`seed`	Optional integer random seed for reproducibility. When supplied, `set.seed(seed)` is called at function entry, jumping the global RNG to the seeded state; it is not restored on exit. Pass `NULL` (the default) to skip the `set.seed()` call and start from the caller's current RNG state. Note that the bootstrap consumes random numbers either way, so the global RNG state will advance during the call – seed = `NULL` avoids the *reset* at entry, not the advance during resampling.

`verbose`	Logical; if `TRUE`, print progress every 50 replicates.
`x`	An `hzr_bootstrap` object.
`digits`	Number of decimal places for formatting.
`...`	Additional arguments (ignored).

Value

A list with class `"hzr_bootstrap"` containing:

replicates Data frame with columns `replicate`, `parameter`, and `estimate` – one row per parameter per successful replicate.

summary Data frame with columns `parameter`, `n`, `pct`, `mean`, `sd`, `min`, `max`, `ci_lower`, `ci_upper` – one row per parameter.

n_success Number of successfully converged replicates.

n_failed Number of replicates that failed to converge.

See Also

`hazard()` for model fitting, `vcov.hazard()` for Hessian-based standard errors.

Examples

```
data(avc)
avc <- na.omit(avc)
fit <- hazard(
  survival::Surv(int_dead, dead) ~ age + mal,
  data  = avc,
  dist  = "weibull",
  theta = c(mu = 0.01, nu = 0.5, 0, 0),
  fit   = TRUE
)
bs <- hzr_bootstrap(fit, n_boot = 50, seed = 123)
print(bs)
```

`hzr_calibrate` *Calibrate a continuous variable against an outcome*

Description

Group a continuous covariate into quantile bins, compute the event probability (or hazard rate) per bin, and apply a link transform (logit, Gompertz, or Cox). This is the R equivalent of the SAS `logit.sas` and `logitgr.sas` macros.

Usage

```
hzr_calibrate(
  x,
  event,
  groups = 10L,
  by = NULL,
  link = c("logit", "gompertz", "cox"),
  time = NULL
)
```

Arguments

`x`	Numeric vector: the continuous covariate to calibrate.
`event`	Numeric vector: event indicator (1 = event, 0 = no event).
`groups`	Integer: number of quantile bins (default 10).
`by`	Optional factor or character vector for stratified calibration (SAS `logitgr.sas` functionality). If provided, calibration is computed within each stratum. Default `NULL` (no stratification).
`link`	Character: transform to apply to event probabilities. One of `"logit"` (default), `"gompertz"` (complementary log-log), or `"cox"`.
`time`	Optional numeric vector: follow-up time, required when `link = "cox"`. The Cox link computes $\log(\text{events}/\sum \text{time})$ (constant hazard rate).

Details

Use this function before model entry to assess whether the covariate relationship with the outcome is approximately linear on the link scale. If the transformed probabilities are roughly linear against the group means, the covariate can enter the model untransformed. Curvature suggests a transformation (log, quadratic) may improve fit.

Value

A data frame with one row per group (or per group-by-stratum combination) and columns:

group Integer group label.

by Stratum level (only present when `by` is provided).

n Number of observations in the group.

events Number of events.

mean Mean of `x` within the group.

min Minimum of `x` within the group.

max Maximum of `x` within the group.

prob Event probability (events / n), or for Cox link: events / sum(time).

link_value Transformed probability on the chosen link scale.

See Also

`hzr_deciles()` for model-based calibration after fitting.

Examples

```
data(avc)
avc <- na.omit(avc)

# Logit calibration of age
cal <- hzr_calibrate(avc$age, avc$dead, groups = 10)
print(cal)


if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)
  ggplot(cal, aes(mean, link_value)) +
    geom_point(size = 3) +
    geom_smooth(method = "lm", formula = y ~ x, se = FALSE,
                linetype = "dashed") +
    labs(x = "Age at repair (months)", y = "Logit(P(death))") +
    theme_minimal()
}
```

`hzr_clamp_prob` *Clamp probabilities away from 0 and 1*

Description

Clamp probabilities away from 0 and 1

Usage

```
hzr_clamp_prob(p, eps = 1e-12)
```

Arguments

`p`	Numeric vector of probabilities.
`eps`	Small positive tolerance.

Value

Numeric vector bounded to `[eps, 1 - eps]`.

Examples

```
hzr_clamp_prob(c(0, 0.5, 1))
```

`hzr_competing_risks` *Competing risks cumulative incidence*

Description

Compute cumulative incidence functions for multiple competing event types using the Aalen-Johansen estimator with Greenwood variance. This is the R equivalent of the SAS `markov.sas` macro.

Usage

```
hzr_competing_risks(time, event)

## S3 method for class 'hzr_competing_risks'
print(x, digits = 4, ...)
```

Arguments

`time`	Numeric vector of follow-up times.
`event`	Integer vector of event type indicators: 0 = censored, 1 = event type 1, 2 = event type 2, etc.
`x`	An `hzr_competing_risks` object.
`digits`	Number of decimal places for formatting.
`...`	Additional arguments (ignored).

Details

Unlike the naive 1 - KM estimator (which overestimates incidence when competing risks exist), this provides the correct marginal cumulative incidence for each event type.

Value

A data frame with one row per unique event time and columns:

time Event time.

n_risk Number at risk.

n_event_1, n_event_2, ... Events of each type at this time.

n_censor Number censored at this time.

surv Overall event-free survival (freedom from all events).

incid_1, incid_2, ... Cumulative incidence for each event type.

se_surv Standard error of overall survival.

se_1, se_2, ... Standard error of each cumulative incidence.

See Also

`hzr_kaplan()` for single-event survival estimation.

Examples

```
data(valves)
valves_cc <- na.omit(valves)
# Combine death and PVE into a competing risks event variable
# 0 = censored, 1 = death, 2 = PVE
event_cr <- ifelse(valves_cc$dead == 1, 1L,
                   ifelse(valves_cc$pve == 1, 2L, 0L))
time_cr <- pmin(valves_cc$int_dead, valves_cc$int_pve)
cr <- hzr_competing_risks(time_cr, event_cr)
head(cr)
```

`hzr_deciles` *Decile-of-risk calibration*

Description

Partition observations into groups (default 10) by predicted risk and compare observed vs. expected event counts in each group. Good calibration means the two track each other across the risk spectrum.

