Package: PAmeasures (via r-universe)

October 28, 2026

Type Package

Title Performance Assessment for Continuous and Survival Outcomes

Version 2.0.0

Date 2026-10-27

Description Evaluates prediction accuracy for continuous outcomes and the predictive performance of survival models in a unified framework. Implements the R-squared, L-squared, pseudo R-squared and related explained variation measures of Li and Wang (2019) <doi:10.1080/01621459.2018.1515079>, the concordance indices of Harrell et al. (1982) <doi:10.1001/jama.1982.03320430047030> and Uno et al. (2011) <doi:10.1002/sim.4154>, the time-dependent area under the receiver operating characteristic curve (AUC) of Heagerty, Lumley and Pepe (2000) <doi:10.1111/j.0006-341X.2000.00337.x>, and the Brier score of Graf et al. (1999) <doi:10.1002/(SICI)1097-0258(19990915/30)18:17/18%3C2529::AID-SIM274%3E3.0.CO;2-5>.
Supports right-censored data; competing risks, following Wolbers et al. (2014) <doi:10.1093/biostatistics/kxt059>, Wu and Li (2018) <doi:10.1002/sim.7806> and Zhuang et al. (2025) <doi:10.48550/arXiv.2507.15040>; and two-phase sampling designs such as case-cohort and nested case-control studies, following Lee, Zeleniuch-Jacquotte and Liu (2021) <doi:10.1080/02664763.2020.1861225>. Revives and expands the original 'PAmeasures' package with a unified interface for survival prediction assessment.

Depends R (>= 4.1.0)

Imports Hmisc, survival, stats, utils, tdROC, survivalROC, yardstick, ggplot2, patchwork, dplyr, magrittr, purrr, tibble, tidyr, pec, pmcalibration (>= 0.2.0), prodlim

URL https://github.com/Larryzza/PAmeasures

BugReports https://github.com/Larryzza/PAmeasures/issues

License GPL-3

Encoding UTF-8

Language en-US

LazyData true

Suggests spelling, testthat (>= 3.0.0), withr, pkgload, covr, randomForestSRC, cmprsk

Config/testthat/edition 3

NeedsCompilation no

Config/roxygen2/version 8.1.0

RoxygenNote 7.3.3

Author Tong Zhu [aut], Zian Zhuang [aut, cre], Wen Su [aut], Xiaowu Dai [aut], Xiaoyan Wang [aut], Gang Li [aut]

Maintainer Zian Zhuang <zianzhuang@ucla.edu>

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

Date/Publication 2026-10-28 21:44:56 UTC

RemoteUrl https://github.com/cran/PAmeasures

RemoteRef HEAD

RemoteSha a8ad0414acd68d7775cb6ce05de8010b0a25f724

Contents

moore . . . 3
pa_calibration . . . 3
pa_case_cohort_weights . . . 6
pa_evaluate_two_phase . . . 6
pa_fit_and_eval . . . 8
pa_metric_names . . . 10
pa_nested_case_control_weights . . . 10
pa_nlm . . . 11
pa_plot_pred . . . 13
pa_plot_summary . . . 14
pa_predict_cif . . . 16
pa_predict_coxph . . . 18
pa_predict_survreg . . . 20
pa_sample_design . . . 21
pa_sim_cox_weibull . . . 23
pa_simulate_fine_gray . . . 25
pa_summarize . . . 27
pa_summarize_cr . . . 28
pa_survival_eval . . . 30
pa_survival_eval_cr . . . 32

Index **36**

moore *Moore's Law Data*

Description

Transistor counts for commercial microprocessors together with their year of introduction. Moore's Law observes that the number of transistors in a dense integrated circuit roughly doubles every two years.

Usage

```
moore
```

Format

A data frame with 48 rows and 3 variables:

year Year of introduction, from 1971 to 2012.

time Years elapsed since 1970.

count Number of transistors.

Source

Compiled from published microprocessor transistor counts.

pa_calibration *Calibration of Predicted Event Probabilities at One Time Point*

Description

Assess calibration from externally supplied probabilities and observed outcomes, without requiring the original prediction model.

Usage

```
pa_calibration(p, event_time, status, t_star, nk = 5, case_weights = NULL)

pa_calibration_cr(
  p,
  event_time,
  status,
  t_star,
  event_type = 1,
  bandwidth = NULL
)
```

Arguments

p	Numeric vector of predicted event probabilities at t_star, in the same subject order as event_time and status. For ordinary survival supply 1 - S(t_star); for competing risks supply the CIF of event_type at t_star.
event_time	Numeric vector of finite, non-negative observed times.
status	Event indicators: 0 for censoring and 1 for an event in pa_calibration(); non-negative integer event codes in pa_calibration_cr() (0 for censoring).
t_star	One positive finite evaluation time, not exceeding the maximum observed follow-up. Use the same time units as event_time.
nk	Number of restricted cubic spline knots (at least 3; default 5).
case_weights	Optional positive finite inverse-inclusion sampling weights, aligned with the subjects. For pa_calibration() only. Used in both the calibration model and the metric summaries.
event_type	Positive integer code for the cause of interest.
bandwidth	Nearest-neighborhood smoothing bandwidth passed to prodlim::prodlim(). NULL uses its automatic bandwidth.

Details

pa_calibration() uses pmcalibration::pmcalibration() with a restricted cubic spline and no confidence interval computation. The knot count is capped at the number of distinct fitting probabilities; with fewer than five distinct probabilities or fewer than six subjects it uses the backend's single-predictor Cox calibration instead. Constant predictions are compared with the marginal Kaplan–Meier risk.

The right-censored backend requires probabilities strictly inside (0, 1). Probabilities within sqrt(.Machine$double.eps) of 0 or 1 are clipped to that distance from the boundary for fitting, with a warning (except for the constant-prediction Kaplan–Meier branch). Metrics compare calibrated values with the original probabilities and retain every subject.

pa_calibration_cr() uses local Aalen–Johansen estimates from prodlim, conditional on the predicted CIF. All nonzero causes other than event_type are pooled as competing events, not censored. Constant or few-valued predictions use the backend's discrete risk groups. This is a different curve estimator from Fine–Gray spline calibration.

For unweighted evaluations, both functions use pmcalibration::cal_metrics() to summarize absolute differences between predicted and curve-implied probabilities at the original subjects. Only ici (the backend's Eavg), e50, and e90 are returned. They are probability-scale errors, with 0 ideal. ICI averages over subjects, not over follow-up time or a uniform risk grid.

With case_weights, pa_calibration() fits a sampling-weighted Cox model directly, using the same complementary log-log transformation and restricted cubic spline knot-placement rule as the ordinary backend. Both coefficients and baseline cumulative hazard come from the weighted fit. Constant predictions use weighted Kaplan–Meier risk; with fewer than five distinct fitting probabilities or six subjects a single linear predictor is used. Knot locations are recorded in knots. ICI is the normalized sampling-weighted mean absolute error. E50 and E90 use weighted order-statistic interpolation, with weights normalized to the sample size (Hmisc::wtd.quantile(..., normwt = TRUE)). Multiplying every sampling weight by the same constant has no effect.

Supply subject inclusion weights for the actual case–cohort/NCC design, rather than weights already combined with censoring adjustments. Only sampled subjects with positive inclusion probabilities should be supplied. The calibration Cox model assumes conditionally independent censoring given its predictor and an appropriate proportional-hazards spline relationship. The competing-risks function remains an unweighted cohort evaluation. No optimism correction or design-based confidence interval is applied; preferably supply held-out or out-of-fold predictions.

Value

A list with `metrics`, a named numeric vector `c(ici, e50, e90)`; `curve`, a data frame with `predicted` and `calibrated` probabilities in input subject order; `t_star`; `event_type`; and `method`, the curve estimator. Weighted results also include the original `case_weights` and spline `knots` on the complementary log-log scale (empty for simpler fits).

