Package: survlearning (via r-universe)

November 26, 2026

Type Package

Title Unified Statistical Learning for Survival and Reliability

Version 0.3.5

Description Provides a small formula-based interface for fitting, tuning, predicting, and comparing statistical learning models for right-censored survival and reliability data. Provides censoring-aware evaluation, reusable stratified, grouped and temporal validation, and nested tuning. Supports paired assessment intervals, censoring-weighted calibration measures and cross-validated convex combinations of survival curves. Distinguishes risk scores, survival probabilities and median survival times. Includes reference models, forests, boosting and neural survival models through integrated engines, with training-only preprocessing and explicit prediction capabilities. Methodological references include Simon et al. (2011) <doi:10.18637/jss.v039.i05> for penalized Cox regression and Ishwaran et al. (2008) <doi:10.1214/08-AOAS169> for random survival forests.

License GPL (>= 3)

URL https://github.com/ramospedroluiz/survlearning

BugReports https://github.com/ramospedroluiz/survlearning/issues

Encoding UTF-8

Depends R (>= 4.3.0)

Imports stats, survival, graphics, utils, methods, parallel, tools, withr (>= 2.5.0), glmnet (>= 5.0), rpart, gbm, Rcpp, irlba, xgboost (>= 3.0.0), MTLR (>= 0.2.2), CoxBoost, ranger, aorsf, splines, nnet, Matrix, quadprog, osqp (>= 1.0.0), clarabel (>= 0.11.3)

Suggests survivalsvm, randomForestSRC, flexsurv, penAFT, testthat (>= 3.1.8), knitr, rmarkdown, callr, pkgload

LinkingTo Rcpp, RcppArmadillo

SystemRequirements OpenMP (optional)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Pedro Luiz Ramos [aut, cre], Hemant Ishwaran [ctb] (Included RF-SRC code), Udaya B. Kogalur [ctb] (Included RF-SRC code), Christopher Jackson [ctb] (Adapted Royston-Parmar initialization), Aaron J. Molstad [ctb] (Adapted Gehan ADMM), Piotr M. Suder [ctb] (Adapted Gehan ADMM)

Maintainer Pedro Luiz Ramos <pedro.ramos@uc.cl>

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

Date/Publication 2026-11-26 01:52:45 UTC

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

RemoteRef HEAD

RemoteSha da42fe2d4942dd51490c1fed6d8ec3cb7fd0d4de

Contents

coef.survlearning_fit . . . 3
plot.survlearning_ale . . . 3
plot.survlearning_calibration . . . 4
plot.survlearning_evaluation . . . 5
plot.survlearning_importance . . . 5
plot.survlearning_reliability . . . 6
predict.survlearning_rsf . . . 6
predict.survlearning_svm . . . 8
print.survlearning_fit . . . 9
print.survlearning_reliability . . . 9
print.survlearning_tune . . . 10
summary.survlearning_fit . . . 10
Surv . . . 11
surv_ale . . . 12
surv_calibrate . . . 13
surv_censor_fit . . . 15
surv_censor_predict . . . 16
surv_compare . . . 16
surv_evaluate . . . 17
surv_fit . . . 19
surv_importance . . . 23
surv_methods . . . 24
surv_parameters . . . 24
surv_reliability . . . 25
surv_resamples . . . 26
surv_score . . . 27
surv_stack . . . 28

surv_tune 30
surv_validate 32

Index **35**

`coef.survlearning_fit` *Classical Coefficients and Covariance*

Description

Classical Coefficients and Covariance

Usage

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

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

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

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

Arguments

`object`	An ordinary Cox or parametric AFT fitted/tuned object.
`...`	Reserved; unused arguments are rejected.

Value

Named coefficients or their covariance matrix, on the original design scale. The AFT covariance excludes the ancillary log-scale parameter to match the returned regression coefficient vector.

`plot.survlearning_ale` *Plot Accumulated Local Effects*

Description

Plot Accumulated Local Effects

Usage

```
## S3 method for class 'survlearning_ale'
plot(x, ...)
```

Arguments

x	Result from [surv_ale()].
...	Arguments passed to [graphics::plot.default()].

Value

Invisibly returns the unchanged ‘survlearning_ale‘ data frame. The side effect is a step plot on the current graphics device.

---

plot.survlearning_calibration

Plot Grouped Survival Calibration

Description

Plot Grouped Survival Calibration

Usage

```
## S3 method for class 'survlearning_calibration'
plot(
  x,
  time = NULL,
  xlab = "Mean predicted survival",
  ylab = NULL,
  xlim = c(0, 1),
  ylim = c(0, 1),
  ...
)
```

Arguments

x	A result from [surv_calibrate()].
time	Optional single horizon to plot; NULL plots all horizons.
xlab, ylab	Axis labels.
xlim, ylim	Axis ranges, defaulting to ‘[0, 1]‘.
...	Additional arguments to [graphics::plot.default()].

Value

Invisibly, ‘x‘.

plot.survlearning_evaluation *Plot a Survival Evaluation Metric*

Description

Plot a Survival Evaluation Metric

Usage

```
## S3 method for class 'survlearning_evaluation'
plot(x, y = NULL, metric = "brier", ...)
```

Arguments

x	A table from [surv_evaluate()].
y	Unused, for S3 compatibility.
metric	One metric present in the evaluation.
...	Additional graphical arguments.

Value

Invisibly, ‘x‘.

plot.survlearning_importance *Plot Permutation Importance*

Description

Plot Permutation Importance

Usage

```
## S3 method for class 'survlearning_importance'
plot(x, ...)
```

Arguments

x	Result from [surv_importance()].
...	Arguments passed to [graphics::barplot()].

Value

Invisibly returns the unchanged ‘survlearning_importance‘ data frame. The side effect is a bar plot on the current graphics device.

`plot.survlearning_reliability` *Plot Conditional Reliability Curves*

Description

Plot Conditional Reliability Curves

Usage

```
## S3 method for class 'survlearning_reliability'
plot(
  x,
  rows = seq_len(min(5L, nrow(x$conditional_survival))),
  type = "survival",
  ...
)
```

Arguments

`x`	Result from [surv_reliability()].
`rows`	Observation indices to plot.
`type`	‘"survival"‘ or ‘"failure"‘.
`...`	Arguments passed to [graphics::matplot()].

Value

Invisibly returns the unchanged ‘survlearning_reliability‘ list. The side effect is a curve plot on the current graphics device.

`predict.survlearning_rsf` *Predict From a Survival Learning Model*

Description

Predict From a Survival Learning Model

Usage

```
## S3 method for class 'survlearning_rsf'
predict(object, newdata, ...)

surv_predict(object, newdata, type = "risk", times = NULL)

## S3 method for class 'survlearning_fit'
predict(object, newdata, type = "risk", times = NULL, ...)

## S3 method for class 'survlearning_tune'
predict(object, newdata, type = "risk", times = NULL, ...)
```

Arguments

object	A fitted object from [surv_fit()] or [surv_tune()].
newdata	A data frame of new predictors. Outcomes are not required.
...	Reserved for S3 compatibility; extra arguments are rejected.
type	‘"risk"‘ (default), ‘"survival"‘, or ‘"time"‘.
times	Non-negative finite time horizons, required for survival curves.