Usage

```
hzr_deciles(object, time, groups = 10L, status = NULL, event_time = NULL)
```

Arguments

`object`	A fitted `hazard` object (with `fit = TRUE`).
`time`	Numeric scalar: the time point at which to evaluate predicted survival / cumulative hazard. For example, `time = 12` evaluates 12-month predictions.
`groups`	Integer: number of risk groups (default 10 for deciles).
`status`	Optional numeric vector of event indicators (1 = event, 0 = censored). If `NULL` (default), extracted from the fitted object's stored data.
`event_time`	Optional numeric vector of observed event/censoring times. If `NULL` (default), extracted from the fitted object.

Details

This implements the workflow of the SAS `deciles.hazard.sas` macro. Patients are ranked by predicted cumulative hazard at a specified time point, grouped into quantile bins, and each bin is tested with a chi-square goodness-of-fit statistic. Subjects censored before the requested horizon are excluded from the observed-vs-expected comparison.

Value

A data frame with one row per risk group and columns:

group Integer group label (1 = lowest risk).

n Number of observations in the group.

events Observed event count by the requested horizon (event indicator = 1 and event time <= `time`).

expected Expected event count by the requested horizon, computed as the sum of individual event probabilities $(1 - S(time))$.

observed_rate Observed event rate (events / n).

expected_rate Expected event rate (expected / n).

chi_sq Chi-square contribution: (events - expected)^2 / expected.

p_value Upper-tail p-value from the chi-square test for this group (1 df).

mean_survival Mean predicted survival probability in the group.

mean_cumhaz Mean predicted cumulative hazard in the group.

An attribute `"overall"` is attached with the overall chi-square statistic, degrees of freedom, and p-value.

See Also

`predict.hazard()` for the prediction types used internally.

Examples

```
data(avc)
avc <- na.omit(avc)
fit <- hazard(
  survival::Surv(int_dead, dead) ~ age + mal,
  data  = avc,
  dist  = "weibull",
  theta = c(mu = 0.01, nu = 0.5, beta_age = 0, beta_mal = 0),
  fit   = TRUE
)
cal <- hzr_deciles(fit, time = 120)
print(cal)
```

`hzr_decompos` *Generalized temporal decomposition*

Description

Computes the cumulative distribution $G(t)$, density $g(t)$, and hazard $h(t) = g(t)/(1 - G(t))$ for the parametric family defined by half-life, time exponent, and shape. This single function generates all temporal phase shapes used in multiphase hazard models.

Usage

```
hzr_decompos(time, t_half, nu, m)
```

Arguments

`time`	Numeric vector of times (must be > 0).
`t_half`	Half-life: time at which $G(t_{1/2}) = 0.5$. Must be > 0.
`nu`	Time exponent controlling rate dynamics. SAS early: `NU`. SAS late: relates to `GAMMA/ETA`.
`m`	Shape exponent controlling the distributional form. SAS early: `M`. SAS late: relates to `GAMMA/ALPHA`.

Value

A named list with three numeric vectors, each the same length as `time`:

G Cumulative distribution $G(t) \in [0, 1]$.

g Density $g(t) = dG/dt \geq 0$. The "early" phase temporal pattern.

h Hazard $h(t) = g(t)/(1 - G(t)) \geq 0$. The "late" phase temporal pattern.

Parameter mapping from SAS/C HAZARD

The original C code used separate parameterizations for early (DELTA, RHO/THALF, NU, M) and late (TAU, GAMMA, ALPHA, ETA) phases. Both collapse onto the three parameters here. See `hzr_argument_mapping()` for the full translation table.

Valid parameter combinations

Six cases are defined by the signs of `nu` and `m`:

Case	Sign	Behavior
1	m > 0, nu > 0	Standard sigmoidal
1L	m = 0, nu > 0	Exponential-like (Weibull CDF)
2	m < 0, nu > 0	Heavy-tailed
2L	m < 0, nu = 0	Exponential decay
3	m > 0, nu < 0	Bounded cumulative
3L	m = 0, nu < 0	Bounded exponential

The combination m < 0 **and** nu < 0 is undefined and raises an error.

References

Blackstone EH, Naftel DC, Turner ME Jr. The decomposition of time-varying hazard into phases, each incorporating a separate stream of concomitant information. *J Am Stat Assoc.* 1986;81(395):615–624. doi:10.1080/01621459.1986.10478314

Rajeswaran J, Blackstone EH, Ehrlinger J, Li L, Ishwaran H, Parides MK. Probability of atrial fibrillation after ablation: Using a parametric nonlinear temporal decomposition mixed effects model. *Stat Methods Med Res.* 2018;27(1):126–141. doi:10.1177/0962280215623583

See Also

`hzr_phase_cumhaz()` for the phase-level cumulative hazard contribution, `hzr_argument_mapping()` for SAS/C parameter mapping, `hzr_phase()` for specifying phases in `hazard()` models.

`vignette("mf-mathematical-foundations")` for the full derivation.

Examples

```
t_grid <- seq(0.1, 10, by = 0.1)

# Case 1: standard sigmoidal (m > 0, nu > 0)
d1 <- hzr_decompos(t_grid, t_half = 3, nu = 2, m = 1)
plot(t_grid, d1$G, type = "l", main = "CDF (m=1, nu=2)")

# Case 1L: Weibull-like (m = 0, nu > 0)
d1L <- hzr_decompos(t_grid, t_half = 3, nu = 2, m = 0)

# Case 2: heavy-tailed (m < 0, nu > 0)
d2 <- hzr_decompos(t_grid, t_half = 3, nu = 2, m = -1)
```

`hzr_decompos_g3` *Late-phase (G3) temporal decomposition*

Description

Computes the cumulative intensity $G_3(t)$ and its derivative $g_3(t) = dG_3/dt$ for the late-phase parametric family used in the original Blackstone C/SAS HAZARD code. Unlike `hzr_decompos()` (which computes the early-phase G1 – a bounded CDF), this function can produce **unbounded** values, making it suitable for modelling increasing late risk.

Usage

```
hzr_decompos_g3(time, tau, gamma, alpha, eta)
```

Arguments

`time`	Numeric vector of times (must be > 0).
`tau`	Positive scalar scale parameter.
`gamma`	Positive scalar time exponent.
`alpha`	Non-negative scalar shape parameter (0 selects limiting case).
`eta`	Positive scalar outer exponent.

Value

A named list with two numeric vectors, each the same length as `time`:

G3 Cumulative intensity $G_3(t) \geq 0$ (may exceed 1).

g3 Derivative $g_3(t) = dG_3/dt \geq 0$.