References

Austin PC, Steyerberg EW (2019). The Integrated Calibration Index (ICI) and related metrics for quantifying the calibration of logistic regression models. Statistics in Medicine 38, 4051–4065. doi:10.1002/sim.8281. Austin PC, Harrell FE Jr, van Klaveren D (2020). Graphical calibration curves and the integrated calibration index (ICI) for survival models. Statistics in Medicine 39, 2714–2742. doi:10.1002/sim.8570. Breslow NE, Wellner JA (2007). Weighted likelihood for semiparametric models and two-phase stratified samples, with application to Cox regression. Scandinavian Journal of Statistics 34, 86–102. doi:10.1111/j.14679469.2006.00523.x. Samuelsen SO (1997). A pseudolikelihood approach to analysis of nested case-control studies. Biometrika 84, 379–394. doi:10.1093/biomet/84.2.379.

Examples

```
set.seed(19)
x <- stats::rnorm(100)
event <- stats::rexp(100, rate = exp(x))
censor <- stats::rexp(100, rate = 0.3)
time <- pmin(event, censor)
status <- as.integer(event <= censor)
p <- 1 - exp(-exp(x) * 0.5)
pa_calibration(p, time, status, t_star = 0.5)$metrics

# Externally supplied CIFs; no fitted model object is needed.
competing <- stats::rexp(100, rate = 0.4)
cr_time <- pmin(event, competing, censor)
cr_status <- ifelse(censor < pmin(event, competing), 0L,
                    ifelse(event <= competing, 1L, 2L))
cif <- exp(x) / (exp(x) + 0.4) * (1 - exp(-(exp(x) + 0.4) * 0.5))
pa_calibration_cr(cif, cr_time, cr_status, t_star = 0.5)$metrics
```

pa_case_cohort_weights *Case-cohort sampling weights*

Description

Computes Prentice-style weights for case-cohort designs. If 'strata' is provided, stratified case-cohort weights are computed.

Usage

```
pa_case_cohort_weights(time, status, subcohort = NULL, strata = NULL)
```

Arguments

time	Numeric vector of follow-up times.
status	Numeric {0,1}, 1 = event, 0 = censored.
subcohort	Logical vector; TRUE if in subcohort.
strata	Optional factor/character for strata (stratified cc).

Value

Numeric vector of weights.

Examples

```
# Compute weights using the full cohort, then retain the random subcohort
# and all cases for analysis. Sampled non-cases represent cohort non-cases.
d <- pa_sim_cox_weibull(n = 150, pi_c = 0.3, v = 2,
                        beta = c(0.5, -0.5), seed = 2025)
set.seed(2025)
w <- pa_case_cohort_weights(time = d$time, status = d$status,
                            subcohort = stats::rbinom(nrow(d), 1, 0.4))
summary(w)
```

pa_evaluate_two_phase *Evaluate Survival Model Metrics for Two-Phase Sampling Designs*

Description

This function constructs a standardized evaluation dataset ('dat1') from the analysis dataset, predicted survival probabilities, and sampling weights, and computes key performance metrics in a tidy table format. Works for common two-phase sampling designs: - Case-cohort (unstratified or stratified) - Nested case-control (matched or unmatched)

Usage

```
pa_evaluate_two_phase(
  pred_results,
  t_star = NULL,
  tau = 1e+11,
  km_cens_fit,
  case_weights,
 metrics = c("pseudo_r2", "harrell_c", "uno_c", "brier_score", "td_auc", "ici"),
  calibration_args = list()
)
```

Arguments

pred_results	A prediction list, such as the output of pa_predict_coxph(), with times (or legacy time), status (0 = censored, 1 = event), pred (predicted mean survival times), and surv_prob (subjects by sorted observed times).
t_star	(Optional) Evaluation time for Brier score, time-dependent AUC, and calibration. If NULL, it defaults to the median survival time.
tau	(Optional) A numeric value specifying the truncation time for calculating explained variation metrics. If NULL, it defaults to the maximum observed survival time.
km_cens_fit	Kaplan-Meier estimate of the censoring distribution, a 'survival::survfit' object. Not required when only calibration is requested.
case_weights	Numeric sampling weights in the same subject order as pred_results. Prediction helpers sort subjects by observed time; reorder weights using order(new_data$time) as well.
metrics	Character vector of metrics. NULL uses the default metrics, including ICI; "all" additionally returns r_square, l_square, e50, and e90.
calibration_args	Optional named list of calibration settings, e.g. list(nk = 3). Sampling weights are supplied by case_weights, not through this list.

Details

Calibration uses the event risk from the last prediction-grid column at or before t_star, with the exact requested outcome horizon. Both the Cox spline calibration model and absolute-error summaries use sampling weights; see [pa_calibration()]. Calibration-only inputs need times, status, and surv_prob, without fitted models or mean time predictions.

Value

A tibble with columns: - 'Metric': name of metric - 'Value': numeric value

Examples

```
d <- pa_sim_cox_weibull(n = 150, pi_c = 0.3, v = 2,
                        beta = c(0.5, -0.5), seed = 2025)
```

```
d <- d[, c("time", "status", "x1", "x2")]
fit <- survival::coxph(survival::Surv(time, status) ~ x1 + x2,
                       data = d, x = TRUE, y = TRUE)
pred <- pa_predict_coxph(model = fit, covs = c("x1", "x2"), new_data = d)

# case-cohort sampling weights, and the censoring distribution they need
set.seed(2025)
in_subcohort <- stats::rbinom(nrow(d), 1, 0.4) == 1
w <- pa_case_cohort_weights(d$time, d$status, subcohort = in_subcohort)
keep <- which(in_subcohort | d$status == 1)
keep <- keep[order(d$time[keep])]
pred <- pa_predict_coxph(fit, c("x1", "x2"), new_data = d[keep, ])
w <- w[keep]
km <- survival::survfit(survival::Surv(d$time, 1 - d$status) ~ 1)

pa_evaluate_two_phase(pred_results = pred, km_cens_fit = km,
                      case_weights = w)
```

`pa_fit_and_eval` *Performance Metrics for Survival Analysis Models*

Description

This function computes a comprehensive set of performance metrics for survival analysis models. It provides metrics such as R_square, L_square, Pseudo_R, Harrell's C, Uno's C, Brier Score, and Time-dependent AUC. Users can specify particular metrics and model types, enabling tailored performance evaluation for various survival models.

Usage

```
pa_fit_and_eval(
  train_data,
  covariates,
  models = "coxph",
  metrics = NULL,
  predicted_data = NULL,
  t_star = NULL,
  tau = NULL,
  calibration_args = list()
)
```

Arguments

`train_data`	A data frame containing the survival data.
`covariates`	A character vector of covariate names to include in the model.
`models`	Character vector of model types: "coxph", "exp", "lognormal", or "weibull". The default fits only "coxph"; use "all" to fit all four.

`metrics` A character string or vector specifying the metrics to compute. The default `NULL` includes ICI and the existing performance metrics; use `"all"` to additionally include E50 and E90. Options include:

- "r_square": R-squared metric.
- "l_square": L-squared metric.
- "pseudo_r2": Pseudo-R-squared metric.
- "harrell_c": Harrell's Concordance Index.
- "uno_c": Uno's Concordance Index.
- "brier_score": Brier Score.
- "td_auc": Time-dependent AUC.
- "ici", "e50", "e90": Absolute calibration error summaries.

`predicted_data` (Optional) A data frame containing validation data. If 'NULL', the function uses the same data as 'data' for model evaluation.

`t_star` (Optional) Positive evaluation time for Brier score, AUC, and calibration; the median evaluation follow-up if omitted.

`tau` (Optional) A time point for truncating the survival time. If provided, the function evaluates predictions up to this time point.

`calibration_args` Optional named list passed to [pa_calibration()]. For sampling-adjusted calibration use [pa_sample_design()] or [pa_calibration()] directly; weights cannot be supplied through this list.