Details

Risk scores always increase with risk. AFT scores are negative log-time predictors; the original SVM scores are negative predicted responses. svm_rank returns its learned risk margin; svm_hybrid and svm_ipcw return negative normalized regression responses. These are risk-only capabilities. RSF mortality scores are not probabilities. Absolute scales must not be compared between methods. XGBoost Cox returns its log-risk margin, whereas XGBoost AFT returns the negative log-time margin. MTLR risk is the negative restricted mean survival up to the largest training follow-up, computed by trapezoidal integration on 101 equally spaced horizons plus the learned knots. This convention does not assume proportional hazards or extrapolate a full mean/median.

‘type = "time"‘ returns the conditional median for parametric AFT and XGBoost AFT, including the extreme-value error median correction. ‘type = "survival"‘ returns native model survival probabilities for AFT, Cox, penalized Cox, RSF, and MTLR. XGBoost Cox uses a training-only Breslow baseline; XGBoost AFT uses its fitted error distribution and scale. Other methods return an informative error; no distribution or calibration is silently added. Nonparametric and semiparametric curves return ‘NA‘ beyond the largest training follow-up. Parametric AFT and XGBoost AFT curves can extrapolate and require model adequacy checks.

Additional native curves are available for Kaplan-Meier, spline Cox, Royston-Parmar, componentwise CoxBoost, ranger, aorsf, neural Cox and PLANN. Royston-Parmar is parametric and can extrapolate; other new engines return NA beyond training follow-up. Kaplan-Meier returns identical curves for all rows and constant zero risk. PLANN survival is a step function on its training interval endpoints; its risk is negative restricted mean on that same grid. Neural Cox returns log hazard with a training-only baseline. Predictions are checked for finite values, valid probability bounds and nonincreasing survival over the requested supported horizons. An invalid curve raises an error rather than being silently clipped or repaired. Fits whose optimizer explicitly reported failure cannot be used for prediction.

Value

Risk and time predictions are numeric vectors. Survival predictions are matrices with one row per observation and one column per requested horizon, in the requested order. Matrix columns are named ‘t=<horizon>‘. Predictions use the outcome time unit supplied for training. Unsupported targets and non-finite log-risks cause errors. Cox survival at zero cumulative hazard is exactly one, including for extreme finite log-risk. Plotting and file output are not performed. Models do not imply a cure probability.

Examples

```
fit <- surv_fit(survival::Surv(time, status) ~ age + karno,
  survival::veteran, method = "aft")
surv_predict(fit, survival::veteran[1:3, ], type = "survival", times = c(30, 90))
```

---

predict.survlearning_svm

Predict Encoded Censored SVM Responses

Description

Predict Encoded Censored SVM Responses

Usage

```
## S3 method for class 'survlearning_svm'
predict(object, newdata, ...)
```

Arguments

`object`	A ‘survlearning_svm‘ engine obtained as the engine of a fitted SVM. Use [surv_predict()] for original, unencoded predictors.
`newdata`	Numeric matrix or data frame of finite encoded and standardized predictors, with the training column names and at least one row.
`...`	Reserved for S3 compatibility; extra arguments are rejected.

Value

A list with numeric vector ‘predicted‘, one value per row. These censored-regression scores are not calibrated survival times. Their negative is the package’s increasing-risk score. Missing/invalid columns cause errors. Failed engines and engines from the former conditioned QP require refitting.

print.survlearning_fit *Print a Survival Learning Fit*

Description

Print a Survival Learning Fit

Usage

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

Arguments

x	A ‘survlearning_fit‘ object from [surv_fit()].
...	Reserved for S3 compatibility.

Value

Invisibly returns ‘x‘ unchanged, after printing the method, engine, training counts, censoring percentage and prediction capabilities.

print.survlearning_reliability *Print Reliability Summaries*

Description

Print Reliability Summaries

Usage

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

Arguments

x	Result from [surv_reliability()].
...	Arguments passed to [print.data.frame()].

Value

Invisibly returns the unchanged ‘survlearning_reliability‘ list, after printing the attained time and the summary data frame.

print.survlearning_tune

Print a Survival Learning Search

Description

Print a Survival Learning Search

Usage

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

Arguments

x	A ‘survlearning_tune‘ object from [surv_tune()].
...	Reserved for S3 compatibility.

Value

Invisibly returns ‘x‘ unchanged, after printing search information and the selected parameter list to the console.

summary.survlearning_fit

Summarize a Survival Learning Fit or Search

Description

Summarize a Survival Learning Fit or Search

Usage

```
## S3 method for class 'survlearning_fit'
summary(object, conf_level = 0.95, ...)

## S3 method for class 'survlearning_tune'
summary(object, conf_level = 0.95, ...)

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

Arguments

`object`	A fitted or tuned survlearning object.
`conf_level`	Confidence level for classical fixed-model Wald intervals.
`...`	Reserved; unused arguments are rejected.
`x`	A summary returned by summary().

Details

Coefficients and covariance are exposed for ordinary Cox and parametric AFT only, with original design-column names. For Cox, the summary additionally reports hazard ratios. AFT coefficients are log-time effects, not log-hazards. Intervals condition on the fitted model and do not include hyperparameter selection or training-sample uncertainty from a full pipeline. No coefficient vector, p-value, or interval is invented for other learners.

Value

‘summary()‘ returns a ‘summary.survlearning_fit‘ list with training information, prediction capabilities, parameters, warnings, numerical diagnostics and, where appropriate, a classical coefficient table. Search summaries also retain the complete candidate table. ‘print()‘ invisibly returns this summary unchanged after console output.

`Surv` *Survival Response Constructor*

Description

Re-exported from survival for use in model formulas after loading only survlearning. This package currently accepts right-censored responses with positive times and event coding 0/1. See `survival::Surv()` for the complete constructor documentation.

Usage

```
Surv(
  time,
  time2,
  event,
  type = c("right", "left", "interval", "counting", "interval2"),
  origin = 0
)
```

Arguments

`time`	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.

time2	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether a transtion to another state occurred at the end of the interval.
event	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death), 1/2 (2=death), or a factor variable. For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. For multiple endpoint data the event variable will always be a factor, whose first level must correspond to "no observed transtition for this subject at this time point". There is no constraint on the labels of the factor; "none", "censored", and "alive" are common choices for the first level. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", "interval2". The default is multi-state if event is a factor, counting process if time2 is present, or right censored, in that order.
origin	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful and is depricated.

Value

An object of class Surv, implemented as a numeric matrix with one row per observation. A right-censored response has two columns, the observed time and event status, with a type = "right" attribute. Time units are those of the inputs. Other constructor types have the structure documented by survival, but are rejected by this package's modeling interface. The constructor does not fit a model or write files.

---

surv_ale *First-Order Accumulated Local Effects*

Description

First-Order Accumulated Local Effects

Usage

```
surv_ale(object, newdata, variable, bins = 10L, type = "risk", time = NULL)
```

Arguments

object	A fitted or tuned model.
newdata	Predictor data; outcomes are not needed.
variable	One original numeric predictor column.

bins	Requested number of empirical quantile intervals.
type	‘"risk"‘ or ‘"failure"‘. The latter requires native survival.
time	One supported horizon when type is ‘"failure"‘.

Details

Computes mean upper-minus-lower prediction differences inside each interval, accumulates them, and centers their upper-bound values by empirical interval frequencies. Repeated quantiles and empty intervals are merged. The returned curve is a binwise approximation; ‘ale‘ is evaluated at ‘upper‘. No smoothing, causal interpretation, or uncertainty interval is added. Risk effects are on the model-specific risk scale; failure effects are on 1-S(time).

Value

A ‘survlearning_ale‘ data.frame with interval endpoints, counts, local differences, and centered accumulated effects.

---

surv_calibrate *Survival Calibration Curves and IPCW Calibration Measures*

Description

Survival Calibration Curves and IPCW Calibration Measures

Usage