Mathematical form

When $\alpha > 0$:

$$G_3(t) = \left(\left((t/\tau)^\gamma + 1\right)^{1/\alpha} - 1\right)^\eta$$

When $\alpha = 0$ (limiting exponential case):

$$G_3(t) = \left(\exp\left((t/\tau)^\gamma\right) - 1\right)^\eta$$

Parameter mapping from SAS/C HAZARD

SAS name	R argument	Role
TAU	`tau`	Scale (time at which $(t/\tau) = 1$)
GAMMA	`gamma`	Power exponent on t/τ
ALPHA	`alpha`	Shape (0 = exponential limiting case)
ETA	`eta`	Outer power exponent

References

Blackstone EH, Naftel DC, Turner ME Jr. The decomposition of time-varying hazard into phases, each incorporating a separate stream of concomitant information. *J Am Stat Assoc*. 1986;81(395):615–624. doi:10.1080/01621459.1986.10478314

See Also

`hzr_decompos()` for the early-phase (G1) decomposition, `hzr_phase_cumhaz()` for phase-level cumulative hazard helpers.

Examples

```
t_grid <- seq(0.1, 10, by = 0.1)

# Weibull-like: alpha = 1 gives G3(t) = (t/tau)^(gamma*eta)
d <- hzr_decompos_g3(t_grid, tau = 1, gamma = 3, alpha = 1, eta = 1)
plot(t_grid, d$G3, type = "l", main = "G3: power law (gamma=3)")

# General case with alpha > 0
d2 <- hzr_decompos_g3(t_grid, tau = 2, gamma = 2, alpha = 0.5, eta = 1)
```

`hzr_gof` *Goodness-of-fit: observed vs. predicted events*

Description

Compare a fitted hazard model against the nonparametric Kaplan-Meier estimate by computing observed and expected (parametric) event counts at each distinct event time. This is the R equivalent of the SAS `hazplot.sas` macro and implements the conservation-of-events diagnostic.

Usage

```
hzr_gof(object, time_grid = NULL)
```

Arguments

`object`	A fitted `hazard` object (with `fit = TRUE`).
`time_grid`	Optional numeric vector of time points at which to evaluate the parametric model. If `NULL` (default), uses the sorted unique event times from the fitted data.

Details

At each observed event time the function computes:

- The Kaplan-Meier survival and cumulative hazard.
- The parametric survival and cumulative hazard from the fitted model (and per-phase components for multiphase models).
- Cumulative observed events vs. cumulative expected events (sum of individual cumulative hazards for those exiting the risk set at each time).
- The running residual (expected minus observed).

Perfect model fit implies the expected and observed event counts track each other (residual near zero). This is the conservation-of-events principle.

Value

A data frame with one row per time point and columns:

time Evaluation time.

n_risk Number at risk (Kaplan-Meier).

n_event Number of events at this time.

n_censor Number censored at this time.

km_surv Kaplan-Meier survival estimate.

km_cumhaz Kaplan-Meier cumulative hazard ($-\log(\text{km\_surv})$).

par_surv Parametric survival from the fitted model.

par_cumhaz Parametric cumulative hazard.

cum_observed Cumulative observed events to this time.

cum_expected Cumulative expected events (sum of individual cumulative hazards for observations exiting the risk set).

residual Expected minus observed (`cum_expected - cum_observed`).

For multiphase models, additional columns are appended for each phase: `par_cumhaz_<phase>`.

An attribute `"summary"` is attached with scalar diagnostics: total observed events, total expected events, and the final residual.

See Also

`hzr_deciles()` for decile-of-risk calibration, `predict.hazard()` for prediction types.

Examples

```
data(avc)
avc <- na.omit(avc)
fit <- hazard(
  survival::Surv(int_dead, dead) ~ age + mal,
  data  = avc,
  dist  = "weibull",
  theta = c(mu = 0.01, nu = 0.5, beta_age = 0, beta_mal = 0),
  fit   = TRUE
)
gof <- hzr_gof(fit)
print(gof)

# Plot observed vs expected events
if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)
  ggplot(gof, aes(x = time)) +
    geom_line(aes(y = cum_observed), colour = "#D55E00") +
    geom_line(aes(y = cum_expected), colour = "#0072B2") +
    labs(x = "Time", y = "Cumulative events") +
    theme_minimal()
}
```

`hzr_kaplan` *Kaplan-Meier survival with exact logit confidence limits*

Description

Compute the product-limit (Kaplan-Meier) survival estimate with logit-transformed confidence limits that respect the $[0, 1]$ boundary. This is the R equivalent of the SAS `kaplan.sas` macro.

Usage

```
hzr_kaplan(time, status, conf_level = 0.95, event_only = TRUE)
```

Arguments

`time`	Numeric vector of follow-up times.
`status`	Numeric event indicator (1 = event, 0 = censored).
`conf_level`	Confidence level for the interval (default 0.95). The SAS default of 0.68268948 corresponds to a 1-SD interval.
`event_only`	Logical; if `TRUE` (default), only return rows at event times (where `n_event` > `0`). If `FALSE`, return rows at all times reported by `survival::survfit()` (events and censoring times).

Details

The standard Greenwood confidence interval can exceed $[0, 1]$ in the tails. The logit-transformed interval avoids this by working on the log-odds scale:

$$\mathrm{CL}_{\text{lower}} = S/\big(S + (1 - S)\,\exp(z_\alpha\,\mathrm{SI})\big)$$
$$\mathrm{CL}_{\text{upper}} = S/\big(S + (1 - S)\,\exp(-z_\alpha\,\mathrm{SI})\big)$$

where $\mathrm{SI} = \sqrt{V_P - 1}/(1 - S)$, V_P is the cumulative Greenwood variance product, and z_α is the normal quantile for the requested confidence level.

Value

A data frame with one row per time point and columns:

time Event/censoring time.

n_risk Number at risk at start of interval.

n_event Number of events at this time.

n_censor Number censored at this time.

survival Kaplan-Meier survival estimate.

std_err Standard error of survival (Greenwood).

cl_lower Lower confidence limit (logit-transformed).

cl_upper Upper confidence limit (logit-transformed).

cumhaz Cumulative hazard $= -\log(S)$.

hazard Interval hazard rate $= \log(S_{t-1}/S_t)/\Delta t$.

density Probability density estimate $= (S_{t-1} - S_t)/\Delta t$.

life Restricted mean survival time (area under curve to this time).

See Also

`hzr_gof()` for parametric vs. nonparametric comparison.

Examples

```
data(cabgkul)
km <- hzr_kaplan(cabgkul$int_dead, cabgkul$dead)
head(km)


if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)
  ggplot(km, aes(time)) +
    geom_step(aes(y = survival * 100)) +
    geom_ribbon(aes(ymin = cl_lower * 100, ymax = cl_upper * 100),
                stat = "identity", alpha = 0.2) +
    labs(x = "Months", y = "Survival (%)") +
    theme_minimal()
}
```

`hzr_log1mexp` *Numerically stable log(1 - exp(-x)) for x > 0*

Description

Numerically stable log(1 - exp(-x)) for x > 0

Usage

```
hzr_log1mexp(x)
```

Arguments

`x`	Numeric vector with positive values.

