Package: prsladder (via r-universe)

November 12, 2026

Title Polygenic Risk Scores and Time to Disease Progression

Version 2026.9.23

Description Measures the association of one or more polygenic risk scores with time to a disease event among persons who have a first diagnosis. Fits a ladder of Cox proportional-hazards models over all subsets of the scores, compares nested models, predicts absolute risk at a horizon with delayed entry, and evaluates discrimination, calibration, and reclassification. With competing events, fits cause-specific hazard models and estimates the cumulative incidence. Draws risk curves with risk tables, hazard ratios across the ladder, and calibration and discrimination, and makes tables of risk by score group. Estimates standardized risk at score percentiles, risk ratios, case shares, and time lost by score group, and net benefit. Includes a simulator with known hazard ratios.

URL https://brettklamer.com/work/prsladder/, https://bitbucket.org/bklamer/prsladder/

License MIT + file LICENSE

Depends R (>= 4.5.0)

Imports ggplot2 (>= 4.0.0), parallel, patchwork (>= 1.3.0), Rdpack, splines, stats, survival (>= 3.8-4), utils

Suggests gtsummary (>= 2.6.1), knitr, rmarkdown, tinytest

RdMacros Rdpack

Language en-US

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Encoding UTF-8

Config/roxygen2/version 8.1.1

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

Date/Publication 2026-11-12 01:33:12 UTC

RemoteUrl https://bitbucket.org/bklamer/prsladder

RemoteRef HEAD

RemoteSha 9ec3ca71663062c3ef43efd054d56408a7d0be29

Contents

assess_calibration . . . 3
assess_discrimination . . . 11
assess_net_benefit . . . 17
assess_stability . . . 23
assess_thresholds . . . 25
bootstrap_plan . . . 33
check_design . . . 37
check_entry . . . 40
compare_ladder . . . 42
cumulative_risk . . . 45
describe_case_mix . . . 54
describe_cohort . . . 56
describe_risk . . . 60
export_ladder . . . 62
fit_ladder . . . 67
horizon_support . . . 75
plot_calibration . . . 78
plot_discrimination . . . 80
plot_gradient . . . 82
plot_ladder . . . 83
plot_net_benefit . . . 85
plot_risk . . . 87
plot_stacked . . . 90
predict_risk . . . 91
prediction_data . . . 98
print.prs_cohort . . . 102
print.prs_export . . . 103
print.prs_ladder . . . 104
prs_cohort . . . 105
simulate_prs_data . . . 113
standardized_risk . . . 123
summary.prs_cohort . . . 130
validation . . . 132

Index **138**

assess_calibration *Assess the Calibration of a Ladder*

Description

The function compares the risks that each model of a ladder predicts with the risks observed in a cohort, at each horizon. The function makes the comparison for all persons together, along a smooth calibration curve, and within groups of predicted risk. Use it to find if a model predicts risks that are too high, too low, or too extreme for new persons. The PRS reporting standards ask for calibration in all analyses, and say that it is most relevant for persons outside the development sample (Wand et al. 2021).

The function has three modes. Without `newdata` and `validation`, it evaluates the development persons, and the estimates are apparent. With `validation`, it corrects the O/E ratio, the calibration slope, the smooth curve, and the summaries of the curve for optimism with the bootstrap. With `newdata`, it evaluates other persons. validation explains when to use each mode.

Usage

```
assess_calibration(
  ladder,
  horizons,
  groups = 10L,
  newdata = NULL,
  validation = NULL,
  min_events = 10L,
  min_at_risk = 50L,
  min_censoring_survival = 0.02
)
```

Arguments

`ladder`	(`prs_ladder` or `prs_export`) A ladder from `fit_ladder()` or its export from `export_ladder()`. An export holds no persons, and it needs `newdata`.
`horizons`	(`numeric()`; range `(0, Inf)`) Distinct finite times since entry, in the units of the age columns.
`groups`	(`integer(1)`; default `10L`; range `[1, Inf)`) Number of quantile groups of predicted risk. A value of `1L` puts all persons in one group. Tied risks can give a model fewer groups (see Details).
`newdata`	(`prs_cohort` or `NULL`; default `NULL`) A different cohort, such as another biobank, a later period, or another ancestry group. Build it with `prs_cohort(data, reference = ladder$cohort)`, or with `prs_cohort(data, reference = ladder)` for an export. Its scores, covariates, and index must be those of the development cohort. If `NULL`, the function evaluates the development persons. The persons must be separate from the development persons and from the GWAS data that gave the scores (Choi et al.

2020). For a ladder, the function gives a warning with the number of persons of `newdata` whose ids are also in `ladder$cohort`. Give `newdata` or `validation`, not both.

`validation` (`prs_bootstrap_plan` or `NULL`; default `NULL`)
A plan from `bootstrap_plan()` for the optimism correction of the estimates on the development persons (see the section "Optimism correction"). If `NULL`, the function gives the apparent or the external estimates.

`min_events` (`integer(1)`; default `10L`; range `[1, Inf)`)
Minimum number of T2 events at or before a horizon.

`min_at_risk` (`integer(1)`; default `50L`; range `[1, Inf)`)
Minimum number of persons whose time since entry is at least the horizon.

`min_censoring_survival`
(`numeric(1)`; default `0.02`; range `(0, 1]`)
Minimum censoring survival $G(h)$ at a horizon h. A value of `1` requires no censoring at or before the horizon.

Details

Persons and windows:

At horizon h, the function evaluates only the persons with a prediction (see `predict_risk()`). The column `n_removed` counts the other persons. The horizon guard of `horizon_support()` applies to the evaluated persons, not to the full cohort. The function leaves out each horizon that fails the guard and gives one warning for all of them.

The window of each person starts at entry and stops at `min(stop, start + h)`. A second event is an observed event only when it occurs in the window.

Calibration-in-the-large:

The function compares the observed and expected numbers of events in the windows. This comparison adapts the observed-to-expected comparison of Crowson et al. (2016), which uses the full follow-up of each person, to windows that stop at h:

- The observed count O is the number of events in the windows. It is the same for each model.
- The expected count E is the sum over persons of the cumulative hazard of the model over the window. The cumulative hazard uses the Breslow baseline hazard of the fit. With this baseline, the sum of E over the development persons equals O also when event times are tied. The Efron baseline of `predict_risk()` gives a larger sum at tied event times. That baseline estimates the hazard of the continuous time that the ties hide. `expected_risk` keeps that baseline, because on the probability scale it matches the Kaplan-Meier `observed_risk`. Without tied event times, the two baselines are equal.
- The ratio `oe_ratio` is O/E. A ratio above one means that the model predicts too few events.
- The 95% confidence interval treats O as Poisson, $O/E \times \exp(\pm 1.96/\sqrt{O})$. The interval is on the log scale. `oe_ratio_se` is the delta-method standard error of the ratio on the ratio scale, $(O/E)/\sqrt{O}$. Because the guard requires at least `min_events` events, O is positive and the interval is finite.

The expected count starts at the entry of each person. Because of this, the ratio needs no assumption about censoring other than the assumption of the Cox model. On the development persons, the ratio is one under the diagnosis index, because every window stops at h on the time scale of

the model. Under the recruitment index, the windows stop at different times, and the ratio is close to one. The apparent interval is still useful. It covers the O/E ratio of the fitted model in the population of the development persons. In a simulation of 200 development samples, it covered that ratio in 93% to 98% of the samples. Use a plan in `validation`, or a different cohort in `newdata`, to get an informative ratio.

With competing events (see `prs_cohort()`), the window of a person with a competing event stops at that event. The expected count then uses the cause-specific hazard of T2 only. Because of this, the ratio measures the calibration of the model of T2, and it does not use the competing model. The calibration slope below also measures the model of T2.

Risk scale:

The risk-scale columns describe the same persons:

- `expected_risk` is the mean predicted risk from `predict_risk()`.
- `observed_risk` is one minus the Kaplan-Meier survival at h on time since entry.
- `observed_risk_se` is the Greenwood standard error of that estimate.

With competing events, `expected_risk` is the mean predicted cumulative incidence, which uses the models of both events. `observed_risk` is then the Aalen-Johansen cumulative incidence of T2 at h (Aalen and Johansen 1978), and `observed_risk_se` is its infinitesimal jackknife standard error. The comparison of these two risks within groups of predicted risk is the calibration check of Wolbers et al. (2009).

The observed risk ignores the predictors. It assumes that censoring on time since entry is independent of the scores and the covariates.

Under the recruitment index, the incident and the prevalent persons can have different censoring on time since entry (see `assess_discrimination()`). When the evaluated persons hold both values of `t1_status`, the observed risk mixes the risks of the two values. Let $\hat{F}_s(h)$ be the Kaplan-Meier or Aalen-Johansen risk of the n_s persons with value s of `t1_status`, and SE_s its standard error. The observed risk and its standard error are

$$\hat{F} = \sum_s \frac{n_s}{n} \hat{F}_s(h),$$

$$\mathrm{SE}(\hat{F}) = \sqrt{\sum_s \frac{n_s^2}{n^2} \mathrm{SE}_s^2 + \frac{1}{n} \sum_s \frac{n_s}{n} \left(\hat{F}_s(h) - \hat{F} \right)^2}.$$

The first term of the variance treats the sizes n_s as constants. The second term is the variance of the shares n_s/n. When the mixture is not `NA`, it equals the mean of the censoring-weighted events, with one censoring distribution for each value of `t1_status`. It assumes that censoring is independent of the entry within each value.

Groups of predicted risk:

For each model, the function puts the persons into `groups` quantile groups of predicted risk. Group 1 holds the lowest risks. In each group, `observed_risk` is the observed risk of the persons in the group, from the same estimator as for all persons. Without competing events, this comparison is the grouped Kaplan-Meier calibration check of Crowson et al. (2016). It is zero for a group with no events in the windows. It is `NA` when nobody in the group is at risk at h and a person is censored at the last time, because no estimate then exists at h. A group whose persons all have T2 or a competing event before h keeps its estimate. Without competing events, this estimate is one. Under the recruitment index, the observed risk of a group mixes the risks of its incident and its

prevalent persons. The weights of the mixture are their shares in the group. It is then also NA when this rule gives NA for the incident or the prevalent persons of the group. The same rules apply to the observed risk of all persons.

Tied risks can make two or more quantiles equal, for example in a model with only a few distinct predicted risks. The groups between equal quantiles are then empty, and the function removes them. Persons with the same risk are always in the same group. That model has fewer than `groups` groups at that horizon. The function gives one warning that names each such model and horizon. The other models keep `groups` groups.

Smooth calibration curve:

The smooth curve gives the observed risk as a function of the predicted risk (Austin et al. 2020). Let $\hat{p}_i$ be the predicted risk of person i at h, and $x_i = \log(-\log(1 - \hat{p}_i))$. The function fits a Cox model of the windows on time since entry, with a natural spline of x. The spline has an inner knot at the 50th percentile of x and outer knots at the 10th and 90th percentiles. This spline is the restricted cubic spline with 3 knots of Austin et al. (2020). The observed risk at $\hat{p}$ is

$$1 - \exp\{-\hat{H}(h)\, e^{\hat{g}(x)}\}.$$

Here $\hat{H}$ is the Efron baseline cumulative hazard and $\hat{g}$ is the spline of this model. Austin et al. (2020) fit the model to the full follow-up and take the curve at h. This function fits it to the windows that stop at h, as the O/E ratio and the slope do. The proportional-hazards assumption of the model then must hold only up to h.

Under the recruitment index with both values of `t1_status`, the Cox model has one baseline $\hat{H}_s$ for each value s. A logistic model of `t1_status` on the same spline of x gives the probability $\hat{P}(s \mid x)$ of each value. The observed risk at $\hat{p}$ is then the mixture

$$\sum_s \hat{P}(s \mid x) \left[1 - \exp\{-\hat{H}_s(h)\, e^{\hat{g}(x)}\}\right].$$

This mixture is the observed risk of all persons with the predicted risk $\hat{p}$. The predicted risk can separate the incident and the prevalent persons, for example when the hazard rises with time since T1. The logistic model then gives $\hat{P}(s \mid x)$ close to 0 or 1. At such an x, the curve uses the baseline of one value of `t1_status`. The ranges of the predicted risks of the two values can then have a gap. Between the two ranges, the curve has no persons, and its shape there comes from the limit of the logistic fit.

Three summaries describe the distance between the curve and the diagonal. Let d_i be the absolute difference between the observed risk at $\hat{p}_i$ and $\hat{p}_i$. `ici` is the mean of d_i over the evaluated persons, the integrated calibration index. `e50` and `e90` are the median and the 90th percentile of d_i. The summaries use the observed risk of each person, not a grid. The element `curve` gives the curve at 200 risks evenly spaced from the 1st to the 99th percentile of the predicted risks.

The curve and its summaries are NA in these cases:

- The cohort declares competing events. The decile groups stay available.
- A predicted risk is 0 or 1, or the three knots are not distinct, for example when a model has few distinct risks.
- The Cox fit of the curve gives a warning. The function then gives one warning for all such models and horizons.
- A coefficient of the Cox fit or of the logistic fit is NA. The function gives no warning.
- The logistic fit of `t1_status` stops with an error, or gives a warning other than the warnings of separation. The function then gives one other warning for all such models and horizons.

The element curve has no rows for such a model and horizon. The summaries have no standard error or interval outside the corrected mode, and their _se, _lower, and _upper columns are then NA.

Calibration slope:

For an external cohort, the calibration slope is the coefficient of the development linear predictor η in a Cox model fitted to the windows of newdata:

- A slope of one means that the spread of the predictions is correct.
- A slope below one means that the predictions are too extreme.
- A slope above one means that the predictions are too moderate.

This slope is the Cox model form of the calibration slope of Crowson et al. (2016). The Cox model has the time scale of the fit, and with design = "index" in fit_ladder() also its strata. Its interval is the Wald interval, the slope plus or minus 1.96 standard errors. The slope is NA when the linear predictor is constant over the persons. When its Cox fit gives a warning, the slope is NA, and the function gives one warning for all such models and horizons. The slope is precise only when the linear predictor varies across persons. Its standard error is about $1/(\mathrm{SD}(\eta)\sqrt{O})$. When the terms of a model have no effect, as can happen for base, the development coefficients are noise. The true slope is then zero, and the interval can contain both zero and one. On the development persons, the slope does not show overfitting. Over all follow-up, the slope is exactly one, because the model comes from these persons. When the hazards are proportional, a window that stops at h moves the slope away from one. But its mean stays near one, also for an overfit model. In a simulation of an overfit model, the mean slope at 5 years was 0.97 on the development persons and 0.37 on new persons. Because of this, the slope columns are NA for apparent evaluation.

Optimism correction:

With a plan from bootstrap_plan() in validation, the function corrects the measures at each horizon for optimism with the bootstrap (Harrell Jr. et al. 1996). These measures are the O/E ratio, the slope, ICI, E50, E90, and the curve. bootstrap_plan() describes the method, the random number streams, and the failed replicates. The column evaluation of models is then "optimism-corrected":

- The function corrects the O/E ratio on the log scale. oe_ratio is $\exp(a - o)$, where a is the apparent log ratio and o the mean optimism of the log ratio. The rows of optimism have the measure "log_oe_ratio".
- slope is the apparent slope at the horizon minus the mean optimism. Because the window moves the apparent slope away from one, the corrected slope at a horizon holds this noise.
- ici, e50, and e90 are each the apparent value minus its mean optimism, truncated at 0. Because the absolute value is not linear, these values differ from the summaries of the corrected curve.
- The corrected curve is the apparent curve minus the mean optimism at each of its 200 points, as in rms::calibrate(). Each replicate gives two curves of the refit at each point. The optimism at the point is the observed risk on the bootstrap sample minus the observed risk on the development persons. The element curve holds the apparent curve with the label "apparent" and the corrected curve with the label "optimism-corrected", on the same points. The optimism of the curve is not in optimism. At a point with fewer than half of the replicates, the corrected curve is NA.
- The element shrinkage gives the corrected slope over all follow-up. Its apparent value is 1, because each model comes from the development persons. This slope is the shrinkage factor

of the linear predictor that new persons need (van Houwelingen and le Cessie 1990; Harrell Jr. et al. 1996). Report it as the measure of overfitting. The column `heuristic_shrinkage` of `fit_ladder()$table` gives an estimate without the bootstrap.

The function does not correct the other columns. `observed_events`, `observed_risk`, and `observed_risk_se` do not depend on a model, and `expected_events` and `expected_risk` keep their apparent values. The element `groups` holds the apparent groups, with the label `"apparent"`.

A replicate in which the Cox fit of a curve fails gives no optimism for that curve and its summaries. The other measures of the replicate stay. The column `n_boot_used` of `optimism` counts the replicates used for each row.

The standard errors and the intervals of `oe_ratio`, `slope`, `ici`, `e50`, `e90`, and `shrinkage` depend on `interval` of the plan:

- `"analytic"` and `"none"`: `NA`. The refit standard error of the slope is too large for the corrected slope. A Poisson interval of the corrected O/E ratio is not yet tested.
- `"two-stage"`: the standard error is the standard deviation of the outer corrected estimates, and the limits are their 2.5% and 97.5% quantiles. For the O/E ratio, the quantiles are on the log scale, and the standard error is on the ratio scale. The limits of `ici`, `e50`, and `e90` are truncated at 0. The curve has no interval.

Value

A list with the six elements below. Because each element has the same columns in every mode, the results of two modes stack with `base::rbind()`. The rows are in the order of horizon, then model as in `ladder$models`, then group or point of the curve. When no horizon passes the guard, `models`, `groups`, and `curve` have zero rows.

`models` One row per horizon and model, with these columns:

`model` Model name.

`horizon` Horizon h.

`evaluation` `"apparent"`, `"optimism-corrected"` with `validation`, or `"external"` with `newdata`.

`n` Number of persons evaluated.

`n_removed` Number of persons without a prediction.

`observed_events` Observed count O.

`expected_events` Expected count E.

`oe_ratio`, `oe_ratio_se`, `oe_ratio_lower`, `oe_ratio_upper` Ratio O/E, its standard error, and its 95% confidence interval.

`expected_risk` Mean predicted risk.

`observed_risk`, `observed_risk_se` Kaplan-Meier risk, or Aalen-Johansen cumulative incidence with competing events, and its standard error. Under the recruitment index, the mixture of these risks over the values of `t1_status` (see Details). `NA` when nobody is at risk at h and a person is censored at the last time, for all persons or, under the recruitment index, for the persons of one value of `t1_status`.

`slope`, `slope_se`, `slope_lower`, `slope_upper` Calibration slope, its standard error, and its 95% confidence interval, `NA` for apparent evaluation.

`ici`, `e50`, `e90` Mean, median, and 90th percentile of the absolute difference between the observed risk of the curve and the predicted risk. Each has the columns `_se`, `_lower`, and `_upper`, which are `NA` except with a two-stage plan.

`groups` One row per horizon, model, and group, with these columns:

- `model`, `horizon`, `evaluation` As in `models`.
- `group` Group number, from 1 for the lowest risks. A model with tied risks can have fewer than `groups` groups.
- `n` Number of persons in the group.
- `n_events` Number of T2 events in the windows of the group.
- `expected_risk` Mean predicted risk of the group.
- `observed_risk`, `observed_risk_se` Observed risk of the group and its standard error.

`curve` The smooth calibration curve at 200 points of each model and horizon, with the columns `model`, `horizon`, `evaluation`, `predicted`, and `observed`. With `validation`, each model and horizon has two curves on the same points, first the apparent curve and then the corrected curve. It has zero rows with competing events.

`shrinkage` One row per model with the corrected slope over all follow-up, and the columns `model`, `slope`, `slope_se`, `slope_lower`, and `slope_upper`. It has zero rows without `validation`.

`optimism` One row per corrected estimate, with the columns of the element `optimism` of `assess_discrimination()`. The rows are in the order of horizon, then the measures `"log_oe_ratio"`, `"slope"`, `"ici"`, `"e50"`, and `"e90"`, then model. The rows of `shrinkage` follow, with the measure `"shrinkage"`, the horizon `NA`, and the apparent value 1. It has zero rows without `validation`.

`settings` A list with the elements of the element `settings` of `assess_discrimination()`.

References

Aalen OO, Johansen S (1978). "An empirical transition matrix for non-homogeneous Markov chains based on censored observations." *Scandinavian Journal of Statistics*, **5**(3), 141–150.

Austin PC, Harrell Jr. FE, van Klaveren D (2020). "Graphical calibration curves and the integrated calibration index (ICI) for survival models." *Statistics in Medicine*, **39**(21), 2714–2742. doi:10.1002/sim.8570.

Choi SW, Mak TS, O'Reilly PF (2020). "Tutorial: a guide to performing polygenic risk score analyses." *Nature Protocols*, **15**(9), 2759–2772. doi:10.1038/s4159602003531.

Crowson CS, Atkinson EJ, Therneau TM (2016). "Assessing calibration of prognostic risk scores." *Statistical Methods in Medical Research*, **25**(4), 1692–1706. doi:10.1177/0962280213497434.

Harrell Jr. FE, Lee KL, Mark DB (1996). "Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors." *Statistics in Medicine*, **15**(4), 361–387. doi:10.1002/(SICI)10970258(19960229)15:4<361::AIDSIM168>3.0.CO;2-4.

van Houwelingen JC, le Cessie S (1990). "Predictive value of statistical models." *Statistics in Medicine*, **9**(11), 1303–1325. doi:10.1002/sim.4780091109.

Wand H, Lambert SA, Tamburro C, Iacocca MA, O'Sullivan JW, Sillari C, Kullo IJ, Rowley R, Dron JS, Brockman D, Venner E, McCarthy MI, Antoniou AC, Easton DF, Hegele RA, Khera AV, Chatterjee N, Kooperberg C, Edwards K, Vlessis K, Kinnear K, Danesh JN, Parkinson H, Ramos EM, Roberts MC, Ormond KE, Khoury MJ, Janssens ACJW, Goddard KAB, Kraft P, MacArthur

JAL, Inouye M, Wojcik GL (2021). "Improving reporting standards for polygenic scores in risk prediction studies." *Nature*, **591**(7849), 211–219. doi:10.1038/s41586021032436.

Wolbers M, Koller MT, Witteman JCM, Steyerberg EW (2009). "Prognostic models with competing risks: methods and application to coronary risk prediction." *Epidemiology*, **20**(4), 555–561. doi:10.1097/EDE.0b013e3181a39056.

See Also

`horizon_support()` to check horizons before the evaluation, validation for the three modes, `bootstrap_plan()` for the optimism correction, `assess_discrimination()`, and `assess_thresholds()`.

Examples

```
#-------------------------------------------------------------------------
# assess_calibration() example
#-------------------------------------------------------------------------
# Fit a ladder on a development cohort.
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
ladder <- fit_ladder(cohort)

# Build an external cohort with the rules of the development cohort.
external <- prs_cohort(
  simulate_prs_data(n = 2000L, seed = 2L, id_prefix = "E"),
  reference = cohort
)

# Assess calibration at 5 and 10 years in 5 groups of predicted risk.
calibration <- assess_calibration(
  ladder,
  horizons = c(5, 10),
  groups = 5L,
  newdata = external
)
calibration$models[c("model", "horizon", "oe_ratio", "slope", "ici")]

# Observed and expected risk by group for the model with both scores.
calibration$groups[calibration$groups$model == "onset+progression", ]

#-------------------------------------------------------------------------
# Optimism correction on the development persons
#-------------------------------------------------------------------------
# A small number of replicates keeps the example fast. An analysis needs
# 200 or more.
```

```
corrected <- assess_calibration(
  ladder,
  horizons = 10,
  validation = bootstrap_plan(n_boot = 20L, seed = 1L)
)
corrected$models[c("model", "oe_ratio", "slope", "ici", "e90")]
corrected$shrinkage
```

`assess_discrimination` *Assess the Discrimination of a Ladder*

Description

The function reports two measures of predictive accuracy for each model of a ladder at each horizon, with standard errors and 95% confidence intervals. The time-dependent AUC measures discrimination, which is how well the predicted risks rank persons with an event above persons without one. The Brier score measures overall accuracy, which is the mean squared error of the predicted risks. The function also reports the difference in each measure between pairs of models, to show the gain from added scores.

The function has three modes. Without `newdata` and `validation`, it evaluates the development persons, and the estimates are apparent. With `validation`, it corrects the apparent estimates for optimism with the bootstrap. With `newdata`, it evaluates other persons. validation explains when to use each mode.

Usage

```
assess_discrimination(
  ladder,
  horizons,
  pairs = NULL,
  newdata = NULL,
  validation = NULL,
  min_events = 10L,
  min_at_risk = 50L,
  min_censoring_survival = 0.02
)
```

Arguments

`ladder`	(`prs_ladder` or `prs_export`) A ladder from `fit_ladder()` or its export from `export_ladder()`. An export holds no persons, and it needs `newdata`.
`horizons`	(`numeric()`; range `(0, Inf)`) Distinct finite times since entry, in the units of the age columns.

`pairs` (`list` of `character(2)` or `NULL`; default `NULL`)
Pairs of models to compare, each as `c(full, reduced)` model names. The names must be in `names(ladder$models)` and must differ. Pairs need not be nested, and a pair must not occur twice. If `NULL`, the function compares every nested pair in the order of `compare_ladder()`.

`newdata` (`prs_cohort` or `NULL`; default `NULL`)
A different cohort, such as another biobank, a later period, or another ancestry group. Build it with `prs_cohort(data, reference = ladder$cohort)`, or with `prs_cohort(data, reference = ladder)` for an export. Its scores, covariates, and index must be those of the development cohort. If `NULL`, the function evaluates the development persons. The persons must be separate from the development persons and from the GWAS data that gave the scores (Choi et al. 2020). For a ladder, the function gives a warning with the number of persons of `newdata` whose ids are also in `ladder$cohort`. Give `newdata` or `validation`, not both.

`validation` (`prs_bootstrap_plan` or `NULL`; default `NULL`)
A plan from `bootstrap_plan()` for the optimism correction of the estimates on the development persons (see the section "Optimism correction"). If `NULL`, the function gives the apparent or the external estimates.

`min_events` (`integer(1)`; default `10L`; range `[1, Inf)`)
Minimum number of T2 events at or before a horizon.

`min_at_risk` (`integer(1)`; default `50L`; range `[1, Inf)`)
Minimum number of persons whose time since entry is at least the horizon.

`min_censoring_survival`
(`numeric(1)`; default `0.02`; range `(0, 1]`)
Minimum censoring survival $G(h)$ at a horizon h. A value of `1` requires no censoring at or before the horizon.

Details

Persons and outcome:

At horizon h, the function evaluates only the persons with a prediction (see `predict_risk()`). The column `n_removed` counts the other persons. The horizon guard of `horizon_support()` applies to the evaluated persons. The function leaves out each horizon that fails the guard and gives one warning for all of them.

The outcome is on time since entry, `stop - start`. At horizon h, each person is in one of three classes:

- A case has an event at or before h.
- A control is event-free at h.
- A person censored at or before h has weight zero.

Cases and controls get inverse probability of censoring weights from the Kaplan-Meier estimate of the censoring distribution, as in `assess_thresholds()`. Under the recruitment index, incident persons enter at T1 and prevalent persons enter at recruitment (see `prs_cohort()`). Because of this, the two groups of `t1_status` can have different censoring on time since entry. When the evaluated persons hold both groups, the function estimates one censoring distribution for each group. The element `censoring_strata` of `settings` records whether the evaluated cohort holds

both groups. With `prevalent = "drop"` and under the diagnosis index, the function estimates one censoring distribution for all persons.

The weights assume that censoring is independent of the scores and the covariates. Under the recruitment index, they also assume that censoring is independent of the entry within each group of `t1_status`.

With competing events (see `prs_cohort()`), a case has T2 at or before h. The controls are the persons free of both events at h and the persons with a competing event at or before h (Blanche et al. 2013). A person with a competing event gets the weight of an event at that time, and the outcome Y_i is 0.

The function computes all measures from the risks of `predict_risk()`.

Measures:

- `auc` is the weighted probability that a case has a higher predicted risk than a control (Blanche et al. 2013). Tied risks count one half. A value of 0.5 means no discrimination and a value of 1 means perfect discrimination.
- `brier` is the mean of $w_i(Y_i - r_i)^2$ over all evaluated persons (Gerds and Schumacher 2006). Here w_i is the censoring weight, Y_i is 1 for a case and 0 for a control, and r_i is the predicted risk. The mean divides by the number of evaluated persons, which includes the censored persons with weight zero. A lower value is better. The Brier score changes with the proportion of cases, which increases with h. Compare Brier scores only at the same horizon.

Differences between models:

Each difference is the measure of the full model minus the measure of the reduced model:

- A positive `delta_auc` favors the full model.
- A negative `delta_brier` favors the full model.

The function gives no p-value for a difference. For a binary outcome, a test of a difference in AUC between nested models is conservative on the development persons (Demler et al. 2012; Pepe et al. 2013). The likelihood-ratio test of `compare_ladder()` is the test of added information.

Standard errors and intervals:

Each standard error is the standard deviation of an influence function over the persons, divided by $\sqrt{n}$. The influence function of an estimate is n times the derivative of the estimate with respect to the weight of one person. This derivative includes the change in the Kaplan-Meier censoring weights. With one censoring distribution for each group of `t1_status`, the weight of a person changes only the censoring weights of its own group. For the censoring distribution, the variance of this influence function is the Greenwood variance. The Kaplan-Meier standard errors of `cumulative_risk()` and `assess_calibration()` also use the Greenwood variance. The influence function of a difference is the difference of the influence functions of the two models on the same persons. Because of this, its standard error includes the correlation between the two models. Each 95% confidence interval is the estimate plus or minus 1.96 standard errors. The function truncates the intervals of `auc` and `brier` to $[0, 1]$. A standard error of 0 gives `NA` limits, because no person of the sample can then change the estimate. The intervals of `delta_auc` and `delta_brier` are not truncated.

The standard errors treat the predicted risks as constant. This assumption holds for persons outside the development sample. On the development persons, the estimates are apparent and tend to be optimistic, and the intervals ignore the uncertainty of the fitted models.

Optimism correction:

With a plan from `bootstrap_plan()` in `validation`, the function corrects each estimate for optimism with the bootstrap (Harrell Jr. et al. 1996). `bootstrap_plan()` describes the method, the random number streams, and the failed replicates. The column `evaluation` is then `"optimism-corrected"`, and each estimate column holds the corrected estimate, the apparent estimate minus the mean optimism. The function truncates the corrected `auc` and `brier` to $[0, 1]$. Because the true value is in this range, the truncation never moves an estimate away from it. The element `optimism` holds the apparent estimate, the optimism, and its Monte Carlo standard error of each estimate. The columns `n` and `n_removed` keep their values.

The standard error and the interval depend on `interval` of the plan:

- `"analytic"`: `auc_se` and `brier_se` are the influence-function standard errors of the apparent estimates. The interval is the corrected estimate plus or minus 1.96 of them, truncated to $[0, 1]$. The differences get `NA` standard errors and limits.
- `"two-stage"`: the standard error is the standard deviation of the outer corrected estimates, and the limits are their 2.5% and 97.5% quantiles.
- `"none"`: the standard errors and limits are `NA`.

For a model with an AUC near 0.5, the corrected AUC can be too high. Its analytic interval then covers less often than 95% (see the section "Status of the intervals" of validation).

The analytic interval of a difference ignores the variance of the optimism and of the fitted models. In a simulation with an added score of no effect, it covered the true difference in only 73% of the samples. Because of this, the function gives no analytic interval for a difference. The likelihood-ratio test of `compare_ladder()` tests whether an added score improves a model. No improvement in performance is the same null hypothesis as a coefficient of zero for the added score (Pepe et al. 2013).

Value

A list with the four elements below. Because each element has the same columns in every mode, the results of two modes stack with `base::rbind()`. The rows are in the order of horizon, then model as in `ladder$models` or pair as in `pairs`. When no horizon passes the guard, the data frames have zero rows.

`models` One row per horizon and model, with these columns:

- `model` Model name.
- `horizon` Horizon h.
- `evaluation` `"apparent"`, `"optimism-corrected"` with `validation`, or `"external"` with `newdata`.
- `n` Number of persons evaluated.
- `n_removed` Number of persons without a prediction.
- `auc`, `auc_se`, `auc_lower`, `auc_upper` Time-dependent AUC, its standard error, and its 95% confidence interval.
- `brier`, `brier_se`, `brier_lower`, `brier_upper` Brier score, its standard error, and its 95% confidence interval.

`pairs` One row per horizon and pair, with these columns:

- `full`, `reduced` Model names of the pair.
- `horizon`, `evaluation`, `n`, `n_removed` As in `models`.

`delta_auc`, `delta_auc_se`, `delta_auc_lower`, `delta_auc_upper` Difference in AUC, full minus reduced, its standard error, and its 95% confidence interval.

`delta_brier`, `delta_brier_se`, `delta_brier_lower`, `delta_brier_upper` Difference in Brier score, full minus reduced, its standard error, and its 95% confidence interval.

`optimism` One row per corrected estimate, in the order of horizon, then the measures `auc`, `brier`, `delta_auc`, and `delta_brier`, then model or pair. It has zero rows without `validation`.

`model` Model name, `NA` for a pair.

`full`, `reduced` Model names of a pair, `NA` for a model.

`horizon` Horizon h.

`measure` Name of the measure.

`apparent` Apparent estimate.

`optimism` Mean optimism over the replicates used. The corrected estimate is `apparent - optimism`, truncated to the range of the measure. `NA` when fewer than half of the replicates remain.

`optimism_mcse` Monte Carlo standard error of `optimism`, the standard deviation of the optimism of the replicates divided by the square root of their number.

`n_boot_used` Integer number of replicates used.

`settings` A list with these elements:

`evaluation` The label of the mode.

`design` The argument `design` of `fit_ladder()`.

`horizons` The horizons that pass the guard.

`min_events`, `min_at_risk`, `min_censoring_survival` The guard arguments.

`censoring_strata` `"t1_status"` when the evaluated cohort holds both values of `t1_status`, else `NULL`. A horizon or a bootstrap resample whose persons hold one value uses one censoring distribution.

`validation` The plan, or `NULL` without `validation`.

`replicates` A data frame with one row per horizon and the columns `horizon`, `n_boot_used`, `failed_refit`, `failed_spline`, `failed_guard`, and `failed_risk`. The integer columns count the replicates used and the replicates that fail from each cause (see `bootstrap_plan()`). It has zero rows without `validation`.

References

Blanche P, Dartigues J, Jacqmin-Gadda H (2013). "Estimating and comparing time-dependent areas under receiver operating characteristic curves for censored event times with competing risks." *Statistics in Medicine*, **32**(30), 5381–5397. doi:10.1002/sim.5958.

Choi SW, Mak TS, O'Reilly PF (2020). "Tutorial: a guide to performing polygenic risk score analyses." *Nature Protocols*, **15**(9), 2759–2772. doi:10.1038/s4159602003531.

Demler OV, Pencina MJ, D'Agostino Sr. RB (2012). "Misuse of DeLong test to compare AUCs for nested models." *Statistics in Medicine*, **31**(23), 2577–2587. doi:10.1002/sim.5328.

Gerds TA, Schumacher M (2006). "Consistent estimation of the expected Brier score in general survival models with right-censored event times." *Biometrical Journal*, **48**(6), 1029–1040.

doi:10.1002/bimj.200610301.

Harrell Jr. FE, Lee KL, Mark DB (1996). "Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors." *Statistics in Medicine*, **15**(4), 361–387. doi:10.1002/(SICI)10970258(19960229)15:4<361::AIDSIM168>3.0.CO;2-4.

Pepe MS, Kerr KF, Longton G, Wang Z (2013). "Testing for improvement in prediction model performance." *Statistics in Medicine*, **32**(9), 1467–1482. doi:10.1002/sim.5727.

See Also

`horizon_support()` to check horizons before the evaluation, validation for the three modes, `bootstrap_plan()` for the optimism correction, `compare_ladder()` for the likelihood-ratio test of added scores, `assess_calibration()`, and `assess_thresholds()`.

Examples

```
#-------------------------------------------------------------------------
# assess_discrimination() example
#-------------------------------------------------------------------------
# Fit a ladder on a development cohort.
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
ladder <- fit_ladder(cohort)

# Build an external cohort with the rules of the development cohort.
external <- prs_cohort(
  simulate_prs_data(n = 2000L, seed = 2L, id_prefix = "E"),
  reference = cohort
)

# Assess discrimination at 5 and 10 years, and the gain from adding the
# progression score to the onset score.
discrimination <- assess_discrimination(
  ladder,
  horizons = c(5, 10),
  pairs = list(c("onset+progression", "onset")),
  newdata = external
)
discrimination$models
discrimination$pairs

#-------------------------------------------------------------------------
# Optimism correction on the development persons
```

```
#--------------------------------------------------------------------------
# A small number of replicates keeps the example fast. An analysis needs
# 200 or more.
corrected <- assess_discrimination(
  ladder,
  horizons = 10,
  validation = bootstrap_plan(n_boot = 20L, seed = 1L)
)
corrected$models
corrected$optimism
```

`assess_net_benefit` *Assess Net Benefit for Decision Curves*

Description

`assess_net_benefit()` gives the net benefit of each model of a ladder at risk thresholds, at each horizon. The net benefit weighs the true positives of a rule against its false positives. The weight of a false positive is the odds of the threshold, $p_t/(1-p_t)$. This weight is the harm of one false positive in units of the benefit of one true positive. Its reciprocal is the largest number of false positives that a clinician accepts for one true positive. A plot of net benefit against the threshold is a decision curve (Vickers and Elkin 2006).

The function also gives the net benefit of treating all persons, the standardized net benefit, and the change in net benefit between pairs of models. `plot_net_benefit()` draws the decision curves.