```
surv_calibrate(
  object,
  newdata,
  times,
  groups = 5L,
  censoring_model = NULL,
  min_g = 0.05,
  n_boot = 0L,
  conf_level = 0.95,
  seed = NULL,
  type = "groups"
)
```

Arguments

object	A fitted or tuned survlearning object with native probabilities.
newdata	Evaluation data including outcomes and predictors.
times	Distinct finite non-negative supported horizons.
groups	Requested number of quantile groups at each horizon.

`censoring_model`
: Optional training/reference fit from [surv_censor_fit()]. When supplied, use normalized IPCW case/control calibration instead of KM.

`min_g`
: Minimum censoring survival; smaller values cause errors, not clipping.

`n_boot`
: Assessment bootstrap repetitions for IPCW intervals; default 0 reports point estimates only. The event/G fits and original groups stay fixed.

`conf_level`
: Interval confidence level.

`seed`
: NULL (default) uses the caller's RNG; an integer >= 0 supplies a local bootstrap seed and restores the caller's RNG state.

`type`
: ‘"groups"‘ (default) retains grouped calibration; ‘"metrics"‘ estimates an IPCW logistic calibration intercept and slope at each horizon.

Details

With ‘type = "metrics"‘, the target is failure by the horizon. The intercept uses logit(predicted failure) as an offset with slope fixed at one. The slope comes from a separate joint intercept/slope fit. Their ideal values are zero and one, respectively. Cases receive 1/G(T-), controls 1/G(t), and earlier censorings zero weight. Default G uses training outcomes, never evaluation outcomes. An explicit conditional G is allowed. Quasibinomial IRLS gives the weighted logistic point estimates; model-based GLM standard errors are not used. Optional percentile intervals resample assessment units, keeping event/G fits fixed. They exclude training and selection uncertainty and are pointwise. Exact boundary probabilities, separation, constant scores and insufficient support are diagnosed, not silently clipped or replaced. This is evaluation, not model recalibration.

Groups are formed separately at each horizon using empirical midrank quantile bins in predicted survival. Identical probabilities are never split; fewer than ‘groups‘ groups can result. Without a censoring model, observed survival is a group-specific KM, with log-log Greenwood intervals at ‘conf_level‘ from [survival::survfit()]. No KM tail extrapolation is performed. Groups with fewer than 10 observations, fewer than 3 events by the horizon, or fewer than 5 at risk trigger a low support warning. Grouping is descriptive, not a test of calibration or a recalibration procedure; within-group censoring must be independent. Without censoring_model, covariate-dependent censoring is not adjusted. With a supplied model, observed survival is the normalized IPCW control mass divided by case plus control mass. Censoring assumptions, positive support and reference transportability are required. Bootstrap intervals exclude event/G fitting and group-selection uncertainty. Risk scores are not converted to fake probabilities. Training-data calibration is resubstitution.

Value

With ‘type = "metrics"‘, a data frame with metric, time, estimate, ideal, lower, upper, n, known outcomes, events by horizon, effective_n, bootstrap_valid, status and message. Undefined estimates/intervals are NA.

A ‘survlearning_calibration‘ data.frame with ‘time‘, ‘group‘, ‘predicted‘ (mean predicted survival), ‘observed‘ (KM survival without a censoring model, normalized IPCW survival otherwise), ‘lower‘, ‘upper‘, ‘n‘, ‘events‘ (events by the horizon), ‘n_risk‘ (just before the horizon), ‘status‘, and ‘message‘. Unsupported group tails are NA. The IPCW branch also returns ‘effective_n‘ (squared total weight divided by the sum of squared weights) and ‘bootstrap_valid‘ (number of usable bootstrap estimates). Its intervals are percentile bootstrap intervals at ‘conf_level‘, or NA when ‘n_boot = 0‘ or bootstrap support is inadequate.

Examples

```
fit <- surv_fit(Surv(time, status) ~ age + karno, survival::veteran)
calibration <- surv_calibrate(fit, survival::veteran, times = 30, groups = 3)
plot(calibration)
```

`surv_censor_fit` *Fit a Reference Censoring Distribution*

Description

Fit a Reference Censoring Distribution

Usage

```
surv_censor_fit(formula, data, method = "cox", impute = "none")
```

Arguments

`formula`	Original event-response formula Surv(time,event) ~ predictors. Event is still 1 for failure, not 1 for censoring.
`data`	Reference training data, not evaluation outcomes.
`method`	‘"km"‘, ‘"cox"‘, or ‘"stratified"‘. The last uses joint strata of the named original categorical predictors, without interactions or transformations in the formula.
`impute`	‘"none"‘ or training-only ‘"median"‘/mode imputation.

Details

KM and stratified KM remove failures before tied censorings, consistently with event = I(T <= C). Cox uses an exact ordering of two phases at each distinct observed time: original failures depart in phase one, censoring events occur in phase two. The phase index is an ordering device for the censoring partial likelihood, not jitter or a change to event-model times. Efron ties and the matching native Cox baseline are used within phase two. A conditional Cox G requires its PH/functional-form assumptions; it does not solve dependent censoring unexplained by the predictors or model a terminal administrative point mass exactly. Stratified KM avoids PH but needs support in every relevant stratum. Unknown strata are rejected.

Value

A serializable ‘survlearning_censoring‘ object. No neural runtime or external engine beyond survival is needed.

surv_censor_predict *Predict Reference Censoring Survival*

Description

Predict Reference Censoring Survival

Usage

```
surv_censor_predict(object, newdata, times, left = FALSE, pairwise = FALSE)
```

Arguments

object	A fit from [surv_censor_fit()].
newdata	Predictor data, with one row per unit.
times	Finite non-negative horizons; order and repetitions are kept.
left	Use G(t-) instead of G(t).
pairwise	If TRUE, one time per row is required and a vector is returned, avoiding a subject-by-subject time matrix.

Value

An n-by-times matrix, or an aligned vector when pairwise = TRUE. Horizons beyond reference/stratum follow-up are rejected.

surv_compare *Compare Models on the Same Held-Out Observations*

Description

Compare Models on the Same Held-Out Observations

Usage

```
surv_compare(
  models,
  newdata,
  ...,
  reference = NULL,
  n_boot = 0L,
  conf_level = 0.95,
  seed = NULL
)
```

Arguments

models	A named list of fitted or tuned models. Models must use the same response expression and must not have been selected using ‘newdata‘.
newdata	A common held-out data frame.
...	Evaluation arguments passed to [surv_evaluate()].
reference	Optional name in ‘models‘. If supplied, return each other model minus this reference instead of separate model evaluations.
n_boot	Number of paired test-case bootstrap repetitions. Use zero for point estimates or at least two; several hundred are usually needed.
conf_level	Confidence level for percentile intervals.
seed	Optional local bootstrap seed, restoring the caller’s RNG.

Details

Metrics are evaluated on the identical rows and requested horizons. Paired intervals resample independent assessment units with replacement, using the same row indices for every model and metric. Predictions and the censoring fit remain fixed. Intervals exclude training, tuning and G-fitting uncertainty and are pointwise, not simultaneous. No fold-level t test or p-value is returned. Do not pass pooled/repeated cross-validation rows or repeated observations of the same unit as independent assessment units. Positive differences favor the model for concordance/AUC; negative differences favor it for Brier/IBS. Undefined replicates are counted. IPCW comparisons require identical training outcomes or one explicit common censoring_data/censoring_model. Risk scales themselves are never compared.

Value

Without ‘reference‘, a data frame of model evaluations. With it, a data frame containing ‘model‘, ‘reference‘, ‘metric‘, ‘time‘, ‘estimate‘, ‘reference_estimate‘, ‘difference‘, ‘lower‘, ‘upper‘, ‘bootstrap_valid‘, ‘n‘, ‘events‘, ‘status‘, and ‘message‘. Limits concern the difference.

Examples