Value

Numeric vector with element-wise `log(1 - exp(-x))`.

Examples

```
hzr_log1mexp(c(0.01, 0.5, 5))
```

`hzr_log1pexp` *Numerically stable log(1 + exp(x))*

Description

Numerically stable log(1 + exp(x))

Usage

```
hzr_log1pexp(x)
```

Arguments

`x`	Numeric vector.

Value

Numeric vector with element-wise `log(1 + exp(x))`.

Examples

```
hzr_log1pexp(c(-50, 0, 50))
```

`hzr_nelson` *Wayne Nelson cumulative hazard estimator with lognormal confidence limits*

Description

Compute the Nelson-Aalen cumulative hazard estimate with lognormal confidence limits. Supports weighted events for severity-adjusted analyses of repeated/recurrent events. This is the R equivalent of the SAS `nelsonl.sas` macro.

Usage

```
hzr_nelson(time, event, weight = NULL, conf_level = 0.95)

## S3 method for class 'hzr_nelson'
print(x, digits = 4, ...)
```

Arguments

`time`	Numeric vector of follow-up times.
`event`	Numeric event indicator (1 = event, 0 = censored).
`weight`	Optional numeric vector of event weights (default 1). Weights are applied only to events (censored observations contribute zero weight). Use for severity-weighted repeated events.
`conf_level`	Confidence level for the interval (default 0.95).
`x`	An `hzr_nelson` object.
`digits`	Number of decimal places for formatting.
`...`	Additional arguments (ignored).

Details

Unlike `survival::survfit()` which uses the Breslow estimator with Greenwood variance, this function uses the Wayne Nelson estimator with lognormal confidence limits that are always non-negative.

Value

A data frame with one row per unique event time and columns:

time Event time.

n_risk Number at risk.

n_event Number of events at this time.

weight_sum Sum of event weights at this time.

cumhaz Nelson cumulative hazard estimate.

std_err Standard error.

cl_lower Lower lognormal confidence limit.

cl_upper Upper lognormal confidence limit.

hazard Interval hazard rate.

cum_events Cumulative (weighted) event count.

See Also

`hzr_kaplan()` for survival estimation.

Examples

```
data(cabgkul)
nel <- hzr_nelson(cabgkul$int_dead, cabgkul$dead)
head(nel)
```

`hzr_phase` *Specify a single hazard phase*

Description

Creates an `hzr_phase` object describing one term in a multiphase additive cumulative hazard model. Pass a list of these to the `phases` argument of `hazard()` when `dist = "multiphase"`.

Usage

```
hzr_phase(
  type = c("cdf", "hazard", "constant", "g3"),
  t_half = 1,
  nu = 1,
  m = 0,
  tau = 1,
  gamma = 1,
  alpha = 1,
  eta = 1,
  formula = NULL,
  fixed = character(0)
)

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

Arguments

`type`	Character; phase type: `"cdf"`, `"hazard"`, `"g3"`, or `"constant"`.
`t_half`	Positive scalar; initial half-life (time at which $G(t_{1/2}) = 0.5$). Used for `"cdf"` and `"hazard"` phases. SAS early: `THALF/RHO`.
`nu`	Numeric scalar; initial time exponent. Used for `"cdf"` and `"hazard"` phases. SAS early: `NU`.
`m`	Numeric scalar; initial shape exponent. Used for `"cdf"` and `"hazard"` phases. SAS early: `M`.
`tau`	Positive scalar; scale parameter for `"g3"` phases. SAS late: `TAU`.
`gamma`	Positive scalar; time exponent for `"g3"` phases. SAS late: `GAMMA`.
`alpha`	Non-negative scalar; shape parameter for `"g3"` phases. When `alpha > 0`, the generic G3 formula is used; `alpha = 0` gives the exponential limiting case. SAS late: `ALPHA`.
`eta`	Positive scalar; outer exponent for `"g3"` phases. SAS late: `ETA`.
`formula`	Optional one-sided formula (e.g. `~ age + nyha`) for phase-specific covariates. When `NULL` (default), the phase inherits the global formula from `hazard()`.

`fixed`	Character vector naming shape parameters to hold fixed during optimization. Valid names for `"cdf"`/`"hazard"`: `"t_half"`, `"nu"`, `"m"`, or `"shapes"` (shorthand for all three). Valid names for `"g3"`: `"tau"`, `"gamma"`, `"alpha"`, `"eta"`, or `"shapes"` (shorthand for all four). Fixed parameters are held at their starting values; only `mu` (and covariates) are estimated. Ignored for `"constant"` phases. This mirrors the SAS/C HAZARD workflow where shapes are typically fixed and only scale parameters are estimated.
`x`	An `hzr_phase` object (for `print.hzr_phase()`).
`...`	Additional arguments (ignored).

Value

An S3 object of class `"hzr_phase"` with elements:

type Phase type string.

t_half Initial half-life (cdf/hazard phases).

nu Initial time exponent (cdf/hazard phases).

m Initial shape exponent (cdf/hazard phases).

tau Scale parameter (g3 phases).

gamma Time exponent (g3 phases).

alpha Shape parameter (g3 phases).

eta Outer exponent (g3 phases).

formula Phase-specific formula or `NULL`.

fixed Character vector of fixed parameter names (may be empty).

Phase types

`"cdf"` Early risk that resolves over time. $\Phi(t) = G(t)$, bounded $[0, 1]$. SAS equivalent: Early / G1 phase.

`"hazard"` Late or aging risk that accumulates. $\Phi(t) = -\log(1 - G(t))$, monotone increasing. SAS equivalent: Late / G3 phase.

`"constant"` Flat background hazard rate. $\Phi(t) = t$. No shape parameters are estimated. SAS equivalent: Constant / G2 phase.

See Also

`hazard()` for fitting multiphase models, `hzr_decompos()` for the underlying parametric family, `hzr_phase_cumhaz()` and `hzr_phase_hazard()` for computing $\Phi(t)$ and $\phi(t)$ from these specifications.

`vignette("fitting-hazard-models")` for multiphase fitting examples, `vignette("mf-mathematical-foundations")` for the mathematical framework.

Examples

```
# Classic 3-phase Blackstone pattern
early <- hzr_phase("cdf",      t_half = 0.5, nu = 2, m = 0)
const <- hzr_phase("constant")
late  <- hzr_phase("g3", tau = 1, gamma = 3, alpha = 1, eta = 1)