Details

Calibration evaluates model-predicted event probabilities exactly at `t_star` (the median evaluation follow-up if omitted).

Rows with missing values in `time`, `status`, or a requested covariate are excluded from training and evaluation with a warning. All requested metrics use the same complete evaluation rows. The warning can be silenced with `suppressWarnings()`.

Value

A data frame containing the selected model's performance metrics.

Examples

```
data(pbc, package = "survival")
pbc <- pbc[!is.na(pbc$trt), ]
pbc$log_albumin <- log(pbc$albumin)
pbc$log_bili    <- log(pbc$bili)
pbc$log_protime <- log(pbc$protime)
pbc$status <- ifelse(pbc$status == 2, 1, 0)
covariates <- c("age", "log_albumin", "log_bili", "log_protime", "edema")
dat <- pbc[, c("time", "status", covariates)]
dat <- dat[stats::complete.cases(dat), ]

# Default Cox model and metrics, including ICI
```

```
results <- pa_fit_and_eval(train_data = dat, covariates = covariates)
results

# A specific subset of models and metrics
results2 <- pa_fit_and_eval(
  train_data = dat,
  covariates = covariates,
  models  = c("lognormal", "weibull"),
  metrics = c("r_square", "l_square", "brier_score")
)
results2
```

`pa_metric_names` *Canonical metric names*

Description

The metric identifiers accepted by the ‘metrics‘ argument of the evaluation functions, and emitted in the ‘Metric‘ column of their results.

Usage

```
pa_metric_names()
```

Value

A character vector of canonical metric names.

Examples

```
pa_metric_names()
```

`pa_nested_case_control_weights`
Nested case-control (NCC) sampling weights

Description

Computes inverse-selection weights for NCC designs. If ‘strata‘ is provided, matched NCC weights are computed.

Usage

```
pa_nested_case_control_weights(time, status, strata = NULL, m = NULL)
```

Arguments

time	Numeric vector of follow-up times.
status	Numeric {0,1}, 1 = event, 0 = censored.
strata	Optional factor/character defining matching strata in the full phase-one cohort, not IDs of sampled case-control sets.
m	Positive integer, number of controls per case (required).

Details

Supply full-cohort follow-up and event information, not only sampled rows. Controls are sampled without replacement within each case's risk set and independently across cases. At tied failure times, all simultaneous cases are excluded from the eligible controls. Events precede censoring at ties. Matching restricts both cases and controls to the same stratum. If fewer than m controls are eligible, all eligible controls are sampled. A non-case with no sampling opportunity has infinite inverse probability; such a subject cannot belong to the sampled analysis dataset. Subset the returned weights to the subjects actually sampled and align their order with the prediction objects before evaluation.

Value

Numeric vector of weights, in full-cohort input order.

Examples

```
# Unmatched nested case-control design with m = 2 controls per case.
d <- pa_sim_cox_weibull(n = 150, pi_c = 0.3, v = 2,
                        beta = c(0.5, -0.5), seed = 2025)
set.seed(2025)
w <- pa_nested_case_control_weights(time = d$time, status = d$status, m = 2)
summary(w)
```

---

pa_nlm *Prediction Accuracy for Continuous Outcomes*

Description

Evaluate observed and predicted continuous responses using the R-squared and L-squared measures of Li and Wang (2019). Predictions may come from a linear or nonlinear model; the original fitted model is not required.

Usage

```
pa_nlm(y, y.predict)

pam.nlm(y, y.predict)
```

Arguments

y	A finite numeric vector of observed continuous responses.
y.predict	A finite numeric vector of predictions, paired with y in the same order and of the same length. At least two pairs are required. Missing and infinite values are rejected.

Details

An ordinary least-squares regression of y on y.predict, with an intercept, gives the corrected predictions $\hat{y}_c$. The measures are

$$R^2 = \frac{\sum_i (\hat{y}_{c,i} - \bar{y})^2}{\sum_i (y_i - \bar{y})^2}$$

and

$$L^2 = \frac{\sum_i (y_i - \hat{y}_{c,i})^2}{\sum_i (y_i - y.predict_i)^2}.$$

R-squared describes explained response variation after affine correction; L-squared compares the corrected and original prediction errors. These are the original PAmeasures definitions, not $1 - L^2$.

Constant predictions use the intercept-only correction mean(y). A constant observed response makes R-squared undefined. Predictions exactly equal to the observations make L-squared undefined because its denominator is zero. Undefined measures are NA_real_, with a warning; any other defined measure is retained. Inputs whose squared deviations overflow or whose nonzero deviations underflow to zero are rejected with an error and should be rescaled to common units.

pam.nlm() is the retained PAmeasures 0.1.0 compatibility entry point. It uses the same calculation, then rounds and formats each result to four decimal places as in the historical package. Its ordinary, defined results retain the historical field names and character format; input validation and undefined-result handling follow pa_nlm(). No deprecation warning is issued for this supported compatibility entry.

Value

A list with R.squared and L.squared. pa_nlm() returns unrounded numeric scalars. pam.nlm() returns the historical formatted character scalars.

References

Li, G. and Wang, X. (2019). Prediction Accuracy Measures for a Nonlinear Model and for Right-Censored Time-to-Event Data. *Journal of the American Statistical Association*, 114, 1815–1825. doi:10.1080/01621459.2018.1515079.

Examples

```
y <- c(1, 2, 4, 5)
prediction <- c(1, 2, 3, 4)
pa_nlm(y, prediction)
pam.nlm(y, prediction)

data(moore)
fit <- stats::lm(log2(count) ~ time, data = moore)
```

```
pa_nlm(moore$count, 2^stats::predict(fit))
```

---

pa_plot_pred *Quick Diagnostic Plot for Survival Model Predicted Probabilities*

Description

Produces a quick diagnostic plot visualizing predicted survival or event probabilities (e.g., from a Weibull AFT model) against the subject-specific linear predictors (risk scores). The plot combines a smoothed line of predicted values and points representing observed times, using shape to indicate censoring status. This visualization provides a simple check of how predicted risks correspond to observed outcomes.

Usage

```
pa_plot_pred(
  data,
  title = NULL,
  xlab = "Risk Score",
  ylab = "Days",
  label_name = c("censored", "event"),
  levels = NULL,
  shape_style = NULL,
  legend_name = "status",
  invert_linear = TRUE,
  sample_index = NULL,
  restrict_time = NULL
)
```

Arguments

data	A data frame (or list-like object) containing at least the following columns: **'linear.pred'** Model linear predictor (risk score). **'pred'** Predicted probability or median survival value. **'times'** Observed event or censoring time. **'status'** Event indicator (0 = censored, 1 = event; optionally can include more levels if defined in label_name).
title	Optional plot title (default = NULL).
xlab, ylab	Character strings specifying the axis labels (defaults are "Risk Score" for x and "Days" for y).
label_name	Character vector of labels corresponding to event status codes (e.g., c("censored", "event")); used in the legend.
levels	Numeric or character vector giving the ordering of status levels; defaults inferred from the length of label_name.

shape_style	Numeric vector of plotting symbols (pch values) corresponding to the levels of status. If NULL, defaults to c(1, 19) for two levels (censored vs event) or c(2, 19, 1) for three levels.
legend_name	Character string for the legend title (default = "status").
invert_linear	Logical; if TRUE (default), plots the negative of linear.pred so that higher values correspond to higher risk. Set to FALSE if you want higher values correspond to lower risk.
sample_index	Optional integer vector selecting which subjects to plot, e.g. sample(seq_len(n), 200) to thin a crowded chart. If NULL (default), every subject is plotted.
restrict_time	Optional numeric. If supplied, observed times above this value are capped at it and a dashed horizontal reference line is drawn, which keeps a few long follow-up times from compressing the plot.

