

Package ‘SuperSurv’

March 26, 2026

Type Package

Title A Unified Framework for Machine Learning Ensembles in Survival Analysis

Version 0.1.1

Description Implements a Super Learner framework for right-censored survival data. The package fits convex combinations of parametric, semiparametric, and machine learning survival learners by minimizing cross-validated risk using inverse probability of censoring weighting (IPCW). It provides tools for automated hyperparameter grid search, high-dimensional variable screening, and evaluation of prediction performance using metrics such as the Brier score, Uno's C-index, and time-dependent area under the curve (AUC). Additional utilities support model interpretation for survival ensembles, including Shapley additive explanations (SHAP), and estimation of covariate-adjusted restricted mean survival time (RMST) contrasts. The methodology is related to treatment-specific survival curve estimation using machine learning described by Westling, Luedtke, Gilbert and Carone (2024) <[doi:10.1080/01621459.2023.2205060](https://doi.org/10.1080/01621459.2023.2205060)>, and the unified ensemble framework described in Lyu et al. (2026) <[doi:10.64898/2026.03.11.711010](https://doi.org/10.64898/2026.03.11.711010)>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.0

Depends R (>= 4.0.0)

LazyData true

Imports survival, nnls, future.apply, stats, dplyr, magrittr

URL <https://github.com/yuelyu21/SuperSurv>,
<https://yuelyu21.github.io/SuperSurv/>

BugReports <https://github.com/yuelyu21/SuperSurv/issues>

Suggests aorsf, BART, CoxBoost, glmnet, gbm, mgcv, randomForestSRC,
ranger, rpart, survivalsvm, xgboost, fastshap, survex, ggplot2,
tidyr, quadprog, ggforce, patchwork, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Yue Lyu [aut, cre]

Maintainer Yue Lyu <yuelyu0521@gmail.com>

Repository CRAN

Date/Publication 2026-03-26 10:00:27 UTC

Contents

estimate_marginal_rmst	3
eval_brier	4
eval_cindex	5
eval_summary	6
eval_timeROC	8
explain_kernel	9
explain_survex	10
list_wrappers	11
metabric	12
plot_beeswarm	13
plot_benchmark	14
plot_calibration	15
plot_dependence	16
plot_global_importance	17
plot_marginal_rmst_curve	19
plot_patient_waterfall	20
plot_predict	21
plot_rmst_vs_obs	22
plot_survival_heatmap	23
predict.SuperSurv	24
predict.surv.aorsf	25
predict.surv.bart	26
predict.surv.coxboost	27
predict.surv.coxph	28
predict.surv.gam	29
predict.surv.gbm	30
predict.surv.glmnet	31
predict.surv.km	32
predict.surv.parametric	33
predict.surv.ranger	34
predict.surv.rfsrc	35
predict.surv.ridge	36
predict.surv.rpart	37
predict.surv.svm	39
predict.surv.xgboost	40
screen.all	41
screen.elasticnet	41
screen.glmnet	43
screen.marg	44
screen.rfsrc	45

screen.var	46
SuperSurv	47
surv.aorsf	49
surv.bart	51
surv.coxboost	52
surv.coxph	54
surv.exponential	55
surv.gam	56
surv.gbm	57
surv.glmnet	59
surv.km	60
surv.loglogistic	61
surv.lognormal	62
surv.parametric	63
surv.ranger	65
surv.rfsrc	66
surv.ridge	68
surv.rpart	69
surv.svm	71
surv.weibull	72
surv.xgboost	73
Index	76

estimate_marginal_rmst

Estimate Causal Restricted Mean Survival Time (RMST)

Description

Calculates the causal treatment effect based on the difference in Restricted Mean Survival Time (RMST) between treatment and control groups up to a specific truncation time.

Usage

```
estimate_marginal_rmst(fit, data, trt_col, times, tau, verbose = FALSE)
```

Arguments

fit	A fitted SuperSurv ensemble object.
data	A data.frame containing the patient covariates and the treatment assignment.
trt_col	Character string. The exact name of the binary treatment indicator column in data (e.g., "treatment").
times	Numeric vector of time points matching the prediction grid.
tau	Numeric. A single truncation time limit up to which the RMST will be calculated.
verbose	Logical; if TRUE, progress messages are shown.

Value

A numeric value representing the estimated causal RMST difference (Treatment - Control).