The function has three modes. Without `newdata` and `validation`, it evaluates the development persons, and the estimates are apparent. With `validation`, it corrects the apparent estimates for optimism with the bootstrap. With `newdata`, it evaluates other persons. validation explains when to use each mode.

Usage

```
assess_net_benefit(
  ladder,
  horizons,
  thresholds,
  pairs = NULL,
  newdata = NULL,
  validation = NULL,
  min_events = 10L,
  min_at_risk = 50L,
  min_censoring_survival = 0.02
)
```

Arguments

ladder	(prs_ladder or prs_export) A ladder from fit_ladder() or its export from export_ladder(). An export holds no persons, and it needs newdata.
horizons	(numeric(); range (0, Inf)) Distinct finite times since entry, in the units of the age columns.
thresholds	(numeric(); range (0, 1)) Strictly increasing risk thresholds p_t. The function applies the same thresholds at every horizon. The argument has no default. See the section "Thresholds" in Details.
pairs	(list of character(2) or NULL; default NULL) Pairs of models to compare, each as c(full, reduced) model names. The names must be in names(ladder$models) and must differ. Pairs need not be nested, and a pair must not occur twice. If NULL, the function compares every nested pair in the order of compare_ladder().
newdata	(prs_cohort or NULL; default NULL) A different cohort, such as another biobank, a later period, or another ancestry group. Build it with prs_cohort(data, reference = ladder$cohort), or with prs_cohort(data, reference = ladder) for an export. Its scores, covariates, and index must be those of the development cohort. If NULL, the function evaluates the development persons. The persons must be separate from the development persons and from the GWAS data that gave the scores (Choi et al. 2020). For a ladder, the function gives a warning with the number of persons of newdata whose ids are also in ladder$cohort. Give newdata or validation, not both.
validation	(prs_bootstrap_plan or NULL; default NULL) A plan from bootstrap_plan() for the optimism correction of the estimates on the development persons (see the section "Optimism correction"). If NULL, the function gives the apparent or the external estimates.
min_events	(integer(1); default 10L; range [1, Inf)) Minimum number of T2 events at or before a horizon.
min_at_risk	(integer(1); default 50L; range [1, Inf)) Minimum number of persons whose time since entry is at least the horizon.
min_censoring_survival	(numeric(1); default 0.02; range (0, 1]) Minimum censoring survival $G(h)$ at a horizon h. A value of 1 requires no censoring at or before the horizon.

Details

Persons and weights:

The function evaluates the persons of assess_thresholds(), with the same horizon guard and the same censoring weights. See the section "Persons and weights" of assess_thresholds(). Under the recruitment index, the weights use one censoring distribution for each group of t1_status, as in assess_discrimination(). With competing events, a person with a competing event by

the horizon is a non-event. The prevalence is then the cumulative incidence of T2 by the horizon (Vickers et al. 2008; van Geloven et al. 2022).

The horizon is time since entry, as in the other `assess_*()` functions. Under the recruitment index or with a positive `entry_lag` in `prs_cohort()`, this window differs from the time since T1 of `cumulative_risk()`.

Net benefit:

Let w_i be the weight of person i. Let e_i and c_i be the indicators of an event and a non-event. Let f_i be the indicator of a risk at or above the threshold p_t. With n evaluated persons and the prevalence $\rho = n^{-1} \sum_i w_i e_i$:

$$NB = \frac{1}{n} \left[\sum_i w_i e_i f_i - \frac{p_t}{1 - p_t} \sum_i w_i c_i f_i \right],$$

$$NB_{\text{all}} = \rho - (1 - \rho) \frac{p_t}{1 - p_t}.$$

Because the weights sum to n, the net benefit also equals $\rho\,\text{TPR} - (1 - \rho)\,\text{FPR}\, p_t/(1 - p_t)$. Here, TPR is the `sensitivity` and FPR is the `false_positive_rate` of `assess_thresholds()`. With each weight equal to 1, this net benefit is the net benefit of Vickers and Elkin (2006) for a binary outcome. The weights extend it to censored data. In a simulation, its mean equaled the mean of the Aalen-Johansen form of Vickers et al. (2008) within the Monte Carlo error.

Treating no person has a net benefit of 0 at every threshold. A model is useful at a threshold when its net benefit is larger than that of treating all persons and larger than 0.

The standardized net benefit is NB/ρ (Kerr et al. 2016). Its largest value is 1, for a rule that flags each event and no non-event. Here, "standardized" means the division by ρ. It is not the standardized score of `prs_cohort()`, and not the average over the cohort of `standardized_risk()`.

The change in net benefit of a pair is $NB_{\text{full}} - NB_{\text{reduced}}$ at the same threshold. For one threshold, it equals the weighted NRI with two categories (Kerr et al. 2014). The weight is $s_1 = 1$ for events and $s_2 = p_t/(1 - p_t)$ for non-events:

$$\Delta NB = \rho\,\text{NRI}_{\text{events}} + (1 - \rho) \frac{p_t}{1 - p_t}\,\text{NRI}_{\text{nonevents}},$$

with the components `nri_events` and `nri_nonevents` of `assess_thresholds()` at that threshold.

Thresholds:

`thresholds` has no default, because the range of reasonable thresholds depends on the clinical decision (Vickers et al. 2016). A decision curve needs many thresholds, such as `seq(0.01, 0.5, 0.01)`. `assess_thresholds()` is for a few thresholds, because many thresholds give many categories of the NRI (Kerr et al. 2014).

Standard errors and intervals:

Each standard error is the standard deviation of an influence function over the persons, divided by $\sqrt{n}$, as in `assess_discrimination()`. The influence function includes the change in the Kaplan-Meier censoring weights. The net benefit and the prevalence are means of weighted terms, and their influence functions have the form of that of the Brier score. The standardized net benefit is a ratio of two weighted sums. The net benefit of treating all persons and its standardized value are functions of the prevalence alone. Each 95% interval is the estimate plus or minus 1.96 standard

errors, because a net benefit can be negative. When a model flags no person at a threshold, its net benefit and the standard error are 0. No person of the sample can then change the estimate, and the limits are NA. The function does not use the random number generator.

The net benefit on the development persons is optimistic, because fit_ladder() fitted the models on the same persons. For apparent evaluation, the intervals also ignore the uncertainty of the fitted models. Use a plan in validation, or a different cohort in newdata, to get a decision curve without this optimism.

Optimism correction:

With a plan from bootstrap_plan() in validation, the function corrects net_benefit, standardized_net_benefit, and delta_net_benefit at each threshold for optimism with the bootstrap (Harrell Jr. et al. 1996; Vickers et al. 2008). bootstrap_plan() describes the method, the random number streams, and the failed replicates. The column evaluation of models and pairs is then "optimism-corrected", and each estimate column holds the corrected estimate, the apparent estimate minus the mean optimism. The element optimism holds the apparent estimate, the optimism, and its Monte Carlo standard error of each estimate. The columns n and n_removed keep their values.

The prevalence and the net benefit of treating all persons do not depend on a model. The element treat_all keeps the values of the development persons, with the label "apparent".

Vickers et al. (2008) recommend repeated 10-fold cross-validation over this correction. In their simulation, the corrected estimates of the two methods differed by less than 0.005 at the thresholds below 60%. The bootstrap had the least bias at 60% and 80%.

The standard error and the interval depend on interval of the plan:

- "analytic": net_benefit_se and standardized_net_benefit_se are the influence-function standard errors of the apparent estimates. The interval is the corrected estimate plus or minus 1.96 of them. When the model flags no person, the apparent standard error is 0, and the limits are NA. The corrected estimate stays, and it can differ from 0, because the refits can flag persons. delta_net_benefit gets an NA standard error and NA limits, as the differences of assess_discrimination() do.
- "two-stage": the standard error is the standard deviation of the outer corrected estimates, and the limits are their 2.5% and 97.5% quantiles.
- "none": the standard errors and limits are NA.

The analytic interval is provisional. A simulation study before the release of the package measures its coverage.

Value

A list with the five elements below. Because each element has the same columns in every mode, the results of two modes stack with base::rbind(). The rows are in the order of horizon, then model as in ladder$models or pair as in pairs, then threshold. When no horizon passes the guard, the data frames have zero rows. Each estimate has three more columns with the suffixes _se, _lower, and _upper, for its standard error and 95% interval (see the section "Standard errors and intervals").

models One row per horizon, model, and threshold, with these columns:

model Model name.

horizon Horizon h.

evaluation "apparent", "optimism-corrected" with validation, or "external" with newdata.

`threshold` Risk threshold p_t.

`n` Number of persons evaluated.

`n_removed` Number of persons without a prediction.

`net_benefit` Net benefit of the model.

`standardized_net_benefit` Net benefit divided by the prevalence.

`treat_all` One row per horizon and threshold, with the columns `horizon`, `evaluation`, `threshold`, `n`, and `n_removed` of `models`, and these columns. With `validation`, `evaluation` is `"apparent"`.

`prevalence` Weighted cumulative incidence of T2 by the horizon, ρ. It is the same at every threshold of a horizon.

`net_benefit` Net benefit of treating all persons.

`standardized_net_benefit` Net benefit of treating all persons divided by the prevalence.

`pairs` One row per horizon, pair, and threshold, with these columns:

`full`, `reduced` Model names of the pair.

`horizon`, `evaluation`, `threshold`, `n`, `n_removed` As in `models`.

`delta_net_benefit` Net benefit of `full` minus net benefit of `reduced`.

`optimism` One row per corrected estimate, with the columns of the element `optimism` of `assess_discrimination()` and the column `threshold` after `horizon`. The rows are in the order of horizon, then the measures `net_benefit`, `standardized_net_benefit`, and `delta_net_benefit`, then model or pair, then threshold. It has zero rows without `validation`.

`settings` A list with the elements of the element `settings` of `assess_discrimination()`.

References

Choi SW, Mak TS, O'Reilly PF (2020). "Tutorial: a guide to performing polygenic risk score analyses." *Nature Protocols*, **15**(9), 2759–2772. doi:10.1038/s4159602003531.

Harrell Jr. FE, Lee KL, Mark DB (1996). "Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors." *Statistics in Medicine*, **15**(4), 361–387. doi:10.1002/(SICI)10970258(19960229)15:4<361::AIDSIM168>3.0.CO;2-4.

Kerr KF, Brown MD, Zhu K, Janes H (2016). "Assessing the clinical impact of risk prediction models with decision curves: guidance for correct interpretation and appropriate use." *Journal of Clinical Oncology*, **34**(21), 2534–2540. doi:10.1200/JCO.2015.65.5654.

Kerr KF, Wang Z, Janes H, McClelland RL, Psaty BM, Pepe MS (2014). "Net reclassification indices for evaluating risk prediction instruments: a critical review." *Epidemiology*, **25**(1), 114–121. doi:10.1097/EDE.0000000000000018.

van Geloven N, Giardiello D, Bonneville EF, Teece L, Ramspek CL, van Smeden M, Snell KIE, van Calster B, Pohar-Perme M, Riley RD, Putter H, Steyerberg E (2022). "Validation of prediction models in the presence of competing risks: a guide through modern methods." *BMJ*, **377**, e069249. doi:10.1136/bmj2021069249.

Vickers AJ, Cronin AM, Elkin EB, Gonen M (2008). "Extensions to decision curve analysis, a novel method for evaluating diagnostic tests, prediction models and molecular markers." *BMC*

Medical Informatics and Decision Making, **8**, 53. doi:10.1186/14726947853.

Vickers AJ, Elkin EB (2006). "Decision curve analysis: a novel method for evaluating prediction models." *Medical Decision Making*, **26**(6), 565–574. doi:10.1177/0272989X06295361.

Vickers AJ, Van Calster B, Steyerberg EW (2016). "Net benefit approaches to the evaluation of prediction models, molecular markers, and diagnostic tests." *BMJ*, **352**, i6. doi:10.1136/bmj.i6.

See Also

validation for the three modes, `plot_net_benefit()` for the decision curves, and `assess_thresholds()` for the sensitivity, the predictive values, and the NRI at a few thresholds.

Examples

```
#-------------------------------------------------------------------------
# assess_net_benefit() example
#-------------------------------------------------------------------------
library(prsladder)

# Fit a ladder on a development cohort.
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
ladder <- fit_ladder(cohort)

# Build an external cohort with the rules of the development cohort.
external <- prs_cohort(
  simulate_prs_data(n = 2000L, seed = 2L, id_prefix = "E"),
  reference = cohort
)

# Net benefit of the 10-year risk at thresholds from 5% to 40%.
net_benefit <- assess_net_benefit(
  ladder,
  horizons = 10,
  thresholds = seq(0.05, 0.4, by = 0.05),
  pairs = list(c("onset+progression", "onset")),
  newdata = external
)
net_benefit$models[
  net_benefit$models$model == "onset+progression",
  c("threshold", "net_benefit", "standardized_net_benefit")
]
net_benefit$treat_all[c("threshold", "prevalence", "net_benefit")]
net_benefit$pairs[c("threshold", "delta_net_benefit")]
```

`assess_stability` *Assess the Stability of the Predicted Risks*

Description

`assess_stability()` measures how much the predicted risk of each person changes when `fit_ladder()` fits the ladder on other samples of the same size. It uses the bootstrap samples of a plan from `bootstrap_plan()`. validation explains the internal validation that gives these samples. For each model and horizon, it reports the mean and the largest mean absolute prediction error over the persons (Riley and Collins 2023).

Usage

```
assess_stability(
  ladder,
  horizons,
  validation,
  min_events = 10L,
  min_at_risk = 50L,
  min_censoring_survival = 0.02
)
```

Arguments

`ladder`	(`prs_ladder`) A ladder from `fit_ladder()`.
`horizons`	(`numeric()`; range `(0, Inf)`) Distinct finite times since entry, in the units of the age columns.
`validation`	(`prs_bootstrap_plan`) A plan from `bootstrap_plan()`. The argument has no default.
`min_events`	(`integer(1)`; default `10L`; range `[1, Inf)`) Minimum number of T2 events at or before a horizon.
`min_at_risk`	(`integer(1)`; default `50L`; range `[1, Inf)`) Minimum number of persons whose time since entry is at least the horizon.
`min_censoring_survival`	(`numeric(1)`; default `0.02`; range `(0, 1]`) Minimum censoring survival $G(h)$ at a horizon h. A value of `1` requires no censoring at or before the horizon.

Details

Mean absolute prediction error:

Let r_i be the risk of person i at horizon h from the model fitted on the development persons D. Let r_{ib} be the risk of the same person from the model fitted again on the bootstrap sample D_b of replicate b. The mean absolute prediction error of person i is

$$\text{MAPE}_i = \frac{1}{B}\sum_{b=1}^{B} |r_{ib} - r_i|,$$

over the B replicates that do not fail at h (Riley and Collins 2023). The function reports the mean and the maximum of MAPE_i over the persons that the other `assess_*()` functions evaluate at h. A large value shows that the risk of a person depends on the sample that gave the model. Such instability is larger in a small cohort, and it is a sign of overfitting.

With competing events, the risk is the cumulative incidence of T2 from the models of both events, as in `predict_risk()`.

Replicates:

The function uses the replicates and the failure rules of the plan (see `bootstrap_plan()`). With the same plan and the same guard arguments, the replicates at a horizon are the replicates of the corrected mode of the other `assess_*()` functions. The function ignores `interval` and `n_inner` of the plan. When fewer than half of the `n_boot` replicates remain at a horizon, `mape_mean` and `mape_max` are `NA`.

The horizon guard of `horizon_support()` applies to the persons with a prediction, as in `assess_discrimination()`. The function leaves out each horizon that fails the guard and gives one warning for all of them.

Value

A list with the two elements below.

`models` A data frame with one row per horizon and model, in the order of horizon, then model as in `ladder$models`. It has zero rows when no horizon passes the guard.

- `model` Model name.
- `horizon` Horizon h.
- `n` Integer number of persons evaluated.
- `n_boot_used` Integer number of replicates used.
- `mape_mean` Mean of the mean absolute prediction error over the persons.
- `mape_max` Maximum of the mean absolute prediction error over the persons.

`settings` A list with the elements `design` (the argument `design` of `fit_ladder()`), `horizons` (the horizons that pass the guard), `min_events`, `min_at_risk`, `min_censoring_survival`, `censoring_strata` (as in `assess_discrimination()`), `validation` (the plan), and `replicates`. `replicates` is a data frame with one row per horizon and the integer columns `n_boot_used`, `failed_refit`, `failed_spline`, `failed_guard`, and `failed_risk` after the column `horizon`. These columns count the replicates used and the replicates that fail from each cause (see `bootstrap_plan()`).

References

Riley RD, Collins GS (2023). "Stability of clinical prediction models developed using statistical or machine learning methods." *Biometrical Journal*, **65**(8), e2200302. doi:10.1002/bimj.202200302.

See Also

validation for the internal validation, `bootstrap_plan()` for the replicates, and `assess_discrimination()` and `assess_calibration()` for the corrected measures.

Examples

```
#-------------------------------------------------------------------------
# assess_stability() example
#-------------------------------------------------------------------------
library(prsladder)

dat <- simulate_prs_data(n = 1000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
ladder <- fit_ladder(cohort)

# A small number of replicates keeps the example fast. An analysis needs
# 200 or more.
plan <- bootstrap_plan(n_boot = 20L, seed = 1L)
assess_stability(ladder, horizons = c(5, 10), validation = plan)$models
```

---

assess_thresholds *Assess Risk Thresholds and Reclassification*

Description

The function reports how each model of a ladder classifies persons at fixed risk thresholds, at each horizon. For each model and threshold, it gives the proportion of persons at or above the threshold, the sensitivity, the false-positive rate, and the positive and negative predictive values. For pairs of models, it gives the categorical net reclassification improvement (NRI) over the risk categories that the thresholds define. The PRS reporting standards ask for these proportions and for the NRI at clinically relevant thresholds (Wand et al. 2021).

The function has three modes. Without `newdata` and `validation`, it evaluates the development persons, and the estimates are apparent. With `validation`, it corrects the apparent estimates for optimism with the bootstrap. With `newdata`, it evaluates other persons. validation explains when to use each mode.

Usage

```
assess_thresholds(
  ladder,
```

```
  horizons,
  thresholds,
  pairs = NULL,
  newdata = NULL,
  validation = NULL,
  min_events = 10L,
  min_at_risk = 50L,
  min_censoring_survival = 0.02
)
```

Arguments

ladder	(prs_ladder or prs_export) A ladder from fit_ladder() or its export from export_ladder(). An export holds no persons, and it needs newdata.
horizons	(numeric(); range (0, Inf)) Distinct finite times since entry, in the units of the age columns.
thresholds	(numeric(); range (0, 1)) Strictly increasing risk thresholds. The function applies the same thresholds at every horizon. Because predicted risk increases with the horizon, the same threshold flags more persons at a longer horizon. To use different thresholds at different horizons, call the function once per horizon.
pairs	(list of character(2) or NULL; default NULL) Pairs of models to compare, each as c(full, reduced) model names. The names must be in names(ladder$models) and must differ. Pairs need not be nested, and a pair must not occur twice. If NULL, the function compares every nested pair in the order of compare_ladder().
newdata	(prs_cohort or NULL; default NULL) A different cohort, such as another biobank, a later period, or another ancestry group. Build it with prs_cohort(data, reference = ladder$cohort), or with prs_cohort(data, reference = ladder) for an export. Its scores, covariates, and index must be those of the development cohort. If NULL, the function evaluates the development persons. The persons must be separate from the development persons and from the GWAS data that gave the scores (Choi et al. 2020). For a ladder, the function gives a warning with the number of persons of newdata whose ids are also in ladder$cohort. Give newdata or validation, not both.
validation	(prs_bootstrap_plan or NULL; default NULL) A plan from bootstrap_plan() for the optimism correction of the estimates on the development persons (see the section "Optimism correction"). If NULL, the function gives the apparent or the external estimates.
min_events	(integer(1); default 10L; range [1, Inf)) Minimum number of T2 events at or before a horizon.
min_at_risk	(integer(1); default 50L; range [1, Inf)) Minimum number of persons whose time since entry is at least the horizon.

`min_censoring_survival`
: (`numeric(1)`; default `0.02`; range `(0, 1]`)
Minimum censoring survival $G(h)$ at a horizon h. A value of `1` requires no censoring at or before the horizon.

Details

Persons and weights:

At horizon h, the function evaluates only the persons with a prediction (see `predict_risk()`). The column `n_removed` counts the other persons. The horizon guard of `horizon_support()` applies to the evaluated persons. The function leaves out each horizon that fails the guard and gives one warning for all of them.

Let T_i be the time since entry of person i, and G the Kaplan-Meier estimate of the censoring distribution on the evaluated persons. Each person is in one of three classes:

- An event has T2 at or before h, with weight $1/G(T_i-)$.
- A non-event has $T_i > h$, with weight $1/G(h)$. With competing events, a person with a competing event at or before h is also a non-event, with weight $1/G(T_i-)$.
- A person censored at or before h has weight zero.

A competing event is not censoring in the estimate of G. Under the recruitment index, the evaluated persons can hold both groups of `t1_status`. G is then the estimate on the evaluated persons of the group of person i. `assess_discrimination()` describes these groups.

These weights are the inverse probability of censoring weights of Blanche et al. (2013), with the left limit $G(T_i-)$ of Gerds and Schumacher (2006). With one censoring distribution for all persons, they are the weights of `riskRegression::Score()` with `cens.model = "km"`. That model of `riskRegression::Score()` uses one censoring distribution also with `strat()` in the formula. The weights assume that censoring is independent of the scores and the covariates. Under the recruitment index, they also assume that censoring is independent of the entry within each group of `t1_status`.

Measures at a threshold:

A risk equal to a threshold counts as at or above it:

- `above` is the unweighted proportion of all evaluated persons at or above the threshold. It includes the censored persons.
- `sensitivity` is the weighted proportion of events at or above the threshold.
- `false_positive_rate` is the weighted proportion of non-events at or above the threshold.
- `ppv` is the positive predictive value, the weighted proportion of events among the persons at or above the threshold.
- `npv` is the negative predictive value, the weighted proportion of non-events among the persons below the threshold.

Let w_i be the weight of person i. Let e_i and c_i be the indicators of an event and a non-event. Let f_i be the indicator of a risk at or above the threshold. Then:

$$\text{PPV} = \frac{\sum_i w_i e_i f_i}{\sum_i w_i f_i}, \qquad \text{NPV} = \frac{\sum_i w_i c_i (1 - f_i)}{\sum_i w_i (1 - f_i)}.$$

Zheng et al. (2008) define the time-dependent PPV and NPV. They estimate them from a Nelson-Aalen, kernel, or Cox estimate of survival, not with censoring weights. This function estimates

them with the weights of Blanche et al. (2013), which those authors give for the sensitivity and the specificity. With competing events, a person with a competing event by the horizon is a non-event (Blanche et al. 2013; van Geloven et al. 2022). ppv is then the cumulative incidence of T2 among the persons at or above the threshold (van Geloven et al. 2022). npv is the probability of no T2 by the horizon among the persons below the threshold.

The weights sum to the number of evaluated persons n. Let $\rho = n^{-1} \sum_i w_i e_i$ be the prevalence. Then the predictive values follow from the other measures:

$$\text{PPV} = \frac{\rho\,\text{TPR}}{\rho\,\text{TPR} + (1-\rho)\,\text{FPR}},$$

where TPR is `sensitivity` and FPR is `false_positive_rate`. Because `above` has no weights, `sensitivity * prevalence / above` is close to `ppv`, but not equal.

`ppv` is `NA` when the persons at or above the threshold have no weight, for example when the threshold is above every risk. `npv` is `NA` when the persons below the threshold have no weight, for example when the threshold is at or below every risk.

The horizon is time since entry, as in the other `assess_*()` functions. Under the recruitment index or with a positive `entry_lag` in `prs_cohort()`, a person can enter after T1. A predictive value at a horizon of 10 then covers the 10 time units after entry. The risk at time 10 of `cumulative_risk()` covers the 10 time units after T1.

Reclassification:

For a pair, the thresholds divide the risk into categories. For each person, the function compares the category of the full model with the category of the reduced model. The move is up when the category of the full model is higher, and down when it is lower. The categorical NRI of Pencina et al. (2008) has these terms:

- `nri_events` is the weighted proportion of events that move up minus the weighted proportion that move down.
- `nri_nonevents` is the weighted proportion of non-events that move down minus the weighted proportion that move up.

Pencina et al. (2008) define the NRI for a binary outcome without censoring. The censoring weights extend it to censored data and to competing events.

The function does not report the sum of the two components. The sum is not a proportion, and each component is a difference of proportions (Kerr et al. 2014). With one threshold, `nri_events` is the change in sensitivity. Also, `nri_nonevents` is the change in false-positive rate with the sign reversed. With three or more categories, the NRI does not account for the clinical importance of each move (Kerr et al. 2014).

The NRI can be positive for an added score with no effect when a model is overfit or poorly calibrated. This bias also occurs on persons outside the development sample. Hilden and Gerds (2014) and Pepe et al. (2015) show this bias for the continuous NRI. Pepe et al. (2015) report the same bias for the categorical NRI. Check calibration with `assess_calibration()`. On persons outside the development sample, the AUC and the Brier score of `assess_discrimination()` do not improve because of a poorly fitting model (Hilden and Gerds 2014; Pepe et al. 2015). The function does not report the continuous NRI or the integrated discrimination improvement, which have the same bias (Hilden and Gerds 2014).

Standard errors and intervals:

Each standard error is the standard deviation of an influence function over the persons, divided by $\sqrt{n}$, as in `assess_discrimination()`. The influence function includes the change in the Kaplan-Meier censoring weights. Each measure except `above` is a ratio of two weighted sums, and its influence function has the form of that of the AUC. `above` has no weights, and its standard error is that of a proportion.

The proportions are `above`, `sensitivity`, `false_positive_rate`, `ppv`, and `npv`. The 95% interval of a proportion is the Wald interval on the logit scale:

$$\operatorname{expit}\left\{\operatorname{logit}(\hat{p}) \pm 1.96 \frac{\widehat{\mathrm{SE}}}{\hat{p}(1-\hat{p})}\right\}.$$

Its limits are `NA` when $\hat{p}$ is 0 or 1. The NRI terms can be negative, and their intervals are the Wald intervals on the raw scale, the estimate plus or minus 1.96 standard errors. The standard error and the limits are `NA` when the estimate is `NA`. The limits are also `NA` when the standard error is 0, such as for an NRI term when no person moves. No person of the sample can then change the estimate. The function does not use the random number generator.

The standard errors treat the predicted risks as constant. This assumption holds for persons outside the development sample. On the development persons, the estimates are apparent and tend to be optimistic, and the intervals ignore the uncertainty of the fitted models.

Optimism correction:

With a plan from `bootstrap_plan()` in `validation`, the function corrects each measure at each threshold for optimism with the bootstrap (Harrell Jr. et al. 1996). `bootstrap_plan()` describes the method, the random number streams, and the failed replicates. The column `evaluation` is then `"optimism-corrected"`, and each estimate column holds the corrected estimate, the apparent estimate minus the mean optimism. The element `optimism` holds the apparent estimate, the optimism, and its Monte Carlo standard error of each estimate. The columns `n` and `n_removed` keep their values.

The function corrects a proportion on the raw scale. A logit scale is not possible, because a refit that flags no person gives a sensitivity of 0, with an infinite logit. The raw difference can be below 0 or above 1 when the apparent value is 0, 1, or near these values. This occurs at a threshold near the edge of the risks of a model. The function then truncates the corrected proportion to 0 or 1, and its analytic limits are `NA`. Because the true value is in $[0, 1]$, the truncation never moves an estimate away from it. The element `optimism` keeps the raw difference.

A measure that is `NA` in a replicate, such as `ppv` when the refit flags no person, does not use that replicate. The column `n_boot_used` of `optimism` counts the replicates used for each row. When fewer than half of the replicates remain, the corrected estimate is `NA`.

The standard error and the interval depend on `interval` of the plan:

- `"analytic"`: the standard error of a proportion is the influence-function standard error of the apparent estimate. The interval is the logit-scale interval of the apparent estimate. The function moves this interval to the corrected estimate:

$$\operatorname{expit}\left\{\operatorname{logit}(\hat{p}_{\text{corr}}) \pm 1.96 \frac{\widehat{\mathrm{SE}}}{\hat{p}_{\text{app}}(1-\hat{p}_{\text{app}})}\right\}.$$

  Its limits are `NA` when $\hat{p}_{\text{app}}$ or $\hat{p}_{\text{corr}}$ is not in $(0, 1)$. The NRI components get `NA` standard errors and limits. Such an interval of a difference between models ignores the variance of the optimism and of the fitted models (see `assess_discrimination()`).

- `"two-stage"`: the standard error is the standard deviation of the outer corrected estimates, and the limits are their 2.5% and 97.5% quantiles. The function truncates the limits of a proportion to $[0, 1]$.
- `"none"`: the standard errors and limits are `NA`.

The analytic interval is provisional. A simulation study before the release of the package measures its coverage.

Value

A list with the four elements below. Because each element has the same columns in every mode, the results of two modes stack with `base::rbind()`. The rows are in the order of horizon, then model as in `ladder$models` and threshold, or pair as in `pairs`. When no horizon passes the guard, the data frames have zero rows. Each measure has three more columns with the suffixes `_se`, `_lower`, and `_upper`, for its standard error and 95% interval (see the section "Standard errors and intervals").

`models` One row per horizon, model, and threshold, with these columns:

- `model` Model name.
- `horizon` Horizon h.
- `evaluation` `"apparent"`, `"optimism-corrected"` with `validation`, or `"external"` with `newdata`.
- `threshold` Risk threshold.
- `n` Number of persons evaluated.
- `n_removed` Number of persons without a prediction.
- `above` Proportion of persons at or above the threshold.
- `sensitivity` Weighted proportion of events at or above the threshold.
- `false_positive_rate` Weighted proportion of non-events at or above the threshold.
- `ppv` Positive predictive value: weighted proportion of events among the persons at or above the threshold.
- `npv` Negative predictive value: weighted proportion of non-events among the persons below the threshold.

`pairs` One row per horizon and pair, with these columns:

- `full`, `reduced` Model names of the pair.
- `horizon`, `evaluation`, `n`, `n_removed` As in `models`.
- `nri_events` NRI component for events.
- `nri_nonevents` NRI component for non-events.

`optimism` One row per corrected estimate, with the columns of the element `optimism` of `assess_discrimination()` and the column `threshold` after `horizon`. The rows are in the order of horizon, then the measures `above`, `sensitivity`, `false_positive_rate`, `ppv`, `npv`, `nri_events`, and `nri_nonevents`, then model and threshold, or pair. `threshold` is `NA` for an NRI component. It has zero rows without `validation`.

`settings` A list with the elements of the element `settings` of `assess_discrimination()`.

References

Blanche P, Dartigues J, Jacqmin-Gadda H (2013). "Estimating and comparing time-dependent areas under receiver operating characteristic curves for censored event times with competing risks." *Statistics in Medicine*, **32**(30), 5381–5397. doi:10.1002/sim.5958.

Choi SW, Mak TS, O'Reilly PF (2020). "Tutorial: a guide to performing polygenic risk score analyses." *Nature Protocols*, **15**(9), 2759–2772. doi:10.1038/s4159602003531.

Gerds TA, Schumacher M (2006). "Consistent estimation of the expected Brier score in general survival models with right-censored event times." *Biometrical Journal*, **48**(6), 1029–1040. doi:10.1002/bimj.200610301.

Harrell Jr. FE, Lee KL, Mark DB (1996). "Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors." *Statistics in Medicine*, **15**(4), 361–387. doi:10.1002/(SICI)10970258(19960229)15:4<361::AIDSIM168>3.0.CO;2-4.

Hilden J, Gerds TA (2014). "A note on the evaluation of novel biomarkers: do not rely on integrated discrimination improvement and net reclassification index." *Statistics in Medicine*, **33**(19), 3405–3414. doi:10.1002/sim.5804.

Kerr KF, Wang Z, Janes H, McClelland RL, Psaty BM, Pepe MS (2014). "Net reclassification indices for evaluating risk prediction instruments: a critical review." *Epidemiology*, **25**(1), 114–121. doi:10.1097/EDE.0000000000000018.

Pencina MJ, D'Agostino Sr. RB, D'Agostino Jr. RB, Vasan RS (2008). "Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond." *Statistics in Medicine*, **27**(2), 157–172. doi:10.1002/sim.2929.

Pepe MS, Fan J, Feng Z, Gerds T, Hilden J (2015). "The net reclassification index (NRI): a misleading measure of prediction improvement even with independent test data sets." *Statistics in Biosciences*, **7**(2), 282–295. doi:10.1007/s1256101491180.

van Geloven N, Giardiello D, Bonneville EF, Teece L, Ramspek CL, van Smeden M, Snell KIE, van Calster B, Pohar-Perme M, Riley RD, Putter H, Steyerberg E (2022). "Validation of prediction models in the presence of competing risks: a guide through modern methods." *BMJ*, **377**, e069249. doi:10.1136/bmj2021069249.

Wand H, Lambert SA, Tamburro C, Iacocca MA, O'Sullivan JW, Sillari C, Kullo IJ, Rowley R, Dron JS, Brockman D, Venner E, McCarthy MI, Antoniou AC, Easton DF, Hegele RA, Khera AV, Chatterjee N, Kooperberg C, Edwards K, Vlessis K, Kinnear K, Danesh JN, Parkinson H, Ramos EM, Roberts MC, Ormond KE, Khoury MJ, Janssens ACJW, Goddard KAB, Kraft P, MacArthur JAL, Inouye M, Wojcik GL (2021). "Improving reporting standards for polygenic scores in risk prediction studies." *Nature*, **591**(7849), 211–219. doi:10.1038/s41586021032436.

Zheng Y, Cai T, Pepe MS, Levy WC (2008). "Time-dependent predictive values of prognostic biomarkers with failure time outcome." *Journal of the American Statistical Association*, **103**(481), 362–368. doi:10.1198/016214507000001481.

See Also

validation for the three modes, `assess_calibration()` to check calibration, `assess_discrimination()` for the AUC and the Brier score, and `compare_ladder()` for the likelihood-ratio test of added scores.

Examples

```
#--------------------------------------------------------------------------
# assess_thresholds() example
#--------------------------------------------------------------------------
# Fit a ladder on a development cohort.
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
ladder <- fit_ladder(cohort)

# Build an external cohort with the rules of the development cohort.
external <- prs_cohort(
  simulate_prs_data(n = 2000L, seed = 2L, id_prefix = "E"),
  reference = cohort
)

# Classify at 10-year risks of 20% and 40%, and compare the model with
# both scores to the model with the onset score only.
thresholds <- assess_thresholds(
  ladder,
  horizons = 10,
  thresholds = c(0.2, 0.4),
  pairs = list(c("onset+progression", "onset")),
  newdata = external
)
thresholds$models
thresholds$pairs

#--------------------------------------------------------------------------
# Optimism correction on the development persons
#--------------------------------------------------------------------------
# A small number of replicates keeps the example fast. An analysis needs
# 200 or more.
corrected <- assess_thresholds(
  ladder,
  horizons = 10,
  thresholds = 0.2,
  pairs = list(c("onset+progression", "onset")),
  validation = bootstrap_plan(n_boot = 20L, seed = 1L)
)
```

```
corrected$models[c("model", "sensitivity", "ppv", "ppv_lower", "ppv_upper")]
```

`bootstrap_plan` *Plan the Bootstrap of an Internal Validation*

Description

`bootstrap_plan()` holds the settings of the bootstrap optimism correction of the `assess_*()` functions. Give the plan to the argument `validation` of `assess_discrimination()`, `assess_calibration()`, `assess_thresholds()`, `assess_net_benefit()`, or `assess_stability()`. validation explains when to use the internal validation. The plan holds no data. The same plan gives the same bootstrap samples in every function.

Usage

```
bootstrap_plan(
  n_boot = 200L,
  seed = NULL,
  interval = c("analytic", "two-stage", "none"),
  n_inner = 50L
)
```

Arguments

`n_boot`	(`integer(1)`; default `200L`; range `[2, Inf)`) Number of bootstrap replicates B. With `interval = "two-stage"`, the number of outer replicates. A double with no fractional part is also accepted.
`seed`	(`integer(1)` or `NULL`; default `NULL`) Seed of the random number streams. If `NULL`, the function draws one seed from the current generator (see the section "Random numbers").
`interval`	(`character(1)`; default `"analytic"`) The interval of the corrected estimates: `"analytic"`, `"two-stage"`, or `"none"` (see the section "Intervals").
`n_inner`	(`integer(1)`; default `50L`; range `[2, Inf)`) Number of inner replicates of each outer replicate with `interval = "two-stage"`. The function checks the value for every `interval`, and uses it only with `"two-stage"`.

Details

Optimism correction:

On the development persons, the measures of a ladder are apparent. They are too optimistic, because `fit_ladder()` fitted the models on the same persons. The bootstrap optimism correction of Efron (1983); Harrell Jr. et al. (1996) estimates this optimism and subtracts it. The correction uses all persons. For a highly overfitted model, the correction can be too small (Efron 1983).

Let D be the development persons and f the ladder from D. Let $\theta(f, E)$ be a measure of f on the persons E. In each replicate $b = 1, \ldots, B$, the function draws a sample D_b of the persons of D

with replacement. It fits every model of the ladder again on D_b, which gives f_b. The optimism of replicate b and the corrected estimate are

$$O_b = \theta(f_b, D_b) - \theta(f_b, D), \qquad \hat{\theta}_{\text{corr}} = \theta(f, D) - \frac{1}{B}\sum_b O_b.$$

The refit repeats every step of `fit_ladder()`, also the knots of each spline, which come from the values of D_b.