```
d <- survival::veteran
models <- list(cox = surv_fit(Surv(time, status) ~ age + karno, d),
  aft = surv_fit(Surv(time, status) ~ age + karno, d, "aft"))
surv_compare(models, d)
```

---

surv_evaluate *Censoring-Aware Evaluation of Survival Predictions*

Description

Censoring-Aware Evaluation of Survival Predictions

Usage

```
surv_evaluate(
  object,
  newdata,
  metrics = "harrell",
  times = NULL,
  tau = NULL,
  censoring_data = NULL,
  min_g = 0.05,
  n_boot = 0L,
  conf_level = 0.95,
  seed = NULL,
  censoring_model = NULL
)
```

Arguments

object	A fitted or tuned survlearning object.
newdata	Evaluation data including outcomes and predictors.
metrics	One or more of '"harrell"', '"uno"', '"auc"', '"brier"', '"ibs"', and '"ipcw_c"'. Higher concordance/AUC and lower Brier/IBS are better.
times	Finite non-negative horizons for AUC/Brier. IBS requires an explicit, strictly increasing grid of at least two supported horizons.
tau	Required finite positive truncation horizon for Uno or ipcw_c.
censoring_data	Optional reference data.frame. Only the outcome side of the fitted formula is evaluated, so predictors are unnecessary. Defaults to 'object$training_outcome'; evaluation outcomes are never substituted.
min_g	Minimum acceptable censoring survival, in '(0, 1]'. Unsupported horizons or smaller G values cause errors, not weight clipping.
n_boot	Number of conditional test-case bootstrap replicates; zero disables confidence intervals. The fitted model and censoring KM stay fixed.
conf_level	Confidence level for percentile bootstrap intervals.
seed	NULL (default) uses the caller's RNG for bootstrap sampling. An integer >= 0 supplies a local seed and restores the caller's RNG state.
censoring_model	Optional training/reference fit from [surv_censor_fit()]. Mutually exclusive with censoring_data. Conditional G is used for Brier, IBS, weighted-case-and-control AUC, and ipcw_c. Uno remains the historical marginal-G statistic and is rejected with a conditional censoring model.

Details

Only right-censoring is supported, with event = I(T <= C). The reference censoring distribution uses reverse Kaplan-Meier: at tied times, observed events depart before censoring, so the censoring hazard denominator is n_risk - d_event, not n_risk. Merely swapping the event indicator in an

ordinary KM is incorrect at event/censor ties. Event weights use the left limit G(T-) = P(C >= T); controls use G(t) = P(C > t), after censoring at t. These marginal IPCW estimates require independent censoring, also independent of predictors, and a transportable censoring distribution between reference and evaluation populations. A supplied conditional G instead requires conditional independent censoring given its predictors, a correctly specified censoring model, transportability, and adequate positive support.

Harrell uses [survival::concordance()] with its tie convention (including event/censor pairs tied in time) and numerical time-tie correction. Uno uses strictly ordered pairs T_i < T_j with event i at T_i <= tau, weighted by 1/G(T_i-)^2. Risk ties receive half credit. Uno requires a declared supported tau; full follow-up is never chosen automatically. Its rank-tree calculation takes O(n log n) time and O(n) memory. The separately named ipcw_c uses strictly ordered pairs with weight 1 / (G(T_i- | x_i) * G(T_i | x_j)). It is not labeled Uno under conditional censoring. Equal-time outcomes are excluded and risk ties receive half credit. Its time-group calculation avoids a dense pair matrix, but is slower than the marginal Uno rank tree.

Cumulative/dynamic AUC uses native risk scores (larger means earlier events), weighted cases with T <= t and an event, and controls with T > t. Case and control weights are 1/G(T-) and 1/G(t); risk ties receive half credit. It does not require survival probabilities and uses sorted controls, not a pair matrix. Brier requires native S(t) and is 'sum(I(T <= t, event) * S(t)^2 / G(T-) + I(T > t) * (1 - S(t))^2 / G(t)) / n'. Censored observations at or before t contribute zero but remain in n. IBS is the trapezoidal Brier integral over the supplied grid divided by its last minus first time. No implicit grid or zero-time interval is added.

Probability predictions are never synthesized from risk scores. Horizons cannot exceed reference or evaluation follow-up. Undefined metrics (e.g., AUC without cases/controls) return NA with diagnostics. Bootstrap intervals condition on the fitted model and estimated G, not training uncertainty. Undefined bootstrap replicates are counted and reported; intervals using fewer than all replicates have an explicit diagnostic status.

Value

A data.frame with class 'survlearning_evaluation', columns 'metric', 'time', 'estimate', 'lower', 'upper', 'n', 'events', 'status', 'message'. Harrell and IBS have NA time; Uno has time tau. Attributes 'censoring', 'settings', and 'assumptions' record G, its source, grids, and limitations.

Examples

```
fit <- surv_fit(Surv(time, status) ~ age + karno, survival::veteran)
surv_evaluate(fit, survival::veteran, c("harrell", "auc", "brier"), times = 30)
```

`surv_fit` *Fit a Survival Learning Model*

Description

All methods use the same formula and training-only predictor encoding. Missing values cause an error by default. Optional median/mode imputation is learned on training predictors only; outcomes are never imputed. 'svm' retains censored absolute-loss regression. Separate 'svm_rank', 'svm_hybrid', and 'svm_ipcw' methods have explicitly different objectives. None uses automatic subsampling.

Usage