Details

The function is primarily designed for quick visual diagnostics in survival model development. The line (pred vs linear.pred) shows the fitted or predicted trend, while the overlaid points (times vs linear.pred) illustrate observed event times, distinguished by censoring status. For more elaborate visualization (e.g., confidence bands, multiple models), use faceting or combine plots with **patchwork**.

Value

A ggplot object that can be further customized.

Examples

```
d <- pa_sim_cox_weibull(n = 150, pi_c = 0.3, v = 2,
                        beta = c(0.5, -0.5), seed = 2025)
d <- d[, c("time", "status", "x1", "x2")]
fit <- survival::coxph(survival::Surv(time, status) ~ x1 + x2,
                       data = d, x = TRUE, y = TRUE)
pred <- pa_predict_coxph(model = fit, covs = c("x1", "x2"), new_data = d)

# observed time against predicted risk score, shaped by event status
pa_plot_pred(pred, title = "Cox model")
```

---

pa_plot_summary *Arrange multiple prediction plots with letter tags*

Description

Builds one panel per data frame using plot_fun (default: pa_plot_pred) and arranges them with letter tags. The function collects a single shared legend and applies global styling (titles, axis labels, legend options) across panels.

Usage

```
pa_plot_summary(
  data_list,
  titles = NULL,
  plot_fun = pa_plot_pred,
  ncol = 2,
  tag_levels = "a",
  invert_linear = TRUE,
  xlab = "Risk Score",
  ylab = "Days",
  label_name = c("Censored", "Event"),
  levels = NULL,
  shape_style = NULL,
  sample_size = NULL,
  restrict_time = NULL,
  legend_name = "Status",
  seed = NULL
)
```

Arguments

data_list	List of data frames, each containing columns: linear.pred, pred, times, and status.
titles	Optional character vector, same length as data_list; if NULL, defaults to "Panel 1", "Panel 2", etc.
plot_fun	Function that creates a single panel (default = pa_plot_pred).
ncol	Number of columns in the panel layout (default = 2).
tag_levels	Letter style for tags: "a", "A", "1", "i", or "I".
invert_linear	Logical or logical vector; if length 1, recycled to all panels.
xlab, ylab	Axis labels applied to all panels (defaults are "Risk Score" and "Days").
label_name	Character vector of labels for status legend (e.g., c("censored", "event")).
levels	Numeric vector for status level ordering.
shape_style	Numeric vector of plotting symbols for censoring/event status.
sample_size	Optional integer. If supplied, that many subjects are sampled at random for each panel using seed; otherwise every subject is plotted.
restrict_time	Optional numeric. If supplied, observed times above this value are capped at it and a dashed horizontal reference line is drawn, which keeps a few long follow-up times from compressing the plot.
legend_name	Character string for the legend title (default = "status").
seed	Optional integer. If supplied, the random number generator is seeded with it before sample_size subjects are drawn, so the sampled points are reproducible. The default NULL uses the current random number stream without resetting it; random draws advance its state.

Value

A patchwork ggplot object combining all panels.

Examples

```
# One panel per model. The previous example referenced undefined objects and
# so could not be executed; this one runs.
d <- pa_sim_cox_weibull(n = 150, pi_c = 0.3, v = 2,
                        beta = c(0.5, -0.5), seed = 2025)
d <- d[, c("time", "status", "x1", "x2")]
cox <- survival::coxph(survival::Surv(time, status) ~ x1 + x2,
                       data = d, x = TRUE, y = TRUE)
wei <- survival::survreg(survival::Surv(time, status) ~ x1 + x2,
                         data = d, dist = "weibull", x = TRUE, y = TRUE)
p_cox <- pa_predict_coxph(model = cox, covs = c("x1", "x2"), new_data = d)
p_wei <- pa_predict_survreg(model = wei, covs = c("x1", "x2"), new_data = d)

pa_plot_summary(
  list(p_cox, p_wei),
  titles = c("Cox PH", "Weibull AFT"),
  ncol = 2,
  legend_name = "Status"
)
```

---

pa_predict_cif *Predict Subject-Specific Cumulative Incidence Function (CIF)*

Description

This function calculates cumulative incidence function (CIF) predictions for individual subjects across multiple types of competing risks models. It supports cause-specific Cox models, accelerated failure time (AFT) models (Weibull, Log-normal, Log-logistic), Fine-Gray models, and competing risks random survival forests.

Usage

```
pa_predict_cif(
  model1 = NULL,
  model2 = NULL,
  fg_model = NULL,
  cr_model = NULL,
  tau = NULL,
  newdata,
  event.type = 1,
  covs
)
```

Arguments

model1	The fitted model for the target cause (either a ‘coxph‘ or ‘survreg‘ object).
model2	The fitted model for competing causes (either a ‘coxph‘ or ‘survreg‘ object).
fg_model	Optional Fine-Gray model object (‘crr‘ from ‘cmprsk‘ package).
cr_model	Optional competing risks random forest model object.
tau	The maximum time horizon for prediction.
newdata	A data frame containing observed event times (‘time‘), event indicators (‘status‘).
event.type	The event type of interest for which CIF is to be predicted.
covs	A character vector specifying the names of the covariates used in the model.

Details

For cause-specific AFT models, CIFs are evaluated by adaptive quadrature on the target cause's probability scale. This handles Weibull densities that are unbounded at time zero, including fixed scales above 1. The time summary integrates one minus the CIF on the returned time grid.

Value

A list with the following components:

times Numeric vector of observed times from `newdata$time`.

status Integer vector of event indicators from `newdata$status`.

cif_pred A matrix where the first column is time, and subsequent columns are predicted CIFs for each subject.

pred Numeric vector of the integral of one minus the CIF from `0` to `tau`.

linear.pred Numeric vector of subject-specific linear predictors.

Examples

```
data(pbc, package = "survival")
pbc <- pbc[!is.na(pbc$trt), ]
pbc$log_albumin <- log(pbc$albumin)
pbc$log_bili    <- log(pbc$bili)
pbc$log_protime <- log(pbc$protime)
covariates <- c("age", "log_albumin", "log_bili", "log_protime", "edema")
dat <- pbc[, c("time", "status", covariates)]
dat <- dat[stats::complete.cases(dat), ]

# Cause-specific Cox models: event of interest (1) and competing event (2)
m1 <- survival::coxph(survival::Surv(time, status == 1) ~ age + log_albumin +
                        log_bili + log_protime + edema,
                      data = dat, x = TRUE, y = TRUE)
m2 <- survival::coxph(survival::Surv(time, status == 2) ~ age + log_albumin +
                        log_bili + log_protime + edema,
                      data = dat, x = TRUE, y = TRUE)
```

```
pred <- pa_predict_cif(model1 = m1, model2 = m2, newdata = dat,
                       covs = covariates, event.type = 1,
                       tau = max(dat$time))
str(pred$pred)

# Fine-Gray backend, available when cmprsk is installed
if (requireNamespace("cmprsk", quietly = TRUE)) {
  fg <- cmprsk::crr(ftime = dat$time, fstatus = dat$status,
                    cov1 = as.matrix(dat[, covariates]),
                    failcode = 1, cencode = 0)
  pred_fg <- pa_predict_cif(fg_model = fg, newdata = dat,
                            covs = covariates, event.type = 1,
                            tau = max(dat$time))
  str(pred_fg$pred)
}
```

`pa_predict_coxph` *Prediction Accuracy Measures for Cox Proportional Hazards Model*

Description

This function calculates a set of measures, R^2 and L^2, $Pseudo-R^2$, for a Cox proportional hazards model. When predict = TRUE, it returns the predicted survival times alongside observed values and event status. R-squared quantifies the proportion of variability in the observed survival times explained by the model's predictions, while L-squared measures the proportion of prediction error explained by a corrected prediction function. Together, these metrics provide a comprehensive evaluation of the predictive power of the Cox model.