Let n be the number of development persons. The corrected value estimates the expected performance on new persons of a ladder that `fit_ladder()` fits on n persons of the same population. It is a poor estimate of the performance of the one ladder f. Bates et al. (2024) prove this for the bootstrap and for cross-validation in the linear model. A report of the corrected estimate for f assumes that f is a typical ladder of that size.

Choices made in advance:

The correction repeats only the steps of `fit_ladder()`. Choose the rung, `spline_df`, the thresholds, and the pairs before the analysis. A choice made from the results of the same persons is not part of the replicates, and the correction is then too small. The construction of each score is also outside the replicates. A score tuned on the same persons needs other persons for its validation (Choi et al. 2020).

The column `heuristic_shrinkage` of `fit_ladder()$table` shows the size of the optimism before any bootstrap. A value near 1 shows a small optimism.

Random numbers:

Replicate b draws its persons from stream b of the L'Ecuyer-CMRG generator (L'Ecuyer et al. 2002). Stream 1 comes from `base::set.seed()` with `seed`, and stream $b+1$ from `parallel::nextRNGStream()` applied to stream b. Each function that gets the plan starts again from `seed`. Because of this, all functions use the same bootstrap samples. Each function sets the generator kinds explicitly, and it restores the generator kinds and state of the session on exit.

When `seed` is `NULL`, `bootstrap_plan()` draws one seed from the current generator and stores it. This draw advances the generator state of the session. Every function that gets the plan then uses the stored seed.

Intervals:

`interval` selects the 95% confidence interval of the corrected estimates:

- `"analytic"` moves the influence-function interval of the apparent estimate to the corrected estimate. Only the measures of one model with an influence function, such as the AUC and the Brier score, get this interval. Other measures, such as a difference between two models, the observed-to-expected ratio, and the calibration slope, get `NA` limits. The interval ignores the variance of the optimism and of the fitted models.
- `"two-stage"` is the interval of Algorithm 2 of Noma et al. (2021). Each outer replicate b repeats the full correction on D_b, with `n_inner` inner replicates from the substreams of stream b (`parallel::nextRNGSubStream()`). The limits are the 2.5% and 97.5% quantiles of the `n_boot` outer corrected estimates, and the standard error is their standard deviation. Every corrected measure gets this interval.
- `"none"` gives no interval.

The limits of `"two-stage"` are quantiles of the outer estimates. Their Monte Carlo error depends on `n_boot`. With 200 outer replicates, the Monte Carlo standard deviation of a limit is about 0.18 standard deviations of the outer estimates. With 1000 outer replicates, it is about 0.084. Because of this, use `n_boot` of 1000 or more with `"two-stage"` (Noma et al. 2021). Noma et al. (2021) found that the limits changed little from 100 to 2000 inner replicates (their e-Appendix B). The default `n_inner = 50L` is below that range. In one simulation setting, the analytic interval of the corrected AUC covered 0.93 to 0.94 for the rungs with a score of moderate effect. It covered 0.89 to 0.90 for the rungs with an AUC near 0.5, because the corrected AUC of such a rung is too high. The analytic interval of the Brier score covered 0.96 to 0.97. validation gives the setting. The coverage of the two-stage interval and of the other settings is not measured.

Failed replicates:

A replicate fails at every horizon when a spline term or a refit fails on D_b. It fails at one horizon when the persons of D_b with a prediction fail the horizon guard of `horizon_support()`. It also fails at one horizon when f_b gives no risk for a person that f evaluates. The follow-up limits of f_b can be narrower than those of f (see the section "Missing risks" of `predict_risk()`). The functions drop each failed replicate and give one warning. A measure can also be `NA` in one replicate. The optimism of a measure then uses the replicates in which both of its terms are not `NA`. When fewer than half of the `n_boot` replicates remain, the corrected estimate and its interval are `NA`.

Time:

Each function refits the ladder in each replicate. A test on one computer used 20,000 persons, four models, and two horizons. A replicate took about 0.4 seconds in `assess_discrimination()` and about 0.7 seconds in `assess_calibration()`. 200 replicates then take a few minutes. With competing events, the cumulative incidence takes most of the time. A replicate of `assess_discrimination()` then took about 13 seconds, and 200 replicates take about 45 minutes. `assess_stability()` needs the same time. A replicate of `assess_thresholds()` or `assess_net_benefit()` took about 11 seconds. The corrected measures of `assess_calibration()` use the model of T2 only. Its replicate then took about 0.4 seconds. With 200 replicates in each of the five functions, the internal validation then takes about 3 hours. Each function repeats the refits and the risks, because each function runs its own replicates. The interval `"two-stage"` takes `n_boot * (n_inner + 1)` replicates. It is fast in a small cohort, where it matters most.

Value

An object of class `prs_bootstrap_plan`, a list with the elements `n_boot` (integer), `seed` (integer), `interval` (character), and `n_inner` (integer).

References

Bates S, Hastie T, Tibshirani R (2024). "Cross-validation: what does it estimate and how well does it do it?" *Journal of the American Statistical Association*, **119**(546), 1434–1445. doi:10.1080/01621459.2023.2197686.

Choi SW, Mak TS, O'Reilly PF (2020). "Tutorial: a guide to performing polygenic risk score analyses." *Nature Protocols*, **15**(9), 2759–2772. doi:10.1038/s4159602003531.

Efron B (1983). "Estimating the error rate of a prediction rule: improvement on cross-validation."

Journal of the American Statistical Association, **78**(382), 316–331. doi:10.1080/01621459.1983.10477973.

Harrell Jr. FE, Lee KL, Mark DB (1996). "Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors." *Statistics in Medicine*, **15**(4), 361–387. doi:10.1002/(SICI)10970258(19960229)15:4<361::AIDSIM168>3.0.CO;2-4.

L'Ecuyer P, Simard R, Chen EJ, Kelton WD (2002). "An object-oriented random-number package with many long streams and substreams." *Operations Research*, **50**(6), 1073–1075. doi:10.1287/opre.50.6.1073.358.

Noma H, Shinozaki T, Iba K, Teramukai S, Furukawa TA (2021). "Confidence intervals of prediction accuracy measures for multivariable prediction models based on the bootstrap-based optimism correction methods." *Statistics in Medicine*, **40**(26), 5691–5701. doi:10.1002/sim.9148.

See Also

validation for the three modes of the `assess_*()` functions, and `assess_discrimination()`, `assess_calibration()`, `assess_thresholds()`, `assess_net_benefit()`, and `assess_stability()`, which accept the plan.

Examples

```
#-------------------------------------------------------------------------
# bootstrap_plan() example
#-------------------------------------------------------------------------
library(prsladder)

dat <- simulate_prs_data(n = 1000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
ladder <- fit_ladder(cohort)

# A small number of replicates keeps the example fast. An analysis needs
# 200 or more.
plan <- bootstrap_plan(n_boot = 20L, seed = 1L)
discrimination <- assess_discrimination(ladder, 10, validation = plan)
discrimination$models
discrimination$optimism
```

check_design *Test the Entry Terms and Pick a Design for Absolute Risks*

Description

`check_design()` tests how the hazard of T2 depends on the entry into a cohort under the recruitment index. From these tests, it picks a value of the argument `design` of `fit_ladder()` for absolute risks. The tests are for T2 and use the model with all scores.

The pick is for absolute risks only. Fit the hazard ratios and the primary tests with the default `design = "t1"`. Choose this workflow before the analysis, and report the three rows of `tests`.

Usage

```
check_design(cohort, spline_df = c(age_t1 = 3L), entry_cuts = c(1, 2, 4, 7))
```

Arguments

`cohort`	(`prs_cohort`) A cohort from `prs_cohort()` under the recruitment index, with incident and prevalent persons and at least one event.
`spline_df`	(named `integer()` or `NULL`; default `c(age_t1 = 3L)`; range `[1, 8]`) Degrees of freedom of the variables with a spline term, as in `fit_ladder()`. Each name must be `age_t1` or a numeric covariate of the cohort. `NULL` or a numeric vector of length zero gives a linear term to every variable. The function also accepts doubles with no fractional part, such as `3`.
`entry_cuts`	(`numeric()`; default `c(1, 2, 4, 7)`; range `(0, Inf)`) Upper limits of the bands of the time since entry, in the time unit of the cohort, without the last band to infinity. The values must be finite and strictly increasing.

Details

Tests:

Let p be 1 for a prevalent person and 0 for an incident person, and let d be `t1_duration`. Let t be the time since T1, and let $u = t - d$ be the time since entry. The element `tests` has one row for each test below.

- `entry_terms` is the likelihood-ratio test of the model of `"t1"` inside the model of `"t1_entry"`. It tests the terms $\gamma p + \delta d$ of `"t1_entry"`. It has 2 degrees of freedom.
- `entry_bands` is a score test at the estimates of `"t1_entry"`. The alternative replaces γp by $p \sum_j \gamma_j 1(u \in B_j)$. The bands are $B_j = (c_{j-1}, c_j]$, with $c_0 = 0$ and the values of `entry_cuts`, and a last band to infinity. The test asks whether the shift of prevalent persons changes with the time since entry. Because the test needs no fit of the alternative, a band with no T2 event needs no special case. The degrees of freedom are the rank of the information matrix of the band terms. This rank is usually one less than the number of bands that hold a prevalent person at the time of a T2 event. A band with no such person adds no information.

- index_duration is the test of survival::cox.zph() with transform = "km" for the duration term of "index" (Grambsch and Therneau 1994). It tests that the effect of d in this model is constant in u. Its degrees of freedom are the value of age_t1 in spline_df.

A competing event is a censoring in each test.

To try the tests on data with a known effect of the entry, use the arguments prevalent_hazard_ratio, duration_hazard_ratio, and prevalent_effect_years of simulate_prs_data().

Rule:

The function applies this rule at the 5% level:

1. If entry_bands does not reject, the pick is "t1_entry".
2. If entry_bands rejects, the pick is "index".

The tests entry_terms and index_duration are diagnostics, and the rule does not use them. The rule cannot pick "t1". A rejection of entry_terms shows that "t1" is not correct. It does not show that "t1_entry" is adequate. With the birth year in the models and a short recruitment period, entry_terms can have little power against an effect that ends after entry. The birth year and the age at T1 then explain most of t1_status and t1_duration. When the pick is "index", a rejection of index_duration shows that its assumption of a constant effect of d can fail.

Pre-specify entry_cuts. The p-value of entry_bands assumes that the cut points do not depend on the data. Use the same spline_df in check_design() and fit_ladder().

Limits:

The rule is for one cohort. Because the pick varies by chance, a study that pools results over cohorts pre-specifies one design for all cohorts. The optimism correction takes the design as fixed. No test of this function targets a score effect that differs by T1 status. Below about 500 events, a result without a rejection says little. One event in a band with very little follow-up can decide entry_bands. The rule was not studied with competing events or with an entry lag. The pick comes from T2, and the competing model uses the same design.

Errors:

The function stops with an error in these cases:

- cohort is not under the recruitment index. Every person then has the same entry.
- All persons of the cohort have the same value of t1_status.
- The cohort has no events.
- A value of t1_status has 10 persons or fewer, as in fit_ladder().
- The data do not support a spline term. A smaller value in spline_df can fix the problem (see the section "Errors and warnings" of fit_ladder()).
- survival::coxph() would reduce the time since entry of a person to zero.
- One of the three fits gives a warning or an aliased coefficient, as in fit_ladder(). The error message names the design.
- entry_bands has no degrees of freedom. The band terms then have no information at the T2 event times. An example is a cohort with no prevalent person in follow-up after the first cut at the time of a T2 event.
- survival::cox.zph() fails, as in fit_ladder(). The error message names the design.

Value

A list with the two elements below.

`tests` A data frame with one row per test, in the order `entry_terms`, `entry_bands`, and `index_duration`, and the columns below.

`test` The name of the test.

`null_model` The value of `design` in `fit_ladder()` of the model under the null hypothesis.

`statistic` The chi-square statistic.

`df` Integer degrees of freedom.

`p_value` The p-value from the chi-square distribution.

`rejected` Logical, `TRUE` when `p_value` is below 0.05.

`design` The pick of the rule, `"t1_entry"` or `"index"`. It is a value for the argument `design` of `fit_ladder()`.

References

Grambsch PM, Therneau TM (1994). "Proportional hazards tests and diagnostics based on weighted residuals." *Biometrika*, **81**(3), 515–526. doi:10.1093/biomet/81.3.515.

See Also

`fit_ladder()` for the designs, `check_entry()` for the test of quasi-independence, and `predict_risk()` for the risks.

Examples

```
#-------------------------------------------------------------------------
# check_design() example
#-------------------------------------------------------------------------
library(prsladder)

dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  index = "recruitment",
  age_at_recruitment = "age_at_recruitment"
)

# The tests and the pick of the rule.
checks <- check_design(cohort)
checks$tests

# The ladder of the absolute risks.
risk_ladder <- fit_ladder(cohort, design = checks$design)
risk_ladder
```

check_entry *Test the Quasi-Independence of Entry and Exit*

Description

check_entry() tests whether the entry time of a cohort is quasi-independent of its exit time. Delayed entry corrects for the selection of prevalent persons only under this condition (see the section "Time scale and entry" of prs_cohort()). The test is the conditional Kendall's tau of Tsai (1990).

Usage

```
check_entry(cohort)
```

Arguments

cohort	(prs_cohort) A cohort from prs_cohort() with two or more distinct entry times, such as a cohort under the recruitment index.

Details

Statistic:

Let L_i be the entry time start of person i, and X_i the exit time stop, both in time since T1. An exit is the first of T2 and the competing event. A censoring is not an exit. For each exit k, the risk set is $R_k = \{j : L_j < X_k \leq X_j\}$, which includes k. A person who enters at X_k is not in R_k, as in the follow-up intervals (start, stop] of prs_cohort(). The statistic is

$$K = \sum_k \sum_{j \in R_k} \text{sign}(L_j - L_k),$$

where the outer sum is over the exits. Each term compares two persons who were both in follow-up at X_k. These pairs are the comparable pairs of Tsai (1990). M is the number of comparable pairs, and the estimate of tau is K/M. A positive tau means that persons who enter later also exit later.

Two exits at the same time are not a comparable pair. Each is in the risk set of the other, and their two terms cancel in K. M does not count them. A pair with the same entry time adds zero to K, but it counts in M.

Variance:

Let x be a time with d_x exits, and let R_x be its risk set, with r_x persons. Under quasi-independence, the d_x persons who exit at x are equally likely to be any d_x persons of R_x. The terms of the times are then uncorrelated. The variance of K is

$$V = \sum_x \frac{d_x(r_x - d_x)}{r_x - 1} \frac{r_x^3 - \sum_g t_g^3}{3r_x}.$$

Here t_g is the size of a group of equal entry times in R_x. The factor $(r_x - d_x)/(r_x - 1)$ is the correction for a sample without replacement. It is one when d_x is one. Without equal entry times and without tied exits, each term is $(r_x^2 - 1)/3$, which is the variance of Tsai (1990). Equal entry times and tied exits make the variance smaller. Under the recruitment index, every incident person has the same entry time. These groups can then be large. The statistic is $z = K/\sqrt{V}$, and the two-sided p-value comes from the standard normal distribution.

Interpretation:

A small p-value is evidence that entry and exit are dependent. The absolute risks of `cumulative_risk()` and `predict_risk()` can then be biased. A large p-value does not show that the condition holds. The test has low power against a dependence that is not monotone (Chiou et al. 2018). The test is marginal. The condition can hold within groups of persons and fail in the full cohort, or the reverse. Because of this, the test can reject when a covariate, such as the age at T1, changes both the entry and the hazard. The designs `"t1"` and `"t1_entry"` of `fit_ladder()` need the condition only given their terms. The estimates of `cumulative_risk()` need the condition within each group. `check_design()` tests the entry given the terms of the models. The test also assumes that censoring is independent of the exit time, given the entry time. Censoring can depend on the entry time, as it does when a biobank ends follow-up on one date. With times in whole years, the test is approximate when censorings fall in the same years as exits. Use ages with decimals (see `prs_cohort()`).

The time of the computation grows with the number of persons times the number of distinct exit times.

Errors:

The function stops with an error when `cohort` is not a `prs_cohort`. It also stops when every person has the same entry time, because the cohort then has no delayed entry to test. This case includes the diagnosis index and `prevalent = "drop"`.

Value

A data frame with one row and the columns below.

`n` Integer number of persons.

`n_exits` Integer number of exits, T2 or the competing event.

`n_pairs` Number of comparable pairs M.

`tau` The estimate K/M, or `NA` when M is zero.

`z` The statistic $K/\sqrt{V}$, or `NA` when V is zero.

`p_value` The two-sided p-value of `z`, or `NA` when `z` is `NA`.

References

Chiou SH, Qian J, Mormino E, Betensky RA (2018). "Permutation tests for general dependent truncation." *Computational Statistics & Data Analysis*, **128**, 308–324. doi:10.1016/j.csda.2018.07.012.

Tsai W (1990). "Testing the assumption of independence of truncation time and failure time." *Biometrika*, **77**(1), 169–177. doi:10.1093/biomet/77.1.169.

See Also

prs_cohort() for the entry times of the two indexes, and check_design() for tests of the entry given the terms of the models.

Examples

```
#-------------------------------------------------------------------------
# check_entry() example
#-------------------------------------------------------------------------
library(prsladder)

dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(progression = "progression_score"),
  index = "recruitment",
  age_at_recruitment = "age_at_recruitment"
)

# With its default arguments, the simulator recruits independently of the
# event times.
check_entry(cohort)
```

---

compare_ladder *Compare Nested Models of a Ladder*

Description

compare_ladder() tests whether the scores that a full model adds to a nested, reduced model improve the fit. For example, the pair c("onset+progression", "onset") tests whether the progression score adds information to the onset score. The function uses the likelihood-ratio test on the development persons. For each pair, it also reports the change in AIC and in Harrell's concordance.

Usage

```
compare_ladder(ladder, pairs = NULL)
```

Arguments

ladder	(prs_ladder) A ladder from fit_ladder().
pairs	(list of character(2), or NULL; default NULL) Pairs of models to test. Each element is c(full, reduced), with the full model first. Each pair must obey these rules:

- Both names are in `names(ladder$models)`, and neither is `NA`.
- The reduced model is nested in the full model.
- The pair occurs only once in the list.

The function ignores the names of the list and of its elements. If `NULL`, the function tests every nested pair. The pairs are then in order of the full model, then of the reduced model, both in the order of `ladder$models`.

Details

Nested pairs:

Two models form a nested pair when the scores of the reduced model are a proper subset of the scores of the full model. For example, `onset` is nested in `onset+progression`, and `base` is nested in every other model. The models `onset` and `progression` do not form a nested pair.

A ladder with k scores has $3^k - 2^k$ nested pairs. For example, two scores give 5 pairs and three scores give 19 pairs. A pair with `base` as the reduced model gives the gain of the scores over the covariates and age term alone. The PRS reporting standards ask for this gain over a conventional risk model (Wand et al. 2021).

Likelihood-ratio test:

For the partial log-likelihoods ℓ_{full} and ℓ_{reduced}, the statistic is

$$\chi^2 = 2(\ell_{\text{full}} - \ell_{\text{reduced}}).$$

The degrees of freedom equal the number of added scores, because each score has one coefficient. Under the null hypothesis that the added scores have no effect, the statistic has an asymptotic chi-squared distribution. The test is valid because every model of a ladder has the same persons, covariates, age term, and design. For the pre-specified tests, use a ladder with the default `design = "t1"` of `fit_ladder()` (see its section "Designs").

The statistic does not change when the analyst rescales a score or reverses its sign. Because of this, the standardization of the scores has no effect on the test. Numerical error in the fits can give a statistic slightly below zero. The function reports that statistic as computed, with a p-value of 1.

With competing events (see `prs_cohort()`), `chisq`, `delta_aic`, and `delta_concordance` compare the models of the cause-specific hazard of T2. They do not use the competing models of `fit_ladder()`. The test is then a test of the association of the added scores with the hazard of T2. A score can also change the cumulative incidence of T2 through the hazard of the competing event.

Selection on the first diagnosis can bias the hazard ratio of a score, most of all an onset score. The section on index event bias in `fit_ladder()` explains this bias. Under this bias, the test applies to the hazard ratio in the selected persons, which can differ from the hazard ratio in the population.

Descriptive measures:

The function gives two descriptive differences, full minus reduced:

- `delta_aic` is the difference in AIC. It equals $-\chi^2 + 2\,\mathrm{df}$. A negative value favors the full model.
- `delta_concordance` is the difference in Harrell's concordance on the development persons. A positive value favors the full model.

Unlike the AIC, `delta_concordance` has no penalty for the added scores. Because it uses the development persons, it tends to overstate the gain in new persons. An accurate estimate of the gain needs out-of-sample persons (Choi et al. 2020). Use `assess_discrimination()` with a plan in `validation` or a different cohort in `newdata` for this estimate (see validation).

The function does not test these differences. On the development persons, a test of a difference in AUC between nested models is conservative (Demler et al. 2012; Pepe et al. 2013). Harrell's concordance is a form of the AUC for survival data. Pepe et al. (2013) study a binary outcome. They show that a null hypothesis of no gain in prediction equals a null hypothesis of no association of the added predictor. The association is conditional on the other predictors. The likelihood-ratio test is a test of this association.

Multiple comparisons:

The function does not adjust p-values for multiple comparisons. Prespecify the primary pairs and give them in `pairs`. The default, all nested pairs, is a descriptive overview.

Value

A data frame with one row per pair, in the order of `pairs`, and the columns below.

`full`, `reduced` Model names of the pair.

`added` Scores that the full model adds, joined by `+`.

`df` Integer number of added scores.

`chisq` Likelihood-ratio statistic.

`p_value` P-value of the test, not adjusted for multiple comparisons.

`delta_aic` AIC of the full model minus AIC of the reduced model.

`delta_concordance` Harrell's concordance of the full model minus that of the reduced model.

References

Choi SW, Mak TS, O'Reilly PF (2020). "Tutorial: a guide to performing polygenic risk score analyses." *Nature Protocols*, **15**(9), 2759–2772. doi:10.1038/s4159602003531.

Demler OV, Pencina MJ, D'Agostino Sr. RB (2012). "Misuse of DeLong test to compare AUCs for nested models." *Statistics in Medicine*, **31**(23), 2577–2587. doi:10.1002/sim.5328.

Pepe MS, Kerr KF, Longton G, Wang Z (2013). "Testing for improvement in prediction model performance." *Statistics in Medicine*, **32**(9), 1467–1482. doi:10.1002/sim.5727.

Wand H, Lambert SA, Tamburro C, Iacocca MA, O'Sullivan JW, Sillari C, Kullo IJ, Rowley R, Dron JS, Brockman D, Venner E, McCarthy MI, Antoniou AC, Easton DF, Hegele RA, Khera AV, Chatterjee N, Kooperberg C, Edwards K, Vlessis K, Kinnear K, Danesh JN, Parkinson H, Ramos EM, Roberts MC, Ormond KE, Khoury MJ, Janssens ACJW, Goddard KAB, Kraft P, MacArthur JAL, Inouye M, Wojcik GL (2021). "Improving reporting standards for polygenic scores in risk prediction studies." *Nature*, **591**(7849), 211–219. doi:10.1038/s41586021032436.

See Also

`fit_ladder()` for the models, and `assess_discrimination()` for the change in AUC and Brier score at a horizon, corrected for optimism or on a different cohort.

Examples

```
#------------------------------------------------------------------------------
# compare_ladder() example
#------------------------------------------------------------------------------
# Fit a ladder with three scores.
dat <- simulate_prs_data(n = 2000L, seed = 1L, score_correlation = 0.3)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(
    onset = "onset_score",
    progression = "progression_score",
    outcome = "outcome_score"
  )
)
ladder <- fit_ladder(cohort)

# Test the prespecified primary pairs.
compare_ladder(
  ladder,
  pairs = list(
    c("onset+progression", "onset"),
    c("progression+outcome", "outcome")
  )
)

# Describe all 19 nested pairs.
compare_ladder(ladder)
```

`cumulative_risk` *Cumulative Risk by Score Group*

Description

`cumulative_risk()` compares the observed risk of the second event (T2) between persons with low and high values of one polygenic risk score. The function divides the cohort into quantile groups of the score. For each group, it estimates the cumulative risk of T2 at given times since the first diagnosis (T1). It also gives the risk difference and the risk ratio to a reference group, and the share of all T2 cases in each group. The other results are the restricted mean time lost to T2 and a test of equal risk across the groups. With `score = NULL`, it estimates the risk of the whole cohort.

The result is a `prs_cumulative_risk` object. `plot_risk()`, `plot_stacked()`, and `plot_gradient()` draw it, and `describe_risk()` makes a table of it. Because these functions use one object, the figures and tables show the same estimates.

The estimates are Kaplan-Meier estimates, or Aalen-Johansen estimates with competing events, with no covariate adjustment. For the risk from a fitted model, use `predict_risk()`.

Usage

```
cumulative_risk(
  cohort,
  score,
  times,
  groups = 5L,
  cut_probs = NULL,
  reference = 1L,
  min_events = 1L,
  min_at_risk = 50L
)

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

Arguments

cohort	(prs_cohort) A cohort from prs_cohort().
score	(character(1) or NULL) The name of one score of the cohort, as given in the names of scores in prs_cohort(). If NULL, the function estimates the risk of the whole cohort. The argument has no default. Because of this, a missing score is an error.
times	(numeric(); range (0, Inf)) Distinct finite times since T1, in the units of the age columns. Each time must be after the origin, the entry_lag of the cohort. The order does not matter. The result gives the times in increasing order.
groups	(integer(1); default 5L; range [2, Inf)) Number of quantile groups of the score. For example, 5L gives quintiles and 4L gives quartiles. Must be missing when score is NULL or when cut_probs is given.
cut_probs	(numeric() or NULL; default NULL; range (0, 1)) Strictly increasing probabilities at which to cut the score into length(cut_probs) + 1 groups. Each value is a probability, not a score in standard deviation units. For example, 0.95 separates the 5% of persons with the highest scores. The probabilities refer to the standardized score, which includes the sign from prs_cohort(). With a sign of -1, cut_probs = 0.95 separates the 5% of persons with the lowest raw scores. If NULL, the function uses groups groups of near-equal size. Must be NULL when score is NULL. See the section "Groups at cut points" in Details.
reference	(integer(1); default 1L; range [1, groups], or [1, length(cut_probs) + 1] with cut_probs) The group whose risk the function subtracts from the risk of each group. The default is the group with the lowest scores. Must be missing when score is NULL.
min_events	(integer(1); default 1L; range [1, Inf)) Minimum number of T2 events in a group at or before the time. The default

hides a risk of 0 with an interval of $[0, 0]$, which occurs only with competing events. Without competing events, the default has no effect, because the third condition in Details requires an event. A larger value selects the groups with early events. The risks that pass are then too high, and their intervals cover less than 95%.

`min_at_risk` (`integer(1)`; default `50L`; range `[1, Inf)`)
Minimum number of persons of a group at risk at the time and at each earlier event time of the group.

`x` (`prs_cumulative_risk`)
A result of `cumulative_risk()`.

`...` Not used.

Details

Score groups:

The function cuts the score at its sample quantiles into `groups` groups of near-equal size. The cut uses the standardized score of the cohort, which includes the sign from `prs_cohort()`. Group 1 has the lowest standardized scores. With a sign of `-1`, group 1 has the highest raw scores. The function stops with an error when tied quantiles give fewer distinct groups than requested.

Groups at cut points:

`cut_probs` gives groups of other sizes. The breaks are `quantile(x, c(0, cut_probs, 1), type = 7)`, where `x` is the standardized score. Group k holds the scores in the interval $(q_{k-1}, q_k]$ of the breaks, and group 1 also holds its lower limit. `groups = 5L` and `cut_probs = 1:4 / 5` give identical breaks. The literal `c(0.2, 0.4, 0.6, 0.8)` also gives identical breaks. `seq(0.2, 0.8, 0.2)` can give breaks that differ by about 10^{-16}, because its third element is $0.6 + 10^{-16}$. A score in that small gap goes to another group.

Two examples:

- `cut_probs = 0.95` compares the 5% of persons with the highest scores (group 2) with the other 95% (group 1).
- `cut_probs = c(0.05, 0.95)` with `reference = 2L` compares each tail with the middle 90%.

The labels of the figures and tables of the result give the percent range of each group, such as `0-95%` and `95-100%`. These ranges refer to the standardized score. With a sign of `-1`, `cut_probs = 0.95` separates the 5% of persons with the lowest raw scores. Each label then also gives the range of the raw score, such as `95-100% (raw 0-5%)`. The footnote or caption states that the groups run against the raw score.

A small group has fewer persons at risk. Because of this, it fails the condition on `min_at_risk` at an earlier time. For example, with 2,000 persons, the top 5% is 100 persons, and the risks at later times can be `NA`. A group with fewer persons than `min_at_risk` fails the condition at every time. Each of its risks is then `NA`, and the function gives a warning. Because the case share needs the risks of all groups, each case share is then also `NA`.

Time scale and risk:

The time scale is time since T1, with delayed entry at `start`. A person is at risk at time t when `start` $< t \leq$ `stop`. The prediction metrics, such as `assess_calibration()`, use time since entry. This function uses time since T1, because a progression curve starts at the diagnosis.

The risk at time t is $1 - \hat{S}(t)$, where $\hat{S}$ is the Kaplan-Meier estimate from `survival::survfit()` on `Surv(start, stop, status)`. The standard error is the Greenwood standard error of $\hat{S}(t)$, which is also the standard error of the risk. With a positive `entry_lag` in `prs_cohort()`, the curve is conditional on being free of T2 at the lag.

When the cohort declares the `competing` status code (see `prs_cohort()`), the risk at time t is the cumulative incidence of T2. This risk is the probability of T2 by time t before a competing event. The function estimates it with the Aalen-Johansen estimator (Aalen and Johansen 1978) from `survival::survfit()` on `Surv(start, stop, state)`, where `state` is 0, 1, or 2. This estimator accepts delayed entry when the entry and censoring times are independent of the event times (Geskus 2011). The standard error is the infinitesimal jackknife standard error of `survival::survfit()`. With a positive `entry_lag`, the curve is conditional on being free of both events at the lag. One minus the Kaplan-Meier estimate is at least the cumulative incidence. The Kaplan-Meier estimate treats a competing event as censoring. It assumes that a person with a competing event keeps the hazard of T2 after that event. In fact, that person can no longer have T2. Because of this, one minus the Kaplan-Meier estimate overestimates the cumulative incidence (Putter et al. 2007; Andersen et al. 2012).

Support of a group at a time:

A group supports a time when all three conditions below are true:

1. At least `min_events` events occur in the group at or before the time.
2. At least `min_at_risk` persons of the group are at risk at the time.
3. At least `min_at_risk` persons of the group are at risk at each event time of the group up to the time.

The events of the first condition are T2 events. With competing events, the event times of the third condition include the times of competing events, because these events also change the curve. The third condition protects against an early risk set of a few prevalent persons. Each step of the curve at such an early time stays in every later estimate. When a group does not support a time, `risk` and `risk_se` are `NA`, never zero. The competing risk uses the same conditions, with competing events in the first condition. A time after the last follow-up of a group has zero persons at risk, and its risk is `NA`.

The default `min_events = 1L` hides only a risk of 0 with an interval of $[0, 0]$. This interval does not include a positive risk. With competing events, a group with competing events and no T2 event by a time has this risk. A larger `min_events` selects the samples with more early events than expected, because the condition passes only in these samples. The risks that pass are then too high, and their intervals cover the risk less often than stated. A simulation used delayed entry and groups of 200 and 1000 persons. Under the default, the 95% intervals covered the risk in 93.6% to 96.7% of the samples in which the conditions passed. With `min_events = 10L`, they covered 53% to 89% at the times where the condition passed in 7% to 27% of samples. The default conditions also select samples at a time with about one expected event, because the third condition needs an event. In the simulation, such a risk of 1.6% was too high by 0.7 points on average, and its interval covered the risk in 95% of samples.

These counts differ from those of `horizon_support()`, which counts on time since entry and also applies a censoring guard.

A row of `curves` uses only the third condition. The function marks an event row as supported when at least `min_at_risk` persons are at risk at each event time of the group up to the row. At an event time, the second condition adds nothing to the third. The first condition does not apply to `curves`, because with an event minimum, a curve starts only in the samples with early events. The

origin row has the value of the first event row. The reason is that the risk before the first event is 0 whatever the size of the risk set. The third condition cannot change from false to true. Because of this, the supported rows of a group are its first rows. `curves` keeps the risks of the rows that are not supported, and `supported` marks the rows that a figure draws.

The three conditions are statistical. They do not protect the privacy of persons. Each step of a curve is one event, and `n_events` gives the event counts. The package does no disclosure control. Apply the rules of the data provider before you share a result.

Curves:

The element `curves` holds the step function of the risk of each group. It has one row at the origin and one row at each event time of either cause. The origin is the `entry_lag` of the cohort, and the risk at the origin is 0. The number at risk at the origin is the number of persons just after it, with `start` $\leq$ origin $<$ `stop`. The rule `start` $< t$ gives an empty risk set at the origin. Under the recruitment index, the number at the origin is the number of incident persons. The number at risk can then increase over time, because prevalent persons enter after T1. At a time in `times`, the last row of `curves` at or before the time has the risk of `risks`, when `risks` shows a risk.

Intervals:

The interval of a risk F with standard error s is a pointwise 95% interval on the log scale of F:

$$\left[Fe^{-zs/F},\ \min\left(1,\ Fe^{zs/F}\right)\right], \quad z = \Phi^{-1}(0.975).$$

Both limits are 0 when F is 0. For example, before the first T2 event of a group, a row of `curves` has a risk of 0 and the interval $[0, 0]$. The same rule applies to the risk of T2 and to the competing risk, in `risks` and in `curves`. The interval at each time holds for that time only, not for the whole curve.

For the Aalen-Johansen estimate, this interval is the default of `survival::survfit()`. For the Kaplan-Meier estimate, the default of `survival::survfit()` is on the log scale of $1 - F$. The `ggsurvfit` and `gtsummary` packages use that default. Because of this, the Kaplan-Meier intervals of this function differ from the intervals of these packages.

Whole cohort:

With `score = NULL`, all persons form group 1. `groups` and `reference` must then be missing, and `cut_probs` must be `NULL`. The columns of the difference, the ratio, the case share, and the difference in time lost are `NA`, and `test` has zero rows.

Differences and test:

The difference is the risk of a group minus the risk of the `reference` group at the same time. Because the groups hold different persons, the variance of a difference is the sum of the two variances. The interval is a pointwise 95% Wald interval, $d \pm z_{0.975}\mathrm{SE}(d)$, with no adjustment for multiple comparisons. A difference is `NA` when either of the two risks is `NA`.

The test is the score test of `coxph(Surv(start, stop, status) ~ group)`, with the group as a factor. It has `groups - 1` degrees of freedom. This test accepts delayed entry. It equals the log-rank test when there is no delayed entry and no tied event times. The test uses all follow-up and does not depend on `times`, `reference`, or the guards.

With competing events, the test compares the cause-specific hazards of T2, which agrees with the models of `fit_ladder()`. A competing event counts as censoring in the test. The curves of the groups can also differ because of the hazard of the competing event, which the test does not examine. The test of Gray (1988) compares the cumulative incidence curves. It applies only to right-censored data without delayed entry.

Ratios and case shares:

The ratio is the risk of a group divided by the risk of the `reference` group at the same time:

$$\widehat{RR}_g = \hat{F}_g / \hat{F}_r, \qquad \mathrm{SE}(\log \widehat{RR}_g) = \sqrt{(s_g/\hat{F}_g)^2 + (s_r/\hat{F}_r)^2},$$

where $\hat{F}_g$ is the risk of group g, s_g is its standard error, and r is the reference group. This standard error comes from the delta method on the log scale. The covariance is zero, because the groups hold different persons. The interval is a pointwise 95% Wald interval on the log scale, $\widehat{RR}_g \exp(\pm z_{0.975}\mathrm{SE})$. The column `log_ratio_se` holds the standard error of $\log \widehat{RR}_g$, not of the ratio. A ratio is `NA` for the reference group, and when either of the two risks is `NA`. Because `min_events` is at least 1, each risk that the function shows is positive, and the logarithm of each ratio is finite.

The case share is the share of all T2 cases by the time that occur in a group. With n_g the number of persons in group g, $a_g = n_g \hat{F}_g$, $v_g = n_g^2 s_g^2$, and $S = \sum_k a_k$:

$$\hat{\pi}_g = a_g / S, \qquad \widehat{\mathrm{Var}}(\hat{\pi}_g) = \frac{(S - a_g)^2 v_g + a_g^2 \sum_{k \neq g} v_k}{S^4}.$$

The variance comes from the delta method. The interval is a pointwise 95% interval on the logit scale, $\mathrm{logit}^{-1}\{\mathrm{logit}(\hat{\pi}_g) \pm z_{0.975}\mathrm{SE}/[\hat{\pi}_g(1 - \hat{\pi}_g)]\}$. The shares of the groups sum to 1 at each time. For the persons above the cut point, the case share is an analogue of the proportion of cases followed of Pfeiffer and Gail (2011). Pfeiffer and Gail rank the persons by the predicted risk of a model, and they compute the proportion from the predicted risks. Here, the score ranks the persons, and the share comes from the observed events. At a time where any group has an `NA` risk, the case shares of all groups are `NA`.

The case share does not use the raw counts of events. Because censoring removes more of the later events, the raw counts give too large a share to a group whose events occur earlier. The top group is such a group. In a simulation, the raw counts overstated the share of the top quintile by 0.42 percentage points, and $n_g \hat{F}_g$ by 0.14 points.

The intervals of the ratio and the case share are conditional on the groups of the sample. They treat the cut points and the group sizes as fixed, but the cut points are sample quantiles that change from sample to sample.

The case share describes a population with the group sizes of the cohort. Under the recruitment index, the cohort holds only the persons who were free of T2 at recruitment. Persons with high scores more often had T2 before recruitment. Because of this, the cohort holds fewer of them than the population at T1. The case share of the top group is then not an unbiased estimate of its share in the population at T1. In a simulation with the default hazard ratios of `simulate_prs_data()`, the share of the top quintile was 0.2646 in the persons of the cohort and 0.2681 in all persons at T1.

Time lost:

The element `time_lost` gives the restricted mean time lost to T2 by each time τ in `times` (Andersen 2013; Conner and Trinquart 2021):

$$\mathrm{RMTL}_g(\tau) = \int_{t_0}^{\tau} \hat{F}_g(u)\, du,$$

where t_0 is the origin and $\hat{F}_g$ is the risk of group g. It is the mean time from T2 to τ, where a person with no T2 by τ adds 0. The unit is the unit of the age columns. For example, take years as

the unit and $\tau = 10$. A time lost of 0.5 then means that, in the first 10 years after T1, a person of the group spends 0.5 years after T2 on average. Without competing events, the time lost is $\tau - t_0$ minus the restricted mean survival time.

With competing events, `competing_rmtl` is the same area under the cumulative incidence of the competing event. The time lost to T2, the time lost to the competing event, and the restricted mean time free of both events sum to $\tau - t_0$.

The function gets the time lost and its standard error from `summary(fit, rmean = tau)` on the fit of the risks. The time lost equals the area under the step function of `curves` from the origin to τ. With competing events, the standard error is the infinitesimal jackknife standard error of `survival::survfit()`. `survival::survfit()` gives this standard error at each time of the fit, and `summary()` interpolates it linearly between these times. In a check, the interpolated value differed from the exact value by less than 0.1%. Without competing events, the standard error is the Greenwood-type standard error of the restricted mean survival time from `survival::survfit()`.

The interval is a pointwise 95% Wald interval, $\mathrm{RMTL} \pm z_{0.975}\mathrm{SE}$, with the lower limit truncated at 0. The difference is the time lost of a group minus the time lost of the `reference` group. Its variance is $s_g^2 + s_r^2$, and its interval is a 95% Wald interval. The time lost at τ is `NA` where the risk at τ is `NA`, because the conditions on `min_at_risk` cover each event time up to τ. For the competing event, the same rule uses the competing risk.

The intervals are conditional on the groups of the sample, as are those of the ratio and the case share. A simulation with competing events used quintile groups, 300 samples of 3,000 persons, and $\tau = 10$. The reference value was the time lost in one sample of 400,000 persons. Without delayed entry, the 95% intervals of the time lost to T2 covered this value in 92.0% to 96.7% of the samples. With delayed entry, they covered it in 94.0% to 96.0% of the samples. The interval of the difference between the top and bottom quintiles covered its reference value in 94.7% and 94.0% of the samples.

Value

An object of class `prs_cumulative_risk`, a list with the elements below.

`risks` A data frame with one row per group and time. The function orders the rows by group, then by increasing time. The columns are below. The columns `n_competing` and `competing_risk` to `competing_risk_upper` occur only when the cohort declares the `competing` status code.

- `group` Integer group number, from 1 to `groups`.
- `label` Character label of the group, as in the figures and tables. With equal groups, it is the group number, with the tags `(lowest)` and `(highest)` for a sign of `1`. With `cut_probs`, it is the percent range of the standardized score, such as `95-100% (highest)`, or `95-100% (raw 0-5%)` for a sign of `-1`. `"All persons"` for the whole cohort.
- `time` Time since T1.
- `n` Integer number of persons in the group.
- `n_events`, `n_competing`, `n_censored` Integer numbers of T2 events, competing events, and censorings in the group at or before `time`.
- `n_at_risk` Integer number of persons in the group at risk at `time`.
- `min_at_risk_before` Integer smallest number at risk at an event time of the group up to `time`, for an event of either cause. `NA` when the group has no event by `time`.
- `risk`, `risk_se` Cumulative risk of T2 at `time` and its standard error. `NA` when the group does not support `time`.

`risk_lower`, `risk_upper` Limits of the 95% interval of the risk.

`competing_risk`, `competing_risk_se`, `competing_risk_lower`, `competing_risk_upper` Cumulative incidence of the competing event at `time`, its standard error, and the limits of its 95% interval.

`difference`, `difference_se` Risk difference from the `reference` group and its standard error. `NA` for the reference group.

`difference_lower`, `difference_upper` Limits of the 95% Wald interval of the difference.

`ratio`, `log_ratio_se` Risk ratio to the `reference` group and the standard error of its logarithm. `NA` for the reference group.

`ratio_lower`, `ratio_upper` Limits of the 95% Wald interval of the ratio on the log scale.

`case_share`, `case_share_se` Share of all T2 cases by `time` that occur in the group, and its standard error. `NA` at a time where any group has an `NA` risk, and for the whole cohort.

`case_share_lower`, `case_share_upper` Limits of the 95% interval of the case share on the logit scale.

`test` A data frame with the columns `statistic`, `df`, and `p_value` of the score test across groups. It has one row, or zero rows when `score` is `NULL`.

`curves` A data frame with one row at the origin and one row at each event time of either cause, for each group. The function orders the rows by group, then by increasing time. The columns are `group`, `time`, `n_at_risk`, `n_events`, `n_competing`, the risk columns of `risks`, and `supported`. `n_events` and `n_competing` are totals at or before `time`. The risks are never `NA`. `supported` is `TRUE` for a row that a figure draws.

`time_lost` A data frame with the rows of `risks`, in the same order, and the columns below. The columns `competing_rmtl` to `competing_rmtl_upper` occur only when the cohort declares the `competing` status code. The values are in the unit of the age columns.

`group`, `label`, `time` As in `risks`. To merge `time_lost` with `risks`, use these three columns as the key.

`rmtl`, `rmtl_se` Restricted mean time lost to T2 from the origin to `time`, and its standard error. `NA` where `risk` is `NA`.

`rmtl_lower`, `rmtl_upper` Limits of the 95% Wald interval of the time lost, with the lower limit truncated at 0.

`competing_rmtl`, `competing_rmtl_se`, `competing_rmtl_lower`, `competing_rmtl_upper` Restricted mean time lost to the competing event, its standard error, and the limits of its 95% interval. `NA` where `competing_risk` is `NA`.

`rmtl_difference`, `rmtl_difference_se` Difference in time lost to T2 from the `reference` group and its standard error. `NA` for the reference group.

`rmtl_difference_lower`, `rmtl_difference_upper` Limits of the 95% Wald interval of the difference.

`settings` A list with the elements `score`, `sign`, `groups`, `cut_probs`, `reference`, `times`, `min_events`, `min_at_risk`, `competing`, `index`, `origin`, `delayed_entry`, and `event_labels`. `sign` is the sign of the score in `prs_cohort()`, or `NULL` for the whole cohort. `groups` is the number of groups, which is `length(cut_probs) + 1` with `cut_probs`. `cut_probs` is the input `cut_probs`, or `NULL` for groups of near-equal size. `competing` is `TRUE` when the cohort declares the `competing` status code. `origin` is the `entry_lag` of the cohort. `delayed_entry` is `TRUE` when a person enters after the origin. `event_labels` is the `event_labels` element of the cohort, which the figures and tables use.

`print()` shows the guards, the risks of at most 20 rows of `risks`, a line that names `time_lost`, and the test. With `cut_probs`, it first gives the label of each group, one per line. With a sign of `-1`, it also states that the raw score runs against the groups. It returns `x` invisibly.

References

Aalen OO, Johansen S (1978). "An empirical transition matrix for non-homogeneous Markov chains based on censored observations." *Scandinavian Journal of Statistics*, **5**(3), 141–150.

Andersen PK (2013). "Decomposition of number of life years lost according to causes of death." *Statistics in Medicine*, **32**(30), 5278–5285. doi:10.1002/sim.5903.

Andersen PK, Geskus RB, de Witte T, Putter H (2012). "Competing risks in epidemiology: possibilities and pitfalls." *International Journal of Epidemiology*, **41**(3), 861–870. doi:10.1093/ije/dyr213.

Conner SC, Trinquart L (2021). "Estimation and modeling of the restricted mean time lost in the presence of competing risks." *Statistics in Medicine*, **40**(9), 2177–2196. doi:10.1002/sim.8896.

Geskus RB (2011). "Cause-specific cumulative incidence estimation and the Fine and Gray model under both left truncation and right censoring." *Biometrics*, **67**(1), 39–49. doi:10.1111/j.1541-0420.2010.01420.x.

Gray RJ (1988). "A class of K-sample tests for comparing the cumulative incidence of a competing risk." *The Annals of Statistics*, **16**(3), 1141–1154. doi:10.1214/aos/1176350951.

Pfeiffer RM, Gail MH (2011). "Two criteria for evaluating risk prediction models." *Biometrics*, **67**(3), 1057–1065. doi:10.1111/j.15410420.2010.01523.x.

Putter H, Fiocco M, Geskus RB (2007). "Tutorial in biostatistics: competing risks and multi-state models." *Statistics in Medicine*, **26**(11), 2389–2430. doi:10.1002/sim.2712.

See Also

- `plot_risk()`, `plot_stacked()`, `plot_gradient()`, and `describe_risk()` for figures and tables of the result.
- `predict_risk()` for the model-based risk from entry.
- `horizon_support()` for the support of prediction horizons.

Examples