```
surv_fit(
  formula,
  data,
  method = "cox",
  params = list(),
  seed = NULL,
  impute = "none"
)
```

Arguments

formula	A formula such as 'Surv(time, event) ~ temperature + age'. Events must be coded 1 and right-censoring 0; times must be positive.
data	A data frame of training observations.
method	One method listed by [surv_methods()].
params	A named list of method-specific parameters. See Details.
seed	NULL (default) uses the caller's RNG normally. An integer >= 0 supplies a local reproducible seed through withr, restoring the caller's RNG state even on error. Forest and XGBoost fits with NULL draw and store an engine seed so that later prediction does not reseed or advance the caller's RNG.
impute	'"none"' (default) or '"median"'. The latter replaces missing numeric predictors by training medians and categorical predictors by training modes, including when predicting or fitting validation folds.

Details

Parameters are: 'aft': 'distribution' (default '"weibull"', also '"lognormal"' or '"loglogistic"'); 'aft_lasso', 'aft_ridge': 'lambda' (0.1), 'max_iter' (10000), 'tolerance' (0.00025); 'cox': none; 'cox_lasso', 'cox_ridge': 'lambda' (0.01); 'tree': 'cp' (0.01), 'min_node' (10), 'max_depth' (5); 'rsf': 'trees' (200), 'mtry' (square root of the encoded predictor count), 'min_node' (15), 'splits' (5), 'threads' (1); 'boosting': 'trees' (300), 'depth' (2), 'learning_rate' (0.03), 'min_node' (10), 'bag_fraction' (0.7); 'svm': 'cost' (0.1), 'kernel' ('"linear"' or '"additive"'), 'solver' ('"clarabel"', default, or '"osqp"'), 'tolerance' (1e-12), 'max_iter' (100000), 'memory_limit_mb' (2048 MiB workspace planning limit, not measured peak memory), and 'prediction_tolerance' (1e-8). The exact censored absolute-loss primal uses all training rows. The additive representation retains every distinct training coordinate and fixes ranges in training. No nearPD projection, spectral truncation or subsampling is used. Solver success, primal/dual feasibility and an independent objective gap are checked before returning a model. Legacy Newton/quadprog fits targeted a different conditioned quadratic program and must be refitted. Predictions are placed on a training-fixed numerical grid of width prediction_tolerance times the maximum training follow-up, preventing solver-scale near ties from producing artificial risk ordering. Set this parameter to zero to inspect raw scores; it does not change fitting. A constant training score is reported explicitly in diagnostics and a warning. This can be a valid optimum of censored absolute loss, not a solver failure; it provides no learned risk discrimination. No alternative objective is selected automatically. Ranking and hybrid methods remain explicit choices. 'collapse_duplicates' is accepted for compatibility and has no effect: all observations have individual loss constraints. Cost must be tuned separately for each kernel,

without using test outcomes. SVM scores are not calibrated times or probabilities. See the SVM optimization vignette.

‘svm_rank‘, ‘svm_hybrid‘, and ‘svm_ipcw‘ use native R/C++ with no additional dependencies: ‘lambda‘ (0.01), ‘kernel‘ (‘"linear"‘ or ‘"additive"‘), ‘tolerance‘ (1e-6), ‘max_iter‘ (2000), ‘memory_limit_mb‘ (2048), and ‘prediction_tolerance‘ (1e-8 in dimensionless score units; zero disables it). These lambdas multiply half the squared RKHS norm; losses are averaged, so their scale is not the original svm cost. Ranking uses every pair with an observed earlier failure and strictly larger later follow-up, excluding tied times. A native Fenwick calculation evaluates the exact squared-hinge objective/gradient without storing all pairs. The pair evaluator centers scores internally to avoid cancellation of a common offset; this does not change the fitted objective or prediction orientation. Harrell evaluation also admits event/censoring ties in recorded time, unlike the strict ranking training pairs. The additive representation retains every distinct training coordinate. L-BFGS-B fits are checked for stationarity; ranking also reports a strong-convexity objective-gap bound. A line-search termination is accepted only when these same accuracy checks pass. It is labeled ‘stationarity_verified‘, with the original optimizer code and message retained; iteration-limit and other failures are rejected. Stationarity means the Euclidean gradient norm is at most tolerance times max(1, parameter norm). Ranking also requires its gap bound to be at most tolerance times max(1, absolute objective). Optimizer success is required. ‘svm_hybrid‘ adds ‘rank_weight‘ (0.5, strictly between zero and one), mixing mean pair loss with mean squared one-sided censored-regression loss. Regression times are divided by maximum training follow-up; risk is the negative normalized regression response. This is not the absolute-loss svm.

‘svm_ipcw‘ requires an explicit positive ‘horizon‘ in the outcome time unit. It fits the restricted outcome min(T,horizon)/horizon with squared epsilon-insensitive loss: ‘epsilon‘ (0), ‘censoring_method‘ (‘"km"‘, default, or ‘"cox"‘), and ‘min_g‘ (0.05). At epsilon zero it is weighted kernel ridge regression, not sparse epsilon-insensitive absolute-loss SVR. The censoring model is refitted inside each training split using the encoded predictors for Cox. Failures before the horizon and all rows followed to the horizon have observed restricted outcomes; earlier censorings inform G. Weights use G(min(Y,horizon)-). Insufficient fitted support raises an error, never clips weights or drops rows. KM needs marginal censoring assumptions; Cox needs a valid conditional PH model. Both require conditional independent censoring and positivity. A smooth Cox G does not repair deterministic administrative support failure. Empirical support is checked at the horizon for every training row as well as at each observed restricted response; finite fitted probabilities alone do not verify population positivity. At epsilon zero, the unpenalized unrestricted population target is the negative normalized restricted mean, but a penalized fitted score is not an unbiased mean estimate. All three methods return risk only, not calibrated survival probabilities or unrestricted lifetime estimates. See the survival SVM learning vignette for objectives, assumptions and examples.

‘aft_elasticnet‘ and ‘cox_elasticnet‘ add ‘alpha‘ (0.5) to their respective penalized AFT/Cox parameters: 1 is LASSO, 0 is ridge. ‘xgboost_cox‘ and ‘xgboost_aft‘ use ‘trees‘ (200), ‘depth‘ (2), ‘learning_rate‘ (0.05), ‘min_child_weight‘ (1), ‘leaf_penalty‘ (1), and ‘threads‘ (1). XGBoost AFT also accepts ‘distribution‘ (‘"normal"‘, ‘"logistic"‘, or ‘"extreme"‘) and ‘scale‘ (1), referring to the error distribution of log-time. ‘mtlr‘ accepts ‘intervals‘ (10), ‘lambda‘ (1, mapped to C1), ‘max_iter‘ (5000), and ‘tolerance‘ (1e-5).

Factors are encoded consistently with training. Numeric design columns are standardized using training means and standard deviations for SVM penalized AFT, MTLR, and Royston-Parmar. Other engines receive the unscaled design; glmnet performs its own training-only standardization. Penalized Cox follows a descending regularization path ending at the exact requested lambda, rather than fitting a weak-penalty cold start in isolation.

Penalized AFT uses the Gehan loss and an integrated ADMM solver adapted from penAFT. It returns a relative risk ordering, not absolute survival times. Cox boosting means gradient-boosted trees with Cox loss (gbm), not the componentwise algorithm in the CoxBoost package. Dependencies are loaded only for the selected engine.

Reference parameters: ‘km‘ has no parameters and ignores the formula’s predictors; ‘cox_spline‘ uses ‘df‘ (3) for natural splines of continuous design columns; ‘flexsurv‘ uses ‘knots‘ (1) and ‘scale‘ (‘"hazard"‘, ‘"odds"‘ or ‘"normal"‘); ‘coxboost‘ uses ‘steps‘ (100) and ‘penalty‘ (100). Royston-Parmar stores its optimizer coefficients in training-standardized coordinates together with their transformation. CoxBoost uses a zero-effect auxiliary column for single-predictor designs to initialize the upstream scalar penalty correctly; the fitted method and observations are unchanged. ‘ranger‘ uses ‘trees‘ (200), ‘mtry‘ (square root of predictor count), ‘min_node‘ (15) and ‘threads‘ (1). ‘aorsf‘ uses the same forest arguments with ‘min_node‘ (5) and ‘splits‘ (5), specifying oblique rather than axis aligned splits. See [surv_parameters()] for inspectable defaults.

‘neural_cox‘ uses ‘hidden‘ (16), ‘epochs‘ (100), ‘learning_rate‘ (0.001), ‘decay‘ (0.0001), ‘ties‘ (‘"efron"‘, also ‘"breslow"‘), and ‘max_weights‘ (100000). It uses native R matrix operations and analytic backpropagation, with full-batch Adam and L2 penalty decay/2 times the squared weight norm. No Torch, Python, external runtime or separate neural engine is needed. All observations at an event time enter the risk set before removing deaths or censorings; only event mass enters the Efron correction. The baseline uses the same ties method as training. ‘threads‘ must remain 1; this setting does not control the R installation’s BLAS threads. Fixed epochs are not a convergence certificate; final objective/gradient diagnostics are reported. ‘neural_discrete‘ uses PLANN through nnet with ‘hidden‘ (8), ‘intervals‘ (10), ‘decay‘ (0.01), ‘max_iter‘ (1000), and ‘max_weights‘ (10000). Its risk is negative restricted mean on the training interval grid. No test-based early stopping is performed and no external runtime is installed automatically. Censoring contributes only completed intervals; events contribute through their failure interval. This explicit discretization loses partial-interval information, rather than inventing survival until the next grid endpoint. XGBoost Cox uses signed time labels for censoring and an explicitly estimated training-only Breslow baseline. XGBoost AFT uses lower/upper censoring bounds, not ordinary regression on censored times. Its initial log-location is the training mean log-follow-up, so initialization respects changes in the outcome time unit. This is a starting value, not an estimate of mean lifetime. MTLR knots and scaling are learned within training, including each tuning fold.

Value

A ‘survlearning_fit‘ object with engine, parameters, design, and training metadata. Access the engine via ‘$engine‘. Neural adapters store serializable R weights alongside fitted components. Training outcomes are retained for IPCW evaluation; [surv_diagnostics()] exposes fit provenance. Components ‘n‘, ‘events‘, ‘max_time‘, ‘seed‘, and ‘fit_seconds‘ describe the training rows, observed failures, largest follow-up (input time unit), seed used, and elapsed fitting seconds. ‘diagnostics‘ records warnings and the engine’s reported convergence, not a universal convergence certificate. Kaplan-Meier accepts fully censored training outcomes and returns survival one within observed support; regression engines require an observed failure. Invalid inputs or unavailable required dependencies cause informative errors. Unpenalized AFT rejects rank-deficient designs and reported nonconvergence, rather than returning unidentified coefficients. Royston-Parmar checks the baseline spline’s slope throughout its domain, including between knots. An invalid fitted distribution is rejected; curves are never repaired by clipping or by silently reducing the number of knots. No files are written and no dependencies are installed automatically.

Examples