Usage

```
pa_predict_coxph(model, covs, tau = 1e+11, new_data = NULL, predict = TRUE)
```

Arguments

`model`	An object of class 'coxph' representing a fitted Cox proportional hazards regression model. The model must be fitted with 'x = TRUE' and 'y = TRUE' to include the design matrix and response vector.
`covs`	A character vector specifying the names of the covariates used in the model.
`tau`	(Optional) A restriction time for computing restricted survival time. If provided, the function evaluates predictions up to this time point. Default is 10e10.
`new_data`	Optional evaluation data. Its outcomes are returned for metric computation and define the evaluation time grid; they never estimate the baseline hazard. All prediction parameters come from the training fit.
`predict`	A logical value indicating whether to return individual predictions. Default is TRUE.

Details

Probabilities use the fitted training baseline, including its case weights, strata, tie method and offsets. Offset centering follows the weighted training mean used by [survival::survfit()]. With 'predict = FALSE', R-squared metrics return missing values with a warning if censoring weights cannot be normalized, for example when the training censoring survival is zero at an observed evaluation event.

Value

A **named list** of model-specific prediction objects. Each element of the returned list corresponds to one fitted model and contains:

- `model` – the fitted survival model object.
- `times` – numeric vector of observed follow-up times for the evaluation dataset.
- `status` – event indicators (1 = event, 0 = censored).
- `surv_prob` – an $n \times K$ matrix of predicted subject-specific survival probabilities on a common time grid.
- `pred` – numerical integral of the training-model survival curve on the evaluation grid, up to its maximum time or tau, whichever is smaller. This preserves the legacy integration rule; it is not an identified unrestricted mean. Fix the grid and tau for comparisons.
- `covs` – character vector of covariate names used for prediction.
- `new_data` – dataset on which the predictions were computed.

References

Li, G., & Wang, X. (2016). Prediction Accuracy Measures for a Nonlinear Model and for Right-Censored Time-to-Event Data. arXiv preprint arXiv:1611.03063. Available at https://arxiv.org/abs/1611.03063.

Examples

```
data(pbc, package = "survival")
pbc <- pbc[!is.na(pbc$trt), ]
pbc$log_albumin <- log(pbc$albumin)
pbc$log_bili    <- log(pbc$bili)
pbc$log_protime <- log(pbc$protime)
pbc$status <- ifelse(pbc$status == 2, 1, 0)
covs <- c("age", "log_albumin", "log_bili", "log_protime", "edema")
df <- pbc[, c("time", "status", covs)]
df <- df[stats::complete.cases(df), ]

set.seed(2025)
idx <- sample(seq_len(nrow(df)), round(2 / 3 * nrow(df)))
train_data <- df[idx, ]
test_data  <- df[-idx, ]

m.cox <- survival::coxph(survival::Surv(time, status) ~ .,
                         data = train_data, x = TRUE, y = TRUE)
cox_pred <- pa_predict_coxph(model = m.cox, covs = covs,
```

```
                                  new_data = test_data)
str(cox_pred$pred)
```

`pa_predict_survreg` *Prediction Accuracy Measures for Parametric Survival Regression Models*

Description

This function computes Calculates a set of measures, R^2 and L^2, $Pseudo-R^2$ for evaluating the predictive accuracy of parametric survival regression models. R-squared quantifies the proportion of variability in the response that is explained by a corrected prediction function derived from the original model. L-squared measures the proportion of the prediction error from the original model that is explained by the corrected prediction function, effectively capturing the improvement in prediction accuracy. Together, these metrics provide a comprehensive assessment of the predictive performance of a survival regression model.

Usage

```
pa_predict_survreg(model, covs, tau = 1e+11, new_data = NULL, predict = TRUE)
```

Arguments

`model`	An object of class survreg representing a fitted parametric survival regression model. The survreg call must include x = TRUE and y = TRUE to ensure the design matrix and response vector are stored in the model object.
`covs`	A vector of covariate names to be used for prediction when new_data is provided.
`tau`	(Optional) A restriction time for computing restricted survival time. If provided, the function evaluates predictions up to this time point. Default is 10e10.
`new_data`	(Optional) A new dataset for evaluating the model. If NULL, the function uses the training data stored in model.
`predict`	(Optional) A logical value indicating whether to return individual predictions. Default is TRUE.

Details

With ‘predict = FALSE‘, R-squared metrics return missing values with a warning if censoring weights cannot be normalized, for example when the training censoring survival is zero at an observed evaluation event. Individual predictions remain available with ‘predict = TRUE‘.

Value

A **named list** of model-specific prediction objects. Each element of the returned list corresponds to one fitted model and contains:

- `model` – the fitted survival model object.
- `times` – numeric vector of observed follow-up times for the evaluation dataset.
- `status` – event indicators (1 = event, 0 = censored).
- `surv_prob` – an $n \times K$ matrix of predicted subject-specific survival probabilities on a common time grid.
- `pred` – predicted mean survival time (restricted or unrestricted, depending on the function).
- `covs` – character vector of covariate names used for prediction.
- `new_data` – dataset on which the predictions were computed.

Examples

```
data(pbc, package = "survival")
pbc <- pbc[!is.na(pbc$trt), ]
pbc$log_albumin <- log(pbc$albumin)
pbc$log_bili    <- log(pbc$bili)
pbc$log_protime <- log(pbc$protime)
pbc$status <- ifelse(pbc$status == 2, 1, 0)
covs <- c("age", "log_albumin", "log_bili", "log_protime", "edema")
df <- pbc[, c("time", "status", covs)]
df <- df[stats::complete.cases(df), ]

set.seed(2025)
idx <- sample(seq_len(nrow(df)), round(2 / 3 * nrow(df)))
train_data <- df[idx, ]
test_data  <- df[-idx, ]

m.wei <- survival::survreg(survival::Surv(time, status) ~ .,
                           data = train_data, dist = "weibull",
                           x = TRUE, y = TRUE)
wei_pred <- pa_predict_survreg(model = m.wei, covs = covs,
                                   new_data = test_data)
str(wei_pred$pred)
```

`pa_sample_design` *Summarize multiple two-phase (case-cohort) models into a wide table*

Description

For a list of models (each providing ‘time‘, ‘status‘, ‘surv_prob‘, ‘pred‘), calls ‘pa_evaluate_two_phase()‘ for each and returns a wide comparison table with metrics as rows and model names as columns.

Usage

```
pa_sample_design(
  models,
  case_weights,
  km_cens,
  metrics = c("pseudo_r2", "harrell_c", "uno_c", "brier_score", "td_auc", "ici"),
  t_star = NULL,
  tau = NULL,
  digits = 2,
  calibration_args = list()
)
```

Arguments

models	A **named list** in which each element corresponds to a fitted model. Each model entry must itself be a list containing: • times – numeric vector of observed follow-up times. • surv_prob – an $n \times K$ matrix (or data frame) of subject-specific predicted survival probabilities on a common time grid. • status – event indicator (1 = event, 0 = censored). • pred – (optional) predicted mean survival time (restricted or unrestricted). • new_data – (optional) dataset used for prediction. • covs – character vector of covariate names used for prediction. • model – (optional) the underlying fitted survival model object. Note: If pred is provided, it will be used to calculate R2 and concordance measure.
case_weights	Numeric vector of inverse-inclusion sampling weights, aligned with each model's evaluation subjects.
km_cens	A Kaplan-Meier fit for censoring (based on training data) used to compute IPCW.
metrics	Character vector of metrics to compute (default shown below).
t_star	Optional numeric scalar, specify the time point to evaluate Brier score and AUC.Default is median of observation time.
tau	Optional numeric scalar, specify the max time horizon for R2 measure and concordance measure (default = 10e10).
digits	Integer number of decimal places to round values (default 2).
calibration_args	Optional named list passed to [pa_calibration()], e.g. list(nk = 3). Weights come from case_weights.