```
#--------------------------------------------------------------------------
# cumulative_risk() example
#--------------------------------------------------------------------------
library(prsladder)

# Simulate a cohort and standardize its progression score.
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
```

```
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(progression = "progression_score")
)

# Risk by score quartile at 5 and 10 years since the first diagnosis.
res <- cumulative_risk(cohort, "progression", times = c(5, 10), groups = 4L)
res
res$risks[c("group", "time", "risk", "difference")]

# Years lost to T2 in the first 5 and 10 years since the first diagnosis.
res$time_lost[c("group", "time", "rmtl", "rmtl_difference")]

# The top 5% of the score against the other 95%. The top 5% holds only 100
# persons, so the times are early enough for 50 persons to be at risk.
top <- cumulative_risk(
  cohort,
  "progression",
  times = c(2, 5),
  cut_probs = 0.95
)
top$risks[c("group", "time", "n", "risk", "difference")]

# Risk of the whole cohort.
cumulative_risk(cohort, NULL, times = c(5, 10))
```

`describe_case_mix` *Compare the Case Mix of Two Cohorts*

Description

`describe_case_mix()` compares the linear predictor of each model of a ladder in the development cohort and in another cohort. Use it with an external evaluation, to find whether the other cohort has the case mix of the development cohort.

A difference in case mix changes the measures of the external evaluation. A larger spread of the linear predictor usually gives a higher AUC, and a shift of its mean can change the O/E ratio. Debray et al. (2015) use these values to interpret an external evaluation. Similar values show the reproducibility of a model in persons like the development persons. Different values show its transportability to other persons.

Usage

```
describe_case_mix(ladder, newdata)
```

Arguments

ladder	(prs_ladder or prs_export) A ladder from fit_ladder() or its export from export_ladder().
newdata	(prs_cohort) A different cohort, such as another biobank, a later period, or another ancestry group. Build it with prs_cohort(data, reference = ladder$cohort), or with prs_cohort(data, reference = export) for an export.

Details

The linear predictor is that of the model of T2 of each rung. With competing events, the function does not use the competing models. The function centers the linear predictor at the covariate means of the fit, as predict(fit, type = "lp", reference = "sample") does. survival::coxph() does not center a column with values in -1, 0, and 1 only, such as a 0/1 covariate or a column of a factor. Because of this, the development mean is 0 only when the model has no such column. The difference of the means does not depend on this centering.

With design = "index" in fit_ladder(), the linear predictor does not hold the difference between the baselines of the values of t1_status. Because of this, the values do not show the effect of a different share of prevalent persons on the risks.

The function uses all persons of each cohort, also the persons outside the follow-up limits of the section "Missing risks" of predict_risk(). For a ladder, it computes the development values from ladder$cohort. For an export, it uses the element development of the export, which export_ladder() computes in the same way. Both give the same result.

The function stops with an error in these cases:

- ladder is not a ladder or an export.
- newdata is not a prs_cohort object.
- newdata has other scores, covariates, or index than the development cohort, or only one of the two declares competing events.

For a ladder, the function gives a warning with the number of persons of newdata whose ids are also in ladder$cohort. An export holds no ids, and this check is then not possible.

Value

A data frame with one row per model, in the order of ladder$models, and the columns below.

model Model name.

lp_mean_development, lp_sd_development Mean and standard deviation of the linear predictor in the development cohort.

lp_mean_external, lp_sd_external Mean and standard deviation of the linear predictor in newdata. The standard deviation is NA when newdata has one person.

lp_mean_difference lp_mean_external minus lp_mean_development.

lp_sd_ratio lp_sd_external divided by lp_sd_development.

References

Debray TPA, Vergouwe Y, Koffijberg H, Nieboer D, Steyerberg EW, Moons KGM (2015). "A new framework to enhance the interpretation of external validation studies of clinical prediction models." *Journal of Clinical Epidemiology*, **68**(3), 279–289. doi:10.1016/j.jclinepi.2014.06.018.

See Also

- `export_ladder()` for the export of a ladder.
- `describe_cohort()` for a table of the characteristics of the two cohorts.
- `assess_discrimination()` and `assess_calibration()` for the external evaluation.

Examples

```
#-------------------------------------------------------------------------
# describe_case_mix() example
#-------------------------------------------------------------------------
library(prsladder)

dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  covariates = "sex"
)
export <- export_ladder(fit_ladder(cohort))

# An external cohort whose raw progression scores are higher and more
# spread. The models with the progression score get a larger mean and a
# larger standard deviation of the linear predictor.
dat_external <- simulate_prs_data(n = 2000L, seed = 2L, id_prefix = "E")
dat_external$progression_score <- 0.5 + 1.5 * dat_external$progression_score
external <- prs_cohort(dat_external, reference = export)
describe_case_mix(export, external)
```

---

describe_cohort *Describe the Participants of a Cohort*

Description

`describe_cohort()` makes the table of participant characteristics that the PRS-RS reporting standard asks for (Wand et al. 2021). The table shows age at the first diagnosis (T1), each score on its raw scale, and each covariate. It has one column for each status of the second event (T2). When you also give an external cohort, it has one column for each cohort.

The function returns a gtsummary table. You can change and export it with the functions of the gtsummary package. The function needs gtsummary 2.6.1 or later.

Usage

```
describe_cohort(cohort, newdata = NULL, statistic = NULL, digits = NULL, ...)
```

Arguments

cohort	(prs_cohort) A cohort from prs_cohort(). When newdata is a cohort, cohort is the development cohort.
newdata	(prs_cohort or NULL; default NULL) A different cohort, from prs_cohort(data, reference = cohort). The function stops with an error when newdata has other scores, covariates, or index than cohort. If NULL, the table has one column for each status of T2, or for each event with competing events.
statistic	(formula, list of formulas, or NULL; default NULL) Statistics for gtsummary::tbl_summary(). The function applies statistic after its default of mean (SD) for continuous rows, and statistic wins. PRS-RS also asks for the range. The range is not the default, because the minimum and maximum are values of single persons. To show it, use statistic = gtsummary::all_continuous() ~ "{mean} ({sd}); {min} to {max}". If NULL, the function uses its defaults.
digits	(formula, list of formulas, or NULL; default NULL) Decimal places for gtsummary::tbl_summary(). The function applies digits after its defaults for scores on a small scale, and digits wins. If NULL, the function uses its defaults.
...	Other arguments to gtsummary::tbl_summary(), for example label, type, or include. The function sets by and stops with an error when ... holds by.

Details

The table describes the persons that prs_cohort() keeps. The n_excluded element of the cohort counts the excluded persons, and the table does not describe them. Because prs_cohort() excludes persons with a missing value, the table has no rows for missing values.

Columns:

The columns depend on newdata:

- If newdata is NULL, the columns are No second event and Second event. With competing events (see prs_cohort()), the columns are Neither event, Second event, and Competing event.
- If newdata is a cohort, the columns are Development and External. The first row then gives the number of persons with T2 in each sample. With competing events, the second row gives the number of persons with a competing event. These rows have the labels Second event and Competing event.

The `event_labels` of `prs_cohort()` replace the names of the events in the columns and rows above. For example, with `c(event = "CAD", competing = "Death")`, the columns without `newdata` are `Neither event`, `CAD`, and `Death`. Without competing events, they are `No CAD` and `CAD`.

Use `gtsummary::add_overall()` to add a column for all persons.

Rows:

The table has the rows below, in this order. Each row has a variable name, which you use in `statistic`, `digits`, `label`, `type`, and `include`.

1. `status`, the persons with T2. This row is present only when `newdata` is a cohort.
2. `competing`, the persons with a competing event. This row is present only when `newdata` is a cohort with competing events.
3. `age_t1`, age at T1.
4. `t1_status`, the persons that were prevalent at recruitment. This row is present only under the recruitment index of `prs_cohort()`.
5. One row for each score, named by the names of `scores` in `prs_cohort()`.
6. One row for each covariate, with the name and type that the cohort holds.

Statistics:

By default, a continuous row shows the mean (SD). A categorical row shows the count (percent), which is the `gtsummary` default.

Each score is on its raw scale, which is the scale of the input data before `prs_cohort()` standardizes the score and applies its sign. The function recovers the raw scores with the `mean`, `sd`, and `sign` columns of `cohort$scores`. For an external cohort, the function uses the same values from the development cohort, as `summary.prs_cohort()` does.

By default, `gtsummary` shows at most 4 decimal places. A score on a very small scale then shows as zero. For a score with a development SD below 0.1, the function shows enough decimal places for three significant digits of the SD.

Rows and tests that the table leaves out:

The table has no row for follow-up. The median of `stop - start` stops at the events and makes the follow-up look shorter than it is. `summary.prs_cohort()` reports the reverse Kaplan-Meier median instead.

The function adds no p-values, for these reasons:

- Between the statuses of T2, the groups differ in follow-up. A test there does not test the association with time to T2. The models of `fit_ladder()` test that association.
- Between a development cohort and an external cohort, a test answers no question of the analysis. With large cohorts, a small difference gives a small p-value. `describe_case_mix()` describes the difference that matters for the predictions.

Covariates that can need a change:

The function does not recode covariates. Three kinds of covariate can need a change:

- A 0/1 covariate, such as `smoker`, shows only the count of 1s under its name. Give it a label, for example `label = list(smoker ~ "Current smoker")`. Or make it a factor before you call `prs_cohort()`, which gives the same model.
- A year shows with a thousands separator, as `1,957`. Use `digits = birth_year ~ function(x) gtsummary::style_number(x, big.mark = "")`.

- A numeric covariate with fewer than 10 distinct values shows one row for each value. The models of `fit_ladder()` use it as a linear term. Use `type = k ~ "continuous"` to show it as one continuous row.

Privacy and export:

The function does not suppress small cells. In biobank data, a small cell can identify a person. The table object holds one row of data for each person, without the identifiers, because `gtsummary::add_overall()` and similar functions need these rows. Share the rendered table, not the R object. To render the table, use `gtsummary::as_kable()`, `gtsummary::as_gt()`, or `gtsummary::as_flex_table()`. When you print the table, it renders as HTML. In a terminal, print `gtsummary::as_kable(tbl)` instead.

Value

A `tbl_summary` object from `gtsummary::tbl_summary()`, with one column for each status of T2 or each sample.

References

Wand H, Lambert SA, Tamburro C, Iacocca MA, O'Sullivan JW, Sillari C, Kullo IJ, Rowley R, Dron JS, Brockman D, Venner E, McCarthy MI, Antoniou AC, Easton DF, Hegele RA, Khera AV, Chatterjee N, Kooperberg C, Edwards K, Vlessis K, Kinnear K, Danesh JN, Parkinson H, Ramos EM, Roberts MC, Ormond KE, Khoury MJ, Janssens ACJW, Goddard KAB, Kraft P, MacArthur JAL, Inouye M, Wojcik GL (2021). "Improving reporting standards for polygenic scores in risk prediction studies." *Nature*, **591**(7849), 211–219. doi:10.1038/s41586021032436.

See Also

`summary.prs_cohort()` for the counts, follow-up, and score distributions without `gtsummary`.

Examples

```
#-------------------------------------------------------------------------------
# describe_cohort() example
#-------------------------------------------------------------------------------
if (requireNamespace("gtsummary", quietly = TRUE)) {
  dat <- simulate_prs_data(n = 3000L, seed = 1L)
  development <- dat[1:2000, ]
  cohort <- prs_cohort(
    development,
    id = "IID",
    status = "exit_status",
    age_at_diagnosis = "age_at_diagnosis",
    age_at_exit = "age_at_exit",
    scores = c(onset = "onset_score", progression = "progression_score"),
    covariates = c("sex", "birth_year")
  )

  # By status of the second event.
  tbl <- describe_cohort(
    cohort,
```

```
  label = list(sex ~ "Sex", birth_year ~ "Year of birth"),
  digits = birth_year ~ function(x) gtsummary::style_number(x, big.mark = "")
)
gtsummary::as_kable(tbl)

# By sample, with the persons of another cohort.
validation <- prs_cohort(dat[2001:3000, ], reference = cohort)
gtsummary::as_kable(describe_cohort(cohort, validation))
}
```

`describe_risk` *Table of Cumulative Risk by Score Group*

Description

`describe_risk()` makes a table of the cumulative risks of `cumulative_risk()`, with one row per score group and one column per time. Each cell shows the risk and its 95% interval by default. The table is a `gtsummary` table, which you can change and export with the functions of the `gtsummary` package. The function needs `gtsummary` 2.6.1 or later.

Usage

```
describe_risk(
  x,
  event = "event",
  statistic = "{risk} ({lower}, {upper})",
  missing = "--"
)
```

Arguments

`x`	(`prs_cumulative_risk`) A result of `cumulative_risk()`.
`event`	(`character(1)`; default `"event"`) One of `"event"` or `"competing"`. The event whose risk the table shows. `"competing"` needs a cohort that declares the `competing` status code.
`statistic`	(`character(1)`; default `"{risk} ({lower}, {upper})"`) The text of each cell. The function replaces each field in braces with its value. The fields are `{risk}`, `{lower}`, `{upper}`, `{n_at_risk}`, `{n_events}`, `{n_competing}`, and `{n_censored}`. The counts are those of the `risks` element of `x`. `{n_competing}` needs a cohort that declares the `competing` status code. `statistic` must have at least one field of the risk: `{risk}`, `{lower}`, or `{upper}`. For the counts alone, use the risk table of `plot_risk()` or the `risks` element of `x`. The spanning header ends with `(95% CI)` only when `statistic` holds both `{lower}` and `{upper}`.
`missing`	(`character(1)`; default `"--"`) The text of a cell whose risk is `NA`.

Details

Rows and columns:

The table has one row per score group, or the row `All persons` for the whole cohort. With a score sign of 1, the first group has the label `1 (lowest)`. The last group has the label `k (highest)`, where `k` is the number of groups. With a score sign of -1, the labels are the group numbers only, and the footnote states that group 1 has the highest raw scores.

For groups at the `cut_probs` of `cumulative_risk()`, each label is the percent range of the group on the standardized score, such as `0-95% (lowest)` and `95-100% (highest)`. With a score sign of -1, each label also gives the range of the raw score in parentheses, such as `95-100% (raw 0-5%)`. The footnote then states that the percentiles refer to the score after its sign. The range of the raw score is in the label because `gtsummary::as_kable()` drops the footnote. The column `N` gives the number of persons in each group. Each time of `x` has one column, with the time as its header.

The function formats a risk as a percent with `gtsummary::style_percent()`, as `gtsummary::tbl_survfit()` does. A risk below 10% has one decimal, and a larger risk has none.

Missing cells:

A cell shows `missing` when the risk is `NA`, because the group does not support the time. The section "Support of a group at a time" of `cumulative_risk()` gives the conditions. The footnote states the conditions with the values of `min_events` and `min_at_risk`.

Export:

The table has a spanning header over the time columns and a footnote on that header. The footnote states the event, the estimator, the interval, and the time scale. It names the event `T2` or `the competing event`, or uses the `event_labels` of `prs_cohort()`. `gtsummary::as_kable()` drops the spanning header and the footnote. For a document, use `gtsummary::as_gt()` or `gtsummary::as_flex_table()`, or state the estimator and the meaning of `missing` in the text. The risk is a Kaplan-Meier estimate, or an Aalen-Johansen estimate with competing events. The interval is the pointwise 95% interval on the log scale of the risk.

For a table of both events, stack the two tables: `gtsummary::tbl_stack(list(t2, competing), group_header = c("T2", "Death"))`.

Privacy:

The table holds only formatted cells, with no row of data for a person. The object of `describe_cohort()` holds one row of data for each person. The function does no disclosure control. A count in a cell, or the product of the risk and the group size, can give a small number of events. Apply the rules of the data provider before you share the table.

Value

A `gtsummary` object from `gtsummary::as_gtsummary()`. Its table body has one row per group and the columns `variable`, `row_type`, `label`, `stat_0` for `N`, and `stat_1` to `stat_m` for the `m` times. The columns `variable` and `row_type` are hidden.

See Also

`plot_risk()` for the curves, and `plot_gradient()` for the risks at one time in a figure.

Examples

```
#------------------------------------------------------------------------------
# describe_risk() example
#------------------------------------------------------------------------------
if (requireNamespace("gtsummary", quietly = TRUE)) {
  dat <- simulate_prs_data(n = 2000L, seed = 1L, competing_risk = 0.2)
  cohort <- prs_cohort(
    dat,
    id = "IID",
    status = "exit_status",
    age_at_diagnosis = "age_at_diagnosis",
    age_at_exit = "age_at_exit",
    scores = c(progression = "progression_score"),
    status_codes = list(censored = 0, event = 1, competing = 2)
  )
  res <- cumulative_risk(cohort, "progression", times = c(5, 10), groups = 4L)

  # Cumulative incidence of the second event by score quartile.
  t2 <- describe_risk(res)
  gtsummary::as_kable(t2)

  # Both events in one table.
  death <- describe_risk(res, event = "competing")
  both <- gtsummary::tbl_stack(list(t2, death), group_header = c("T2", "Death"))
  gtsummary::as_kable(both)

  # The risk with the number of T2 events by each time.
  gtsummary::as_kable(describe_risk(res, statistic = "{risk} ({n_events} events)"))
}
```

`export_ladder` *Export a Ladder for External Validation*

Description

`export_ladder()` writes the models of a ladder into an object that holds no person-level data. Use it when another biobank validates the models and the data of the development persons cannot leave their biobank.

Save the export with `base::saveRDS()` and send the file to the other biobank. There, `prs_cohort()` with `reference = export` builds the cohort of the other persons with the rules of the development cohort. The `assess_*()` functions then evaluate the models of the export on this cohort, and `describe_case_mix()` compares the two cohorts. `predict_risk()` and `prediction_data()` also accept the export and its models.

Usage

```
export_ladder(ladder, min_cell_count = 10L)
```

Arguments

`ladder`	(`prs_ladder`) A ladder from `fit_ladder()`.
`min_cell_count`	(`integer(1)`; default `10L`; range `[1, Inf)`) Least number of events in each step of a baseline. It is also the least number of persons with each level or value of a covariate (see the section "Rare values"). The function also accepts a double with no fractional part, such as `10`. A value above 10 also narrows the follow-up limits (see the section "Follow-up limits").

Details

Contents and disclosure:

A fit of a ladder holds the model matrix, the response, and other values of each person. Because of this, a ladder cannot leave its biobank. The export holds only the values below. A print of the export lists them for the disclosure review.

- `models`: the terms, the coefficients, and the covariate means of each model, and its two baselines and follow-up limits. The knots of each spline are percentiles of the values of its variable (see the section "Spline terms" of `fit_ladder()`). With the default `spline_df`, the variable is the age at T1. A percentile can equal the value of one person, and it is otherwise the weighted mean of the values of two persons. With `design = "index"` in `fit_ladder()`, the knots are percentiles of the ages at entry and of the T1 durations above the smallest value. Each value of `t1_status` then has its own two baselines and follow-up limits.
- `rules`: the rules of the development cohort. These rules are the column names, the status codes, and the event labels. They also include the mean, standard deviation, and sign of each score. Last, they include the type and the factor levels of each covariate, the index, the entry lag, and the `prevalent` setting.
- `development`: the number of persons and of events, and the mean and standard deviation of the linear predictor of each model.
- `settings`: `min_cell_count`, the argument `design` of `fit_ladder()`, the version of the package, and the date.

The export holds no variance matrix, because no measure of the external evaluation uses one. The final rule for each value belongs to the disclosure review of the biobank.

Follow-up limits:

Each model of the export has the follow-up limits of the section "Missing risks" of `predict_risk()`, with k = `max(10, min_cell_count)` in place of 10. With `design = "index"` in `fit_ladder()`, each value of `t1_status` has these limits from its own persons, on the time since entry. The lower limit is midway between the k-th and the $(k + 1)$-th smallest entry, `start`, of the development persons. The upper limit is midway between the k-th and the $(k + 1)$-th largest exit, `stop`. Each limit is then the mean of the values of two persons. When more than k persons share the smallest entry, such as the entry lag, the lower limit is that value, and it shows no single person. In other cases, a person who knows their own value can compute the value of the other person.

- With the default `min_cell_count = 10`, the export and the ladder have the same limits, and they evaluate the same persons.
- A `min_cell_count` above 10 gives narrower limits. The export then evaluates fewer persons than the ladder.

- A `min_cell_count` below 10 keeps the limits of the ladder, and only the steps of the baselines get smaller. The export then never predicts a risk for a person that the ladder does not keep.

Merged baselines:

The exact baseline of a fit has one step at each event time. Each step then shows the time of about one event. For each model and each baseline, the export does these steps, also for each value of `t1_status` with `design = "index"` in `fit_ladder()`:

1. It uses only the event times between the lower and the upper limit.
2. It merges consecutive event times into steps of at least `min_cell_count` events. A short last step joins the step before it.
3. It puts the first knot at the lower limit and the last knot at the upper limit. Each other knot is midway between two steps. No inner knot is then an event time, but it is the mean of the times of two events.
4. It stores the exact cumulative hazard at each knot minus its value at the lower limit. The first stored value is then 0.
5. Between two knots, the cumulative hazard is linear.

The export applies these steps to the Efron baseline of the risks and the Breslow baseline of the expected counts. With competing events, it applies them also to the baselines of the competing models. The events before the lower limit do not enter the export. Each risk and each expected count uses a difference of the cumulative hazard at two times between the limits, which the export keeps.

Accuracy:

The merged baselines change the predicted risks a little. The values below come from cohorts of `simulate_prs_data()` with 2000 or 20,000 persons. The development cohort has the seed 1, and the other persons have the seed 2. The cohorts have the three scores, the covariates `sex`, `PC1`, and `PC2`, and no entry lag. The values are the absolute changes of the net risk of the model with three scores, from the exact baselines to the export, on the other persons:

Index	Persons	Horizon	Largest change	Mean change
Diagnosis	2000	5	0.0001	0.00003
Diagnosis	2000	10	0.0020	0.0013
Diagnosis	2000	20	0.0100	0.0094
Diagnosis	20,000	10	0.0001	0.00006
Diagnosis	20,000	20	0.0003	0.00025
Recruitment	2000	5	0.0082	0.0005
Recruitment	2000	10	0.020	0.0017
Recruitment	20,000	10	0.0010	0.00007
Recruitment	20,000	20	0.0034	0.00062

At 1 year with 2000 persons, the mean change was 3% to 4% of the mean risk. Under the diagnosis index with an entry lag of 1 year, it was 7%. At 5 to 20 years with 2000 persons, the O/E ratio of `assess_calibration()` changed by at most 0.0068. At 1 year, it changed by up to 0.044, and by up to 0.048 with the entry lag of 1 year. These changes are small against the Poisson interval of the ratio at 1 year. This interval was 0.82 to 1.37 for 60 events, and 0.53 to 1.04 for 34 events with the entry lag. The change is largest at a short horizon. It is also large where few

persons remain in follow-up, because a step of min_cell_count events can then span years. The reference of these values is the exact estimate, which is itself noisy when few persons remain. The development biobank does not use the export. Its apparent and optimism-corrected results use the exact baselines.

Rare values:

The fit estimates the coefficient of a factor level from the persons with that level. For a rare level, the estimate then shows the outcomes of few persons. The function stops with an error when a level of a factor covariate has fewer than min_cell_count persons. The message names the covariate and the level. Merge the level with another level in the data, then build the cohort and the ladder again.

The same rule applies to a numeric covariate with exactly two distinct values. Such a covariate has the coefficient of the equivalent factor. When its values are not 0 and 1, survival::coxph() also stores its mean. The mean and the number of persons then give the count of each value. For example, 4 of 1500 persons have the value 2 of a 1/2 coding. The mean is then 1.00267, and $1500 \times 0.00267 = 4$.

The same rule applies to t1_status with design = "t1_entry" in fit_ladder(), because the term has a coefficient. With design = "index", t1_status is a stratum, and the export shows the baselines and the follow-up limits of each value. The rule then applies because these values come from the persons with that value. The rule also applies to the persons with t1_duration above its smallest value. The smallest value is the entry lag for every incident person, and the fit estimates the coefficients of t1_duration from the persons above it. Their count must be at least min_cell_count. The message then asks for a fit with design = "t1", which has no term for these columns.

Errors:

The function stops with an error in these cases:

- ladder is not a ladder from fit_ladder().
- min_cell_count is not a whole number of 1 or more.
- A covariate, t1_status, or t1_duration has a rare value (see the section "Rare values").
- A model of T2 or a competing model has fewer than min_cell_count events between its follow-up limits. With design = "index" in fit_ladder(), this rule applies to each value of t1_status.

Value

An object of class prs_export, a list with the elements below.

models A named list with one object of class prs_risk_model per model, in the order of ladder$models. Each object holds the terms with the knots of each spline, the factor levels, the coefficients, and the covariate means. It also holds the merged baselines baseline and the follow-up limits lower and upper. With design = "index" in fit_ladder(), its element parts holds these values, with one object for each value of t1_status. Its element competing is the competing model of the rung, an object of the same class. Without competing events, competing is NULL.

rules The rules of the development cohort, a list with the elements columns, status_codes, event_labels, scores, covariates, levels, index, entry_lag, and prevalent. levels has one element per covariate. The element holds the factor levels, or NULL for a numeric covariate. The other elements are those of prs_cohort().

`development` A data frame with one row per model and the columns `model`, `scores` (a list column of score names), `n`, `n_events`, `lp_mean`, and `lp_sd`. `lp_mean` and `lp_sd` are the mean and the standard deviation of the linear predictor of the model of T2. They use all development persons (see `describe_case_mix()`).

`settings` A list with `min_cell_count` (integer), `design` (the argument `design` of `fit_ladder()`), `version` (the version of prsladder, a string), and `date` (the date of the export).

See Also

- `prs_cohort()` with `reference` to build the cohort of other persons.
- `assess_calibration()`, `assess_discrimination()`, `assess_thresholds()`, and `assess_net_benefit()` for the external evaluation.
- `describe_case_mix()` to compare the linear predictors of the two cohorts.
- `predict_risk()` for the risk of one model of the export.

Examples

```
#-------------------------------------------------------------------------
# export_ladder() example
#-------------------------------------------------------------------------
library(prsladder)

# At the development biobank: fit the ladder and export it.
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  covariates = "sex"
)
ladder <- fit_ladder(cohort)
export <- export_ladder(ladder)
export

# The file that goes to the other biobank.
file <- tempfile(fileext = ".rds")
saveRDS(export, file)

# At the other biobank: the rules of the export build the cohort.
imported <- readRDS(file)
external <- prs_cohort(
  simulate_prs_data(n = 2000L, seed = 2L, id_prefix = "E"),
  reference = imported
)
calibration <- assess_calibration(
  imported,
  horizons = 10,
  newdata = external
```

```
)
calibration$models[c("model", "evaluation", "oe_ratio", "slope")]
```

`fit_ladder` *Fit the Ladder of Cox Models*

Description

`fit_ladder()` fits one Cox proportional-hazards model for each subset of the polygenic risk scores in a cohort. The subsets go from the base model, which has no scores, to the full model, which has all the scores. Every model has the same persons, covariates, and age term. Because of this, `compare_ladder()` can compare any two nested models.

Other functions of the package take the result, or one of its models, as their input. The section "See Also" lists them by task.

Usage

```
fit_ladder(
  cohort,
  spline_df = c(age_t1 = 3L),
  design = c("t1", "t1_entry", "index")
)
```

Arguments

`cohort`	(`prs_cohort`) A cohort from `prs_cohort()` with at least one event.
`spline_df`	(named `integer()` or `NULL`; default `c(age_t1 = 3L)`; range `[1, 8]`) Degrees of freedom of the variables with a spline term (see the section "Spline terms"). Each name must be `age_t1` or a numeric covariate of the cohort, and the names must be unique. The value `1` gives a linear term, and a larger value gives a natural cubic spline. A variable that the argument does not name has a linear term. `NULL` or a numeric vector of length zero gives a linear term to every variable. The function also accepts doubles with no fractional part, such as `3`. The function stops with an error for a missing value. The element `age_t1` also sets the degrees of freedom of the duration term of `design = "index"`. Each degree of freedom above 1 adds one coefficient to every model. For `age_t1` with `design = "index"`, it adds two coefficients. The upper limit of 8 stops an input error, such as `30` in place of `3`, before the function fits any model. The function stops with an error when the data do not support a spline term (see the section "Errors and warnings").
`design`	(`character(1)`; default `"t1"`) One of `"t1"`, `"t1_entry"`, or `"index"`. The model family of every model and competing model of the ladder (see the section "Designs"). `"t1"` gives the models of the section "Models". `"t1_entry"` adds the terms `t1_status`

and t1_duration, the columns of a cohort under the recruitment index (see prs_cohort()). "index" uses the time since entry as its time scale, with one baseline hazard for each value of t1_status. Its age term is a function of the age at entry, age_t1 + t1_duration, and it adds a term for t1_duration. The argument newdata of predict_risk() needs t1_status and t1_duration for "t1_entry" and "index".

Details

Models:

With k scores, the ladder has 2^k models. With the default design = "t1", each model has this formula:

```
survival::Surv(start, stop, status) ~ <scores in the subset> +
  <covariates> + <age term>
```

The argument spline_df selects the variables with a spline term (see the section "Spline terms"). With the default spline_df, the age term is natural_spline(age_t1, df = 3), and each covariate has a linear term or treatment contrasts. A covariate that spline_df names has a spline term in the same form. The age term is age_t1 when spline_df does not name age_t1, or gives it the value 1. The function fits each model with survival::coxph() and the Efron method for ties. The time scale is the time since the first diagnosis (T1). Each person enters the risk set at start, which gives delayed entry. prs_cohort() sets start from the index and the entry lag.

Let x_i be the standardized scores of person i, z_i the covariates, and a_i the age at T1. The hazard at time t is

$$h(t \mid i) = h_0(t) \exp\{\beta^\top x_i + \alpha^\top z_i + f(a_i)\},$$

where h_0 is the baseline hazard and f is the age term. A factor covariate enters z_i as treatment contrasts. A covariate with a spline term enters z_i as its spline columns. The section "Designs" gives the models of the other designs.

Designs:

Under the recruitment index, a prevalent person enters the cohort after T1. The argument design selects how the models treat this entry. Every model and competing model of a ladder has the same design.

design	Time scale	Use
"t1"	Time since T1	Hazard ratios and primary tests
"t1_entry"	Time since T1	Absolute risks, when check_design() picks it
"index"	Time since entry	Absolute risks, when check_design() picks it

The pre-specified analysis of the hazard ratios uses "t1", because a test must not depend on a design that the data selected. Choose this workflow before the analysis. check_design() gives the tests and the pick for the absolute risks. The other two designs need the recruitment index, with incident and prevalent persons.

Let p_i be 1 for a prevalent person and 0 for an incident person. Let d_i be `t1_duration`, and let $u = t - d_i$ be the time since entry. Let $\eta_i = \beta^\top x_i + \alpha^\top z_i$. The hazards of the three designs are

$$\begin{aligned} \text{t1}: &\quad h(t \mid i) = h_0(t) \exp\{\eta_i + f(a_i)\}, \\ \text{t1_entry}: &\quad h(t \mid i) = h_0(t) \exp\{\eta_i + f(a_i) + \gamma p_i + \delta d_i\}, \\ \text{index}: &\quad h(u \mid i) = h_{0p_i}(u) \exp\{\eta_i + f_1(a_i + d_i) + f_2(d_i)\}. \end{aligned}$$

The formulas of `"t1_entry"` and `"index"` are:

```
survival::Surv(start, stop, status) ~ <scores in the subset> +
  <covariates> + t1_status + t1_duration + <age term>
survival::Surv(0 * start, stop - start, status) ~ <scores in the subset> +
  <covariates> + <age term at entry> + <duration term> + strata(t1_status)
```

The term `t1_status` of `"t1_entry"` has the coefficient `t1_statusprevalent`, with incident persons as the reference. The coefficient compares a prevalent person with `t1_duration` equal to the entry lag to an incident person at the same time since T1.

Each design has its own assumption on the hazard after entry:

- `"t1"` assumes one hazard for an incident and a prevalent person with the same terms at the same time since T1. Delayed entry then corrects for the selection of the prevalent persons.
- `"t1_entry"` assumes that the hazards of the two groups differ by a constant factor at the same time since T1. The factor does not change with the time since entry. The design also assumes that the log hazard is linear in `t1_duration`. Gail et al. (2009) state that left truncation must always be handled. They also state that an adjustment for the entry is needed when the baseline hazard depends on the entry. A selection of healthy persons at entry can cause this dependence.
- `"index"` assumes that the effect of `t1_duration` does not change with the time since entry. Each value of `t1_status` has its own baseline hazard on this time. For prevalent persons, the model is similar to a landmark model at entry (van Houwelingen 2007). A landmark model keeps the time since T1 as its time scale. This design measures the time from entry. When the hazard depends on the time since T1, the assumption needs a log hazard that is linear in this time. Korn et al. (1997) derive this condition for time on study in place of age, with the age at entry as a covariate. Thiébaut and Bénichou (2004) confirm it by simulation. Both recommend age as the time scale.

To simulate data with an effect of the entry on the hazard, use the arguments `prevalent_hazard_ratio`, `duration_hazard_ratio`, and `prevalent_effect_years` of `simulate_prs_data()`.

A risk from T1 for a prevalent person is less precise under `"t1_entry"` and `"index"`. This prediction sets `t1_duration` to the entry lag, and it keeps the age at T1 and the birth year of the person. It then describes an incident person with the calendar year of T1 of the prevalent person. With a short recruitment period, most prevalent persons have a T1 before the first T1 of an incident person. The prediction then extrapolates. `prediction_data()` without `age_at_entry` and `standardized_risk()` give risks from T1.

Simulations measured this cost, with 200 cohorts in each setting and the birth year in each model. With no effect of the entry, the bias of the mean risk from T1 was at most 0.005 in each design. The standard deviation of this mean over the cohorts depended on the recruitment period and on the share of incident persons. For a period of 4 years, it was 2.7 to 2.9 times the value of `"t1"` with about 30% incident persons. It was 1.6 to 2.4 times this value with about half incident persons. For a period of 23 years with about half incident persons, it was 1.2 to 1.4 times this value. The

mean risk from entry had no such cost, with a standard deviation at most 1.1 times the value of `"t1"`.

This cost is not a reason to use `"t1"` for absolute risks. Other simulations multiplied the hazard of the prevalent persons by 1.5, or by a factor that fell after recruitment. The target was the mean risk from T1 with the hazard of an incident person, with a true value of about 0.28. The bias of `"t1"` was 0.03 to 0.05, and more persons did not reduce it. The bias of `"t1_entry"` was at most 0.004 with the factor 1.5, and 0.009 to 0.013 with the other factor. The bias of the pick of `check_design()` was at most 0.007.

Do not compare the designs with the columns of `table`:

- The risk sets of `"index"` differ from the risk sets of the other designs. Because of this, `loglik` and `aic` of `"index"` are not comparable with the values of `"t1"` and `"t1_entry"`.
- `concordance` of `"index"` is a value within strata. It compares only persons with the same value of `t1_status`, on the time since entry.

Apparent performance also favors `"index"` (see validation).

Spline terms:

The argument `spline_df` gives the degrees of freedom of each variable with a spline term. A name is `age_t1`, for the age term, or a numeric covariate of the cohort. The value 1 gives a linear term. A value of 2 or more gives a natural cubic spline with that number of degrees of freedom from `splines::ns()`. A variable that `spline_df` does not name has a linear term. The default gives a spline with 3 degrees of freedom to the age at T1 and a linear term to each covariate.

A value of `spline_df` replaces the default. For example, `spline_df = c(birth_year = 3)` gives a spline term for the covariate `birth_year` and a linear age term. To keep the spline of the age term, use `spline_df = c(age_t1 = 3, birth_year = 3)`. Choose `spline_df` before the analysis.

The outer knots of a spline are the 5th and the 95th percentiles of the values of its variable. Its inner knots are quantiles of the values between the outer knots. The spline is linear below and above the outer knots. The default knots of `splines::ns()` would put the outer knots at the smallest and the largest value, which are the values of two persons. A percentile is an aggregate value, but it can still equal the value of one person.

A spline term has these limits:

- A spline approximates the effect of its variable. A strongly curved effect, such as a U shape, can need more than 3 degrees of freedom.
- A variable with a large share of the persons at one value does not support a spline term. An example is a count with many persons at zero. The function then stops with an error (see the section "Errors and warnings").
- A score always has a linear term, because the ladder reports the hazard ratio per standard deviation of each score.
- A factor covariate and the entry terms `t1_status` and `t1_duration` cannot have an element in `spline_df`.

With `design = "index"`, the age term uses the age at entry, `age_t1 + t1_duration`, in place of the age at T1. The models also have a duration term with the degrees of freedom of `age_t1`. With a linear age term, the two terms are `age_t1` and `t1_duration`, which span the linear terms of the age at entry and of the duration. With a spline of the age term, the duration term is a natural cubic spline. Its knots come from the persons with `t1_duration` above its smallest value, by the rule of the other splines. The spline is linear below and above the outer knots. Every incident person has the same value of the term, and the baseline of these persons absorbs it.

Each fit stores its knots. A prediction for new persons then uses the knots of the fit. The formula of each fit holds the internal function `natural_spline()`, or `duration_spline()` for the duration term, which finds the knots again in other data. A refit with `formula(fit)` on other persons, such as a bootstrap sample, then gets the knots of those persons.

Heuristic shrinkage:

The column `heuristic_shrinkage` of `table` is the heuristic shrinkage factor of van Houwelingen and le Cessie (1990):

$$\hat{\gamma} = \frac{\chi^2_{LR} - p}{\chi^2_{LR}},$$

where χ^2_{LR} is the likelihood-ratio statistic of the model against the model with no coefficients, and p is the number of coefficients. The count p includes each spline column and each factor contrast. A value near 1 shows little overfitting. A value near 0, or below 0, shows that the fit is mostly noise. The function does not truncate the value. The value depends on the strength of the predictors, not only on the number of events per coefficient. With competing events, the value describes the model of T2.

Competing events:

When the cohort declares the `competing` status code (see `prs_cohort()`), the function also fits a Cox model for the competing event in each rung:

```
survival::Surv(start, stop, competing) ~ <same terms as the rung>
```

With `design = "index"`, the response is `survival::Surv(0 * start, stop - start, competing)`. Each model of T2 treats a competing event as censoring, and each model of the competing event treats T2 as censoring. These models are the cause-specific hazards models, and separate fits for each cause give the same estimates as one joint fit of both causes (Putter et al. 2007). The models of T2 are the same as without the competing role. The likelihood-ratio tests of `compare_ladder()` are then tests on the cause-specific hazard of T2. The concordance, the hazard ratios, and the proportional-hazards test of `table` and `coefficients` are for the cause-specific hazard of T2.

Each rung has its own competing model, with the same terms as the model of T2. The `base` rung then has a competing model with no scores. Because of this, a comparison of two rungs by cumulative incidence includes the change in the competing model.

Each model of T2 holds the competing model of its rung as the attribute `"competing"`. `predict_risk()` uses this attribute to give the cumulative incidence. To get the competing models, use `lapply(ladder$models, attr, "competing")`. The function `stats::update()` on a model of T2 drops the attribute and the class `prs_fit`, and `predict_risk()` does not accept its result.

Model names and order:

- The model without scores has the name `base`.
- The name of each other model is its score names joined by `+`, for example `onset+progression`.
- The models are in order of the number of scores, then in the order of the scores in `cohort$scores`. For three scores `a`, `b`, and `c`, the order is `base`, `a`, `b`, `c`, `a+b`, `a+c`, `b+c`, and `a+b+c`.

Errors and warnings:

The function stops with an error in these cases:

- `design` is not `"t1"` under the diagnosis index. Every person then has the same entry, and the entry terms have no variation.

- design is not "t1", and all persons of the cohort have the same value of t1_status. An example is a cohort with prevalent = "drop".
- design is "index", and a value of t1_status has 10 persons or fewer. Each follow-up limit of a value leaves 10 persons outside (see the section "Missing risks" of predict_risk()).
- design is "index", and survival::coxph() would reduce the time since entry of a person to zero.
- The cohort has no events.
- The cohort declares the competing status code but has no competing events.
- The cohort has only one distinct age at T1.
- A variable with a spline term has equal 5th and 95th percentiles. Equal percentiles need more than 90% of the persons at the same value. The value 1 in spline_df gives a linear term.
- splines::ns() cannot build the spline of a variable, gives a warning, or gives collinear columns. This occurs when the variable has fewer distinct values than its degrees of freedom plus 1. It also occurs when a large share of the persons has the same value. A smaller value in spline_df can fix the problem.
- design is "index", and the age term is a spline. The same conditions then hold for the ages at entry, age_t1 + t1_duration, and for the T1 durations above the smallest value.

The function checks the spline terms before it fits a model.

- survival::coxph() gives a warning for a model, for example when the fit does not converge. The error message names the model. For a competing model, the name ends with (competing event). A warning can refer to a coefficient by its number, as in "Loglik converged before variable 2". The message then also gives the name of each such coefficient, for example siteB for the level B of a factor site. A term that separates persons with and without the event, such as a factor level with no events, is a frequent cause.
- A model has an aliased coefficient because a score, a covariate, or an entry term is collinear with other terms. An aliased coefficient would change the degrees of freedom of the likelihood-ratio tests without notice.
- Two coefficients of a model have the same name. An example is a factor covariate grp with the level b and a numeric covariate grpb. predict_risk() finds a coefficient by its name, and the risks would be wrong without notice.
- survival::cox.zph() fails for a model.

The function gives a warning in these cases:

- The cohort has more than four scores. The number of models doubles with each added score.
- The full model has fewer than 10 events per coefficient. The count of coefficients includes each spline column and each factor contrast. The function checks the full competing model in the same way.

Proportional hazards:

The function tests the proportional-hazards assumption of every model with survival::cox.zph() and transform = "km". The global p-value of each test is in the ph_p_value column of table. This p-value is advisory, and a small value does not stop the function. A small value shows that a hazard ratio can change over follow-up. The estimated hazard ratio is then an average over follow-up.

Stored fits:

Each fit has the class `c("prs_fit", "coxph")`. The class `prs_fit` marks a fit with Efron ties, which `predict_risk()` needs. Only the design `"index"` has strata. Because the class keeps `"coxph"`, the methods of survival, such as `stats::coef()`, `stats::vcov()`, and `survival::cox.zph()`, work on each fit.

The environment of each formula is the namespace of the package, which R saves by name only. With `design = "index"`, the environment is a child of the namespace that holds only the function `strata()` of survival. Each fit then holds no analysis data in its environment. Each fit keeps its model matrix. Because of this, `predict_risk()` and `stats::predict()` work on a fit without the analysis data. Each fit also keeps vectors with one element per person. For 20,000 persons, two scores, and three covariates, each fit uses about 5.5 MB of memory.

Index event bias:

The cohort includes only persons with a first diagnosis. This selection can bias the hazard ratio of a score that affects T1, such as an onset score. The bias occurs when factors that the model does not include affect both T1 and T2. Dudbridge et al. (2019) call this bias index event bias. The package does not correct this bias, and a report should state it.

Value

An object of class `prs_ladder`, a list with the elements below.

`models` A named list of `survival::coxph()` fits of class `c("prs_fit", "coxph")`, one per model, in the order given in Details. With competing events, each fit holds the competing model of its rung as the attribute `"competing"`.

`table` A data frame with one row per model and the columns below.

- `model` The model name.
- `scores` A list column with the score names of the model.
- `n_scores` Integer number of scores.
- `n` Integer number of persons.
- `n_events` Integer number of events.
- `loglik` The partial log-likelihood.
- `aic` The AIC.
- `concordance` Harrell's concordance on the development persons. It is the apparent value, which is too high for new persons.
- `concordance_se` The standard error of `concordance`.
- `events_per_parameter` The number of events divided by the number of coefficients.
- `heuristic_shrinkage` The heuristic shrinkage factor (see the section "Heuristic shrinkage").
- `ph_p_value` The global p-value of the proportional-hazards test.

`coefficients` A data frame with one row per model and coefficient, and the columns below.

- `model` The model name.
- `term` The coefficient name.
- `estimate` The log hazard ratio.
- `se` The standard error of `estimate`.
- `hr` The hazard ratio.
- `hr_lower`, `hr_upper` The limits of the Wald 95% confidence interval of `hr`.

`p_value` The p-value of the two-sided Wald test.

A factor covariate has one row per contrast. A spline term has one row per basis column. For a score, `hr` is per one standard deviation of the raw score, in the direction of the score sign.

`competing_coefficients` `NULL`, or with competing events, a data frame with the columns of `coefficients` for the competing models.

`cohort` The input `cohort`, unchanged.

References

Dudbridge F, Allen RJ, Sheehan NA, Schmidt AF, Lee JC, Jenkins RG, Wain LV, Hingorani AD, Patel RS (2019). "Adjustment for index event bias in genome-wide association studies of subsequent events." *Nature Communications*, **10**, 1561. doi:10.1038/s4146701909381w.

Gail MH, Graubard B, Williamson DF, Flegal KM (2009). "Comments on 'Choice of time scale and its effect on significance of predictors in longitudinal studies' by Michael J. Pencina, Martin G. Larson and Ralph B. D'Agostino, Statistics in Medicine 2007; 26:1343–1359." *Statistics in Medicine*, **28**(8), 1315–1317. doi:10.1002/sim.3473.

Korn EL, Graubard BI, Midthune D (1997). "Time-to-event analysis of longitudinal follow-up of a survey: choice of the time-scale." *American Journal of Epidemiology*, **145**(1), 72–80. doi:10.1093/oxfordjournals.aje.a009034.

Putter H, Fiocco M, Geskus RB (2007). "Tutorial in biostatistics: competing risks and multi-state models." *Statistics in Medicine*, **26**(11), 2389–2430. doi:10.1002/sim.2712.

Thiébaut ACM, Bénichou J (2004). "Choice of time-scale in Cox's model analysis of epidemiologic cohort data: a simulation study." *Statistics in Medicine*, **23**(24), 3803–3820. doi:10.1002/sim.2098.

van Houwelingen HC (2007). "Dynamic prediction by landmarking in event history analysis." *Scandinavian Journal of Statistics*, **34**(1), 70–85. doi:10.1111/j.14679469.2006.00529.x.

van Houwelingen JC, le Cessie S (1990). "Predictive value of statistical models." *Statistics in Medicine*, **9**(11), 1303–1325. doi:10.1002/sim.4780091109.

See Also

Before the fit:

- `check_design()` for the choice of a design.

Functions that take the ladder:

- `summary.prs_ladder()` and `plot_ladder()` for the score hazard ratios.
- `compare_ladder()` for the nested tests.
- `standardized_risk()` for the absolute risk at score percentiles.
- `assess_discrimination()`, `assess_calibration()`, `assess_thresholds()`, and `assess_net_benefit()` for the performance of the models. validation gives an overview.

- assess_stability() for the stability of the predicted risks.
- export_ladder() for an export that holds no person-level data, and describe_case_mix() for the case mix of two cohorts.

Functions that take one model of the ladder:

- predict_risk() for the absolute risk of a person.

Examples

```
#------------------------------------------------------------------------------
# fit_ladder() example
#------------------------------------------------------------------------------
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  covariates = c("sex", "PC1", "PC2", "birth_year")
)
ladder <- fit_ladder(cohort)
ladder

# Hazard ratios of the scores in every model.
summary(ladder)

# The fitted formula of a model.
formula(ladder$models[["onset+progression"]])

# A spline term for the birth year, with the default spline of the age term.
flexible <- fit_ladder(cohort, spline_df = c(age_t1 = 3, birth_year = 3))
formula(flexible$models[["onset+progression"]])
```

---

horizon_support *Check the Support of Prediction Horizons*

Description

horizon_support() checks whether a cohort has enough data to evaluate a risk prediction at each horizon. A horizon is a time since entry, such as 5 or 10 years. For each horizon, the function counts the events and the persons still in follow-up. It also estimates the probability that a person is not yet censored. It then compares these values with three minimums.

The functions assess_calibration(), assess_discrimination(), assess_net_benefit(), assess_stability(), and assess_thresholds() apply the same check. They skip each horizon that fails the check and give a warning. Use horizon_support() before these functions to select horizons, and to find which minimum a skipped horizon fails.

Usage

```
horizon_support(
  cohort,
  horizons,
  min_events = 10L,
  min_at_risk = 50L,
  min_censoring_survival = 0.02
)
```

Arguments

`cohort`	(`prs_cohort`) A cohort from `prs_cohort()`.
`horizons`	(`numeric()`; range `(0, Inf)`) Distinct finite times since entry, in the units of the age columns.
`min_events`	(`integer(1)`; default `10L`; range `[1, Inf)`) Minimum number of T2 events at or before a horizon.
`min_at_risk`	(`integer(1)`; default `50L`; range `[1, Inf)`) Minimum number of persons whose time since entry is at least the horizon.
`min_censoring_survival`	(`numeric(1)`; default `0.02`; range `(0, 1]`) Minimum censoring survival $G(h)$ at a horizon h. A value of `1` requires no censoring at or before the horizon.

Details

Time scale:

All counts use time since entry, `stop - start`, which is the time scale of a prediction from `predict_risk()`. Under the recruitment index of `prs_cohort()`, time since entry differs from time since the first diagnosis.

Conditions for support:

The cohort supports a horizon h when all three conditions below are true:

1. At least `min_events` T2 events occur at or before h.
2. At least `min_at_risk` persons have a time since entry of at least h.
3. The censoring survival $G(h)$ is at least `min_censoring_survival`.

Each comparison includes equality. A horizon after the last follow-up has zero persons at risk, and the cohort does not support it.

The event minimum selects samples. Consider a horizon where the expected number of events is near `min_events`. At this horizon, the first condition passes more often in samples with more events than expected. At a horizon that passes by a small margin, the observed event count then tends to be too high. The observed-to-expected ratio of `assess_calibration()` also tends to be too high there. At a horizon with many more events than `min_events`, the condition passes in almost every sample and selects nothing. The effect on the other metrics is not measured. Choose horizons with enough events for a validation, not horizons that pass the minimum by a small

margin. For a binary outcome, Vergouwe et al. (2005) suggest at least 100 events and 100 non-events. Riley et al. (2022) give a simulation-based sample size for a time-to-event outcome at one horizon. It targets the precision of the calibration slope, and it checks the precision of the other metrics. Each horizon needs its own calculation. In the examples of the article, the calculation gave 366 to 1430 events.

Censoring survival:

$G(h)$ is the probability that a person is not censored by time h. The function estimates it with the reverse Kaplan-Meier estimator, which treats censoring as the event. At a time with both events and censorings, the events leave the censoring risk set first. This order is the convention of `riskRegression::Score()` and `prodlim::prodlim(..., reverse = TRUE)`. `survfit(Surv(time, 1 - status) ~ 1)` gives different values at tied times. With competing events (see `prs_cohort()`), a competing event ends follow-up and is not censoring.

Under the recruitment index, the incident and the prevalent persons can have different censoring on time since entry. When the cohort holds both groups of `t1_status`, the function estimates G_s for each group s. The censoring weights of `assess_discrimination()` use the same groups. The largest weight in group s is $1/G_s(h)$. Because of this, the guard uses the smallest $G_s(h)$.

The prediction metrics weight each person by the inverse of G (Gerds and Schumacher 2006; Blanche et al. 2013). The largest weight at horizon h is $1/G(h)$. Because of this, the default `min_censoring_survival = 0.02` limits each weight to at most 50.

Difference from the metric functions:

The metric functions apply the check at each horizon only to the persons with a prediction at that horizon. `predict_risk()` gives `NA` for a person outside the follow-up limits of the fit. The section "Missing risks" of `predict_risk()` defines these limits. Because the metric functions leave out these persons, a metric function can skip a horizon that `horizon_support()` supports. It can also evaluate a horizon that `horizon_support()` does not support.

Value

A data frame with one row per horizon, in the order of `horizons`, and the columns below.

`horizon` The horizon, as given in `horizons`.

`n` Integer number of persons in the cohort. This value is the same in each row.

`n_events` Integer number of T2 events at or before the horizon.

`n_at_risk` Integer number of persons whose time since entry is at least the horizon.

`censoring_survival` Reverse Kaplan-Meier estimate of $G(h)$. When the cohort holds both values of `t1_status`, the smaller of the two estimates. It is `0` after the last follow-up of a group when that follow-up ends in censoring.

`supported` `TRUE` when the horizon satisfies all three conditions, else `FALSE`.

References

Blanche P, Dartigues J, Jacqmin-Gadda H (2013). "Estimating and comparing time-dependent areas under receiver operating characteristic curves for censored event times with competing risks." *Statistics in Medicine*, **32**(30), 5381–5397. doi:10.1002/sim.5958.

Gerds TA, Schumacher M (2006). "Consistent estimation of the expected Brier score in general survival models with right-censored event times." *Biometrical Journal*, **48**(6), 1029–1040. doi:10.1002/bimj.200610301.

Riley RD, Collins GS, Ensor J, Archer L, Booth S, Mozumder SI, Rutherford MJ, van Smeden M, Lambert PC, Snell KIE (2022). "Minimum sample size calculations for external validation of a clinical prediction model with a time-to-event outcome." *Statistics in Medicine*, **41**(7), 1280–1295. doi:10.1002/sim.9275.

Vergouwe Y, Steyerberg EW, Eijkemans MJC, Habbema JDF (2005). "Substantial effective sample sizes were required for external validation studies of predictive logistic regression models." *Journal of Clinical Epidemiology*, **58**(5), 475–483. doi:10.1016/j.jclinepi.2004.06.017.

See Also

`cumulative_risk()` for a guard on time since the first diagnosis.

Examples

```
#------------------------------------------------------------------------
# horizon_support() example
#------------------------------------------------------------------------
library(prsladder)

# Simulate a cohort.
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(progression = "progression_score")
)

# Horizons of 5 to 20 years pass. At 21.5 years, too few persons remain in
# follow-up.
horizon_support(cohort, horizons = c(5, 10, 20, 21.5))
```

---

plot_calibration *Plot Observed Against Predicted Risk*

Description

`plot_calibration()` draws the calibration of `assess_calibration()` at one horizon. The main display is the smooth calibration curve, with the observed risk on the y-axis and the predicted risk on the x-axis. The groups of predicted risk are a second display, as points. A curve near the diagonal shows a model whose risks are right. The figure is a `ggplot2` figure, which you can change with the functions of the `ggplot2` package.

Usage

```
plot_calibration(calibration, horizon, model = NULL)
```

Arguments

calibration	(list) The output of assess_calibration().
horizon	(numeric(1)) One horizon of the output.
model	(character() or NULL; default NULL) Distinct names of one or more models of the output. If NULL, the figure draws every model, in the order of the output.

Details

The curve is the element curve of calibration. The function draws the curve over the points. The points are the columns expected_risk and observed_risk of the groups element. The vertical bars are pointwise 95% intervals of the observed risk. The intervals are on the log scale of the risk, as in the section "Intervals" of cumulative_risk(). A group with no event has an observed risk of 0 and the interval $[0, 0]$. The figure draws this point. A group with nobody at risk at the horizon has no observed risk, and the caption names it. A model without a curve, for example with competing events, has only the points, and the caption names it.

Both axes start at 0 and end at the same value. The dashed diagonal marks equal observed and predicted risks. Several models give one panel each, with the same axes, in the order of model. The subtitle states whether the evaluation is apparent, optimism-corrected, or external.

In the optimism-corrected mode, the figure draws the corrected curve as a solid line and the apparent curve as a dotted line. The points are the apparent groups, and the caption states this.

For an external or a corrected evaluation, the strip of each panel gives the observed-to-expected (O/E) ratio and the calibration slope. The strip gives the 95% interval of the O/E ratio only when both limits are finite. It gives the standard error of the slope only when that value is finite. With an analytic plan, the corrected O/E ratio and slope have no interval. The O/E ratio compares event counts, not risks. With competing events, the O/E ratio and the slope measure only the cause-specific model of T2, as in assess_calibration(). The points compare the cumulative incidence of T2. For an apparent evaluation, the strip gives only the model name. On the development persons, the O/E ratio is close to 1, and assess_calibration() gives no slope (see its Details).

The time scale is time since entry, as in assess_calibration(). Under the recruitment index or with a positive entry_lag in prs_cohort(), this time scale differs from the time since T1 of plot_risk().

Value

A ggplot object.

See Also

assess_calibration() for the values, and plot_discrimination() for the AUC and Brier score.

Examples

```
#--------------------------------------------------------------------------
# plot_calibration() example
#--------------------------------------------------------------------------
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
ladder <- fit_ladder(cohort)
external <- prs_cohort(
  simulate_prs_data(n = 2000L, seed = 2L, id_prefix = "E"),
  reference = cohort
)
calibration <- assess_calibration(ladder, 10, newdata = external)
plot_calibration(calibration, 10, model = c("onset", "onset+progression"))
```

---

plot_discrimination *Plot the AUC or Brier Score of the Ladder*

Description

`plot_discrimination()` draws the time-dependent AUC or the Brier score of each model from `assess_discrimination()`, with its 95% interval. With `measure = "delta_auc"` or `"delta_brier"`, it draws the difference between the two models of each pair instead. The figure is a `ggplot2` figure, which you can change with the functions of the `ggplot2` package.

Usage

```
plot_discrimination(discrimination, measure = "auc")
```

Arguments

`discrimination`	(list) The output of `assess_discrimination()`.
`measure`	(`character(1)`; default `"auc"`) One of `"auc"`, `"brier"`, `"delta_auc"`, or `"delta_brier"`. Each value names a column of the `models` or `pairs` element of `discrimination`. `"auc"` and `"brier"` draw the values of the models, from the `models` element. `"delta_auc"` and `"delta_brier"` draw the differences of the pairs, from the `pairs` element.

Details

The figure has one panel per horizon, with one row per model or pair. The models are in the order of `ladder$models` from the top, and the pairs are in the order of `discrimination$pairs`. The subtitle states whether the evaluation is apparent, optimism-corrected, or external. A row gets a bar only when both limits of its interval are finite. With an analytic plan, a corrected difference has no interval, and the figure draws only its point.

An AUC of 0.5 is no discrimination. The figure has no line at 0.5, because the line would stretch the axis and hide the differences between the models.

A raw Brier score depends on the event rate at the horizon. Because of this, each panel of the Brier score has its own x-axis. Use the differences of `measure = "delta_brier"`, which compare models on the same persons, instead of the raw scores. A Brier difference below 0 favors the full model, and an AUC difference above 0 favors it.

For a difference, a dashed line marks 0. Consider a test of a difference in AUC between nested models on the development persons. This test is conservative when the added scores have no effect or a small effect (Demler et al. 2012; Pepe et al. 2013). An interval of the difference that includes 0 is then weak evidence of no gain. This problem does not occur when the added scores are associated with T2 (Demler et al. 2012). For an apparent evaluation of the AUC, the figure has a caption that states this. The two studies do not examine the Brier score, and its figure has no caption. `compare_ladder()` gives the likelihood-ratio test of the added scores. For an optimism-corrected difference, the caption refers to this test.

Value

A `ggplot` object.

References

Demler OV, Pencina MJ, D'Agostino Sr. RB (2012). "Misuse of DeLong test to compare AUCs for nested models." *Statistics in Medicine*, **31**(23), 2577–2587. doi:10.1002/sim.5328.

Pepe MS, Kerr KF, Longton G, Wang Z (2013). "Testing for improvement in prediction model performance." *Statistics in Medicine*, **32**(9), 1467–1482. doi:10.1002/sim.5727.

See Also

`assess_discrimination()` for the values, and `plot_calibration()` for the calibration.

Examples

```
#-------------------------------------------------------------------------
# plot_discrimination() example
#-------------------------------------------------------------------------
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
```

```
  scores = c(onset = "onset_score", progression = "progression_score")
)
ladder <- fit_ladder(cohort)
external <- prs_cohort(
  simulate_prs_data(n = 2000L, seed = 2L, id_prefix = "E"),
  reference = cohort
)
discrimination <- assess_discrimination(
  ladder,
  horizons = c(5, 10),
  pairs = list(c("onset+progression", "onset")),
  newdata = external
)
plot_discrimination(discrimination)
plot_discrimination(discrimination, measure = "delta_auc")
```

`plot_gradient` *Plot the Risk at One Time by Score Group*

Description

`plot_gradient()` shows the cumulative risk of each score group at one time, with its 95% interval. The figure shows how the risk rises over the groups of a score. It is the usual figure for many groups, such as deciles, where curves are hard to tell apart. The figure is a `ggplot2` figure, which you can change with the functions of the `ggplot2` package.

Usage

```
plot_gradient(x, time, event = "event")
```

Arguments

`x`	(`prs_cumulative_risk`) A result of `cumulative_risk()` with a score. The whole cohort has no gradient, and the function stops with an error.
`time`	(`numeric(1)`) One of the `times` of `x`.
`event`	(`character(1)`; default `"event"`) One of `"event"` or `"competing"`. The event whose risk the figure shows. `"competing"` needs a cohort that declares the `competing` status code.

Details

The points and intervals are the `risk`, `risk_lower`, and `risk_upper` columns of the `risks` element of `x` at `time`. With `event = "competing"`, they are the columns of the competing risk. The interval is the pointwise 95% interval on the log scale of the risk, as in the section "Intervals" of `cumulative_risk()`.

A group that does not support `time` has no point. The x-axis keeps its place, and the caption names the group. All three conditions of the section "Support of a group at a time" of `cumulative_risk()` apply. With the default `min_events = 1L`, the rule is that of the curves of `plot_risk()`, except for a group with no event by `time`.

Group 1 has the lowest standardized scores. With a score sign of -1 in `prs_cohort()`, group 1 has the highest raw scores, and the axis title states this.

For groups at the `cut_probs` of `cumulative_risk()`, the tick labels give the percent range of each group on the standardized score, and the axis title is `Group of <score>`. With a score sign of -1, each tick label also gives the range of the raw score. The caption then states that the percentiles refer to the score after its sign.

The y-axis title names the event `T2` or the `competing event`, or uses the `event_labels` of `prs_cohort()`.

Value

A `ggplot` object.

See Also

`plot_risk()` for the curves, and `describe_risk()` for a table.

Examples

```
#------------------------------------------------------------------------------
# plot_gradient() example
#------------------------------------------------------------------------------
dat <- simulate_prs_data(n = 4000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(progression = "progression_score")
)

# Risk at 10 years by decile of the score.
deciles <- cumulative_risk(cohort, "progression", times = 10, groups = 10L)
plot_gradient(deciles, time = 10)
```

`plot_ladder` *Plot the Hazard Ratios of the Scores Across the Ladder*

Description

plot_ladder() draws a forest plot of the hazard ratio of each score in each model of a ladder from fit_ladder(). The figure shows how the hazard ratio of a score changes when the model adds the other scores. The figure is a ggplot2 figure, which you can change with the functions of the ggplot2 package.

Usage

```
plot_ladder(ladder, event = "event")
```

Arguments

ladder	(prs_ladder) A ladder from fit_ladder().
event	(character(1); default "event") One of "event", "competing", or "both". The event whose hazard ratios the figure shows. "both" gives one row of panels per event. "competing" and "both" need a cohort that declares the competing status code.

Details

The figure has one column of panels per score and one row of panels per event. Each row of a panel is a model of the ladder, in the order of ladder$models from the top. A model without the score has no point in the column of that score. Each point is the hazard ratio per standard deviation of the score, with its Wald 95% interval, as in summary.prs_ladder(). Because prs_cohort() standardizes the scores, all panels share one log x-axis. A dashed line marks a hazard ratio of 1.

A change in the hazard ratio of a score from one model to another comes mainly from the correlation among the scores. A Cox hazard ratio also changes when the model adds a score that is not correlated with it, because the hazard ratio is not collapsible. This change is small at the risks of this application. A simulation had 20 samples of 200,000 persons and a risk of 15% at 10 years. A score had a hazard ratio of 1.25, and an independent score had 1.30. When the model added the second score, the estimate of the first changed by 0.0018 on average, with a range of 0.0002 to 0.0036.

With competing events, the strip of each event states that the hazard ratios are cause-specific. The hazard ratios of the competing event come from ladder$competing_coefficients. The strips name the events T2 and Competing event, or the event_labels of prs_cohort().

The figure has no p-values. compare_ladder() gives the likelihood-ratio tests of the added scores.

Value

A ggplot object.

See Also

summary.prs_ladder() for the hazard ratios as a table, and compare_ladder() for the tests.

Examples

```
#-------------------------------------------------------------------------
# plot_ladder() example
#-------------------------------------------------------------------------
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
plot_ladder(fit_ladder(cohort))
```

---

plot_net_benefit *Plot Decision Curves*

Description

`plot_net_benefit()` draws the decision curves of `assess_net_benefit()` at one horizon. Each curve gives the net benefit of a strategy against the risk threshold. The strategies are the models, treating all persons, and treating no person. A model is useful at a threshold when its curve is above both reference curves. The figure is a `ggplot2` figure, which you can change with the functions of the `ggplot2` package.

Usage

```
plot_net_benefit(
  net_benefit,
  horizon,
  model = NULL,
  standardized = FALSE,
  conf_int = TRUE
)
```

Arguments

`net_benefit`	(list) The output of `assess_net_benefit()`.
`horizon`	(`numeric(1)`) One horizon of the output.
`model`	(`character()` or `NULL`; default `NULL`) Distinct names of one or more models of the output. If `NULL`, the figure draws every model.

`standardized` (`logical(1)`; default `FALSE`)
Whether to draw the standardized net benefit, which is the net benefit divided by the prevalence.

`conf_int` (`logical(1)`; default `TRUE`)
Whether to draw the 95% bands.

Details

The curves are the columns `net_benefit` of the `models` and `treat_all` elements of `net_benefit` at `horizon`. With `standardized = TRUE`, they are the columns `standardized_net_benefit`. The curve of treating no person is 0 at every threshold. With a single threshold, the figure draws points in place of curves.

The y-axis shows the range from -0.1ρ to ρ, where ρ is the `prevalence` of `treat_all` at `horizon`. This range is the default of the `dcurves` package. Because the largest net benefit of a model is ρ, the range keeps the curves of the models readable at a low prevalence. With `standardized = TRUE`, the range is from -0.1 to 1. The function sets the range with `ggplot2::coord_cartesian()`, which limits the view and keeps every row and band. The curve of treating all persons falls below the range at high thresholds.

The bands are the 95% intervals of the models and of treating all persons from `assess_net_benefit()`. They hold at each threshold, not for the whole curve at once. The figure draws a band only for the rows with two finite limits. With an analytic plan, the band of a corrected net benefit is the apparent band, moved to the corrected estimate. The curve and the band of treating all persons do not depend on a model. In the optimism-corrected mode, they are those of the development persons.

The x-axis is the threshold of the risk by `horizon` since entry, as in `assess_net_benefit()`. The subtitle states whether the evaluation is apparent, optimism-corrected, or external.

Value

A `ggplot` object.

See Also

`assess_net_benefit()` for the values, and `plot_calibration()` to check that the predicted risks are right.

Examples