```
fit <- surv_fit(survival::Surv(time, status) ~ age + karno,
  survival::veteran, method = "aft")
surv_predict(fit, survival::veteran[1:3, ], type = "time")
```

`surv_importance` *Permutation Importance Using Harrell Concordance*

Description

Permutation Importance Using Harrell Concordance

Usage

```
surv_importance(
  object,
  newdata,
  variables = NULL,
  n_repeats = 10L,
  seed = NULL
)
```

Arguments

`object`	A fitted or tuned model.
`newdata`	Assessment data including outcomes and original predictors.
`variables`	Predictor names, or a named list of groups of predictor names permuted together. Defaults to the original formula predictors.
`n_repeats`	Number of independent permutations per variable/group.
`seed`	NULL (default) uses the caller's RNG. An integer >= 0 supplies a local seed and restores the caller's RNG state.

Details

Importance is original Harrell C minus C after permuting the selected raw columns using a common row permutation within each group. Formula-based transformations are recomputed by prediction. No models are refitted and no rows are sampled away. The same permutations are used across groups. Negative importance is retained. Correlated predictors complicate marginal permutation importance; this is not conditional importance or causality. Use independent assessments or outer-fold assessments, not the final test set repeatedly for feature selection. SD describes permutation variability, not model-fitting uncertainty or a confidence interval.

Value

A data.frame with class 'survlearning_importance'. Attributes retain individual permutation results, groups, baseline concordance and settings.

surv_methods *Available Survival Learning Methods*

Description

List engines, prediction capabilities, and installed dependency availability. A risk score always increases with risk; it is not a failure probability. Penalized AFT methods use the semiparametric Gehan estimator, not a penalized parametric likelihood. The ‘installed‘ column describes the default configuration. Native SVM kernels need no external SVM engine. The original regression solver uses Clarabel or OSQP; ranking, hybrid and IPCW use native R/C++ and stats. All supported engine and solver packages are required installation dependencies.

Usage

```
surv_methods()
```

Value

A data frame with one row per supported method.

Examples

```
surv_methods()
```

surv_parameters *Inspect Parameters and Fit Diagnostics*

Description

Inspect Parameters and Fit Diagnostics

Usage

```
surv_parameters(method)

surv_diagnostics(object)
```

Arguments

method	A method listed by [surv_methods()].
object	A fitted or tuned survlearning object.

Value

‘surv_parameters()‘ returns the named default parameter list. ‘surv_diagnostics()‘ returns provenance, warnings, and the convergence status actually reported by the engine, not a universal convergence test.

Examples

```
surv_parameters("aft")
```

`surv_reliability` *Reliability Summaries Conditional on Survival to an Attained Time*

Description

Reliability Summaries Conditional on Survival to an Attained Time

Usage

```
surv_reliability(
  object,
  newdata,
  horizons,
  age = 0,
  probs = c(0.1, 0.5),
  grid_points = 201L,
  extrapolate = FALSE
)
```

Arguments

`object`	A fitted or tuned model providing native survival curves.
`newdata`	Original predictors for the units of interest.
`horizons`	Distinct positive future durations, in the model's time unit.
`age`	One non-negative attained time on the model's time scale.
`probs`	Distinct probabilities in (0,1) for remaining-time quantiles.
`grid_points`	Number of equally spaced integration/quantile grid points. Requested horizons and zero are also included.
`extrapolate`	Allow parametric tail extrapolation explicitly. Default FALSE requires age + max(horizons) within training follow-up.

Details

With fixed input covariates, conditional survival is S(age+u|x)/S(age|x). This conditions on survival, not on newly observed longitudinal predictors. The time origin remains the model's origin, not necessarily installation. Failure probability is one minus conditional survival. The restricted mean remaining time integrates this curve from zero to each supplied horizon by the trapezoidal rule; it is not an unrestricted life expectancy. Quantiles are the first grid points reaching their CDF probability within the largest horizon. NA means not reached on this grid, not infinite life, cure, or a prediction interval. Accuracy depends on grid resolution. No probabilities are constructed for risk-only models. Extrapolation is available only for aft, xgboost_aft and flexsurv, and requires model adequacy.

Value

A ‘survlearning_reliability‘ list with ‘summary‘, ‘quantiles‘, ‘times‘, ‘conditional_survival‘, and ‘settings‘.

`surv_resamples` *Explicit, Reusable Survival Resampling Plans*

Description

Explicit, Reusable Survival Resampling Plans

Usage

```
surv_resamples(
  formula,
  data,
  folds = 5L,
  repeats = 1L,
  type = "stratified",
  group = NULL,
  time = NULL,
  seed = NULL
)
```

Arguments

`formula`	Two-sided formula with a right-censored survival response.
`data`	Data frame. Only outcomes are inspected, allowing missing predictors.
`folds`	Number of assessment folds, at least two.
`repeats`	Number of repetitions, at least one.
`type`	One of ‘"stratified"‘, ‘"grouped"‘, or ‘"temporal"‘.
`group`	Column name or row-aligned group vector for grouped splitting.
`time`	Column name or row-aligned numeric, Date, or POSIXct vector giving observation order, not the survival response, for temporal splitting.
`seed`	NULL (default) uses the caller's RNG. An integer >= 0 supplies a local seed and restores the caller's RNG state.

Details

Stratification allocates events and censored observations separately. Grouped folds balance group sizes without dividing groups. Temporal plans partition ordered unique times into ‘folds + 1‘ blocks: block one is initial training; each subsequent block is assessed using all preceding blocks. Time ties are never divided. Temporal plans require ‘repeats = 1‘: deterministic duplicate splits must not be counted as additional validation replicates. Chronological row order alone does not establish when labels or predictors became available.

Splits contain integer ‘analysis‘ and ‘assessment‘ row indices. The plan stores ‘data.n‘, formula, outcomes, row names, structural vectors, and sample/event diagnostics. Consumers recheck signatures and disjointness. The signature detects changed/reordered outcomes and row names, not arbitrary predictor changes. Preserve row order. No-event splits are retained, not redrawn.

Value

A ‘survlearning_resamples‘ list with ‘splits‘, ‘diagnostics‘, and metadata. ‘fold_id‘ is available for a single complete CV partition.

Examples