Value

A data.frame: rows = metrics, columns = model names.

Examples

```
d <- pa_sim_cox_weibull(n = 150, pi_c = 0.3, v = 2,
                        beta = c(0.5, -0.5), seed = 2025)
d <- d[, c("time", "status", "x1", "x2")]
fit <- survival::coxph(survival::Surv(time, status) ~ x1 + x2,
                       data = d, x = TRUE, y = TRUE)
pred <- pa_predict_coxph(model = fit, covs = c("x1", "x2"), new_data = d)

# case-cohort sampling weights, and the censoring distribution they need
set.seed(2025)
in_subcohort <- stats::rbinom(nrow(d), 1, 0.4) == 1
w <- pa_case_cohort_weights(d$time, d$status, subcohort = in_subcohort)
keep <- which(in_subcohort | d$status == 1)
keep <- keep[order(d$time[keep])]
pred <- pa_predict_coxph(fit, c("x1", "x2"), new_data = d[keep, ])
w <- w[keep]
km <- survival::survfit(survival::Surv(d$time, 1 - d$status) ~ 1)

# the wide summary of the same two-phase evaluation
pa_sample_design(models = list(cc = pred), case_weights = w, km_cens = km)
```

`pa_sim_cox_weibull` *Simulate Cox survival data with optional interaction terms and targeted censoring*

Description

Generates survival data under a Cox proportional hazards model with a Weibull baseline cumulative hazard $H_0(t) = (0.5t)^v$. Event times are sampled via inverse transform:

$$Y = H_0^{-1}(-\log U \cdot \exp(-\beta^\top Z)), \quad U \sim \mathrm{Unif}(0,1),$$

where $H_0^{-1}(s) = 2s^{1/v}$.

Covariates are simulated from a standard normal distribution. When `interact = FALSE`, the design matrix Z contains only main effects. When `interact = TRUE`, the function automatically detects whether `length(beta)` equals 3 or 7:

- If `length(beta) = 3`, two covariates (x_1, x_2) and their two-way interaction (x_1x_2) are included.
- If `length(beta) = 7`, three covariates (x_1, x_2, x_3), their three two-way interactions (x_1x_2, x_1x_3, x_2x_3), and their three-way interaction $(x_1x_2x_3)$ are included.

Right-censoring times are generated as $C_i = \exp(\mu + r_i)$ with $r_i \sim \mathcal{N}(0, \mathrm{sd}^2)$. By default, `sd` is set to $\mathrm{sd}(\log Y)$, and `mu` is calibrated so that the expected censoring proportion equals `pi_c`. If `mu` and/or `sd` are supplied, calibration is skipped accordingly:

- **mu and sd provided:** use them directly (no calibration).
- **sd only:** solve for `mu` using the given `sd`.
- **mu only:** use the given `mu` and estimate $\mathrm{sd}(\log Y)$.
- **neither:** estimate both $\mathrm{sd}(\log Y)$ and `mu`.

Usage

```
pa_sim_cox_weibull(
  n,
  pi_c,
  v,
  beta,
  mu = NULL,
  sd = NULL,
  seed = NULL,
  interact = FALSE,
  nonlinear = FALSE
)
```

Arguments

n	Integer. Sample size.
pi_c	Numeric in $[0, 0.99]$. Target censoring proportion. Ignored if both mu and sd are provided.
v	Positive numeric. Weibull shape in $H_0(t) = (0.5t)^v$.
beta	Numeric vector of regression coefficients. If interact = FALSE, its length defines the number of main effects. If interact = TRUE, its length must be either 3 (two main + one interaction) or 7 (three main + four interactions).
mu	Optional numeric. Mean of $\log C$. If supplied with sd, no calibration is performed.
sd	Optional positive numeric. Standard deviation used in $\log C = \mu + r_i$. If NULL, defaults to $\mathrm{sd}(\log Y)$.
seed	Optional integer seed for reproducibility.
interact	Logical (default FALSE). If TRUE, include pre-defined interaction terms as described above.
nonlinear	Logical. If 'TRUE', generates two covariates plus their interaction and a quadratic term; 'beta' must then have length 4.

Value

A data.frame with columns:

- time: observed time $T_i = \min(Y_i, C_i)$.
- status: event indicator $1\{Y_i \leq C_i\}$.
- x1, x2, ...: simulated covariates.
- x1:x2, x1:x3, ...: interaction terms (if interact = TRUE).
- y_true: latent event time Y_i.
- cens_time: censoring time C_i.

Attributes include:

- mu: censoring shift used.

- sd_log: standard deviation used in $\log C$.
- sd_logY: $\mathrm{sd}(\log Y)$ from the generated events.
- pi_c_target: requested censoring rate.
- pi_c_observed: realized censoring rate.
- v, beta: model parameters.
- interact: logical flag indicating whether interactions were included.
- nonlinear: logical flag indicating whether polynomial and interactions term were included.

Examples

```
# (1) Main effects only
set.seed(1)
d0 <- pa_sim_cox_weibull(500, 0.3, v = 1.2, beta = c(0.4, -0.2))

# (2) Two covariates + their interaction
d1 <- pa_sim_cox_weibull(500, 0.3, v = 1.2,
                              beta = c(0.5, -0.3, 0.2),
                              interact = TRUE)

# (3) Three covariates + all two-way and three-way interactions
b <- c(0.4, -0.2, 0.1, 0.15, -0.05, 0.08, 0.02)
d2 <- pa_sim_cox_weibull(500, 0.3, v = 1.0, beta = b, interact = TRUE)
```

---

pa_simulate_fine_gray *Simulate data from a two-cause Fine-Gray model*

Description

This function generates a simulated dataset for a competing risks model with two causes, following a Fine-Gray-like model specification.

Usage

```
pa_simulate_fine_gray(
  n,
  v,
  beta1,
  beta2,
  lambda1 = 1,
  X = NULL,
  mu = 0,
  p = 0.7,
  c_scale = 1,
  censor = 0,
  sd.time = NULL,
```

```
  mu.c.time = NULL,
  independent_c = TRUE,
  report.mu_and_sd = FALSE,
  seed = NULL
)
```

Arguments

n	An integer specifying the number of subjects to simulate.
v	A numeric value for the Weibull shape parameter.
beta1	A numeric vector of coefficients for cause 1.
beta2	A numeric vector of coefficients for cause 2.
lambda1	A numeric value for the Weibull scale parameter for cause 1.
X	A matrix of covariates, if not provided, covariates are generated from a standard normal distribution.
mu	A numeric value for the mean of the generated covariates.
p	A numeric value for the marginal probability of a cause 1 event.
c_scale	A numeric value to scale the standard deviation of censoring times.
censor	A numeric value representing the desired censoring proportion.
sd.time	A numeric value for the standard deviation of censoring times. If NULL, it's calculated based on event times.
mu.c.time	A numeric value for the mean of censoring times. If NULL and 'censor > 0', it is calculated using 'uniroot' to achieve the desired censoring.
independent_c	A logical value. If TRUE, censoring times are independent of covariates.
report.mu_and_sd	A logical value. If TRUE, a summary of censoring parameters and event time quantiles is returned instead of the full dataset.
seed	Optional integer. If supplied, the random number generator is seeded with it for reproducibility. The default NULL uses the current random number stream without resetting it; random draws advance its state.

Value

A data frame containing the simulated data with columns 'obs.times', 'obs.event', and the covariates 'X'. Alternatively, if 'report.mu_and_sd' is TRUE, it returns a data frame with censoring parameters and event time quantiles.