# Fix all shapes (C/SAS-style: only estimate mu)
early_fixed <- hzr_phase("cdf", t_half = 0.5, nu = 2, m = 0,
                          fixed = "shapes")
late_fixed  <- hzr_phase("g3", tau = 1, gamma = 3, alpha = 1, eta = 1,
                          fixed = "shapes")

# Fix only some parameters
early_partial <- hzr_phase("cdf", t_half = 0.5, nu = 2, m = 0,
                            fixed = c("nu", "m"))

# Phase with specific covariates
early_cov <- hzr_phase("cdf", t_half = 0.5, nu = 2, m = 0,
                        formula = ~ age + shock)

# Use in hazard():
# hazard(Surv(time, status) ~ age, data = dat,
#        dist = "multiphase",
#        phases = list(early = early, constant = const, late = late))
```

`hzr_phase_cumhaz` *Cumulative hazard contribution from a single phase*

Description

Computes $\Phi_j(t)$ for one phase in the additive model $H(t|x) = \sum_j \mu_j(x)\Phi_j(t)$.

Usage

```
hzr_phase_cumhaz(
  time,
  t_half = 1,
  nu = 1,
  m = 0,
  type = c("cdf", "hazard", "constant")
)
```

Arguments

`time`	Numeric vector of times (> 0).
`t_half`	Half-life parameter (> 0).
`nu`	Time exponent.

`m`	Shape parameter.
`type`	Phase type: `"cdf"` (early – uses $G(t)$), `"hazard"` (late – uses cumulative hazard from $h(t)$), or `"constant"` (flat rate – $\Phi = t$).

Details

- `"cdf"`: $\Phi(t) = G(t)$. Bounded $[0, 1]$. Models early risk that resolves over time.
- `"hazard"`: $\Phi(t) = -\log(1 - G(t))$. Monotone increasing. Models late or aging risk. This is the cumulative hazard derived from the hazard function $h(t)$, since $\int_0^t h(s)\, ds = -\log(1 - G(t))$.
- `"constant"`: $\Phi(t) = t$. Ignores `t_half`, `nu`, `m`. Equivalent to exponential (constant hazard rate).

Value

Numeric vector of cumulative hazard contributions $\Phi(t)$, same length as `time`.

See Also

`hzr_decompos()` for the underlying parametric family, `hzr_phase_hazard()` for the instantaneous hazard contribution.

Examples

```
t_grid <- seq(0.1, 10, by = 0.1)
phi_early <- hzr_phase_cumhaz(t_grid, t_half = 2, nu = 2, m = 0,
                              type = "cdf")
phi_late  <- hzr_phase_cumhaz(t_grid, t_half = 5, nu = 1, m = 0,
                              type = "hazard")
phi_const <- hzr_phase_cumhaz(t_grid, type = "constant")
```

`hzr_phase_hazard` *Instantaneous hazard contribution from a single phase*

Description

Computes $\phi_j(t) = d\Phi_j/dt$ for one phase – the derivative of the cumulative hazard contribution returned by `hzr_phase_cumhaz()`.

Usage

```
hzr_phase_hazard(
  time,
  t_half = 1,
  nu = 1,
  m = 0,
  type = c("cdf", "hazard", "constant")
)
```

Arguments

`time`	Numeric vector of times (> 0).
`t_half`	Half-life parameter (> 0).
`nu`	Time exponent.
`m`	Shape parameter.
`type`	Phase type: `"cdf"` (early – uses $G(t)$), `"hazard"` (late – uses cumulative hazard from $h(t)$), or `"constant"` (flat rate – $\Phi = t$).

Details

- `"cdf"`: $\phi(t) = g(t)$ (density).
- `"hazard"`: $\phi(t) = h(t) = g(t)/(1 - G(t))$.
- `"constant"`: $\phi(t) = 1$.

Value

Numeric vector of instantaneous hazard contributions $\phi(t)$, same length as `time`.

See Also

`hzr_decompos()` for the underlying parametric family, `hzr_phase_cumhaz()` for the cumulative version.

Examples

```
t_grid <- seq(0.1, 10, by = 0.1)
phi_early <- hzr_phase_hazard(t_grid, t_half = 2, nu = 2, m = 0,
                               type = "cdf")
phi_late  <- hzr_phase_hazard(t_grid, t_half = 5, nu = 1, m = 0,
                               type = "hazard")
```

`hzr_stepwise` *Stepwise covariate selection for a parametric hazard model*

Description

Run forward, backward, or two-way stepwise selection on an existing `hazard` fit using Wald p-values or AIC deltas as the entry / retention criterion. Phase-specific entry is supported for multi-phase models: a covariate can enter one phase and not another.

Usage

```
hzr_stepwise(
  fit,
  scope = NULL,
  data,
  direction = c(”both”, ”forward”, ”backward”),
  criterion = c(”wald”, ”aic”),
  slentry = 0.3,
  slstay = 0.2,
  max_steps = 50L,
  max_move = 4L,
  force_in = character(),
  force_out = character(),
  trace = TRUE,
  ...
)

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

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

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

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

Arguments

`fit`	A fitted `hazard` object built via the `formula = Surv(...) ~ predictors, data = df` interface.
`scope`	Candidate set. `NULL` (default) uses every data-frame column not already in the model for every phase. For single-distribution fits, pass a one-sided formula (`~ age + nyha`) or a character vector of names. For multiphase fits, pass a named list of one-sided formulas keyed by phase.
`data`	Data frame the base fit was built on. Required for refits.
`direction`	One of `"both"` (default), `"forward"`, `"backward"`.
`criterion`	One of `"wald"` (default) or `"aic"`. SAS-style p-value thresholds apply to Wald; AIC uses `DeltaAIC < 0` uniformly.
`slentry`	Entry p-value threshold for the Wald criterion. Default `0.30` matches SAS `SLENTRY`.
`slstay`	Retention p-value threshold for the Wald criterion. Default `0.20` matches SAS `SLSTAY`.
`max_steps`	Hard cap on total accepted actions. Emits a `warning()` if hit. Default `50`.

`max_move`	Per-variable oscillation cap. When a variable has entered + exited more than `max_move` times it is frozen for the remainder of the run. Default `4`.
`force_in`	Character vector of variables that must remain in the model. Such variables are still scored and reported in the selection trace, but are never dropped.
`force_out`	Character vector of variables that may never be considered as candidates.
`trace`	Logical; print step-by-step progress to the console. Default `TRUE`.
`...`	Unused.
`x`	An `hzr_stepwise` object.
`object`	An `hzr_stepwise` object.