```
plan <- surv_resamples(survival::Surv(time, status) ~ age + karno,
  survival::veteran, folds = 3)
plan$diagnostics
```

---

surv_score *Evaluate Harrell's Concordance Index*

Description

Evaluate Harrell's Concordance Index

Usage

```
surv_score(object, newdata)
```

Arguments

object	A fitted or tuned survlearning object.
newdata	Evaluation data including the outcome and predictors.

Value

A single numeric C-index. Larger risk scores always mean earlier events. This is discrimination, not probability calibration or an event detection rate. Training-data scores are resubstitution estimates. An error is raised for non-finite scores or when no comparable pair exists, including a completely censored assessment sample.

Examples

```
fit <- surv_fit(survival::Surv(time, status) ~ age + karno,
  survival::veteran, method = "cox")
surv_score(fit, survival::veteran)
```

surv_stack *Combine Native Survival Curves Using Out-of-Fold IPCW Brier Loss*

Description

Combine Native Survival Curves Using Out-of-Fold IPCW Brier Loss

Usage

```
surv_stack(
  formula,
  data,
  methods = c("cox", "rsf"),
  times,
  grids = NULL,
  params = NULL,
  folds = 5L,
  inner_folds = 3L,
  resamples = NULL,
  seed = NULL,
  impute = "none",
  censoring = NULL,
  min_g = 0.05
)

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

Arguments

formula	Two-sided formula with a right-censored response.
data	Training data, not the final assessment sample.
methods	Distinct method identifiers providing native survival curves.
times	Explicit positive increasing horizons for learning the weights.
grids	Optional named tuning grids. Each supplied grid is selected within each weight-training fold by inner Harrell-C validation. No grid is invented for methods without one; their parameters stay fixed.
params	Named list of fixed parameter lists, one per method.
folds	Number of out-of-fold partitions for learning the weights.
inner_folds	Number of inner folds for supplied tuning grids.
resamples	Optional single-repeat stratified plan from [surv_resamples()].
seed	Optional local seed, restoring the caller's RNG state.
impute	‘"none"‘ or training-fold median/mode imputation.

censoring	Formula/list for G, refitted in every training fold. NULL uses marginal reverse KM. Global fitted G objects are not accepted.
min_g	Smallest supported G; weights are never clipped.
x	A fitted stacking object.
...	Reserved; unused arguments are rejected.

Details

Each training unit receives predictions from models that did not use its outcomes or predictors for fitting, preprocessing or tuning. One nonnegative weight per method, summing to one and constant over time, minimizes the mean IPCW Brier loss over the supplied horizons. Earlier censorings have zero outcome-loss weight but remain in the denominator and contribute to training and G estimation. All rows are retained. Clarabel, already required by survlearning, solves the small convex QP. Feasibility and a simplex first-order objective-gap bound are verified. Tiny solver feasibility errors are projected onto the simplex before checking the bound. Duplicate curves may have nonunique weights.

Models are then refitted on all training data. The OOF loss used to choose weights is a training criterion, not an unbiased performance estimate of the stack. Evaluate on an independent test set or nest the whole call in outer validation. Base grids and censoring estimation remain inside each training fold. Unsupported horizons, failed fits or risk-only methods stop the fit rather than dropping observations or learners.

Survival predictions use the same convex weights at all times, preserving probability bounds and monotonicity. Risk is negative restricted mean, approximated by trapezoidal integration over zero plus ‘times‘. It is not a Cox log hazard or an unrestricted mean. No median, extrapolation, Python or Torch is added. Mixing uncalibrated risk scores is not supported.

Value

A ‘survlearning_stack‘ inheriting from ‘survlearning_fit‘, usable by [surv_predict()], [surv_score()], [surv_evaluate()], [surv_compare()], [surv_calibrate()] and [surv_reliability()]. Includes named ‘weights‘, refitted ‘models‘, OOF curves/weights, resamples, fold tuning diagnostics, training loss and numerical checks. Weights are not variable importance.

Examples

```
d <- survival::veteran
stack <- surv_stack(Surv(time, status) ~ age + karno, d,
  methods = c("cox", "aft"), times = c(30, 60), folds = 3, seed = 11)
stack$weights
surv_predict(stack, d[1:2, ], "survival", times = c(30, 60))
```

surv_tune *Tune a Survival Learning Model With a Common Validation Protocol*

Description

Tune a Survival Learning Model With a Common Validation Protocol

Usage

```
surv_tune(
  formula,
  data,
  method = "cox",
  grid = NULL,
  params = list(),
  folds = 5L,
  seed = NULL,
  resamples = NULL,
  metric = "harrell",
  times = NULL,
  tau = NULL,
  impute = "none",
  cache = TRUE,
  censoring = NULL,
  workers = 1L,
  checkpoint = NULL,
  resume = TRUE,
  retry_failed = FALSE,
  progress = FALSE
)
```

Arguments

formula	A formula such as 'Surv(time, event) ~ temperature + age'. Events must be coded 1 and right-censoring 0; times must be positive.
data	A data frame of training observations.
method	One method listed by [surv_methods()].
grid	Data frame of candidates or named list expanded as a Cartesian grid. No automatic search range is imposed. NULL means one fixed setting.
params	A named list of method-specific parameters. See Details.
folds	Number of event-stratified validation folds (default 5).
seed	NULL (default) uses the caller's RNG normally. An integer >= 0 supplies a local reproducible seed through withr, restoring the caller's RNG state even on error. Forest and XGBoost fits with NULL draw and store an engine seed so that later prediction does not reseed or advance the caller's RNG.

resamples	Reusable plan from [surv_resamples()]. When NULL, historical event-stratified allocation and minimum event counts apply.
metric	Scalar tuning objective. Non-Harrell metrics are delegated to [surv_evaluate()]. Brier and integrated Brier (‘"ibs"‘) are minimized; concordance and AUC are maximized.
times	Evaluation horizons. Time-specific tuning requires one horizon.
tau	Truncation horizon passed to [surv_evaluate()].
impute	‘"none"‘ or ‘"median"‘, learned within every training fit.
cache	Reuse SVM preprocessing, sparse penalties and constraint matrices within each analysis split and kernel (default TRUE). Never shared across splits; at most one kernel workspace is retained. FALSE provides a cold reference path. Other methods are unaffected.
censoring	NULL for marginal IPCW, or a formula/list specification refitted by [surv_censor_fit()] on each analysis fold. Never a global fit.
workers	Number of local PSOCK workers. Default 1 is serial. Work is grouped by analysis fold, retaining within-fold SVM matrix reuse.
checkpoint	NULL, or a directory owned by this search. Individual candidate-fold records are committed atomically; no fold models are stored.
resume	Reuse compatible completed records (default TRUE).
retry_failed	Recompute failed checkpoint records (default FALSE).
progress	Report completion after each serial fold or parallel wave.