Examples

```
# Simulate a dataset with 100 subjects
sim_data <- pa_simulate_fine_gray(
  n = 100, v = 1, beta1 = c(0.5, -0.2), beta2 = c(-0.3, 0.4)
)
head(sim_data)
```

```
# Simulate with a specific censoring proportion and report parameters
params_summary <- pa_simulate_fine_gray(
  n = 1000, v = 1, beta1 = c(0.5, -0.2), beta2 = c(-0.3, 0.4),
  censor = 0.2, report.mu_and_sd = TRUE
)
params_summary
```

`pa_summarize` *Summarize multiple survival models into a wide comparison table*

Description

Calls `pa_survival_eval()` for each model in a named list and outputs a wide table: each row corresponds to one metric and each column corresponds to a model.

Usage

```
pa_summarize(
  models,
  metrics = NULL,
  t_star = NULL,
  tau = 1e+11,
  digits = 2,
  calibration_args = list()
)
```

Arguments

`models`	A **named list** in which each element corresponds to a fitted model. Each model entry must itself be a list containing: • `times` – numeric vector of observed follow-up times. • `surv_prob` – an $n \times K$ matrix (or data frame) of subject-specific predicted survival probabilities on a common time grid. • `status` – event indicator (1 = event, 0 = censored). • `pred` – (optional) predicted mean survival time (restricted or unrestricted). • `new_data` – (optional) dataset used for prediction. • `covs` – character vector of covariate names used for prediction. • `model` – (optional) the underlying fitted survival model object. Note: If `pred` is provided, it will be used to calculate R2 and concordance measure.
`metrics`	Optional character vector of metrics to compute (passed through).
`t_star`	Optional evaluation time for point metrics and calibration. Defaults to the median observed follow-up.
`tau`	Optional numeric scalar, specify the max time horizon for R2 measure and concordance measure (default = 10e10).

`digits` Number of decimal places to round the metric values (default = 2).

`calibration_args`
Optional named list passed to [pa_calibration()].

Value

A data frame with:

- Each row = metric
- Each column = model name
- Cell values = corresponding rounded metric values

Examples

```
d <- pa_sim_cox_weibull(n = 150, pi_c = 0.3, v = 2,
                        beta = c(0.5, -0.5), seed = 2025)
d <- d[, c("time", "status", "x1", "x2")]
fit <- survival::coxph(survival::Surv(time, status) ~ x1 + x2,
                       data = d, x = TRUE, y = TRUE)
pred <- pa_predict_coxph(model = fit, covs = c("x1", "x2"), new_data = d)

# one column per model, metrics down the rows
pa_summarize(list(cox = pred))
```

---

pa_summarize_cr *Summarize multiple competing-risks models into a wide comparison table*

Description

Calls `pa_survival_eval_cr()` for each model in a named list and returns a wide table where rows are metrics and columns are model names.

Usage

```
pa_summarize_cr(
  models,
  metrics = NULL,
  t_star = NULL,
  tau = NULL,
  event_type = 1,
  digits = 2,
  censoring = c("cox", "km"),
  censor_covariates = NULL,
  calibration_args = list()
)
```

Arguments

`models`	A **named list**; each element is a list with components: • `pred_cif` – numeric matrix of predicted CIF values (rows: time grid; cols: subjects) • `time.cif` – numeric vector of the time grid corresponding to `pred_cif` rows • `event_time` – numeric vector of observed times • `status` – integer vector of event codes (0=censoring; `event_type`=target; others=competing)
`metrics`	Character vector of metrics to compute (passed through to `pa_survival_eval_cr()`); use `"all"` for all supported.
`t_star`	(Optional) Evaluation time for Brier score, AUC, point R2, and calibration. Defaults to the median observed follow-up.
`tau`	(Optional) The maximum prediction time horizon for R2 and C index. Default is maximum of observation time.
`event_type`	Integer code for the cause of interest (passed through).
`digits`	Number of decimal places to round the metric values (default = 2).
`censoring`	Censoring estimator for generalized R-squared metrics, ‘"cox"‘ (default) or ‘"km"‘; passed to [pa_survival_eval_cr()].
`censor_covariates`	Optional numeric censoring-model covariates shared across models. Rows must match each model’s subject order. Passed to [pa_survival_eval_cr()]; used only with ‘censoring = "cox"‘.
`calibration_args`	Optional named list passed to [pa_calibration_cr()].

Details

See [pa_survival_eval_cr()] for the event-observation weighting rule, original-time support requirements and probability floor.

Value

A data.frame with:

- each row = metric
- each column = model name
- cell values = metric values

Examples

```
# Competing risks: cause 1 is the event of interest, cause 2 the competing
# risk. Both cause-specific hazards are modelled.
cr <- pa_simulate_fine_gray(n = 150, v = 2, beta1 = c(0.5, -0.5),
                            beta2 = c(-0.3, 0.3), censor = 0.3, seed = 2025)
cr$time <- cr$obs.times
```

```
cr$status <- cr$obs.event
m1 <- survival::coxph(survival::Surv(time, status == 1) ~ X1 + X2,
                      data = cr, x = TRUE, y = TRUE)
m2 <- survival::coxph(survival::Surv(time, status == 2) ~ X1 + X2,
                      data = cr, x = TRUE, y = TRUE)
p <- pa_predict_cif(model1 = m1, model2 = m2, newdata = cr,
                    covs = c("X1", "X2"), event.type = 1, tau = max(cr$time))

pa_summarize_cr(list(csh = p), event_type = 1)
```

`pa_survival_eval` *Compute Survival Model Evaluation Metrics*

Description

This function calculates various predictive performance metrics for survival models based on predicted survival probabilities and observed survival data. It supports a range of evaluation measures, including explained variation, concordance indices, Brier Score, and time-dependent AUC.

Usage

```
pa_survival_eval(
  model,
  event_time,
  predicted_probability,
  pred_mean_survival = NULL,
  status,
  covariates,
  new_data = NULL,
  metrics = NULL,
  t_star = NULL,
  tau = NULL,
  calibration_args = list()
)
```

Arguments

`model` A fitted survival model object. Optional for the metrics computed directly from `predicted_probability`.

`event_time` A numeric vector of observed survival times.

`predicted_probability`
Numeric matrix of predicted survival probabilities, with

- rows = subjects.
- columns = sorted observed follow-up times, including duplicates

`pred_mean_survival`
Optional numeric vector of predicted mean survival times, used for explained variation and concordance. Predicted probabilities are still needed for point metrics, Brier score, AUC, and calibration.

`status` A numeric vector indicating event occurrence (1 for event, 0 for censoring).

`covariates` A character vector specifying the names of the covariates used in the model.

`new_data` Optional data frame used by metrics that require refitting or prediction from `model`. If supplied, it should contain the variables needed by those procedures and columns `time` and `status` coded as above.

`metrics` A character vector specifying the evaluation metrics to compute. Options include:

- "pseudo_r2" - Pseudo R-squared measure
- "r_square" - Explained variation R^2
- "l_square" - L-squared measure
- "Harrell's C" - Harrell's concordance index
- "Uno's C" - Uno's concordance index
- "brier_score" - Overall probability prediction error
- "td_auc" - Time-dependent area under the curve (AUC)
- "ici", "e50", "e90" - Absolute calibration error summaries

The default `NULL` includes ICI but omits E50 and E90. Use `"all"` to include every supported metric.

`t_star` (Optional) Evaluation time for point R-squared, Brier score, time-dependent AUC, and calibration. If NULL, it defaults to the median survival time.

`tau` (Optional) A numeric value specifying the truncation time for calculating explained variation metrics. If NULL, it defaults to the maximum observed survival time.

`calibration_args`
Optional named list passed to [pa_calibration()], e.g. `list(nk = 3)`. These settings affect calibration only. For sampling weights use [pa_sample_design()] or [pa_calibration()] directly, rather than passing weights through this list.

Details

Point R-squared uses the event indicator 'I(T <= t_star)' and the predicted event probability '1 - S' at the selected prediction-grid column. Events at the evaluation time are included. Calibration uses `1 - S(t_star)` from the last supplied grid column at or before `t_star`, interpreting the grid as a right-continuous step function (risk zero before its first time). Its outcome horizon is exactly `t_star`, not the nearest future grid time. See [pa_calibration()] for the estimator and boundary-probability policy. A failed calibration fit warns and returns missing calibration metrics; other requested metrics are retained.