Details

The `steps` data frame has columns:

`step_num` Integer sequence starting at 1.

`action` `"enter"`, `"drop"`, or `"frozen"`.

`variable` Variable affected.

`phase` Phase name (multiphase) or `NA_character_`.

`criterion` `"wald"` or `"aic"`.

`score` Winning score used for the decision.

`stat`, `df` Wald statistic and degrees of freedom.

`p_value`, `delta_aic` Always populated when computable, regardless of the active criterion.

`logLik`, `aic`, `n_coef` Goodness-of-fit diagnostics of the model *after* this step.

Value

An object of class `c("hzr_stepwise", "hazard")` – the final fit augmented with:

`steps` Data frame with one row per accepted / frozen action; see Details.

`scope` Record of the candidate scope, plus `force_in`, `force_out`, and the frozen set.

`criteria` Named list of the threshold / direction settings actually applied.

`trace_msg` Character vector of the trace lines, captured regardless of the `trace` flag.

`elapsed` `difftime` from start to finish.

`final_call` The call that produced this result.

`print.hzr_stepwise` returns `x` invisibly.

`summary.hzr_stepwise` returns a `summary.hzr_stepwise` object (extends `summary.hazard`) with `$stepwise_steps` and `$stepwise_trace` appended.

`print.summary.hzr_stepwise` returns `x` invisibly.

`as.data.frame.hzr_stepwise` returns the `$steps` data frame.

Examples

```
data(avc)
avc <- na.omit(avc)
base <- hazard(survival::Surv(int_dead, dead) ~ age,
               data = avc, dist = "weibull", fit = TRUE)

sw <- hzr_stepwise(base, scope = ~ age + nyha,
                   data = avc, direction = "forward",
                   control = list(n_starts = 1))
print(sw)
```

`is_hzr_phase` *Test if an object is an hzr_phase*

Description

Test if an object is an hzr_phase

Usage

```
is_hzr_phase(x)
```

Arguments

`x`	Object to test.

Value

Logical scalar.

Examples

```
is_hzr_phase(hzr_phase("cdf"))
is_hzr_phase("not a phase")
```

`omc` *OMC: Open Mitral Commissurotomy*

Description

Data for 339 patients who underwent open mitral commissurotomy at the University of Alabama Birmingham. Contains repeated thromboembolic events (up to 3 per patient) with left censoring, exercising the interval censoring likelihood.

Usage

```
omc
```

Format

A data frame with 339 rows and 7 variables:

study Patient identifier

te1 Indicator for first thromboembolic event

te2 Indicator for second thromboembolic event

te3 Indicator for third thromboembolic event

int_dead Follow-up interval to death or last contact (months)

dead Death indicator (1 = dead, 0 = censored)

opdjul Operation date (Julian)

Source

University of Alabama Birmingham cardiac surgery registry.

See Also

Other datasets: `avc`, `cabgkul`, `tga`, `valves`

`predict.hazard` *Predict from a hazard model object*

Description

Produces prediction outputs from a `hazard` object. Supports multiple prediction types including linear predictor, hazard, survival probability, and cumulative hazard.

Usage

```
## S3 method for class 'hazard'
predict(
  object,
  newdata = NULL,
  type = c("hazard", "linear_predictor", "survival", "cumulative_hazard"),
  decompose = FALSE,
  se.fit = FALSE,
  level = 0.95,
  ...
)
```

Arguments

`object`	A `hazard` object.
`newdata`	Optional matrix or data frame of predictors. For types requiring time (e.g., "survival", "cumulative_hazard"), newdata should include a `time` column, or time will be taken from the fitted object's data.
`type`	Prediction type: • `"linear_predictor"`: Linear predictor eta = x*beta (not available for multiphase) • `"hazard"`: Hazard scale exp(eta) (not available for multiphase) • `"survival"`: Survival probability S(t|x) = exp(-H(t|x)) • `"cumulative_hazard"`: Cumulative hazard H(t|x) at event times
`decompose`	Logical; if `TRUE` and the model is multiphase, return a data frame with per-phase cumulative hazard contributions alongside the total. Ignored for single-distribution models. Default `FALSE`.
`se.fit`	Logical; if `TRUE`, compute delta-method standard errors and confidence limits for each prediction. The return value becomes a data frame with columns `fit`, `se.fit`, `lower`, `upper`. Default `FALSE`. CLs are computed on the log-hazard / log-cumhaz scale and on the log(-log(survival)) scale so lower/upper stay inside the valid range of each prediction type; `linear_predictor` uses symmetric natural-scale CLs. Not compatible with `decompose = TRUE`.
`level`	Numeric confidence level in `(0, 1)`; default `0.95`. Only used when `se.fit = TRUE`.
`...`	Unused; included for S3 compatibility.

Details

For Weibull models with survival or cumulative_hazard predictions:

- Cumulative hazard: H(t|x) = (mu*t)^nu * exp(eta)
- Survival: S(t|x) = exp(-H(t|x))

Time values must be positive and finite. If newdata contains a `time` column, it will be used; otherwise, the time vector from the fitted object is used. For models fit with `time_windows`, predictions for `type = "linear_predictor"` or `"hazard"` also require time values (via `newdata$time` or fitted-time fallback) so window-specific coefficients can be selected.

Value

Numeric vector of predictions.

Known limitations

`decompose = TRUE` and `se.fit = TRUE` cannot be used together. The delta-method Jacobian has not yet been extended to per-phase contributions. To obtain confidence limits for a multiphase model, request point predictions first (`decompose = TRUE, se.fit = FALSE`), then separately request CLs on the total prediction (`decompose = FALSE, se.fit = TRUE`).

See Also

`hazard()` for model fitting, `summary.hazard()` for model summaries, `hzr_phase()` for multiphase temporal shapes.

`vignette("prediction-visualization")` for detailed prediction workflows including decomposed hazard plots and patient-specific curves.