```
#-------------------------------------------------------------------------
# plot_net_benefit() example
#-------------------------------------------------------------------------
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
```

```
ladder <- fit_ladder(cohort)
external <- prs_cohort(
  simulate_prs_data(n = 2000L, seed = 2L, id_prefix = "E"),
  reference = cohort
)
net_benefit <- assess_net_benefit(
  ladder,
  horizons = 10,
  thresholds = seq(0.02, 0.5, by = 0.02),
  newdata = external
)

# Decision curves of all models, then of two models standardized.
plot_net_benefit(net_benefit, 10)
plot_net_benefit(
  net_benefit,
  10,
  model = c("onset", "onset+progression"),
  standardized = TRUE
)
```

`plot_risk` *Plot Cumulative Risk Curves by Score Group*

Description

`plot_risk()` draws the cumulative risk curves of `cumulative_risk()`, one curve per score group, with pointwise 95% bands. A risk table under the curves gives the number of persons at risk and the number of events at each time of `times`. The figure is a `ggplot2` figure, which you can change with the functions of the `ggplot2` package.

Usage

```
plot_risk(x, event = "event", conf_int = TRUE, risk_table = NULL)
```

Arguments

`x`	(`prs_cumulative_risk`) A result of `cumulative_risk()`.
`event`	(`character(1)`; default `"event"`) One of `"event"`, `"competing"`, or `"both"`. The event whose risk the figure shows. `"both"` draws one panel per event. The two panels share one y-axis, which puts the two risks on the same scale. `"competing"` and `"both"` need a cohort that declares the `competing` status code.
`conf_int`	(`logical(1)`; default `TRUE`) Whether to draw the 95% bands.

`risk_table` (`character()` or `NULL`; default `NULL`)
The rows of the risk table, a subset of `"n_at_risk"`, `"n_events"`, `"n_competing"`, and `"n_censored"`. Each statistic has one row per group. `"n_competing"` needs a cohort that declares the `competing` status code. If `NULL`, the rows are the number at risk and the number of events of each event that the figure shows. See the section "Risk table" in Details. Use `character(0)` for no risk table.

Details

Curves and bands:

Each curve is a step function from the origin, the `entry_lag` of the cohort, with risk 0. The curves go up, as Pocock et al. (2002) advise. The function draws the rows of the `curves` element of `x` that have `supported = TRUE`. The section "Support of a group at a time" of `cumulative_risk()` gives the rule. A curve stops at the last time of `times`. After its last event row, a curve continues flat to the last time of `times` at which at least `min_at_risk` persons are at risk. The `times` of the call set how far the curves go, and the event times set their shape.

A group with no supported event row has no curve, and the function gives a warning that names the group. This case occurs when few persons are at risk at the first event times. An example is a cohort with few incident persons under the recruitment index of `prs_cohort()`. A positive `entry_lag` then starts the curves after the early risk sets.

The bands are pointwise 95% intervals on the log scale of the risk, as in the section "Intervals" of `cumulative_risk()`. A band holds at each time, not for the whole curve at once. For the Kaplan-Meier estimate, the bands differ from those of the `ggsurvfit` package, which uses the log scale of one minus the risk.

Risk table:

The risk table has one column at the origin and one column at each time of `times`. Its numbers are the columns of the `risks` element of `x`. The number at risk at the origin is the number of persons just after the origin. The event and censoring numbers are totals at or before each time. Because the table has one column per time, use 3 to 5 times when you want a risk table. The curves themselves do not depend on the number of times. The default rows follow the extended risk table of KMunicate, which most respondents of its survey preferred (Morris et al. 2019). They are the number at risk and the number of events of each event that the figure shows:

- `"event"` gives the rows `"n_at_risk"` and `"n_events"`.
- `"competing"` gives the rows `"n_at_risk"` and `"n_competing"`.
- `"both"` gives the rows `"n_at_risk"`, `"n_events"`, and `"n_competing"`.

The default does not have the row of censored persons. The rows of each statistic list the highest group at the top, as in the legend. With `event = "both"`, the table repeats under each panel, because the counts do not depend on the panel.

Under the recruitment index, the number at risk can increase over time, because prevalent persons enter after T1. The figure then has a caption that states this.

For groups at the `cut_probs` of `cumulative_risk()`, the legend gives the percent range of each group on the standardized score. With a score sign of -1, each label also gives the range of the raw score, and the caption states that the percentiles refer to the score after its sign.

Labels, colors, and changes:

The panels, the y-axis title, and the rows of event counts name the events `T2` and `Competing event`. The argument `event_labels` of `prs_cohort()` gives other names, such as `CAD` and `Death`.

The curves use the default colors of ggplot2, one per group, and the legend lists the highest group at the top. With more than 5 groups, the curves are hard to tell apart in any palette. Use plot_gradient() instead.

Without a risk table, the function returns a ggplot object. Add a theme, scale, or labels to it with +. With a risk table, the function returns a patchwork object. Use & to add a scale or theme elements to every part of it. Use + to change only the risk table. A complete theme, such as ggplot2::theme_bw(), added with & also replaces the theme of the risk table.

Value

A ggplot object when risk_table is character(0), else a patchwork object with the curves above the risk table.

References

Morris TP, Jarvis CI, Cragg W, Phillips PPJ, Choodari-Oskooei B, Sydes MR (2019). "Proposals on Kaplan-Meier plots in medical research and a survey of stakeholder views: KMunicate." *BMJ Open*, **9**(9), e030215. doi:10.1136/bmjopen2019030215.

Pocock SJ, Clayton TC, Altman DG (2002). "Survival plots of time-to-event outcomes in clinical trials: good practice and pitfalls." *The Lancet*, **359**(9318), 1686–1689. doi:10.1016/S0140-6736(02)08594X.

See Also

plot_stacked() for the risks of both events in one stack, plot_gradient() for the risks at one time, and describe_risk() for a table.

Examples

```
#------------------------------------------------------------------------
# plot_risk() example
#------------------------------------------------------------------------
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(progression = "progression_score")
)
res <- cumulative_risk(cohort, "progression", times = c(5, 10, 15), groups = 3L)

# Curves with a risk table.
plot_risk(res)

# Curves only, with a new axis title.
plot_risk(res, risk_table = character(0)) +
  ggplot2::labs(x = "Years since diagnosis")
```

plot_stacked *Plot the Cumulative Incidence of Both Events as a Stack*

Description

plot_stacked() shows how the risk of each score group divides between the second event (T2) and the competing event. In each panel, the lower area is the cumulative incidence of T2. The area above it is the cumulative incidence of the competing event (Putter et al. 2007). The figure is a ggplot2 figure, which you can change with the functions of the ggplot2 package.

Usage

```
plot_stacked(x)
```

Arguments

x	(prs_cumulative_risk) A result of cumulative_risk() on a cohort that declares the competing status code.

Details

The figure has one panel per score group, with one y-axis for all panels. The panels show whether a high score changes the share of T2 among the events. With score = NULL in cumulative_risk(), the figure has one panel. For groups at the cut_probs of cumulative_risk(), the panel titles give the percent range of each group on the standardized score. With a score sign of -1, each title also gives the range of the raw score, and the caption states that the percentiles refer to the score after its sign. The legend names the events T2 and Competing event, or the event_labels of prs_cohort().

The Aalen-Johansen probabilities of the three states sum to 1. Because of this, the top of the stack is one minus the probability of being free of both events. The figure does not draw the area free of both events. The y-axis then shows the two event areas at full height.

The areas use the same rows of curves as the curves of plot_risk(). These rows are the supported rows up to the last time of times. After the last event row, an area continues flat to the last time at which at least min_at_risk persons are at risk. A group with no supported event row has no area, and the function gives a warning that names the group. The figure has no bands and no risk table, because a band around a stacked area is hard to interpret. Use plot_risk() with event = "both" for the bands and the numbers at risk.

Value

A ggplot object.

References

Putter H, Fiocco M, Geskus RB (2007). "Tutorial in biostatistics: competing risks and multi-state models." *Statistics in Medicine*, **26**(11), 2389–2430. doi:10.1002/sim.2712.

See Also

plot_risk() for the curves with bands and a risk table.

Examples

```
#-------------------------------------------------------------------------
# plot_stacked() example
#-------------------------------------------------------------------------
dat <- simulate_prs_data(n = 2000L, seed = 1L, competing_risk = 0.2)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(progression = "progression_score"),
  status_codes = list(censored = 0, event = 1, competing = 2)
)

# The whole cohort, then the score groups.
plot_stacked(cumulative_risk(cohort, NULL, times = c(5, 10, 15)))
plot_stacked(
  cumulative_risk(cohort, "progression", times = c(5, 10, 15), groups = 3L)
)
```

---

predict_risk *Predict Absolute Risk at a Horizon*

Description

predict_risk() gives the absolute risk of the second event (T2) within a horizon after entry, from one Cox model of a ladder. The risk is conditional on the person being free of T2 at entry. With competing events, the risk is the cumulative incidence of T2. It is then conditional on the person being free of T2 and of the competing event at entry.

Use predict_risk() for the persons of a cohort, or for new persons that prediction_data() prepares. The assess_*() functions evaluate a model with these risks.

Usage

```
predict_risk(
  fit,
  newdata,
  horizon,
  event = c("event", "competing"),
  competing = NULL
)
```

Arguments

fit	(prs_fit or prs_risk_model) A model from the models element of a fit_ladder() result, or its competing model, attr(model, "competing"). Or a model from the models element of an export_ladder() result, or its competing model, model$competing. These fits have the class c("prs_fit", "coxph") and Efron ties, and the method of the section "Method" is exact for them. The design "index" of fit_ladder() has one baseline for each value of t1_status, on the time since entry. The function does not accept other survival::coxph() fits. Examples are a fit with Breslow ties, a fit with a strata() term, and the result of stats::update() on a model of a ladder. For these fits, use survival::survfit().
newdata	(data.frame) One row per person, with start and the predictor columns of fit. See the section "Input data" in Details.
horizon	(numeric(1); range (0, Inf)) Time since entry at which to evaluate the risk. The units are those of the time scale of fit. For a ladder, these units are the units of the age columns.
event	(character(1); default "event") One of "event" or "competing". The event whose risk the function gives: T2, or the competing event. "competing" needs a fit that holds a competing model, and competing must then be NULL or FALSE. See the section "Competing events" in Details.
competing	(prs_fit, prs_risk_model, FALSE, or NULL; default NULL) The model of the competing event. If NULL, the function uses the competing model that fit holds, and gives the net risk when fit holds none. If FALSE, the function gives the net risk. A model of a ladder, of class prs_fit, must have the persons and follow-up times of fit, with the competing event as its event. With a model of an export as fit, competing must be a model of the same export, of class prs_risk_model. See the section "Competing events" in Details.

Details

Method:

The time scale of the fit is the time since the first diagnosis (T1). The section "Design index" gives the changes for design = "index" in fit_ladder(). Let Λ_0 be the baseline cumulative hazard of the fit, and $S_0 = \exp(-\Lambda_0)$ the baseline survival. For person i, let η_i be the linear predictor and s_i the entry time, start. The expected number of events from s_i to $s_i + h$ is

$$E_i(h) = [\Lambda_0(s_i + h) - \Lambda_0(s_i)] \exp(\eta_i).$$

The risk within the horizon h after entry is

$$r_i(h) = 1 - \exp\{-E_i(h)\} = 1 - [S_0(s_i + h)/S_0(s_i)]^{\exp(\eta_i)}.$$

When $s_i = 0$, the risk is $1 - S_0(h)^{\exp(\eta_i)}$.

The function computes Λ_0 from the risk-set sums of the fit. It uses the Efron estimator, which is the ties method of every model of a ladder. The function centers Λ_0 and η_i at the covariate means of the fit, as predict(fit, type = "lp", reference = "sample") does. Each spline

term uses the knots of the fit. The risk then equals `-expm1(-predict(fit, newdata, type = "expected"))` with `stop = start + horizon`, to rounding error. The function computes $r_i(h)$ with `base::expm1()`, which keeps precision for small risks. This risk is the net risk, which is the risk if no competing event occurs. With competing events declared as censoring, this interpretation needs the independence assumption in the section "Status codes and competing events" of `prs_cohort()`.

Competing events:

`competing` selects the model of the competing event:

- `NULL` uses the attribute `"competing"` of `fit`. `fit_ladder()` sets this attribute when the cohort declares the `competing` status code (see `prs_cohort()`). A fit without the attribute gives the net risk.
- `FALSE` gives the net risk.
- A model of a ladder, of class `prs_fit`, gives the cumulative incidence with that model.

A model of an export from `export_ladder()` holds its competing model as its element `competing`, and `NULL` uses it. With a model of an export as `fit`, a `competing` model must come from the same export.

With a competing model, the risk is the cumulative incidence of T2, which is the probability of T2 before the competing event (Benichou and Gail 1990; Putter et al. 2007). Let Λ_1 and Λ_2 be the cumulative hazards of T2 and of the competing event for person i. Each is the baseline cumulative hazard of its fit times $\exp(\eta)$. Let $d\Lambda_k(u)$ be the jump of Λ_k at time u. The risk is the sum over the jump times u of Λ_1 with $s_i < u \le s_i + h$:

$$r_i(h) = \sum_u e^{-[\Lambda_1(u-)-\Lambda_1(s_i)]-[\Lambda_2(u-)-\Lambda_2(s_i)]} \frac{d\Lambda_1(u)}{d\Lambda_1(u)+d\Lambda_2(u)} \left(1 - e^{-[d\Lambda_1(u)+d\Lambda_2(u)]}\right).$$

The first factor is the probability to be free of both events just before u. The other factors are the probability of T2 at u. This formula has the form of equation 2.3 of Benichou and Gail (1990), with the jumps of the Cox cumulative hazards in place of the piecewise-constant hazards. The risk equals the cumulative incidence that `survival::survfit()` gives for a multi-state `survival::coxph()` fit of both events with the same terms. With tied event times and Efron ties, the two differ slightly, because `survival::survfit()` does not use the Efron estimator of the baseline hazard for a multi-state fit. When the competing model has no jump in the window, the risk equals the net risk. Otherwise, the risk is at most the net risk.

`event = "competing"` gives the cumulative incidence of the competing event. The function then uses the competing model that `fit` holds as the model of the event, and `fit` as its competing model. The same result comes from giving the competing model as `fit` and the model of T2 as `competing`. Report the cause-specific hazard ratios and the cumulative incidence of both events (Latouche et al. 2013).

The function computes Λ_1 and Λ_2 with the Efron estimator and the centering of the section "Method". The time to compute the risk increases with the number of persons times the number of T2 event times in the window. For 16,500 persons and 4,200 event times, one call takes about 2.5 seconds. The net risk of the same persons takes less than 0.1 seconds. Each `assess_*()` function computes the risks once for each model and each horizon.

Input data:

`newdata` needs the column `start` and each variable in the formula of `fit`, such as `age_t1`, the scores, and the covariates. With a competing model, `newdata` also needs each variable in its

formula. Each score must be on the standardized scale of the development cohort. The function ignores all other columns. A tibble or a data.table gives the same result as a data frame.

- For the persons of a cohort, use the `data` element of a `prs_cohort()` result. For the persons of another cohort with outcome data, first call `prs_cohort(data, reference = cohort)`, or `prs_cohort(data, reference = export)` for an export.
- For persons without outcome data, `prediction_data()` builds `newdata` from raw data with the rules of the development cohort.

With `design = "t1_entry"` or `"index"` in `fit_ladder()`, `newdata` also needs `t1_status` and `t1_duration`. These columns describe the person at entry, and `start` gives the time of the prediction:

- `start` is the time since T1 at which the prediction starts.
- `t1_duration` is the time from T1 to entry.
- `t1_status` is `"incident"` or `"prevalent"`.

Every person of a cohort has `start` equal to `t1_duration`. The two columns differ only for a prediction after entry. You can treat a patient as an incident person followed to time s since T1. This person has `t1_status` equal to `"incident"`, `t1_duration` equal to the `entry_lag` of the cohort, and `start` equal to s. `prediction_data()` with `t1_status = "incident"` gives these columns. This rule is the same for both designs. The section "Persons seen after T1" of `prediction_data()` gives the reason for this choice.

Missing risks:

The fit estimates the baseline hazard only within the follow-up of the development persons. At the ends of follow-up, few persons are at risk, and the estimate rests on the values of a few persons. Because of this, the function uses two follow-up limits of the fit:

- The lower limit is midway between the 10th and the 11th smallest entry time, `start`, of the development persons. When more than 10 persons share the smallest entry, such as `0` or the `entry_lag` of `prs_cohort()`, the lower limit is that entry.
- The upper limit is midway between the 10th and the 11th largest exit time, `stop`, of the development persons.

At least 10 development persons then enter at or before the lower limit, and at least 10 exit at or after the upper limit. A limit is not the value of one person, but it can be the mean of two values. A fit with 10 persons or fewer has no limits, and the function stops. A model of an export has the limits of the section "Follow-up limits" of `export_ladder()`. With the default `min_cell_count`, these limits are those of its ladder.

The function returns `NA` for a person in these cases:

- `start` is before the lower limit. Under the diagnosis index, the lower limit is the `entry_lag` of the cohort.
- `start + horizon` is after the upper limit. `survival::predict.coxph()` would carry the last value forward. Under the recruitment index, with the designs `"t1"` and `"t1_entry"`, these persons are prevalent persons with a long time from T1 to entry. A `horizon` after the upper limit gives `NA` for every person.
- `start` is before `t1_duration`, with `design = "t1_entry"` or `"index"` in `fit_ladder()`. The terms of these designs describe the person at entry, and a prediction cannot start before entry.
- A predictor is `NA` or `NaN`. The risk is then `NA` or `NaN`. With a competing model, this rule applies to the predictors of both models.

Both limits include equality. A person with `start` equal to the lower limit gets a risk. A person with `start + horizon` equal to the upper limit also gets a risk. Use `horizon_support()` to select a horizon with enough follow-up.

Without a competing model, a linear predictor of `-Inf` gives a risk of `0`. A linear predictor of `Inf` gives a risk of `1` when the window holds an event time of the fit, and `NaN` when it does not. The sign of the coefficient sets the sign of the linear predictor. With a competing model, the cumulative incidence has no general limit, and an infinite linear predictor stops the function. Because `prs_cohort()` and `prediction_data()` do not accept infinite values, only a `newdata` made in another way can cause this error.

Design index:

With `design = "index"` in `fit_ladder()`, the time scale of the fit is the time since entry, and each value of `t1_status` has its own baselines. The sections "Method", "Competing events", and "Missing risks" apply with these changes:

- s_i is `start - t1_duration`, the time since entry at the prediction. For a person of a cohort, this value is 0.
- Λ_0 is the baseline of the value of `t1_status` of the person.
- Each value of `t1_status` has its own follow-up limits, from the times since entry of its persons. The lower limit is 0, so a `start` before `t1_duration` gives `NA`.
- The risk equals the value from `stats::predict()` only when `start` equals `t1_duration`.

Every person of a cohort has the same time since entry at the prediction. Because of this, a `horizon` after the upper limit of a value of `t1_status` gives `NA` for all persons of a cohort with that value. With one end of follow-up, incident persons enter after recruitment, and they reach their limit first. The `assess_*()` functions then evaluate only the other persons, and they give a warning.

Models of an export:

A model of an export has the same terms and coefficients as the model of its ladder, and the follow-up limits of the section "Missing risks". Each step of its baselines holds at least `min_cell_count` events, and the cumulative hazard is linear between two knots (see `export_ladder()`). The risk uses the formula of the section "Method" with these baselines. It then differs a little from the risk of the ladder. With a competing model, each cause has a constant hazard between two knots of the two models. Let Δ be the length of such an interval. Let λ_1 and λ_2 be the hazards of T2 and of the competing event of the person on the interval. The cumulative incidence then increases by

$$\frac{\lambda_1}{\lambda_1 + \lambda_2} S(t) \left(1 - e^{-(\lambda_1 + \lambda_2)\Delta}\right),$$

where $S(t)$ is the probability to be free of both events at the start t of the interval. An interval with $\lambda_1 = 0$ adds 0.

Errors:

The function stops with an error in these cases:

- `fit` is not a model of a ladder, of class `prs_fit`, or a model of an export, of class `prs_risk_model` (see the argument `fit`). For other Cox models, use `survival::survfit()`.
- `competing` is not `NULL`, `FALSE`, or a model of the same kind as `fit`.
- A `competing` model has another `design` of `fit_ladder()` than `fit`.

- A `competing` model of an export has other follow-up limits than `fit`.
- A `competing` fit has other persons or follow-up times than `fit`, or a person has an event in both fits. The function does not check the attribute that `fit_ladder()` sets.
- With a competing model, a linear predictor is infinite.
- The fit has 10 persons or fewer. Such a fit has no follow-up limits.
- `horizon` is not one finite positive number.
- `newdata` is not a data frame, or does not have a needed column.
- `start` is not numeric, or has a value that is `NA`, `NaN`, infinite, or negative.
- A factor predictor has a level that is not in the fit.

A `newdata` with zero rows and all needed columns gives `numeric(0)`.

Value

A numeric vector with one risk per row of `newdata`, in the same order and without names. Each risk is in `[0, 1]`, or is `NA` or `NaN` as the section "Missing risks" in Details describes. The risks are point estimates without standard errors. `standardized_risk()` gives bootstrap intervals for average risks.

References

Benichou J, Gail MH (1990). "Estimates of absolute cause-specific risk in cohort studies." *Biometrics*, **46**(3), 813–826. doi:10.2307/2532098.

Latouche A, Allignol A, Beyersmann J, Labopin M, Fine JP (2013). "A competing risks analysis should report results on all cause-specific hazards and cumulative incidence functions." *Journal of Clinical Epidemiology*, **66**(6), 648–653. doi:10.1016/j.jclinepi.2012.09.017.

Putter H, Fiocco M, Geskus RB (2007). "Tutorial in biostatistics: competing risks and multi-state models." *Statistics in Medicine*, **26**(11), 2389–2430. doi:10.1002/sim.2712.

See Also

- `prediction_data()` to build `newdata` for persons without outcome data.
- `horizon_support()` to check whether a cohort supports a horizon.
- `assess_calibration()`, `assess_discrimination()`, and `assess_thresholds()` to evaluate these risks.
- `cumulative_risk()` for the observed risk by score group, without a model.

Examples

```
#-------------------------------------------------------------------------
# predict_risk() example
#-------------------------------------------------------------------------
library(prsladder)

# Fit a ladder on a simulated cohort under the recruitment index.
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
```

```
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  covariates = "sex",
  index = "recruitment",
  age_at_recruitment = "age_at_recruitment"
)
ladder <- fit_ladder(cohort)
full <- ladder$models[["onset+progression"]]

# Five-year risk from entry for the first persons of the cohort.
predict_risk(full, cohort$data[1:5, ], horizon = 5)

# Persons without outcome data, with raw scores and the column names of the
# development data. They are seen 1, 2.5, and 6 years after the first
# diagnosis.
persons <- data.frame(
  age_at_diagnosis = c(48, 55, 61),
  age_now = c(49, 57.5, 67),
  onset_score = c(-1, 0, 2),
  progression_score = c(0.5, 1, -0.3),
  sex = c("Male", "Female", "Female")
)
newdata <- prediction_data(persons, cohort, age_at_entry = "age_now")
predict_risk(full, newdata, horizon = 5)

# With deaths as competing events, each model holds its competing model.
# The cumulative incidence is less than the net risk.
dat_death <- simulate_prs_data(n = 2000L, seed = 1L, competing_risk = 0.2)
cohort_death <- prs_cohort(
  dat_death,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  covariates = "sex",
  status_codes = list(censored = 0, event = 1, competing = 2)
)
full_death <- fit_ladder(cohort_death)$models[["onset+progression"]]
predict_risk(full_death, cohort_death$data[1:5, ], horizon = 5)
predict_risk(
  full_death,
  cohort_death$data[1:5, ],
  horizon = 5,
  competing = FALSE
)

# The cumulative incidence of death before T2.
predict_risk(
```

```
  full_death,
  cohort_death$data[1:5, ],
  horizon = 5,
  event = "competing"
)
```

`prediction_data` *Build Prediction Data for Persons Without Outcome Data*

Description

`prediction_data()` prepares the `newdata` argument of `predict_risk()` for persons who have no outcome data, such as a newly diagnosed patient. Because these persons have no status and no age at exit, `prs_cohort()` cannot take them.

The function applies the rules of a development cohort to the raw data:

- It standardizes each score with the mean, standard deviation, and sign of the cohort.
- It gives each factor covariate the factor levels of the cohort.
- It sets `start`, the entry time since the first diagnosis (T1).
- Under the recruitment index, it sets `t1_status` and `t1_duration`.

Usage

```
prediction_data(data, cohort, age_at_entry = NULL, t1_status = "prevalent")
```

Arguments

`data`	(data.frame) One row per person, with the age at T1, score, and covariate columns of the development data. A tibble or a data.table gives the same result as a data frame.
`cohort`	(`prs_cohort` or `prs_export`) The development cohort of the model, such as `ladder$cohort`, or the export of the ladder from `export_ladder()`. The export holds the rules of the development cohort, and the result is the same for both.
`age_at_entry`	(`character(1)` or `NULL`; default `NULL`) Name of the numeric column with the age at which the prediction starts. If `NULL`, the prediction starts at T1.
`t1_status`	(`character(1)`; default `"prevalent"`) One of `"prevalent"` or `"incident"`. The status of a person whose age at entry is at or after the age at T1, under the recruitment index. `"prevalent"` follows the rule of `prs_cohort()`. `"incident"` gives every person the status `"incident"` and a `t1_duration` equal to the `entry_lag` of `cohort`. `"incident"` needs a cohort under the recruitment index and a column name in `age_at_entry`. Without one of them, the function stops with an error. See the section "Persons seen after T1" in Details.

Details

Input columns:

`data` must have these columns of the development data, under the same names:

- The age at T1, `cohort$columns[["age_at_diagnosis"]]`.
- Each score column in `cohort$scores$column`.
- Each covariate in `cohort$covariates`.

For an export from `export_ladder()`, these elements are in `cohort$rules`, such as `cohort$rules$scores$column`.

The function ignores all other columns. Because the result has no identifier column, match its rows to the rows of `data` by their order.

Entry time:

`start` is the larger of the entry time and the `entry_lag` of `cohort`, as in `prs_cohort()`. The entry time depends on `age_at_entry`:

- If `age_at_entry` is `NULL`, the entry time is zero.
- Otherwise, the entry time is the age at entry minus the age at T1. A negative difference becomes zero, which means entry at T1.

`predict_risk()` then gives the risk of the second event (T2) within the horizon after `start`, for a person who is free of T2 at `start`. Give `age_at_entry` for a person seen some time after T1, for example the current age of a patient diagnosed three years ago. This rule is the same under both indexes. The index of `cohort` changes only which development persons entered the model.

Under the recruitment index, the result also has the columns `t1_status` and `t1_duration`, which the models of `design = "t1_entry"` and `design = "index"` in `fit_ladder()` use. With the default of the argument `t1_status`, the column `t1_duration` equals `start`. The column `t1_status` then depends on `age_at_entry`:

- If `age_at_entry` is `NULL`, every person is `"incident"`.
- Otherwise, a person is `"prevalent"` when the age at entry is at or after the age at T1, as in `prs_cohort()`. Every other person is `"incident"`.

With `t1_status = "incident"`, every person is `"incident"`, and `t1_duration` is the `entry_lag` of `cohort`. The column `start` does not change.

Persons seen after T1:

The designs `"t1_entry"` and `"index"` give a prevalent person the effect of `t1_status`. This effect comes from the biobank volunteers who joined the development cohort after T1. It has the same size for any age at entry after the age at T1, also for a very short time after T1. A patient seen after T1 can differ from these volunteers. This difference can occur in the health system of the biobank, without a transport to another setting. The model cannot show which persons the patient resembles. The analyst owns this choice.

The argument `t1_status` sets the choice for all persons of `data`:

- `"prevalent"`, the default, treats a person seen after T1 as a volunteer who joined the cohort at the age at entry.
- `"incident"` treats each person as an incident person whom the cohort followed from T1 to `start`.

With "incident", predict_risk() gives the conditional risk of an incident person. This risk is the risk within the horizon after start, for a person who has no event before start. The two values give different risks under the designs "t1_entry" and "index". They give the same risk under the design "t1", which has no term for the entry. The section "Input data" of predict_risk() describes the columns that the two designs use.

Without age_at_entry, a person diagnosed long before the recruitment of the cohort also gets a less precise risk (see the section "Designs" of fit_ladder()).

The tie rule has a related effect. A person with age_at_entry equal to the age at T1 is "prevalent", as in prs_cohort(). Without age_at_entry, the same person is "incident". Both persons have the same start, and the two designs give them different risks. The design "t1" gives them the same risk. The package keeps this rule, because both paths then give identical predictor columns (see the section "Persons with outcome data"). With t1_status = "incident", this person has the same row with and without age_at_entry.

Missing and out-of-range values:

The function stops with an error in these cases:

- A named column is not in data.
- An age is not numeric, or is NA, NaN, or infinite. The function needs both ages to set start.
- A score column is not numeric, or has an infinite value. The function accepts a logical column when all its values are NA, as persons$onset_score <- NA gives.
- A covariate does not have the type it has in cohort, or has an infinite value. A factor covariate of cohort accepts factor, character, or logical input.
- A covariate has a value that is not a factor level of cohort.

A missing score or covariate stays missing in the result. predict_risk() then returns NA for that person, or NaN when the missing value is NaN. Because of this rule, a score that the model does not use can be NA. For example, set the other score columns to NA for a model with one score.

The function does not check whether an age or a score is outside the range of the development persons. In that case, the model extrapolates, and the uncertainty of the risk increases. predict_risk() returns NA when start is before the lower follow-up limit of the fit, or start plus the horizon is after its upper limit (see the section "Missing risks" of predict_risk()).

Persons with outcome data:

The result is not a prs_cohort. Because it has no outcomes, the assess_*() functions cannot use it. For other persons with outcome data, use prs_cohort(data, reference = cohort). On the same persons and the same entry, both paths give identical predictor columns with the default t1_status.

Value

A data frame with one row per row of data, in the same order, and row names 1 to nrow(data). The columns are the predictor columns of the data of the development cohort, in the same order:

start The entry time since T1.

age_t1 The age at T1. Every model of the ladder uses this age in its age term.

t1_status "incident" or "prevalent". Only the recruitment index has this column.

t1_duration The double value of start. With t1_status = "incident", the double value of the entry_lag of cohort. Only the recruitment index has this column.

One column per score The standardized score, named by score name.

One column per covariate The covariate, with the factor levels of the development cohort for a factor.

A zero-row `data` gives a zero-row result with the same columns.

See Also

- `predict_risk()` for the risk.
- `prs_cohort()` with `reference` for other persons with outcome data.

Examples

```
#-------------------------------------------------------------------------
# prediction_data() example
#-------------------------------------------------------------------------
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  covariates = "sex"
)
ladder <- fit_ladder(cohort)
full <- ladder$models[["onset+progression"]]

# Two persons diagnosed today, with the column names of the development
# data and raw scores.
persons <- data.frame(
  age_at_diagnosis = c(48, 61),
  onset_score = c(-1, 2),
  progression_score = c(0.5, -0.3),
  sex = c("Male", "Female")
)
newdata <- prediction_data(persons, cohort)
newdata
predict_risk(full, newdata, horizon = 5)

# The same persons three years later, still free of the second event.
persons$age_now <- persons$age_at_diagnosis + 3
later <- prediction_data(persons, cohort, age_at_entry = "age_now")
later$start
predict_risk(full, later, horizon = 5)

#-------------------------------------------------------------------------
# Persons seen after T1, under the recruitment index
#-------------------------------------------------------------------------
recruited <- prs_cohort(
  dat,
```

```
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  covariates = "sex",
  index = "recruitment",
  age_at_recruitment = "age_at_recruitment"
)
columns <- c("start", "t1_status", "t1_duration")

# By default, the persons are prevalent, as volunteers who joined the cohort
# three years after T1.
prediction_data(persons, recruited, age_at_entry = "age_now")[columns]

# The same persons as incident persons, followed from T1 for three years.
prediction_data(
  persons,
  recruited,
  age_at_entry = "age_now",
  t1_status = "incident"
)[columns]
```

`print.prs_cohort` *Print a Cohort*

Description

`print()` shows a short description of a cohort from `prs_cohort()`:

- The index, and the number of persons and events. With the `competing` role, also the number of competing events.
- The `prevalent` setting, under the recruitment index only.
- The entry lag, when it is positive.
- The scores, with their columns and signs.
- The covariates, or `none`.
- The exclusion counts that are not zero.

Usage

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

Arguments

`x`	(`prs_cohort`) A cohort from `prs_cohort()`.
`...`	Not used.

Value

x, invisibly.

`print.prs_export` *Print an Export*

Description

`print()` lists every element of an export from `export_ladder()`. The disclosure review of the biobank can then check each value:

- A header line with the number of models, the index, and the argument `design` of `fit_ladder()`. With competing events, the line ends with `with competing events`.
- The settings: `min_cell_count`, the version of the package, and the date.
- The rules: the columns, the status codes, and the event labels. Then the entry lag, the `prevalent` setting, the score table, and the type and levels of each covariate.
- The table `development`, without its list column `scores`.
- Each model, with its terms, coefficients, number of knots, and follow-up limits, and the same values of its competing model.

The print gives the number of knots of each baseline, not the knots. The knots and the cumulative hazards are in the element `baseline` of each model, such as `x$models[["base"]]$baseline`.

Usage

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

Arguments

`x`	(`prs_export`) An export from `export_ladder()`.
`...`	Not used.

Value

x, invisibly.

Examples

```
#-------------------------------------------------------------------------
# print.prs_export() example
#-------------------------------------------------------------------------
library(prsladder)

dat <- simulate_prs_data(n = 1000L, seed = 1L)
cohort <- prs_cohort(
```

```
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(progression = "progression_score")
)
export <- export_ladder(fit_ladder(cohort))
print(export)

# The knots and the cumulative hazards of the Efron baseline of one model.
head(as.data.frame(export$models[["progression"]]$baseline))
```

`print.prs_ladder` *Print or Summarize a Ladder*

Description

`print()` shows a header line and the model table of a ladder. The header gives the number of models, the number of scores, the index of the cohort, and the argument `design` of `fit_ladder()`. With competing events, the header ends with `with competing events`. The table omits the list column `scores`.

`summary()` returns the rows of `coefficients` for the scores. These rows give the hazard ratio per standard deviation of each score in each model.

Usage

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

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

Arguments

`x, object`	(`prs_ladder`) A ladder from `fit_ladder()`.
`...`	Not used.

Value

`print()` returns `x` invisibly.

`summary()` returns a data frame with the columns of `object$coefficients`. The data frame has one row for each score in each model. The `base` model has no rows.

Examples

```
#-------------------------------------------------------------------------
# summary.prs_ladder() example
#-------------------------------------------------------------------------
dat <- simulate_prs_data(n = 1000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
ladder <- fit_ladder(cohort)
print(ladder)
summary(ladder)
```

`prs_cohort` *Build a Cohort for a Ladder of Polygenic Risk Score Models*

Description

`prs_cohort()` prepares a data frame of persons with a first diagnosis (T1) for the survival analysis of a second event (T2). The function checks the columns and removes the persons who cannot enter the analysis. It then computes the follow-up interval of each person and standardizes each polygenic risk score.

Use the result as the input to `fit_ladder()`, `cumulative_risk()`, `describe_cohort()`, and `prediction_data()`. To prepare the persons of another cohort with the same rules, give the result, or its export from `export_ladder()`, as `reference` in a second call.

Usage

```
prs_cohort(
  data,
  id,
  status,
  age_at_diagnosis,
  age_at_exit,
  scores,
  covariates = NULL,
  status_codes = list(censored = 0, event = 1),
  event_labels = NULL,
  signs = NULL,
  index = c("diagnosis", "recruitment"),
  age_at_recruitment = NULL,
  entry_lag = 0,
```