Details

All candidates share assessment indices. Any failed or undefined split makes a candidate ineligible. Errors and warnings remain in ‘fold_results‘; warnings, including convergence warnings, are not suppressed. Ties select the first grid row. Explicit engine ‘reported_failure‘ diagnostics invalidate a fit; warnings alone and ‘not_reported‘ convergence do not. The winner is refitted on the supplied data. This selection criterion is not an unbiased estimate of the search procedure; use [surv_validate()] or an external test set for final evaluation.

Historical ‘fold_id‘, ‘cindex‘, ‘mean_cindex‘, and ‘sd_cindex‘ fields remain for default Harrell tuning. For other metrics concordance fields are NA; use ‘estimate‘, ‘mean_estimate‘, and ‘sd_estimate‘. With repeated or temporal plans ‘fold_id‘ is NULL; use explicit indices. All-candidate failure raises a ‘survlearning_tune_error‘ retaining ‘results‘, ‘fold_results‘, and ‘resamples‘. Parallel/checkpoint formulas support standard base/stats/survival/splines functions and data columns, not custom closures or captured constants. Checkpoints match all data/predictors/order, formulas, folds, settings and recorded R/platform/engine versions. Workers and progress can change when resuming. Use only one live search per directory. Ordinary interrupts/errors release the lock; after a hard process crash verify no run is active before manually removing the reported lock directory. Incomplete/corrupt records cannot silently win. Warnings/errors are retained. A completed winning model is also cached. Resuming missing fits rebuilds numerical workspaces; no solver warm-start history is persisted, so numerical tolerances still apply. Parallelism duplicates data/workspaces across workers. Avoid nested engine threads, and choose workers according to available memory. No universal speedup or automatic memory cap is promised. The same per-fit seeds and stable result ordering are used across schedules. With seed = NULL a search seed is drawn once from the caller's RNG and stored in the result. Checkpoint searches require an explicit seed so that their identity is stable on resume.

Value

A ‘survlearning_tune‘ with historical ‘best_model‘, ‘best_params‘, ‘results‘, ‘fold_results‘, ‘grid‘, ‘fold_id‘, ‘seed‘, and ‘best_candidate‘, plus ‘resamples‘, ‘metric‘, warnings and failure diagnostics.

Examples

```
tuned <- surv_tune(survival::Surv(time, status) ~ age + karno,
  survival::veteran, method = "aft",
  grid = list(distribution = c("weibull", "lognormal")), folds = 3)
tuned$results
```

---

surv_validate *Nested, Paired Validation of Survival Learning Methods*

Description

Nested, Paired Validation of Survival Learning Methods

Usage

```
surv_validate(
  formula,
  data,
  methods = c("cox", "aft"),
  grids = NULL,
  params = NULL,
  resamples = NULL,
  folds = 5L,
  repeats = 1L,
  inner_folds = 3L,
  tune = TRUE,
  metrics = "harrell",
  times = NULL,
  tau = NULL,
  seed = NULL,
  impute = "none",
  censoring = NULL,
  metric = "harrell",
  selection_time = NULL
)
```

Arguments

formula	Two-sided formula with a right-censored survival response.
data	Data frame. Only outcomes are inspected, allowing missing predictors.
methods	Character vector of method identifiers.

grids	Named list of explicit grids, one per method. Missing entries use one fixed configuration; no search ranges are invented.
params	Named list of method-specific parameter lists.
resamples	Explicit outer plan from [surv_resamples()].
folds	Number of assessment folds, at least two.
repeats	Number of repetitions, at least one.
inner_folds	Number of inner folds when ‘tune = TRUE‘.
tune	Whether to tune on each outer analysis sample only.
metrics	Metrics reported on outer assessments. Their order never changes parameter selection.
times	Evaluation horizons passed to [surv_evaluate()].
tau	Truncation horizon passed to [surv_evaluate()].
seed	NULL (default) uses the caller’s RNG. An integer >= 0 supplies a local seed and restores the caller’s RNG state.
impute	‘"none"‘ or ‘"median"‘; learned separately within each training fit.
censoring	A formula/list censoring specification refitted on each analysis sample, including each inner fold. NULL retains marginal IPCW.
metric	Scalar inner selection objective (default "harrell"), independent of metrics. For AUC/Brier selection provide one selection_time when outer evaluation uses several times.
selection_time	Optional single horizon for AUC/Brier tuning only.

Details

All methods share outer and inner splits. Outer assessments never enter parameter selection or preprocessing. Nested tuning with grouped or temporal outer plans is explicitly rejected. Use ‘tune = FALSE‘ with prespecified parameters for these plans; ordinary inner CV would violate their structure. Explicit optimizer ‘reported_failure‘ diagnostics invalidate a fit. Warnings alone do not; ‘not_reported‘ is not interpreted as confirmed convergence.

Undefined and unsupported evaluations remain NA with diagnostics. An analysis sample without events cannot be fitted; an assessment without events can still have meaningful Brier scores. Warnings remain visible and are recorded. Summary ‘mean‘/‘sd‘ use each method’s finite splits; ‘mean_common‘/‘sd_common‘ use the intersection across all methods. Do not compare available-case means as if they used identical samples. Pairwise summaries use each pair’s own intersection and describe method-one minus method-two. All SDs are descriptive; no standard errors or confidence intervals are invented from correlated folds.

Value

A ‘survlearning_validate‘ list with ‘results‘, ‘summary‘, ‘paired_summary‘, ‘warnings‘, ‘failures‘, ‘resamples‘, ‘inner_resamples‘, and ‘tuning‘ diagnostics. No final deployment model is fitted.

Examples

```
validation <- surv_validate(survival::Surv(time, status) ~ age + karno,
  survival::veteran, folds = 3, inner_folds = 2)
validation$summary
```

Index

coef.survlearning_fit, 3
coef.survlearning_tune
        *(*coef.survlearning_fit*)*, 3

plot.survlearning_ale, 3
plot.survlearning_calibration, 4
plot.survlearning_evaluation, 5
plot.survlearning_importance, 5
plot.survlearning_reliability, 6
predict.survlearning_fit
        *(*predict.survlearning_rsf*)*, 6
predict.survlearning_rsf, 6
predict.survlearning_svm, 8
predict.survlearning_tune
        *(*predict.survlearning_rsf*)*, 6
print.summary.survlearning_fit
        *(*summary.survlearning_fit*)*, 10
print.survlearning_fit, 9
print.survlearning_reliability, 9
print.survlearning_stack *(*surv_stack*)*,
        28
print.survlearning_tune, 10

summary.survlearning_fit, 10
summary.survlearning_tune
        *(*summary.survlearning_fit*)*, 10
Surv, 11
surv_ale, 12
surv_calibrate, 13
surv_censor_fit, 15
surv_censor_predict, 16
surv_compare, 16
surv_diagnostics *(*surv_parameters*)*, 24
surv_evaluate, 17
surv_fit, 19
surv_importance, 23
surv_methods, 24
surv_parameters, 24
surv_predict
        *(*predict.survlearning_rsf*)*, 6
surv_reliability, 25
surv_resamples, 26
surv_score, 27
surv_stack, 28
surv_tune, 30
surv_validate, 32
survival::Surv(), *11*

vcov.survlearning_fit
        *(*coef.survlearning_fit*)*, 3
vcov.survlearning_tune
        *(*coef.survlearning_fit*)*, 3