Examples

```
# -- Basic predictions ------------------------------------------------
set.seed(1)
fit <- hazard(time = rexp(50, 0.3), status = rep(1L, 50),
              theta = c(0.3, 1.0), dist = "weibull", fit = TRUE)
predict(fit, type = "survival")
predict(fit, newdata = data.frame(time = c(1, 2, 5)),
        type = "cumulative_hazard")

# -- Patient-specific survival curves ---------------------------------
set.seed(1001)
n   <- 180
dat <- data.frame(
  time   = rexp(n, rate = 0.35) + 0.05,
  status = rbinom(n, size = 1, prob = 0.6),
  age    = rnorm(n, mean = 62, sd = 11),
  nyha   = sample(1:4, n, replace = TRUE),
  shock  = rbinom(n, size = 1, prob = 0.18)
)
fit2 <- hazard(
  survival::Surv(time, status) ~ age + nyha + shock,
  data  = dat,
  theta = c(mu = 0.25, nu = 1.10, beta1 = 0, beta2 = 0, beta3 = 0),
  dist  = "weibull", fit = TRUE
)

new_patients <- data.frame(
  time = c(0.5, 1.5, 3.0),
  age  = c(50, 65, 75),
  nyha = c(1, 3, 4),
  shock = c(0, 0, 1)
)
# Compute predictions from the clean covariate frame before adding columns
```

```
surv   <- predict(fit2, newdata = new_patients, type = "survival")
cumhaz <- predict(fit2, newdata = new_patients, type = "cumulative_hazard")
new_patients$survival          <- surv
new_patients$cumulative_hazard <- cumhaz
new_patients


# -- Grouped survival curves --------------------------------------
if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)

  t_grid <- seq(0.05, max(dat$time), length.out = 80)
  profiles <- data.frame(
    label = c("Low risk (age 50, NYHA I)",
              "High risk (age 75, NYHA IV)"),
    age   = c(50, 75),
    nyha  = c(1, 4),
    shock = c(0, 1)
  )

  curve_list <- lapply(seq_len(nrow(profiles)), function(i) {
    nd <- data.frame(
      time  = t_grid,
      age   = profiles$age[i],
      nyha  = profiles$nyha[i],
      shock = profiles$shock[i]
    )
    nd$survival <- predict(fit2, newdata = nd, type = "survival") * 100
    nd$profile  <- profiles$label[i]
    nd
  })
  curve_df <- do.call(rbind, curve_list)

  ggplot(curve_df, aes(time, survival, colour = profile)) +
    geom_line() +
    scale_y_continuous(limits = c(0, 100)) +
    labs(x = "Months after surgery",
         y = "Freedom from death (%)",
         title = "Predicted survival by risk profile",
         colour = NULL) +
    theme_minimal()
}



# -- Multiphase predictions with decomposition --------------------
set.seed(42)
n   <- 200
dat <- data.frame(
  time   = rexp(n, rate = 0.25) + 0.01,
  status = rbinom(n, size = 1, prob = 0.65)
)
fit_mp <- hazard(
```

```
  survival::Surv(time, status) ~ 1,
  data   = dat,
  dist   = "multiphase",
  phases = list(
    early = hzr_phase("cdf", t_half = 0.5, nu = 2, m = 0,
                      fixed = "shapes"),
    late  = hzr_phase("cdf", t_half = 5,   nu = 1, m = 0,
                      fixed = "shapes")
  ),
  fit     = TRUE,
  control = list(n_starts = 5, maxit = 1000)
)

t_grid <- seq(0.01, max(dat$time) * 0.9, length.out = 100)
nd     <- data.frame(time = t_grid)

# Overall survival
predict(fit_mp, newdata = nd, type = "survival")

# Per-phase decomposed cumulative hazard
decomp <- predict(fit_mp, newdata = nd,
                  type = "cumulative_hazard", decompose = TRUE)
head(decomp)
```

`print.hzr_calibrate` *Print method for hzr_calibrate*

Description

Print method for hzr_calibrate

Usage

```
## S3 method for class 'hzr_calibrate'
print(x, digits = 3, ...)
```

Arguments

`x`	An `hzr_calibrate` object.
`digits`	Number of decimal places for formatting.
`...`	Additional arguments (ignored).

Value

The object `x` of class `c("hzr_calibrate", "data.frame")`, invisibly. The data frame has one row per quantile group and columns: `group` (group index), `n` (group size), `events` (event count), `mean`, `min`, `max` (covariate summary within group), `prob` (observed event probability), `link_value` (transformed probability on the link scale). When stratified via the `by` argument, a `by` column is also present. Attributes: `"link"` (the transform applied, e.g. `"logit"`) and `"groups"` (number of quantile bins).

`print.hzr_deciles` *Print method for hzr_deciles*

Description

Print method for hzr_deciles

Usage

```
## S3 method for class 'hzr_deciles'
print(x, digits = 3, ...)
```

Arguments

`x`	An `hzr_deciles` object.
`digits`	Number of decimal places for formatting.
`...`	Additional arguments (ignored).

Value

The object `x` of class `c("hzr_deciles", "data.frame")`, invisibly. The data frame has one row per risk group and columns: `group` (integer group index, 1 = lowest risk), `n` (group size), `events` (observed event count), `expected` (expected event count from model predictions), `observed_rate`, `expected_rate` (events / n), `chi_sq` (per-group (O-E)^2/E contribution), `p_value` (1-df chi-square upper-tail p), `mean_survival`, `mean_cumhaz` (mean predicted values in group). An `"overall"` attribute contains the omnibus chi-square test (fields: `chi_sq`, `df`, `p_value`, `time`, `groups`, `total_events`, `total_expected`, `n_included`, `n_excluded`).

`print.hzr_gof` *Print method for hzr_gof*

Description

Print method for hzr_gof

Usage

```
## S3 method for class 'hzr_gof'
print(x, digits = 3, ...)
```

Arguments

`x`	An `hzr_gof` object.
`digits`	Number of decimal places for formatting.
`...`	Additional arguments (ignored).

Value

The object `x` of class `c("hzr_gof", "data.frame")`, invisibly. The data frame has one row per time point and columns: `time`, `n_risk`, `n_event`, `n_censor`, `km_surv` (Kaplan-Meier survival), `km_cumhaz`, `par_surv` (parametric survival), `par_cumhaz`, `cum_observed` (cumulative observed events), `cum_expected` (cumulative expected events from model), `residual` (cum_expected - cum_observed). Multiphase models additionally include `par_cumhaz_<phase>` columns for per-phase cumulative hazard contributions. A `"summary"` attribute contains scalar diagnostics: `total_observed`, `total_expected`, `final_residual`, `dist`, `n`.

`print.hzr_kaplan` *Print method for hzr_kaplan*

Description

Print method for hzr_kaplan

Usage

```
## S3 method for class 'hzr_kaplan'
print(x, digits = 4, n = 20, ...)
```

Arguments

`x`	An `hzr_kaplan` object.
`digits`	Number of decimal places for formatting.
`n`	Maximum rows to print (default 20).
`...`	Additional arguments (ignored).