```
  prevalent = c("keep", "drop"),
  reference = NULL
)
```

Arguments

data	(data.frame) One row per person with a first diagnosis. A tibble or a data.table gives the same result as a data frame.
id	(character(1)) Name of the person identifier column. The column can have any type. Its values must be unique and not missing.
status	(character(1)) Name of the status column, with the codes of status_codes. For numeric codes, the column must be integer, double, or logical. For character codes, the column must be character or factor. The function removes a person with NA or NaN as missing.
age_at_diagnosis	(character(1)) Name of the numeric column with the age at T1.
age_at_exit	(character(1)) Name of the numeric column with the age at T2, at the competing event, or at censoring.
scores	(character()) Names of one or more numeric score columns, with no column repeated. The names of the vector become the score names, for example c(onset = "onset_score"). An unnamed element uses its column name as the score name. Each score name must obey these rules: • It is unique and syntactically valid. • It is not "base". • It is not one of the derived column names id, start, stop, status, competing, age_t1, t1_status, or t1_duration.
covariates	(character() or NULL; default NULL) Names of the covariate columns. Each column must be numeric, integer, logical, factor, or character. Covariate names must be unique, syntactically valid, and different from the score names and the derived column names. If NULL, the models have no covariates.
status_codes	(named list; default list(censored = 0, event = 1)) The codes of the status column for each role. The roles censored and event are required, and the role competing is optional. Each role holds one or more codes with no NA, and no code occurs in two roles. All codes are numeric, or all codes are character. The function does not convert a code. For example, "1" does not match 1. For example, list(censored = 0, event = 1, competing = 2) declares death as a competing event, and list(censored = c(0, 2), event = 1) censors it. See the section "Status codes and competing events" in Details.

event_labels (named character() or NULL; default NULL)
Names of the events in the figures and tables, named by role: event for T2 and competing for the competing event. Either name can be absent, and an absent label keeps its default name. Each label is one non-empty string, and the two labels differ. The competing label needs the competing role in status_codes. If NULL, the figures and tables use the default names. See the section "Event labels" in Details.

signs (named numeric() or NULL; default NULL)
A sign, 1 or -1, for some or all scores, named by score name. A score without an entry has sign 1. If NULL, every score has sign 1.

index (character(1); default "diagnosis")
One of "diagnosis" or "recruitment". The event that starts follow-up.

age_at_recruitment
(character(1) or NULL; default NULL)
Name of the numeric column with the age at recruitment into the biobank. index = "recruitment" requires this argument. index = "diagnosis" requires NULL. Use the age at recruitment, not the start of register follow-up. The start of register follow-up would make every person incident.

entry_lag (numeric(1); default 0; range [0, Inf))
Earliest entry time for every person, in time since T1. The value 0 adds no delay to the entry time that index sets.

prevalent (character(1); default "keep")
One of "keep" or "drop". The treatment of the prevalent persons. "drop" removes them, which gives an incident-only analysis. Only index = "recruitment" accepts the value "drop".

reference (prs_cohort, prs_export, or NULL; default NULL)
A development cohort from prs_cohort(), or an export from export_ladder(). If given, the function applies the rules of reference to data. If NULL, the function sets the rules from the other arguments.

Details

Time scale and entry:

The models of fit_ladder() use time since T1 as the time scale, with delayed entry, unless design = "index". A person enters the risk set at start and leaves it at stop. Both times are in the units of the age columns.

- stop is the age at exit minus the age at T1.
- start is the larger of the entry time and entry_lag.

The entry time depends on index:

"diagnosis" Zero for every person.

"recruitment" Zero for an incident person, who was recruited before T1. For a prevalent person, who was recruited at or after T1, the entry time is the age at recruitment minus the age at T1. A prevalent person is in the cohort only because they were free of T2 at recruitment. Delayed entry corrects for this selection.

This correction holds only when the entry time is quasi-independent of the time to the first of T2 and the competing event (Tsai 1990; Geskus 2011). Quasi-independence means that the two times are independent among the persons who enter. A selection that depends on the remaining time breaks this condition. An example is a biobank that persons join less often when their exit is near. The risks are then biased, and delayed entry does not correct them. `check_entry()` tests this condition.

Under the recruitment index, the incident and the prevalent persons enter at different times since T1. Because of this, their censoring on time since entry can differ. The censoring weights and the observed risks of the `assess_*()` functions use one censoring distribution for each group of `t1_status`. The median follow-up of all persons in `summary.prs_cohort()` uses the same groups. With `prevalent = "drop"`, every person is incident, and these functions use one censoring distribution for all persons.

Under the recruitment index, `t1_duration` is the time from T1 to entry. It equals `start` for every person of the cohort. The two columns have different roles. `start` is the clock variable of a fit, and `t1_duration` is a baseline covariate for the designs of `fit_ladder()`.

Under the recruitment index, keep the birth year in `covariates`. At the same time since T1, a prevalent person usually lived in an earlier calendar period than an incident person. The calendar year at time t since T1 is the birth year plus the age at T1 plus t. A log-linear calendar trend is then linear in the birth year and in the age at T1. The baseline hazard absorbs the part in t. A nonlinear trend is not additive in these terms, and the terms then only approximate it. Without the birth year, the test `entry_terms` of `check_design()` can reject because of the calendar trend. The test `entry_bands` rarely rejects. The rule of that function then does not correct this trend, and the risks can have a bias.

Compute every age with decimals from dates, including the age at recruitment. With ages in whole years, the order of two events in the same year is unknown. An age at T1 in completed years is at or before the true age. A T2 between the two ages then counts as a T2 after T1, and no check can find it.

`entry_lag` sets the earliest entry, for example to skip events recorded during the work-up of T1. A person whose entry time is before `entry_lag` enters at `entry_lag` instead. A prevalent person whose entry time is at or after `entry_lag` keeps it. Under the diagnosis index, a positive lag gives a landmark analysis. This analysis includes only the persons who are free of T2 and in follow-up after the lag.

For a person who enters at `entry_lag`, the horizon of `predict_risk()` starts at the lag. The risk is for a person who has no event before the lag. This risk and a risk from T1 in a cohort without the lag have different time windows. Do not compare them. To get the same window from the cohort without the lag, give `prediction_data()` a column with the age at T1 plus the lag as `age_at_entry`. Under the recruitment index, also set `t1_status = "incident"` for an incident person.

Status codes and competing events:

The status column holds one code per person. `status_codes` gives the role of each code:

`censored` Follow-up stopped at the age at exit, free of T2.

`event` T2 occurred at the age at exit.

`competing` An event that prevents T2, such as death, occurred at the age at exit.

The function derives two 0/1 columns from the roles. `status` is 1 for T2, and `competing` is 1 for a competing event. The column `competing` occurs only when `status_codes` has the `competing` role.

The declaration selects the risk that the later functions estimate:

- With the `competing` role, the risk of T2 is the cumulative incidence. This risk is the probability of T2 before a competing event. `fit_ladder()` fits a cause-specific Cox model for each event (Putter et al. 2007).
- When `status_codes` gives the competing codes the role `censored`, the risk of T2 is the net risk. This risk is the probability of T2 if the competing event did not occur. The estimate is this risk only if the competing event is independent of T2, given the terms of the model. Without further assumptions, the data cannot test this independence (Tsiatis 1975; Andersen et al. 2012).

The two declarations give the same Cox models for T2. They give different absolute risks. When a role has two or more codes, such as `competing = c(2, 3)`, the function merges these codes into one event with one hazard model.

Event labels:

The figures and tables name the events `T2` and `Competing event` by default. `event_labels` gives other names, such as `c(event = "CAD", competing = "Death")`. The labels change only the text of `plot_risk()`, `plot_stacked()`, `plot_gradient()`, `plot_ladder()`, `describe_risk()`, and `describe_cohort()`. They do not change the analysis. A label replaces the default name both where it stands alone, as in a panel title, and inside a phrase, as in `Cumulative incidence of CAD` or `CAD events`. Choose the case that reads well in both places. The `print()` and `summary()` methods keep the default names. A cohort built with `reference` keeps the labels of `reference`.

Exclusions:

The function removes persons for the reasons below, in this order. Each removed person counts once, under the first reason that applies. `n_excluded` stores the count for each reason.

`missing` `NA` or `NaN` in the status, the age at T1, the age at exit, a score, or a covariate.

`prior_event` T2 at or before T1, which includes T2 at the age at T1. The person is not free of T2 at T1, so the person cannot be at risk of T2 after T1. Both indexes.

`no_recruitment_age` `NA` in the age at recruitment. Recruitment index only.

`historical` A prevalent person with T2 after T1 and at or before recruitment. Recruitment index only. A competing event at or before recruitment gives `stop` at or below `start`, and counts under `nonpositive_followup`.

`prevalent_dropped` A prevalent person when `prevalent = "drop"`. Recruitment index only.

`nonpositive_followup` `stop` at or below `start`. This reason also covers an interval so short that `survival::aeqSurv()` reduces its length to zero, which makes `survival::coxph()` stop with an error.

The order of the first five reasons does not change the set of persons that remain. The order changes only the reason recorded for a removed person.

The function gives a warning with the counts in two cases:

- Tied ages: T2 at the age at T1, or exit at entry. The function removes these persons as `prior_event`, `historical`, or `nonpositive_followup`. Ties often come from ages in whole years, and the order of the two events is then unknown. For a sensitivity analysis, move each tied exit a short time after the age that it ties with, and compare the two results.
- Exit before T1: censoring or a competing event before T1. This order is not possible. The data have an error. The function removes these persons under the first reason that applies.

Errors:

The function stops with an error in these cases:

- A named column is not in `data`.
- A column has two roles, for example a score that is also a covariate.
- The identifier has a missing or repeated value.
- `status_codes` breaks a rule of the argument.
- The status column has a type that does not match the codes.
- The status column has a value that is not missing and has no role in `status_codes`.
- An age, score, or covariate column has the wrong type or an infinite value.
- No person remains after the exclusions, for example when `data` has zero rows.
- A score has zero standard deviation in the remaining persons, which includes the case of one remaining person. This check does not apply when `reference` is given.

Scores:

After the exclusions, the function standardizes each score in three steps:

1. Subtract the mean of the remaining persons.
2. Divide by the standard deviation of the remaining persons.
3. Multiply by the sign of the score.

The `scores` element stores the mean and standard deviation on the raw scale. Because of this standardization, a hazard ratio for a score is per one standard deviation of the raw score, in the direction of its sign. Specify the signs before the analysis, for example from allele harmonization. A sign chosen from the association between score and outcome in the same data causes outcome leakage.

Covariates:

- Character and logical covariates become factors.
- The function drops the factor levels that no remaining person has.
- The function removes a covariate that is constant in the remaining persons, and it gives a warning.

The models use a numeric covariate as a linear term and a factor covariate as treatment contrasts. When a linear term is not correct for a numeric covariate, give the covariate a spline term with the argument `spline_df` of `fit_ladder()`. A transformation of the covariate in `data` before the call is an alternative, for example a logarithm.

Other cohorts:

When `reference` is given, every argument other than `data` must be missing. An argument set to its default value also stops the function. The function then applies the rules of `reference` to `data`:

- The column names, scores, covariates, and settings come from `reference`.
- The same exclusions apply.
- The function standardizes each score with the mean, standard deviation, and sign of `reference`.
- Each factor covariate gets the factor levels of `reference`. The function stops when a remaining person has a level that `reference` does not have, which includes a level that `reference` dropped.

- A covariate that is constant in the new persons stays in the result.

An export from `export_ladder()` holds the same rules as its development cohort. At another biobank, `reference = export` then gives the result of `reference = cohort` without the data of the development persons.

Give the result as `newdata` to `assess_calibration()`, `assess_discrimination()`, `assess_thresholds()`, `assess_net_benefit()`, `describe_case_mix()`, or `describe_cohort()`. The new persons must be separate from the development persons and from the GWAS data that gave the scores (Choi et al. 2020).

Value

An object of class `prs_cohort`, a list with these elements:

`data` A data frame with one row per remaining person. Its columns are `id`, `start`, `stop`, `status` (integer 0 or 1), `competing` (integer 0 or 1), `age_t1`, `t1_status`, and `t1_duration`. `competing` occurs only when `status_codes` has the `competing` role. `t1_status` is `"incident"` or `"prevalent"`. `t1_duration` is the double value of `start`, for use as a baseline covariate in `fit_ladder()`. Only the recruitment index has `t1_status` and `t1_duration`. The remaining columns are one standardized column per score, named by score name, and one column per covariate.

`scores` A data frame with one row per score and the columns `name`, `column`, `sign`, `mean`, and `sd`. `mean` and `sd` are on the raw scale.

`covariates` `character()`, the names of the covariates that remain.

`columns` Named `character()`, the input column names for `id`, `status`, `age_at_diagnosis`, and `age_at_exit`. The recruitment index adds `age_at_recruitment`.

`status_codes` The status codes, with the roles in the order `censored`, `event`, and `competing`.

`event_labels` `NULL`, or the event labels, with the names in the order `event` and `competing`.

`index`, `entry_lag`, `prevalent` The settings in use.

`n_excluded` Named `integer(6)`, the number of persons removed for each reason, in the order of the exclusions.

References

Andersen PK, Geskus RB, de Witte T, Putter H (2012). "Competing risks in epidemiology: possibilities and pitfalls." *International Journal of Epidemiology*, **41**(3), 861–870. doi:10.1093/ije/dyr213.

Choi SW, Mak TS, O'Reilly PF (2020). "Tutorial: a guide to performing polygenic risk score analyses." *Nature Protocols*, **15**(9), 2759–2772. doi:10.1038/s4159602003531.

Geskus RB (2011). "Cause-specific cumulative incidence estimation and the Fine and Gray model under both left truncation and right censoring." *Biometrics*, **67**(1), 39–49. doi:10.1111/j.1541-0420.2010.01420.x.

Putter H, Fiocco M, Geskus RB (2007). "Tutorial in biostatistics: competing risks and multi-state models." *Statistics in Medicine*, **26**(11), 2389–2430. doi:10.1002/sim.2712.

Tsai W (1990). "Testing the assumption of independence of truncation time and failure time." *Biometrika*, **77**(1), 169–177. doi:10.1093/biomet/77.1.169.

Tsiatis A (1975). "A nonidentifiability aspect of the problem of competing risks." *Proceedings of the National Academy of Sciences*, **72**(1), 20–22. doi:10.1073/pnas.72.1.20.

See Also

- `fit_ladder()` to fit the models.
- `summary.prs_cohort()` for the counts, follow-up, and score distributions.
- `describe_cohort()` for a table of participant characteristics.
- `prediction_data()` to prepare persons without outcome data.

Examples

```
#------------------------------------------------------------------------------
# prs_cohort() example
#------------------------------------------------------------------------------
dat <- simulate_prs_data(n = 2000L, seed = 1L)

# Diagnosis index: every person enters at T1.
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(
    onset = "onset_score",
    progression = "progression_score",
    outcome = "outcome_score"
  ),
  covariates = c("sex", "PC1", "PC2")
)
cohort
cohort$scores

# The recruitment index enters prevalent persons at recruitment.
recruit <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(progression = "progression_score"),
  index = "recruitment",
  age_at_recruitment = "age_at_recruitment"
)
recruit$n_excluded
summary(recruit)
```

```
# An external cohort follows the rules of the development cohort.
dat_new <- simulate_prs_data(n = 1000L, seed = 2L, id_prefix = "E")
external <- prs_cohort(dat_new, reference = cohort)
external$scores

# Deaths before T2 as competing events.
dat_death <- simulate_prs_data(n = 2000L, seed = 1L, competing_risk = 0.2)
with_death <- prs_cohort(
  dat_death,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(progression = "progression_score"),
  status_codes = list(censored = 0, event = 1, competing = 2)
)
with_death
```

`simulate_prs_data` *Simulate a Polygenic Risk Score Cohort*

Description

The function simulates a cohort of participants with a first diagnosis who are then followed for a second event. Each participant gets one or more polygenic risk scores, an event time, a censoring time, covariates, and an age at recruitment. The output also holds the true risk of each participant. Pass the output to `prs_cohort()`, then compare the estimates with the known hazard ratios and risks. Two optional arguments give sex, age, and the principal components a hazard ratio for each event.

Usage

```
simulate_prs_data(
  n = 3000L,
  seed = NULL,
  hazard_ratios = c(onset = 1.1, progression = 1.3, outcome = 1.15),
  score_correlation = 0,
  baseline_risk = 0.3,
  horizon = 10,
  weibull_shape = 1.2,
  competing_risk = 0,
  competing_hazard_ratios = NULL,
  covariate_hazard_ratios = NULL,
  competing_covariate_hazard_ratios = NULL,
  diagnosis_age_range = c(40, 70),
  incident_probability = 0.5,
  recruitment_offset_range = c(0.5, 8),
```

```
  recruitment_year_range = c(2006, 2010),
  end_year = 2022,
  prevalent_hazard_ratio = 1,
  duration_hazard_ratio = 1,
  prevalent_effect_years = Inf,
  female_probability = 0.5,
  n_pcs = 5L,
  id_prefix = "S"
)
```

Arguments

n	(integer(1); default 3000L; range [1, .Machine$integer.max]) Number of participants. A double with no fractional part is also accepted.
seed	(integer(1) or NULL; default NULL) Seed for the random-number generator. Details gives the effect on the generator state.
hazard_ratios	(named numeric(); default c(onset = 1.10, progression = 1.30, outcome = 1.15); range (0, Inf)) Hazard ratio per one standard deviation of each score. The length must be at least one. Names must be unique and unchanged by base::make.names(). Each name gives a score column with the suffix _score.
score_correlation	(numeric(1); default 0; range [0, 1)) Correlation between each pair of scores. A value of 0 gives independent scores.
baseline_risk	(numeric(1); default 0.30; range (0, 1)) Cumulative risk of the second event within horizon years for a participant whose scores are all zero. With covariate_hazard_ratios, the participant also has the reference values of the covariates, which Details gives.
horizon	(numeric(1); default 10; range (0, Inf)) Time in years at which baseline_risk and true_risk apply.
weibull_shape	(numeric(1); default 1.2; range (0, Inf)) Shape of the Weibull baseline hazard. A value of one gives a constant hazard. A value above one gives a hazard that increases with time since diagnosis.
competing_risk	(numeric(1); default 0; range [0, 1)) Net risk of the competing event within horizon years for a participant whose scores are all zero. With competing_covariate_hazard_ratios, the participant also has the reference values of the covariates, which Details gives. The value 0 gives no competing events.
competing_hazard_ratios	(named numeric() or NULL; default NULL; range (0, Inf)) Hazard ratio of the competing event per one standard deviation of each score, named by names of hazard_ratios. A score without an entry has a hazard ratio of 1. If NULL, no score changes the hazard of the competing event.
covariate_hazard_ratios	(named numeric() or NULL; default NULL; range (0, Inf)) Hazard ratio of the second event for one unit of each covariate. The names

must be unique, and each name must be sex, age_at_diagnosis, or a principal component from PC1 to PC<n_pcs>. The unit of age_at_diagnosis is one year. A covariate without an entry has a hazard ratio of 1. If NULL, no covariate changes the hazard of the second event.

competing_covariate_hazard_ratios
: (named numeric() or NULL; default NULL; range (0, Inf))
Hazard ratio of the competing event for one unit of each covariate. The names follow the rule of covariate_hazard_ratios. A covariate without an entry has a hazard ratio of 1. If NULL, no covariate changes the hazard of the competing event. The argument has no effect when competing_risk is 0.

diagnosis_age_range
: (numeric(2); default c(40, 70))
Minimum and maximum of the uniform age at diagnosis. The first element must be less than the second. The midpoint is the reference age of the hazard ratios of age_at_diagnosis.

incident_probability
: (numeric(1); default 0.5; range [0, 1])
Probability that a participant is recruited before diagnosis. A value of 0.8 gives a cohort in which about 80 percent of participants are incident.

recruitment_offset_range
: (numeric(2); default c(0.5, 8); range [0, Inf))
Minimum and maximum of the uniform gap between recruitment and diagnosis, in years. The first element must be less than the second. When incident_probability is positive, the second element must be less than end_year minus the second element of recruitment_year_range. Every incident diagnosis then comes before the end of the study.

recruitment_year_range
: (numeric(2); default c(2006, 2010))
Minimum and maximum of the uniform calendar year of recruitment. The first element must be less than the second.

end_year
: (numeric(1); default 2022)
Calendar year at which follow-up ends for every participant. It must be after the second element of recruitment_year_range.

prevalent_hazard_ratio
: (numeric(1); default 1; range (0, Inf))
Hazard ratio of the second event after recruitment for a prevalent participant, against an incident participant at the same time since diagnosis. duration_hazard_ratio multiplies this ratio once for each year of the gap from diagnosis to recruitment. With both ratios at 1, recruitment does not change the hazard.

duration_hazard_ratio
: (numeric(1); default 1; range (0, Inf))
Hazard ratio of the second event after recruitment for each year of the gap from diagnosis to recruitment of a prevalent participant. The product prevalent_hazard_ratio * duration_hazard_ratio^gap must be positive and finite at each end of recruitment_offset_range.

prevalent_effect_years
: (numeric(1); default Inf; range (0, Inf])
Years after recruitment during which the two hazard ratios apply. After these

years, a prevalent participant has the hazard of an incident participant again. With `Inf`, the ratios apply for all follow-up after recruitment.

`female_probability`
: (`numeric(1)`; default `0.5`; range `[0, 1]`)
Probability that `sex` is `"Female"`.

`n_pcs`
: (`integer(1)`; default `5L`; range `[0, .Machine$integer.max]`)
Number of independent standard-normal ancestry principal components. A value of `0` gives no principal component columns.

`id_prefix`
: (`character(1)`; default `"S"`)
Prefix of the participant identifiers in `IID`. Give a second cohort another prefix, such as `"E"` for an external cohort. Its identifiers then differ from those of the first cohort.

Details

Events and risks:

The scores and the covariates act on the hazard of the second event through a proportional-hazards model. Let x_i be the vector of scores for participant i, with one element per entry of `hazard_ratios`. Let β be the vector of log hazard ratios of the scores. Let w_i be the vector of covariates, which the section on covariates defines, and α the vector of their log hazard ratios. The linear predictor is $\eta_i = \beta^\top x_i + \alpha^\top w_i$. With the default `covariate_hazard_ratios = NULL`, α is zero. The survival function at time t since diagnosis is

$$S(t \mid x_i, w_i) = \exp\{-\lambda t^k \exp(\eta_i)\},$$

where k is `weibull_shape`. Let p_0 be `baseline_risk` and τ be `horizon`. A participant with $\eta_i = 0$ has cumulative risk p_0 at τ years when

$$\lambda = \frac{-\log(1 - p_0)}{\tau^k}.$$

In the parameterization of `stats::rweibull()`, the event time has shape k and scale $\{\lambda \exp(\eta_i)\}^{-1/k}$. The function draws each event time by inversion of S at a uniform random number (Bender et al. 2005). Follow-up ends at the end of the study, `end_year`, which censors every participant who is still free of the events. The censoring time is independent of the scores and the covariates. It depends on the gap between diagnosis and recruitment, so its distribution differs between incident and prevalent participants. Without an effect of recruitment on the hazard, the censoring time is also independent of the event times. The observed follow-up time is the smaller of the event time and the censoring time. Without an effect of recruitment, a Cox model of `Surv(time_to_exit, exit_status)` estimates the supplied hazard ratios. The model must have all score columns and each covariate with a hazard ratio for the second event.

When `competing_risk` is positive, each participant also has a latent time to a competing event, such as death. Its hazard is $\lambda_2 k t^{k-1} \exp(\eta_{2i})$, with the same shape k as the second event. Its linear predictor is $\eta_{2i} = \beta_2^\top x_i + \alpha_2^\top w_i$. Here β_2 holds the log of `competing_hazard_ratios`, and α_2 holds the log of `competing_covariate_hazard_ratios`. The rate is $\lambda_2 = -\log(1 - q_0)/\tau^k$ for q_0 equal to `competing_risk`. The observed follow-up time is then the smallest of the three times. `exit_status` is 2 when the competing event comes first. A Cox model of the second event, with `exit_status == 1` as the event, still estimates the hazard ratios of the second event. This model is the cause-specific hazards model. The function draws the competing times after all other

draws. Because of this, a seed gives the same participants with and without competing events. With competing events, some participants have the competing event before the second event.

Each score has a standard-normal marginal distribution. Let ρ be `score_correlation`. When ρ is positive, score j is $\sqrt{\rho}\, z_0 + \sqrt{1-\rho}\, z_j$. Here z_0 is a standard-normal draw that all scores share, and z_j is an independent standard-normal draw for score j. Each pair of scores then has correlation ρ. Use correlated scores to study whether one score adds information to another. With independent scores, each score with a hazard ratio other than one always adds information. With one score, a positive ρ does not change the distribution of the score. It changes the random draws, and the output differs from the output for $\rho = 0$ at the same seed.

The `true_risk` column is the cumulative incidence of the second event within τ years of diagnosis. This risk is the probability of the second event before the competing event. Let $a_i = \lambda \exp(\eta_i)$ and $b_i = \lambda_2 \exp(\eta_{2i})$. The value of `true_risk` for participant i is

$$r_i = \frac{a_i}{a_i + b_i}\left[1 - (1-p_0)^{\exp(\eta_i)}(1-q_0)^{\exp(\eta_{2i})}\right].$$

When `competing_risk` is zero, $b_i = 0$, and the risk is $1 - (1-p_0)^{\exp(\eta_i)}$. `baseline_risk` stays the net risk of the second event, which is the risk if the competing event did not occur. With competing events, the cumulative incidence at $\eta_i = \eta_{2i} = 0$ is less than `baseline_risk`. `true_risk` does not depend on censoring or recruitment. With an effect of recruitment, it is the risk at the hazard of an incident participant. It is the reference value for a calibration check.

Covariates:

The vector w_i has `sex`, the centered age at diagnosis, and the principal components `PC1` to `PC<n_pcs>`. The centered age is `age_at_diagnosis` minus the midpoint of `diagnosis_age_range`, which is 55 years with the default range. `covariate_hazard_ratios` and `competing_covariate_hazard_ratios` give the hazard ratio for one unit of a covariate. The unit is one year for `age_at_diagnosis`. For `sex`, the hazard ratio compares `"Female"` with `"Male"`. In w_i, `sex` is 1 for `"Female"` and 0 for `"Male"`. A Cox model with the factor `sex` names this term `sexFemale`. Each principal component has a standard deviation of 1. The birth year has no hazard ratio.

A participant has $\alpha^\top w_i = 0$ at the reference values of the covariates. The reference values are `"Male"` for `sex`, the midpoint of `diagnosis_age_range`, and 0 for each principal component. `baseline_risk` and `competing_risk` are then the net risks of a participant with these values and all scores at zero.

A Cox model that omits a covariate with a hazard ratio for its event does not estimate the hazard ratios of the scores. `fit_ladder()` adjusts each model for the covariates of the cohort and for the age at diagnosis.

The hazard ratios of the covariates change no random draw. A seed gives the same participants with and without them. Only `time_to_exit`, `exit_status`, `age_at_exit`, `recruited`, and `true_risk` can change.

Recruitment and calendar time:

The function recruits a participant before diagnosis with probability `incident_probability`, and after diagnosis otherwise. The gap between recruitment and diagnosis is uniform on `recruitment_offset_range`. The calendar year of recruitment is uniform on `recruitment_year_range`. The birth year is the recruitment year minus the age at recruitment. In time since diagnosis, follow-up ends at the signed gap from diagnosis to recruitment plus `end_year` minus the recruitment year. Because of this, censoring always comes after recruitment, and a participant recruited later has a shorter

follow-up. An incident participant also has a shorter follow-up after diagnosis than a prevalent participant recruited in the same year.

The function simulates every participant from diagnosis, also the prevalent participants who did not live to recruitment. The column `recruited` marks the participants that a biobank holds. It is `FALSE` only for a competing event, such as death, at or before recruitment. The column is `TRUE` for a participant with the second event before recruitment. The function ends follow-up at the second event and draws no later competing event. The column therefore assumes that every participant with the second event before recruitment lived to recruitment. `prs_cohort()` with `index = "recruitment"` removes this participant as `historical`. Without competing events, the column is `TRUE` for every participant.

When `weibull_shape` is not one, the hazard changes with time since diagnosis. For a participant recruited after diagnosis, the hazard at recruitment then depends on the gap.

Effect of recruitment on the hazard:

With the default arguments, recruitment does not change the hazard of the second event. Three arguments change this hazard after recruitment for a prevalent participant. Let θ be `prevalent_hazard_ratio`, ψ be `duration_hazard_ratio`, and c be `prevalent_effect_years`. Let d_i be the gap in years from diagnosis to recruitment of prevalent participant i. The hazard of the second event for this participant is

$$h(t \mid i) = \lambda k t^{k-1} \exp(\eta_i) \times \begin{cases} 1 & t \le d_i, \\ \theta\psi^{d_i} & d_i < t \le d_i + c, \\ 1 & t > d_i + c. \end{cases}$$

Before recruitment, a prevalent participant has the hazard of an incident participant. The participants with the second event before recruitment are then the same with and without the effect, and the column `recruited` does not change. The hazard of an incident participant and the hazard of the competing event do not change.

The function uses the same uniform random number for an event time with and without the effect. The cumulative hazard is piecewise linear in t^k, and the function inverts it exactly (Austin 2012). A seed then gives the same participants with and without the effect. Only `time_to_exit`, `exit_status`, and `age_at_exit` can change, and only for a prevalent participant with no event at or before recruitment.

An effect makes the event time depend on the gap. Under the recruitment index of `prs_cohort()`, the entry time is then not independent of the exit time, and `check_entry()` can reject.

The three arguments set which `design` of `fit_ladder()` is correct for a cohort under the recruitment index. The model of `"t1_entry"` has the terms $\gamma p_i + \delta d_i$, where p_i is 1 for a prevalent participant and 0 for an incident participant. With $c = \infty$ and no entry lag, the coefficient `t1_statusprevalent` estimates $\gamma = \log\theta$, and the coefficient `t1_duration` estimates $\delta = \log\psi$. The table gives the designs with a correct model of the hazard after entry:

Setting	`"t1"`	`"t1_entry"`	`"index"`
$\theta = 1$ and $\psi = 1$	Correct	Correct, with $\gamma = \delta = 0$	Correct if $k = 1$, approximate if not
$\theta \neq 1$ or $\psi \neq 1$, with $c = \infty$	Not correct	Correct	Correct if $k = 1$, approximate if not
$\theta \neq 1$ and $\psi = 1$, with finite c	Not correct	Not correct	Correct if $k = 1$, approximate if not
$\psi \neq 1$, with finite c	Not correct	Not correct	Not correct

Arguments and the random number generator:

No argument accepts NA or NaN. Only prevalent_effect_years accepts Inf. Only seed, competing_hazard_ratios, covariate_hazard_ratios, and competing_covariate_hazard_ratios accept NULL. When seed is not NULL, the function calls base::set.seed() with seed and the generator kinds "Mersenne-Twister", "Inversion", and "Rejection", which are the defaults of R. A session with other kinds, such as sample.kind = "Rounding", then gets the same data. On exit, the function restores the previous kinds and generator state. If no generator state exists before the call, the function removes the state on exit. When seed is NULL, the function uses the current generator state and advances it.

Value

A data.frame with n rows and 11 + length(hazard_ratios) + n_pcs columns, in the order below.

IID Character participant identifier, id_prefix and the row number padded with zeros to the number of digits in n.

<name>_score Numeric standard-normal score, one column per entry of hazard_ratios, in the same order.

sex Factor with the levels "Male" and "Female", in this order. The value is "Female" with probability female_probability and "Male" otherwise. Both levels exist also when one has no participant.

PC1 **to** PC<m> Numeric standard-normal principal components, where m is n_pcs.

birth_year Numeric calendar year of birth, with decimals.

age_at_diagnosis Numeric age at first diagnosis.

time_to_exit Numeric time in years from diagnosis to the second event, the competing event, or censoring.

exit_status Integer, 1 if the second event is observed, 2 if the competing event is observed, and 0 if censored. The value 2 occurs only when competing_risk is positive.

age_at_exit Numeric, age_at_diagnosis + time_to_exit.

age_at_recruitment Numeric age at recruitment.

recruitment_status Character, "incident" if recruited before diagnosis and "prevalent" otherwise.

recruited Logical, FALSE for a competing event at or before recruitment and TRUE otherwise.

true_risk Numeric cumulative incidence of the second event within horizon years of diagnosis.

References

Austin PC (2012). "Generating survival times to simulate Cox proportional hazards models with time-varying covariates." *Statistics in Medicine*, **31**(29), 3946–3958. doi:10.1002/sim.5452.

Bender R, Augustin T, Blettner M (2005). "Generating survival times to simulate Cox proportional hazards models." *Statistics in Medicine*, **24**(11), 1713–1723. doi:10.1002/sim.2059.

See Also

prs_cohort() to build an analysis cohort from the output.

Examples

```
#-------------------------------------------------------------------------
# simulate_prs_data() example
#-------------------------------------------------------------------------
library(prsladder)

# A biobank of persons with a first diagnosis (T1), followed for a second
# event (T2). The data have one row per person.
dat <- simulate_prs_data(n = 2000L, seed = 1L)
str(dat)

# The next step is prs_cohort(). Under the recruitment index, a prevalent
# person enters follow-up at recruitment. The cohort removes a prevalent
# person with T2 before recruitment as `historical`.
table(dat$recruitment_status)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(
    onset = "onset_score",
    progression = "progression_score",
    outcome = "outcome_score"
  ),
  index = "recruitment",
  age_at_recruitment = "age_at_recruitment"
)
cohort

# The model with all scores estimates the hazard ratios of the simulator.
# The defaults are 1.10, 1.30, and 1.15.
hr <- summary(fit_ladder(cohort))
hr[
  hr$model == "onset+progression+outcome",
  c("term", "hr", "hr_lower", "hr_upper")
]

#-------------------------------------------------------------------------
# A score that adds no information
#-------------------------------------------------------------------------
# The onset score has a hazard ratio of 1 and a correlation of 0.5 with the
# progression score. Alone, the onset score has a hazard ratio above 1. In
# the model with both scores, its hazard ratio is near 1.
dat_null <- simulate_prs_data(
  n = 5000L,
  seed = 2L,
  hazard_ratios = c(onset = 1, progression = 1.4),
  score_correlation = 0.5
)
cohort_null <- prs_cohort(
```

```
  dat_null,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  index = "recruitment",
  age_at_recruitment = "age_at_recruitment"
)
summary(fit_ladder(cohort_null))[c("model", "term", "hr", "hr_lower")]

#------------------------------------------------------------------------------
# Competing events and the true risk
#------------------------------------------------------------------------------
# Deaths before T2, with a hazard that increases with the onset score.
# `exit_status` is 1 for T2, 2 for a death, and 0 for the end of follow-up.
dat_death <- simulate_prs_data(
  n = 5000L,
  seed = 3L,
  competing_risk = 0.2,
  competing_hazard_ratios = c(onset = 1.2)
)
table(dat_death$exit_status)

# A biobank does not hold a person who died before recruitment. Keep the
# recruited persons, then pass them to prs_cohort() with `status_codes`.
table(dat_death$recruitment_status, recruited = dat_death$recruited)
biobank <- dat_death[dat_death$recruited, ]

# `true_risk` is the cumulative incidence of T2 within `horizon` years of
# T1. It does not depend on the end of follow-up. With an end in 2100, no
# person is censored before 10 years. The share of the persons with T2
# within 10 years then estimates the mean of `true_risk`.
dat_long <- simulate_prs_data(
  n = 5000L,
  seed = 3L,
  competing_risk = 0.2,
  competing_hazard_ratios = c(onset = 1.2),
  end_year = 2100
)
identical(dat_long$true_risk, dat_death$true_risk)
c(
  observed = mean(dat_long$exit_status == 1L & dat_long$time_to_exit <= 10),
  true_risk = mean(dat_long$true_risk)
)

#------------------------------------------------------------------------------
# Covariates that change the hazard
#------------------------------------------------------------------------------
# By default, sex and age do not change the hazard of T2. Here the hazard is
# 4% higher for each year of age at T1, and 30% lower for women.
dat_covariate <- simulate_prs_data(
  n = 5000L,
```

```
  seed = 6L,
  covariate_hazard_ratios = c(age_at_diagnosis = 1.04, sex = 0.7)
)
cohort_covariate <- prs_cohort(
  dat_covariate,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(
    onset = "onset_score",
    progression = "progression_score",
    outcome = "outcome_score"
  ),
  covariates = "sex",
  index = "recruitment",
  age_at_recruitment = "age_at_recruitment"
)

# `spline_df = NULL` gives the age at T1 a linear term, `age_t1`. The model
# without scores, `base`, estimates the two ratios.
base <- fit_ladder(cohort_covariate, spline_df = NULL)$coefficients
base[base$model == "base", c("term", "hr", "hr_lower", "hr_upper")]

#-------------------------------------------------------------------------
# An effect of recruitment on the hazard
#-------------------------------------------------------------------------
# By default, a prevalent person and an incident person have the same hazard
# at the same time since T1. This is the assumption of `design = "t1"` in
# fit_ladder(). Here the hazard of a prevalent person after recruitment is
# 1.5 times as large, and 1.1 times as large again for each year from T1 to
# recruitment. At the largest gap of 8 years, the ratio is 1.5 * 1.1^8 = 3.2.
dat_entry <- simulate_prs_data(
  n = 10000L,
  seed = 4L,
  hazard_ratios = c(prs = 1.3),
  prevalent_hazard_ratio = 1.5,
  duration_hazard_ratio = 1.1
)
cohort_entry <- prs_cohort(
  dat_entry,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(prs = "prs_score"),
  index = "recruitment",
  age_at_recruitment = "age_at_recruitment"
)

# The entry terms of `design = "t1_entry"` estimate the two ratios.
entry <- fit_ladder(cohort_entry, design = "t1_entry")$coefficients
is_entry_term <- entry$model == "prs" &
```

```
  entry$term %in% c("t1_statusprevalent", "t1_duration")
entry[is_entry_term, c("term", "hr", "hr_lower", "hr_upper")]

# A seed gives the same persons with and without the effect. Only the exit
# of a prevalent person can change.
dat_plain <- simulate_prs_data(
  n = 10000L,
  seed = 4L,
  hazard_ratios = c(prs = 1.3)
)
table(
  status = dat_plain$recruitment_status,
  changed = dat_plain$time_to_exit != dat_entry$time_to_exit
)

# A healthy-volunteer effect. The hazard of a prevalent person is half as
# large for 3 years after recruitment. No constant factor describes this
# effect, and check_design() picks `design = "index"`.
dat_volunteer <- simulate_prs_data(
  n = 10000L,
  seed = 4L,
  hazard_ratios = c(prs = 1.3),
  prevalent_hazard_ratio = 0.5,
  prevalent_effect_years = 3
)
check_design(prs_cohort(dat_volunteer, reference = cohort_entry))

#-------------------------------------------------------------------------------
# A second cohort for external validation
#-------------------------------------------------------------------------------
# Another seed gives other persons, and `id_prefix` gives them other
# identifiers. This cohort has a higher baseline risk.
dat_external <- simulate_prs_data(
  n = 1000L,
  seed = 5L,
  baseline_risk = 0.4,
  id_prefix = "E"
)
head(dat_external$IID, 3)
c(
  development = mean(dat$true_risk),
  external = mean(dat_external$true_risk)
)
```

`standardized_risk` *Standardized Risk at Score Percentiles*

Description

`standardized_risk()` gives the absolute risk of the second event (T2) at given percentiles of one polygenic risk score, from one Cox model of a ladder. For each percentile, the function sets the

score of every person of the cohort to the score at that percentile. It keeps the other predictors of each person, predicts the risk of each person, and averages the risks. This average is the standardized risk, also called the direct adjusted risk (Makuch 1982; Gail and Byar 1986).

The result translates a hazard ratio per standard deviation into absolute risks. An example is the 10-year risk at the 10th and the 90th percentile of the score. It also gives the difference and the ratio of each risk to the risk at a reference percentile.

Usage

```
standardized_risk(
  ladder,
  score,
  times,
  model = score,
  probs = c(0.1, 0.5, 0.9),
  reference = probs[1L],
  n_boot = 0L,
  seed = NULL,
  min_at_risk = 50L
)

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

Arguments

ladder	(prs_ladder) A ladder from fit_ladder().
score	(character(1)) The name of one score of ladder$cohort, as given in the names of scores in prs_cohort().
times	(numeric(); range (entry_lag, Inf)) Distinct finite times since T1, in the units of the age columns. Each time must be after the origin, the entry_lag of the cohort, and at or before the upper follow-up limit of the fit. The section "Missing risks" of predict_risk() defines this limit. The order does not matter. The result gives the times in increasing order.
model	(character(1); default score) The name of a model in ladder$models that contains score. The default is the model with score as its only score. Because fit_ladder() fits every subset of the scores, this model always exists.
probs	(numeric(); default c(0.1, 0.5, 0.9); range (0, 1)) Strictly increasing probabilities of the standardized score, with no NA. The function sets the score to the sample quantile at each probability. With a sign of -1, probs = 0.9 is the 10th percentile of the raw score.
reference	(numeric(1); default probs[1L]) One value of probs, the percentile of the differences and ratios. For example, reference = 0.5 selects the median. To use a percentile that is not in probs as

the reference, add it to probs. The function matches the value with a tolerance of sqrt(.Machine$double.eps). With this tolerance, reference = 0.7 finds the fourth element of seq(0.1, 0.9, 0.2).

n_boot (integer(1); default 0L; range [0, Inf))
Number of bootstrap replicates. If 0L, the standard errors and intervals are NA. See the section "Bootstrap" in Details for the time this takes.

seed (integer(1) or NULL; default NULL)
Seed for the random number streams of the bootstrap (see the section "Bootstrap" in Details). When n_boot is positive, the function restores the previous generator kinds and state on exit. If no generator state exists before the call, the function removes the state on exit. When seed is NULL, the function first draws one seed from the current generator, which advances its state. Then base::set.seed() before the call gives reproducible intervals. When n_boot is 0L, the function does not use the generator.

min_at_risk (integer(1); default 50L; range [1, Inf))
Minimum number of persons of the cohort at risk at the time and at each event time of either cause up to the time. With design = "index" in fit_ladder(), these persons are the incident persons of the cohort.

x (prs_standardized_risk)
A result of standardized_risk().