Value

A data frame with two columns:

`Metric` The name of the computed evaluation metric.

`Value` The corresponding computed value.

References

Schemper, M. and R. Henderson (2000). Predictive accuracy and explained variation in Cox regression. Biometrics 56, 249–255.

Lusa, L., R. Miceli and L. Mariani (2007). Estimation of predictive accuracy in survival analysis using R and S-PLUS. Computer Methods and Programs in Biomedicine 87, 132–137.

Potapov, S., Adler, W., Schmid, M., Bertrand, F. (2024). survAUC: Estimating Time-Dependent AUC for Censored Survival Data. R package version 1.3-0. DOI: doi:10.32614/CRAN.package.survAUC. Available at https://CRAN.R-project.org/package=survAUC.

Examples

```
# A Cox model evaluated on right-censored data. The simulator is seeded,
# so the reported metric values are reproducible.
d <- pa_sim_cox_weibull(n = 150, pi_c = 0.3, v = 2,
                        beta = c(0.5, -0.5), seed = 2025)
d <- d[, c("time", "status", "x1", "x2")]
fit <- survival::coxph(survival::Surv(time, status) ~ x1 + x2,
                       data = d, x = TRUE, y = TRUE)
pred <- pa_predict_coxph(model = fit, covs = c("x1", "x2"), new_data = d)

# the default metric set
pa_survival_eval(
  model = fit, event_time = pred$times,
  predicted_probability = pred$surv_prob, status = pred$status,
  covariates = c("x1", "x2"), new_data = d
)

# a chosen subset; see pa_metric_names() for the accepted identifiers
pa_survival_eval(
  model = fit, event_time = pred$times,
  predicted_probability = pred$surv_prob, status = pred$status,
  covariates = c("x1", "x2"), new_data = d,
  metrics = c("harrell_c", "brier_score")
)
```

`pa_survival_eval_cr` *Evaluate Survival Prediction Performance with Competing Risks*

Description

This function evaluates the predictive performance of a survival model in the presence of competing risks. It calculates various performance metrics including Pseudo R-squared, R-squared, L-squared, C-index, Brier Score, and Time-Dependent AUC.

Usage

```
pa_survival_eval_cr(
  pred_cif,
  event_time,
  time.cif,
  status,
  metrics = NULL,
  t_star = NULL,
  tau = NULL,
  event_type = 1,
  censoring = c("cox", "km"),
  censor_covariates = NULL,
  calibration_args = list()
)
```

Arguments

pred_cif	Numeric matrix of cumulative incidence functions with one row per value of time.cif and one column per subject.
event_time	A numeric vector of observed event or censoring times for subjects.
time.cif	A numeric vector of time points corresponding to the predicted CIFs.
status	A numeric vector indicating the event type for each subject: by default • '0' for censored cases. • '1' for the primary event of interest. • '2' (or other values) for competing events.
metrics	A character vector specifying the performance metrics to calculate. The default NULL computes pseudo R-squared (integrated and point), C-index, Brier score, time-dependent AUC, and ICI. Use "all" to include the components and E50/E90 as well. Valid options include: • '"pseudo_r2"' • '"r_square"' • '"l_square"' • '"pseudo_r2_point"' • '"r2_point"' • '"l2_point"' • '"c_index"' • '"brier_score"' • '"td_auc"' • '"ici"', '"e50"', '"e90"'
t_star	Optional numeric value specifying a reference time for evaluating point-version Pseudo_R_square, the Brier score, time-dependent AUC, and calibration. Defaults to the median of 'event_time'.
tau	Optional numeric value specifying a restricted time horizon for model evaluation. Defaults to the maximum observed follow-up time.

`event_type` An integer indicating the primary event type of interest (default = ‘1‘).

`censoring` Censoring estimator for the generalized R-squared metrics: ‘"cox"‘ (default, optionally conditional on ‘censor_covariates‘) or ‘"km"‘. Other metrics retain their own censoring adjustments.

`censor_covariates`
Optional numeric vector, matrix or data frame of censoring-model covariates, in the same subject order as ‘event_time‘ and the columns of ‘pred_cif‘. Requires ‘censoring = "cox"‘. With ‘NULL‘, the Cox censoring model has no covariates. The Cox censoring model uses Breslow ties, so without covariates it equals exp(-Nelson-Aalen).

`calibration_args`
Optional named list passed to [pa_calibration_cr()], e.g. `list(bandwidth = 0.2)`. These settings affect calibration only.

Details

The generalized R-squared metrics use event-observation IPCW. Censoring is estimated on the original follow-up times, treating any observed event type as an observed event. The unnormalized weight is $I(status_i > 0)/\widehat{G}(Y_i- \mid X_i)$; weights are normalized to sum to one. A censored subject has zero weight even if follow-up exceeds ‘tau‘. Restricting the working outcome therefore does not limit censoring support requirements to ‘tau‘: support at contributing original event times is needed. Cox weights require conditional independent censoring and a valid censoring model; marginal weights require the corresponding marginal independence. Censoring survival probabilities are floored at 10^{-4}. The interval outcome is the target-event time capped at ‘tau‘, with competing events assigned to ‘tau‘. The point outcome is ‘I(T <= t_star, D = event_type)‘, including target events at the evaluation time, as in the CIF. Normalized ratio estimates are not asserted to be exactly unbiased in finite samples. Calibration uses the target-event CIF at or before `t_star` on the supplied grid and evaluates the outcome at `t_star`. It uses the local Aalen–Johansen estimator described in [pa_calibration_cr()]; `censoring` and `censor_covariates` do not control this estimator. A failed curve fit warns and returns missing calibration metrics without discarding others.

Value

A data frame with two columns:

- ‘"Metric"‘: Name of the evaluated metric.
- ‘"Value"‘: Computed value of the corresponding metric.

Examples

```
# Competing risks: cause 1 is the event of interest, cause 2 the competing
# risk. Both cause-specific hazards are modelled.
cr <- pa_simulate_fine_gray(n = 150, v = 2, beta1 = c(0.5, -0.5),
                            beta2 = c(-0.3, 0.3), censor = 0.3, seed = 2025)
cr$time <- cr$obs.times
cr$status <- cr$obs.event
m1 <- survival::coxph(survival::Surv(time, status == 1) ~ X1 + X2,
                      data = cr, x = TRUE, y = TRUE)
m2 <- survival::coxph(survival::Surv(time, status == 2) ~ X1 + X2,
```

```
                       data = cr, x = TRUE, y = TRUE)
p <- pa_predict_cif(model1 = m1, model2 = m2, newdata = cr,
                    covs = c("X1", "X2"), event.type = 1, tau = max(cr$time))

pa_survival_eval_cr(
  pred_cif = p$cif_pred[, -1], event_time = p$times,
  time.cif = p$cif_pred[, 1], status = p$status, event_type = 1
)
```

Index

∗ **datasets**
moore, 3

moore, 3

pa_calibration, 3
pa_calibration_cr *(*pa_calibration*)*, 3
pa_case_cohort_weights, 6
pa_evaluate_two_phase, 6
pa_fit_and_eval, 8
pa_metric_names, 10
pa_nested_case_control_weights, 10
pa_nlm, 11
pa_plot_pred, 13
pa_plot_summary, 14
pa_predict_cif, 16
pa_predict_coxph, 18
pa_predict_survreg, 20
pa_sample_design, 21
pa_sim_cox_weibull, 23
pa_simulate_fine_gray, 25
pa_summarize, 27
pa_summarize_cr, 28
pa_survival_eval, 30
pa_survival_eval_cr, 32
pam.nlm *(*pa_nlm*)*, 11