Value

The object `x` of class `c("hzr_kaplan", "data.frame")`, invisibly. The data frame has one row per event time (or all times when `event_only = FALSE`) and columns: `time` (follow-up time), `n_risk` (number at risk), `n_event` (events in interval), `n_censor` (censored observations in interval), `survival` (Kaplan-Meier survival estimate), `std_err` (Greenwood standard error on log-hazard scale), `cl_lower`, `cl_upper` (logit-transformed confidence limits on the survival scale), `cumhaz` (Nelson-Aalen cumulative hazard), `hazard` (interval hazard estimate), `density` (estimated event density), `life` (life-table life expectancy contribution).

`stepwise_trace` *Extract the captured console trace from an `hzr_stepwise` fit*

Description

Every run of `hzr_stepwise()` records the header, per-step lines, and final summary regardless of the `trace` flag. This accessor returns the full character vector for display or logging.

Usage

```
stepwise_trace(fit)
```

Arguments

`fit`	An `hzr_stepwise` object.

Value

Character vector, one element per console line.

Examples

```
data(avc)
avc <- na.omit(avc)
base <- hazard(survival::Surv(int_dead, dead) ~ age,
               data = avc, dist = "weibull", fit = TRUE)

sw <- hzr_stepwise(base, scope = ~ age + nyha,
                   data = avc, direction = "forward",
                   control = list(n_starts = 1))
cat(stepwise_trace(sw), sep = "\n")
```

`summary.hazard` *Summarize a hazard model*

Description

Returns a compact summary of a `hazard` object, including model metadata, fit diagnostics, and coefficient-level statistics when available.

Usage

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

Arguments

`object`	A hazard object.
`...`	Unused; for S3 compatibility.

Value

An object of class `summary.hazard`.

See Also

`hazard()` for model fitting, `predict.hazard()` for predictions.

`vignette("fitting-hazard-models")` for fitting workflows, `vignette("inference-diagnostics")` for bootstrap CIs and diagnostics.

Examples

```
# -- Single-phase Weibull summary ------------------------------------
fit <- hazard(time = rexp(30, 0.5), status = rep(1L, 30),
              theta = c(0.3, 1.0), dist = "weibull", fit = TRUE)
summary(fit)


# -- Multiphase model summary ----------------------------------------
set.seed(42)
n   <- 200
dat <- data.frame(
  time   = rexp(n, rate = 0.25) + 0.01,
  status = rbinom(n, size = 1, prob = 0.65)
)
fit_mp <- hazard(
  survival::Surv(time, status) ~ 1,
  data   = dat,
  dist   = "multiphase",
  phases = list(
    early = hzr_phase("cdf", t_half = 0.5, nu = 2, m = 0,
                      fixed = "shapes"),
    late  = hzr_phase("cdf", t_half = 5,   nu = 1, m = 0,
                      fixed = "shapes")
  ),
  fit     = TRUE,
  control = list(n_starts = 5, maxit = 1000)
)
summary(fit_mp)
```

`tga` *TGA: Transposition of the Great Arteries*

Description

Survival data for 470 patients who underwent the arterial switch operation for transposition of the great arteries at Boston Children's Hospital and Children's Hospital of Philadelphia. Used for sensitivity analysis and internal validation examples.

Usage

```
tga
```

Format

A data frame with 470 rows and 14 variables:

study Patient identifier

simple Simple TGA indicator (0/1)

dextroin D-looped transposition indicator (0/1)

ca_1rl2c Coronary artery pattern indicator

hyaaproc Hybrid approach procedure indicator (0/1)

no_tca No total circulatory arrest indicator (0/1)

tca_time Total circulatory arrest time (minutes)

age_days Age at operation (days)

arciopyr Aortic cross-clamp time per year

dead Death indicator (1 = dead, 0 = censored)

int_dead Follow-up interval to death or last contact (months)

source Source institution (BCH or CHOP)

ca1_2_l Coronary artery configuration (1/2/L)

opyear Year of operation

Source

Boston Children's Hospital and Children's Hospital of Philadelphia.

See Also

Other datasets: `avc`, `cabgkul`, `omc`, `valves`

`valves` *Valves: Primary Heart Valve Replacement*

Description

Data for 1,533 patients who underwent primary heart valve replacement. The largest multivariable example dataset with multiple endpoints including death, prosthetic valve endocarditis (PVE), bioprosthesis degeneration, and reoperation.

Usage

```
valves
```

Format

A data frame with 1533 rows and 19 variables:

age_cop Age at operation (years)

nyha NYHA functional class (1–4)

mitral Mitral valve position indicator (0/1)

double_ Double valve replacement indicator (0/1)

ao_pinc Aortic position, incompetence (0/1)

black Black race indicator (0/1)

i_path Ischemic pathology indicator (0/1)

nve Native valve endocarditis indicator (0/1)

mechvalv Mechanical valve indicator (0/1)

male Male sex indicator (0/1)

int_dead Follow-up interval to death or last contact (months)

dead Death indicator (1 = dead, 0 = censored)

int_pve Follow-up interval to PVE or last contact (months)

pve PVE indicator (1 = PVE, 0 = censored)

bio Bioprosthesis indicator (0/1)

int_rdg Follow-up interval to degeneration or last contact (months)

reop_dg Reoperation for degeneration indicator (0/1)

int_reop Follow-up interval to reoperation or last contact (months)

reop Reoperation indicator (0/1)

Source

Cleveland Clinic Foundation heart valve replacement registry.

See Also

`vignette("fitting-hazard-models")`, `vignette("prediction-visualization")`

Other datasets: `avc`, `cabgkul`, `omc`, `tga`

Examples

```
data(valves)
valves_cc <- na.omit(valves)

# Kaplan-Meier for two endpoints
km_death <- survival::survfit(
  survival::Surv(int_dead, dead) ~ 1, data = valves_cc)
km_pve <- survival::survfit(
  survival::Surv(int_pve, pve) ~ 1, data = valves_cc)

plot(km_death, xlab = "Months after valve replacement", ylab = "Survival",
     main = "Valves: Death and PVE endpoints")
lines(km_pve, col = "red")
legend("bottomleft", c("Death", "PVE"), col = c("black", "red"), lty = 1)
```

`vcov.hazard` *Extract variance-covariance matrix from hazard model*

Description

Returns the estimated variance-covariance matrix of the fitted coefficients.

Usage

```
## S3 method for class 'hazard'
vcov(object, ...)
```

Arguments

`object`	A hazard object.
`...`	Unused; for S3 compatibility.

Value

A numeric matrix containing the estimated variance-covariance matrix of the fitted coefficients, or `NA` if the model has not been fitted or the covariance matrix is unavailable.

Examples

```
fit <- hazard(time = rexp(30, 0.5), status = rep(1L, 30),
              theta = c(0.3, 1.0), dist = "weibull", fit = TRUE)
vcov(fit)
```

Index