... Not used.

Details

Estimand:

Let z_p be the sample quantile of the standardized score at probability p, of type 7 in stats::quantile(). Let t_0 be the origin, the entry_lag of the cohort. For a time t since T1, the standardized risk at the percentile is

$$\bar{F}_p(t) = \frac{1}{n}\sum_{i=1}^{n} r_i(t - t_0 \mid z_p, x_i).$$

Here, r_i is the risk of predict_risk() for person i with start set to t_0. In this prediction, the score is z_p, and the other predictors x_i of person i keep their values. The columns t1_status and t1_duration are the exception (see the section "Time scale"). This risk is the mean risk if every person of the cohort had the score at the percentile. It has the form of the G-formula estimator of Ozenne et al. (2020), with a value of the score in place of the level of a binary treatment. That paper covers right censoring and competing events, but not delayed entry. riskRegression::ate() implements this estimator. In a test without delayed entry, the standardized risk agreed with the average of riskRegression::predictRisk() on a cause-specific Cox fit of riskRegression::CSC() (Ozenne et al. 2017) to within 0.0002. The small differences come from the baseline hazards at tied times. A simulation used the default hazard ratios of simulate_prs_data(), 20,000 persons, and competing events. In it, the 10-year standardized risk at the 5th, 50th, and 95th percentile was within 0.005 of the true value.

Here, "standardized" means the average over the persons of the cohort, as in epidemiology. It is not the standardized score of prs_cohort() in standard deviation units, and not the standardized net benefit of assess_net_benefit().

The other scores and the covariates of the model keep their observed values. Because of this, the contrast between two percentiles is conditional on the other scores of the model. A covariate, such as an ancestry principal component, can correlate with the score. The standardized risk at a percentile is then not the mean risk of the persons near that percentile. A simulation used a correlation of 0.6 between the score and a covariate that also changed the risk. In it, the 10-year standardized risk at the 90th percentile was 0.39. The mean risk of the persons from the 85th to the 95th percentile, with the score set to its 90th percentile, was 0.52. The default model has the score as its only score, and it adjusts for the covariates and the age at T1. To also adjust for the other scores, give a larger model, such as `model = "onset+progression"`.

Percentiles and the sign of the score:

`probs` refers to the standardized score, which includes the sign from `prs_cohort()`. The models and `cut_probs` in `cumulative_risk()` use the same scale. With a sign of `-1`, `probs = 0.9` is the 10th percentile of the raw score, and the column `raw_score` decreases as `prob` increases.

Time scale:

`times` are times since T1, as in `cumulative_risk()`. Every person starts at the origin. Because of this, the risk at time t is the risk from the origin to t. Under the recruitment index, the function also sets `t1_status` to `"incident"` and `t1_duration` to the origin for every person. The estimand is the mean over all persons of the risk from T1 with the hazard of an incident person. With `design = "index"` in `fit_ladder()`, the risk comes from the baseline of the incident persons. The follow-up limits and the support of a time then use these persons only. The design `"t1"` has no term for these columns. The standardized risk and the observed risk of `cumulative_risk()` are on the same time scale. When the hazards of the two groups differ, the result with `"t1_entry"` or `"index"` does not match the pooled curve of `cumulative_risk()`. That curve also holds the prevalent persons. A prevalent person keeps the age at T1 and the covariates, and this prediction can extrapolate (see the section "Designs" of `fit_ladder()`). The bootstrap fits the model again, and its intervals include this variance.

The function finds the two columns by their names. A column that you build for the same purpose and give in `covariates` of `prs_cohort()` keeps its observed values. The result then does not use the hazard of an incident person, even when the model has the same coefficients as `"t1_entry"`. Use the argument `design` of `fit_ladder()` to get this estimand.

The horizons of the `assess_*()` functions are times since entry, and they differ from `times` under the recruitment index or with a positive `entry_lag`.

Competing events:

When the cohort declares the `competing` status code (see `prs_cohort()`), the risk is the cumulative incidence of T2. The function then also gives the standardized cumulative incidence of the competing event. Because the score enters the models of both causes, a change in the score changes both cause-specific hazards. The contrast between two percentiles then includes the effect of the score on the competing event.

Population:

The average is over the persons of the cohort. Under the recruitment index, the cohort holds only the persons who were free of T2 at recruitment. Persons with high scores more often had T2 before recruitment. Because of this, the cohort holds fewer of them than the population at T1. The result is then not the standardized risk of all persons at T1. In a simulation with the default hazard ratios of `simulate_prs_data()`, the standardized risks of the two populations differed by about 0.001 at each percentile. A score with a larger hazard ratio gives a larger difference.

Support of a time:

The cohort supports a time t when both conditions below are true:

1. At least `min_at_risk` persons of the cohort are at risk at t.
2. At least `min_at_risk` persons of the cohort are at risk at each event time of either cause up to t.

These conditions are conditions 2 and 3 of `cumulative_risk()` for the whole cohort, and the columns `n_at_risk` and `min_at_risk_before` equal those of `cumulative_risk(cohort, NULL, times)`. With `design = "index"` in `fit_ladder()`, the risk rests on the risk sets of the incident persons alone. The two conditions and the two columns then use the incident persons of the cohort, and the columns do not equal those of `cumulative_risk()`. The second condition protects against an early risk set of a few prevalent persons. With 23 persons at risk at the origin, one early event adds about 1/23 to the baseline cumulative hazard. This step stays in every later risk. At a time that the cohort does not support, the risk, difference, and ratio columns are `NA`, and the function gives no warning. The function decides the support once, on these persons.

The function stops when fewer than 11 persons of the fit enter at the origin. The lower follow-up limit of the fit is then after the origin, and `predict_risk()` gives `NA` for every person (see the section "Missing risks" of `predict_risk()`). This case can occur under the recruitment index, because only the incident persons enter at the origin. A positive `entry_lag` in `prs_cohort()` moves the origin.

Differences and ratios:

The difference is the standardized risk at a percentile minus the standardized risk at the `reference` percentile, at the same time. The ratio is the standardized risk at a percentile divided by the standardized risk at the `reference` percentile. Both are `NA` at the reference percentile. The ratio is also `NA` when the risk at the reference percentile is 0. This case occurs with competing events when no T2 event occurs by the time.

Bootstrap:

A search found no published variance for a standardized cumulative incidence with delayed entry. The variance of Gail and Byar (1986) is for a direct adjusted survival curve over discrete strata, without competing events. The influence-function variance of Ozenne et al. (2020) accepts competing events, but not delayed entry, and it is for a binary treatment. Because of this, the function uses the nonparametric bootstrap. Each replicate resamples the persons of the cohort with replacement. Then it refits the model, and with competing events the model of the competing event. It then computes the quantiles of the score, the risks, the differences, and the ratios on the resampled persons. The standard error is the standard deviation of the replicates. The interval is the 95% percentile interval. If `n_boot` is `0L`, these columns are `NA`, and the function does not use the random number generator.

Replicate b draws its persons from stream b of the L'Ecuyer-CMRG generator (L'Ecuyer et al. 2002). Stream 1 comes from `base::set.seed()` with `seed`, and stream $b+1$ from `parallel::nextRNGStream()` applied to stream b. The function sets the generator kinds explicitly. A session with other kinds, such as `sample.kind = "Rounding"`, then gets the same intervals. The bootstrap of the `assess_*()` functions uses the same streams (see `bootstrap_plan()`).

A replicate fails when a refit fails or when it gives `NA` at a supported time. A refit fails when `survival::coxph()` gives a warning, or when the fit has an aliased coefficient, as in `fit_ladder()`. With `design = "index"`, a refit also fails when a value of `t1_status` has 10 persons or fewer. A refit can fail in a resample by chance, for example when a rare factor level is absent. The risks of

a refit are NA when the lower follow-up limit of the resample is after the origin. They are also NA when the upper follow-up limit of the resample is before a time of times. The section "Missing risks" of predict_risk() defines these limits. The function drops each failed replicate and gives one warning with the number of failed replicates. The element n_boot_used of settings gives the number of replicates that remain. When fewer than half of the replicates remain, the interval columns are NA. Within the replicates that remain, an NA ratio in one replicate makes the interval of that ratio NA.

A simulation checked the intervals without competing events. It used 200 samples of 2,000 persons and 200 replicates, under both indexes. The 95% intervals of the risk covered the true risk in 94% of the samples. The intervals of the difference and the ratio were about 8% too narrow. Because of this, they covered the true value in 90% to 94% of the samples. The simulation did not check the intervals with competing events.

A second simulation used 300 new samples under the diagnosis index. In it, the bootstrap standard error of the coefficient of the score equaled its standard error from survival::coxph(). Both were about 5% smaller than the standard deviation of the coefficient over the samples. Because of this, the narrow intervals are not specific to the bootstrap. The Wald interval of the hazard ratio covered the true value in 93% of the samples, and the percentile interval of the difference in 92%. The basic and the normal bootstrap intervals did not cover better.

A third simulation checked the intervals with design = "t1_entry", design = "index", and the pick of check_design(), under the recruitment index. It used 200 samples of 1,500 persons in each of three settings of the entry. Each sample had a recruitment period of 4 years, 200 replicates, and no competing events. The prediction of a prevalent person then extrapolates (see the section "Designs" of fit_ladder()). The intervals of the risk at the median covered the true value in 91.5% to 95.5% of the samples. At the 10th and the 90th percentile, the coverage was 90% to 95.5%. The coverage of the difference was 90.5% to 95.5%. The Monte Carlo standard error of each coverage is about 1.5 percentage points.

The time for the bootstrap increases with the number of persons times the number of event times, for each percentile. With competing events, the function computes the cumulative incidence of both events at all times in one pass. Because of this, the number of times has almost no effect on the time for the bootstrap. In a test with competing events, 20,000 persons, 3 percentiles, and times up to 10 years, each replicate took about 5 seconds. For 200 replicates, the bootstrap then takes about 16 minutes. A replicate takes less time than the estimate on the cohort, which took about 9 seconds, because a resample has fewer distinct event times. Without competing events, the estimate took less than 1 second.

Value

A list of class prs_standardized_risk with the elements below.

risks A data frame with one row per probability and time. The function orders the rows by probability, then by increasing time. The columns competing_risk to competing_risk_upper occur only when the cohort declares the competing status code. Each estimate has three more columns with the suffixes _se, _lower, and _upper, for its bootstrap standard error and 95% percentile interval.

prob Probability of the standardized score, from probs.

score_value The standardized score at prob, in standard deviation units.

raw_score The same score on the raw scale.

time Time since T1.

`n_at_risk` Integer number of persons of the cohort at risk at `time`. With `design = "index"` in `fit_ladder()`, the number of incident persons.

`min_at_risk_before` Integer smallest number at risk at an event time of either cause up to `time`. With `design = "index"`, the number and the event times are those of the incident persons. `NA` when the cohort has no event by `time`.

`risk` Standardized risk of T2 at `time`. `NA` when the cohort does not support `time`.

`competing_risk` Standardized cumulative incidence of the competing event at `time`.

`difference` Risk difference from the `reference` percentile.

`ratio` Risk ratio to the `reference` percentile.

`settings` A list with the elements `score`, `sign`, `model`, `design`, `probs`, `reference`, `times`, `n_boot`, `n_boot_used`, `min_at_risk`, `competing`, `index`, and `origin`. `sign` is the sign of the score in `prs_cohort()`. `design` is the argument `design` of `fit_ladder()`. `reference` is the element of `probs` that matches the input. `times` is in increasing order. `n_boot_used` is the number of replicates that did not fail, and `0L` when the cohort supports no time. `origin` is the `entry_lag` of the cohort.

`print()` shows the settings and at most 20 rows of `risks`. The rows give the risk, its interval when `n_boot` is positive, the competing risk with competing events, the difference, and the ratio. It returns `x` invisibly.

References

Gail MH, Byar DP (1986). "Variance calculations for direct adjusted survival curves, with applications to testing for no treatment effect." *Biometrical Journal*, **28**(5), 587–599. doi:10.1002/bimj.4710280508.

L'Ecuyer P, Simard R, Chen EJ, Kelton WD (2002). "An object-oriented random-number package with many long streams and substreams." *Operations Research*, **50**(6), 1073–1075. doi:10.1287/opre.50.6.1073.358.

Makuch RW (1982). "Adjusted survival curve estimation using covariates." *Journal of Chronic Diseases*, **35**(6), 437–443. doi:10.1016/00219681(82)900583.

Ozenne B, Sørensen AL, Scheike T, Torp-Pedersen C, Gerds TA (2017). "riskRegression: predicting the risk of an event using Cox regression models." *The R Journal*, **9**(2), 440–460. doi:10.32614/RJ2017062.

Ozenne BMH, Scheike TH, Stærk L, Gerds TA (2020). "On the estimation of average treatment effects with right-censored time to event outcome and competing risks." *Biometrical Journal*, **62**(3), 751–763. doi:10.1002/bimj.201800298.

See Also

- `cumulative_risk()` for the observed risk by score group, without a model.
- `predict_risk()` for the risk of each person.

Examples

```
#-------------------------------------------------------------------------
# standardized_risk() example
#-------------------------------------------------------------------------
library(prsladder)

# Fit a ladder on a simulated cohort.
dat <- simulate_prs_data(n = 2000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  covariates = "sex"
)
ladder <- fit_ladder(cohort)

# The 5- and 10-year risk at the 10th, 50th, and 90th percentile of the
# progression score, from the model with this score alone.
res <- standardized_risk(ladder, "progression", times = c(5, 10))
res$risks[c("prob", "time", "risk", "difference", "ratio")]

# The same risks from the model that also adjusts for the onset score,
# against the median.
res <- standardized_risk(
  ladder,
  "progression",
  times = 10,
  model = "onset+progression",
  reference = 0.5
)
res$risks[c("prob", "raw_score", "risk", "ratio")]
```

`summary.prs_cohort` *Summarize a Cohort*

Description

`summary()` gives the cohort items that a polygenic score study reports:

- The number of persons and events.
- The median follow-up.
- The distribution of each raw score.
- The raw scores by status of the second event (T2).
- The exclusion counts.

The PRS-RS reporting standard asks for a description of the score distribution (Wand et al. 2021).

Usage

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

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

Arguments

object	(prs_cohort) A cohort from prs_cohort().
...	Not used.
x	(summary.prs_cohort) A summary from summary.prs_cohort().

Details

The median follow-up is the median of the reverse Kaplan-Meier estimate of the time since entry, stop - start. The reverse Kaplan-Meier estimate treats T2 and a competing event as censoring of the follow-up. At a tied time, events leave the risk set before censorings, as in prodlim::prodlim(..., reverse = TRUE). The median is NA when the estimate does not fall to one half, for example when every person has T2.

Under the recruitment index with incident and prevalent persons, the two groups can have different follow-up since entry. Let G_s be the reverse Kaplan-Meier estimate of the n_s persons of group s, of n persons in all. The row "all" then uses the mixture $\sum_s (n_s/n) G_s(t)$. The median is the first time at which the mixture is at or below one half. The censoring weights of the assess_*() functions use the same groups (see assess_discrimination()).

Under the recruitment index, counts also gives one row for each value of t1_status.

The function recovers the raw scores from the standardized scores with the mean, sd, and sign columns of object$scores. The raw scores are the values in the input data, before the function applies the sign.

Value

summary() returns an object of class summary.prs_cohort, a list with these elements:

index "diagnosis" or "recruitment", the index of the cohort.

counts A data frame with the columns group, n, n_events, and median_followup. With the competing role, the column n_competing follows n_events. It has the row "all" and, under the recruitment index, the rows "incident" and "prevalent".

scores A data frame with one row per score and the columns name, mean, sd, q25, median, and q75, on the raw scale.

scores_by_status A data frame with one row per score and status, and the columns name, status, n, mean, and sd, on the raw scale. status is 0 for censored and 1 for T2. With the competing role, status is 2 for a competing event.

n_excluded The n_excluded element of object.

print() returns x invisibly.

References

Wand H, Lambert SA, Tamburro C, Iacocca MA, O'Sullivan JW, Sillari C, Kullo IJ, Rowley R, Dron JS, Brockman D, Venner E, McCarthy MI, Antoniou AC, Easton DF, Hegele RA, Khera AV, Chatterjee N, Kooperberg C, Edwards K, Vlessis K, Kinnear K, Danesh JN, Parkinson H, Ramos EM, Roberts MC, Ormond KE, Khoury MJ, Janssens ACJW, Goddard KAB, Kraft P, MacArthur JAL, Inouye M, Wojcik GL (2021). "Improving reporting standards for polygenic scores in risk prediction studies." *Nature*, **591**(7849), 211–219. doi:10.1038/s41586021032436.

See Also

`describe_cohort()` for a table of participant characteristics, by status of T2 or by sample.

Examples

```
#--------------------------------------------------------------------------
# summary.prs_cohort() example
#--------------------------------------------------------------------------
dat <- simulate_prs_data(n = 1000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score"),
  index = "recruitment",
  age_at_recruitment = "age_at_recruitment"
)
summary(cohort)
```

---

validation *Internal and External Validation of a Ladder*

Description

A model predicts better for the persons of its fit than for new persons. Validation estimates how well the models of a ladder predict for new persons. The `assess_*()` functions have three modes for this purpose. This page tells you which mode to use, and why a random split of one cohort is not recommended.

Details

Three modes:

`assess_discrimination()`, `assess_calibration()`, `assess_thresholds()`, and `assess_net_benefit()` select the mode from the arguments `validation` and `newdata`:

validation	newdata	ladder	Column evaluation
NULL	NULL	ladder	"apparent"
a plan	NULL	ladder	"optimism-corrected"
NULL	a cohort	ladder or export	"external"

Here a ladder comes from `fit_ladder()`, an export from `export_ladder()`, a plan from `bootstrap_plan()`, and a cohort from `prs_cohort()`. The functions stop with an error when you give both `validation` and `newdata`. They also stop for an export without `newdata`, because an export holds no persons. Each result has the same elements and columns in every mode. Because of this, the same element of two results stacks with `base::rbind()`.

Apparent mode:

The apparent mode evaluates the models on the development persons. The estimates are optimistic, because `fit_ladder()` fitted the models to the same persons. The intervals also ignore the uncertainty of the fitted models. Use the apparent mode to describe the fit, not to report the performance on new persons. The column `concordance` of `fit_ladder()$table` is also apparent.

Internal validation:

With a plan in `validation`, the functions correct the apparent estimates for optimism with the bootstrap (Harrell Jr. et al. 1996). Each replicate fits the ladder again on a bootstrap sample of the development persons. The correction uses all persons for the fit and for the evaluation. Let n be the number of development persons. The corrected value estimates the expected performance on new persons of a ladder that `fit_ladder()` fits on n persons of the same population (Bates et al. 2024). Use this mode for every analysis of one cohort. `bootstrap_plan()` gives the method, the intervals, and the time.

Give the same plan to each function. Each function then uses the same bootstrap samples, and its results pair with the results of the other functions. `assess_stability()` uses the same samples to measure how much the predicted risk of each person changes between refits (Riley and Collins 2023). It needs a plan.

The correction repeats only the steps of `fit_ladder()`. Choose the rung, `spline_df`, the signs of the scores, the thresholds, and the pairs before the analysis. A choice made from the results is not part of the replicates, and the correction is then too small. The correction takes the argument `design` as fixed. It does not repeat the selection of `check_design()`. The column `heuristic_shrinkage` of `fit_ladder()$table` shows the size of the optimism before any bootstrap.

External validation:

With a cohort in `newdata`, the functions evaluate the models on other persons. Use this mode for a different cohort, such as another biobank, a later period, or another ancestry group. The result then shows whether the models transport to that setting. Build the cohort with `prs_cohort(data, reference = cohort)`. The cohort then has the rules of the development cohort.

When the person-level data of the development cohort must stay in its biobank, use `export_ladder()`. The export holds the models and the rules of the cohort, but no person-level data. The other biobank builds its cohort with `prs_cohort(data, reference = export)` and gives the export to the `assess_*()` functions. `describe_case_mix()` compares the linear predictors of the two cohorts. A difference in their spread or their mean helps to explain a difference in discrimination or in calibration (Debray et al. 2015).

The functions give no warning about the size of the external cohort. Collins et al. (2016) recommend at least 100 events, and preferably 200 or more. Riley et al. (2022) give a sample size calculation for a time-to-event outcome at one horizon.

Comparison of designs:

Compare the values of the argument `design` of `fit_ladder()` with optimism-corrected or external values, not with apparent values. With `design = "index"`, each value of `t1_status` has its own baseline. The expected events of each value then sum to its own observed events. Its apparent O/E ratio is close to 1 at each horizon by construction. Because of this, apparent values favor `"index"`.

Random split of one cohort:

A random split of one cohort into a development part and a test part is not recommended. The split has two costs:

- The fit and the likelihood-ratio tests of `compare_ladder()` lose the events of the test part.
- The estimate from the test part is imprecise, because the test part is small.

In the study of Steyerberg et al. (2001), the split gave pessimistic estimates with large variability. The bootstrap gave stable estimates with low bias. Austin and Steyerberg (2017) found that the estimates of a split have the bias and variance of a sample that is smaller by the share of persons in the test part. Steyerberg and Harrell Jr. (2016) recommend the bootstrap for internal validation. A random test part also comes from the same population as the development part. Because of this, it does not show whether the models transport to another setting. For a cohort of several biobanks, centers, or periods, Steyerberg and Harrell Jr. (2016) recommend internal-external validation. Each unit takes one turn as the test unit. A model from the other units then gives the estimates on the test unit. The final model uses all units. With the external mode, fit a ladder on all units but the test unit, and give the test unit in `newdata`.

Other functions:

- `compare_ladder()` has no validation. Its likelihood-ratio tests are inference on the fitted associations. With pairs chosen before the analysis, the tests are valid on the development persons.
- `standardized_risk()` estimates a quantity of the population, not the performance of a model. Its bootstrap fits the model again in each replicate.
- `cumulative_risk()` and its figures and tables use no fitted model.

Status of the intervals:

One simulation check measured the coverage of the analytic interval, the default of `bootstrap_plan()`. The check used these settings:

- `simulate_prs_data()` with two scores and the covariates sex, PC1, and PC2, under the diagnosis index.
- 500 development samples of 1200 persons, with about 490 events in each.
- `assess_discrimination()` at 10 years, with `bootstrap_plan(n_boot = 200L)`.
- For each sample, a reference value from the external mode on 20,000 new persons of the same simulator.

Rung	Mean reference AUC	Coverage of the AUC	Coverage of the Brier score

base	0.500	0.892	0.966
onset	0.519	0.904	0.964
progression	0.579	0.940	0.960
onset+progression	0.584	0.932	0.958

The Monte Carlo standard error of each coverage is 0.008 to 0.014. The coverage of the mean reference over the samples differs by at most 0.02 from these values.

For a rung with an AUC near 0.5, the interval of the AUC covers less often than 95%. The cause is the bias of the corrected AUC, not the width of the interval. For the rung `base`, the mean corrected AUC was 0.513, and the mean reference was 0.500. For a model with little signal, the optimism correction removes only part of the optimism. A check with logistic regression, without this package, gave the same bias. Efron (1983) found that the bootstrap correction of an error rate can have a large downward bias in highly overfitted situations. Smith et al. (2014) found an optimistic bootstrap correction of the C statistic for a model with a true C statistic of 0.61, with 10 or more events per variable. The interval of the Brier score is slightly too wide.

The check did not measure other settings, such as the recruitment index, competing events, smaller cohorts, or other horizons. It also did not measure these intervals:

- The intervals of `assess_calibration()`, `assess_thresholds()`, and `assess_net_benefit()`.
- The two-stage interval.
- The intervals of the differences between rungs.

References

Austin PC, Steyerberg EW (2017). "Events per variable (EPV) and the relative performance of different strategies for estimating the out-of-sample validity of logistic regression models." *Statistical Methods in Medical Research*, **26**(2), 796–808. doi:10.1177/0962280214558972.

Bates S, Hastie T, Tibshirani R (2024). "Cross-validation: what does it estimate and how well does it do it?" *Journal of the American Statistical Association*, **119**(546), 1434–1445. doi:10.1080/01621459.2023.2197686.

Collins GS, Ogundimu EO, Altman DG (2016). "Sample size considerations for the external validation of a multivariable prognostic model: a resampling study." *Statistics in Medicine*, **35**(2), 214–226. doi:10.1002/sim.6787.

Debray TPA, Vergouwe Y, Koffijberg H, Nieboer D, Steyerberg EW, Moons KGM (2015). "A new framework to enhance the interpretation of external validation studies of clinical prediction models." *Journal of Clinical Epidemiology*, **68**(3), 279–289. doi:10.1016/j.jclinepi.2014.06.018.

Efron B (1983). "Estimating the error rate of a prediction rule: improvement on cross-validation." *Journal of the American Statistical Association*, **78**(382), 316–331. doi:10.1080/01621459.1983.10477973.

Harrell Jr. FE, Lee KL, Mark DB (1996). "Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors." *Statistics in Medicine*, **15**(4), 361–387. doi:10.1002/(SICI)10970258(19960229)15:4<361::AIDSIM168>3.0.CO;2-4.

Riley RD, Collins GS (2023). "Stability of clinical prediction models developed using statistical or machine learning methods." *Biometrical Journal*, **65**(8), e2200302. doi:10.1002/bimj.202200302.

Riley RD, Collins GS, Ensor J, Archer L, Booth S, Mozumder SI, Rutherford MJ, van Smeden M, Lambert PC, Snell KIE (2022). "Minimum sample size calculations for external validation of a clinical prediction model with a time-to-event outcome." *Statistics in Medicine*, **41**(7), 1280–1295. doi:10.1002/sim.9275.

Smith GCS, Seaman SR, Wood AM, Royston P, White IR (2014). "Correcting for optimistic prediction in small data sets." *American Journal of Epidemiology*, **180**(3), 318–324. doi:10.1093/aje/kwu140.

Steyerberg EW, Harrell Jr. FE (2016). "Prediction models need appropriate internal, internal-external, and external validation." *Journal of Clinical Epidemiology*, **69**, 245–247. doi:10.1016/j.jclinepi.2015.04.005.

Steyerberg EW, Harrell Jr. FE, Borsboom GJJM, Eijkemans MJC, Vergouwe Y, Habbema JDF (2001). "Internal validation of predictive models: efficiency of some procedures for logistic regression analysis." *Journal of Clinical Epidemiology*, **54**(8), 774–781. doi:10.1016/S08954356(01)00341-9.

See Also

- `bootstrap_plan()` for the internal validation.
- `export_ladder()` and `describe_case_mix()` for the external validation.
- `horizon_support()` to check horizons before an evaluation.

Examples

```
#-------------------------------------------------------------------------
# validation example
#-------------------------------------------------------------------------
library(prsladder)

# Fit a ladder on a development cohort.
dat <- simulate_prs_data(n = 1000L, seed = 1L)
cohort <- prs_cohort(
  dat,
  id = "IID",
  status = "exit_status",
  age_at_diagnosis = "age_at_diagnosis",
  age_at_exit = "age_at_exit",
  scores = c(onset = "onset_score", progression = "progression_score")
)
ladder <- fit_ladder(cohort)

# Build an external cohort with a higher baseline risk and the rules of the
# development cohort.
external <- prs_cohort(
  simulate_prs_data(n = 1000L, seed = 2L, baseline_risk = 0.40,
```

```
                    id_prefix = "E"),
  reference = cohort
)

# The three modes. A small number of replicates keeps the example fast. An
# analysis needs 200 or more.
plan <- bootstrap_plan(n_boot = 20L, seed = 1L)
apparent <- assess_discrimination(ladder, horizons = 10)
corrected <- assess_discrimination(ladder, horizons = 10, validation = plan)
external_result <- assess_discrimination(
  ladder,
  horizons = 10,
  newdata = external
)

# The results of the three modes stack.
results <- rbind(apparent$models, corrected$models, external_result$models)
results[c("model", "evaluation", "auc", "auc_lower", "auc_upper")]
```

Index

assess_calibration, 3
assess_calibration(), *13*, *16*, *25*, *28*, *32*, *33*, *35*, *36*, *47*, *56*, *64*, *66*, *74–76*, *78*, *79*, *96*, *111*, *132*, *135*
assess_discrimination, 11
assess_discrimination(), *5*, *9*, *10*, *18–21*, *24*, *25*, *27–30*, *32*, *33*, *35*, *36*, *44*, *56*, *66*, *74*, *75*, *77*, *80*, *81*, *96*, *111*, *131*, *132*, *134*
assess_net_benefit, 17
assess_net_benefit(), *33*, *35*, *36*, *66*, *74*, *75*, *85*, *86*, *111*, *125*, *132*, *135*
assess_stability, 23
assess_stability(), *33*, *35*, *36*, *75*, *133*
assess_thresholds, 25
assess_thresholds(), *10*, *12*, *16*, *18*, *19*, *22*, *33*, *35*, *36*, *66*, *74*, *75*, *96*, *111*, *132*, *135*

base::expm1(), *93*
base::make.names(), *114*
base::rbind(), *8*, *14*, *20*, *30*, *133*
base::saveRDS(), *62*
base::set.seed(), *34*, *119*, *125*, *127*
bootstrap_plan, 33
bootstrap_plan(), *4*, *7*, *10*, *12*, *14–16*, *18*, *20*, *23–26*, *29*, *127*, *133*, *134*, *136*

check_design, 37
check_design(), *41*, *42*, *68*, *70*, *74*, *108*, *128*, *133*
check_entry, 40
check_entry(), *39*, *108*, *118*
compare_ladder, 42
compare_ladder(), *12–14*, *16*, *18*, *26*, *32*, *67*, *71*, *74*, *81*, *84*, *134*
cumulative_risk, 45
cumulative_risk(), *13*, *19*, *28*, *41*, *60*, *61*, *78*, *79*, *82*, *83*, *87*, *88*, *90*, *96*, *105*, *126*, *127*, *129*, *134*

describe_case_mix, 54
describe_case_mix(), *58*, *62*, *66*, *75*, *111*, *133*, *136*
describe_cohort, 56
describe_cohort(), *56*, *61*, *105*, *109*, *111*, *112*, *132*
describe_risk, 60
describe_risk(), *45*, *53*, *83*, *89*, *109*

export_ladder, 62
export_ladder(), *3*, *11*, *18*, *26*, *55*, *56*, *75*, *92–95*, *98*, *99*, *103*, *105*, *107*, *111*, *133*, *136*

fit_ladder, 67
fit_ladder(), *3*, *7*, *11*, *15*, *18*, *20*, *23*, *24*, *26*, *33*, *34*, *37–39*, *41–44*, *49*, *55*, *58*, *59*, *63–66*, *84*, *92–96*, *99*, *100*, *103–105*, *107–112*, *117*, *118*, *124–129*, *133*, *134*

ggplot2::coord_cartesian(), *86*
ggplot2::theme_bw(), *89*
gtsummary::add_overall(), *58*, *59*
gtsummary::as_gtsummary(), *61*
gtsummary::as_kable(), *59*, *61*
gtsummary::style_percent(), *61*
gtsummary::tbl_summary(), *57*, *59*
gtsummary::tbl_survfit(), *61*

horizon_support, 75
horizon_support(), *4*, *10*, *12*, *16*, *24*, *27*, *35*, *48*, *53*, *95*, *96*, *136*

parallel::nextRNGStream(), *34*, *127*
parallel::nextRNGSubStream(), *34*
plot_calibration, 78
plot_calibration(), *81*, *86*
plot_discrimination, 80
plot_discrimination(), *79*
plot_gradient, 82

plot_gradient(), *45*, *53*, *61*, *89*, *109*
plot_ladder, 83
plot_ladder(), *74*, *109*
plot_net_benefit, 85
plot_net_benefit(), *17*, *22*
plot_risk, 87
plot_risk(), *45*, *53*, *60, 61*, *79*, *83*, *90, 91*, *109*
plot_stacked, 90
plot_stacked(), *45*, *53*, *89*, *109*
predict_risk, 91
predict_risk(), *4, 5*, *12, 13*, *24*, *27*, *35*, *39*, *41*, *45*, *53*, *55*, *62, 63*, *66*, *68*, *71–73*, *75–77*, *98–101*, *108*, *124, 125*, *127–129*
prediction_data, 98
prediction_data(), *62*, *69*, *91*, *94–96*, *105*, *108*, *112*
print.prs_cohort, 102
print.prs_cumulative_risk *(*cumulative_risk*)*, 45
print.prs_export, 103
print.prs_ladder, 104
print.prs_standardized_risk *(*standardized_risk*)*, 123
print.summary.prs_cohort *(*summary.prs_cohort*)*, 130
prs_cohort, 105
prs_cohort(), *5*, *12, 13*, *19*, *28*, *37*, *40–43*, *46–48*, *52*, *57, 58*, *61, 62*, *65–68*, *71*, *76, 77*, *79*, *83, 84*, *88*, *90*, *93–95*, *98–102*, *113*, *118, 119*, *124–127*, *129*, *131*, *133*

simulate_prs_data, 113
simulate_prs_data(), *38*, *50*, *64*, *69*, *125, 126*
splines::ns(), *70*, *72*
standardized_risk, 123
standardized_risk(), *19*, *69*, *74*, *96*, *134*
stats::coef(), *73*
stats::predict(), *73*, *95*
stats::quantile(), *125*
stats::rweibull(), *116*
stats::update(), *71*, *92*
stats::vcov(), *73*
summary.prs_cohort, 130
summary.prs_cohort(), *58, 59*, *108*, *112*, *131*
summary.prs_ladder *(*print.prs_ladder*)*, 104
summary.prs_ladder(), *74*, *84*
survival::aeqSurv(), *109*
survival::cox.zph(), *38*, *72, 73*
survival::coxph(), *38*, *55*, *65*, *68*, *72, 73*, *92, 93*, *109*, *127, 128*
survival::predict.coxph(), *94*
survival::survfit(), *48, 49*, *51*, *92, 93*, *95*

validation, *3*, *10, 11*, *14*, *16, 17*, *22, 23*, *25*, *32, 33*, *35, 36*, *44*, *70*, *74*, 132