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:80, ]
x_cols <- grep("^x", names(dat))[1:5]
X <- dat[, x_cols, drop = FALSE]
new.times <- seq(20, 120, by = 20)

fit <- SuperSurv(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = X,
  new.times = new.times,
  event.library = c("surv.coxph", "surv.glmnet"),
  cens.library = c("surv.coxph"),
  control = list(saveFitLibrary = TRUE)
)

rmst_res <- estimate_marginal_rmst(
  fit = fit,
  data = dat,
  trt_col = "x4",
  times = new.times,
  tau = 100
)
rmst_res$ATE_RMST
```

 eval_brier

IPCW Brier Score and Integrated Brier Score (IBS)

Description

Calculates the Inverse Probability of Censoring Weighted (IPCW) Brier Score over a grid of times, and computes the Integrated Brier Score (IBS) using trapezoidal integration.

Usage

```
eval_brier(time, event, S_mat, times, tmin = min(times), tmax = max(times))
```

Arguments

time	Numeric vector of observed follow-up times.
event	Numeric vector of event indicators (1 = event, 0 = censored).
S_mat	A numeric matrix of predicted survival probabilities (rows = observations, columns = time points).

times Numeric vector of evaluation times matching the columns of `S_mat`.
tmin Numeric. Lower bound for IBS integration. Defaults to `min(times)`.
tmax Numeric. Upper bound for IBS integration. Defaults to `max(times)`.

Value

A list containing:

- `brier_scores`: A numeric vector of Brier scores at each time point.
- `ibs`: The Integrated Brier Score over the range `{tmin, tmax}`.
- `times`: The time grid used.

Examples

```

data("metabric", package = "SuperSurv")
dat <- metabric[1:40, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:10, , drop = FALSE]
times <- seq(50, 150, by = 50)

```

```

fit <- surv.coxph(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
  obsWeights = rep(1, nrow(dat)),
  id = NULL
)

```

```

eval_brier(
  time = dat$duration[1:10],
  event = dat$event[1:10],
  S_mat = fit$pred,
  times = times
)

```

eval_cindex

Calculate Concordance Index (Harrell's or Uno's)

Description

Calculate Concordance Index (Harrell's or Uno's)

Usage

```
eval_cindex(time, event, S_mat, times, eval_time, method = "uno")
```

Arguments

time	Numeric vector of observed follow-up times.
event	Numeric vector of event indicators (1 = event, 0 = censored).
S_mat	A numeric matrix of predicted survival probabilities.
times	Numeric vector of evaluation times matching the columns of S_mat.
eval_time	Numeric. The specific time point at which to extract predictions.
method	Character. Either "harrell" or "uno". Defaults to "uno".

Value

A numeric value representing the chosen C-index.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:40, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:10, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.coxph(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
  obsWeights = rep(1, nrow(dat)),
  id = NULL
)

eval_cindex(
  time = dat$duration[1:10],
  event = dat$event[1:10],
  S_mat = fit$pred,
  times = times,
  eval_time = 100,
  method = "uno"
)
```

eval_summary

Evaluate SuperSurv predictions on test data

Description

Computes the integrated Brier score (IBS), Uno C-index, and integrated area under the curve (iAUC) for the SuperSurv ensemble and all individual base learners.

Usage

```
eval_summary(
  object,
  newdata,
  time,
  event,
  eval_times,
  risk_time = stats::median(eval_times),
  verbose = FALSE
)
```

Arguments

object	A fitted SuperSurv object.
newdata	A data.frame of test covariates.
time	Numeric vector of observed follow-up times for the test set.
event	Numeric vector of event indicators for the test set.
eval_times	Numeric vector of times at which to evaluate survival predictions.
risk_time	Numeric. The specific time horizon used when extracting risk scores for Uno C-index. Defaults to the median of eval_times.
verbose	Logical; if TRUE, progress messages are shown.

Value

An object of class "SuperSurv_eval" containing benchmark metrics for the ensemble and base learners.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:80, ]
x_cols <- grep("^x", names(dat))[1:3]

fit <- SuperSurv(
  time = dat$duration,
  event = dat$event,
  X = dat[, x_cols, drop = FALSE],
  new.times = seq(50, 200, by = 50),
  event.library = c("surv.coxph", "surv.km"),
  cens.library = c("surv.coxph", "surv.km")
)

res <- eval_summary(
  object = fit,
  newdata = dat[, x_cols, drop = FALSE],
  time = dat$duration,
  event = dat$event,
  eval_times = seq(50, 200, by = 50)
```

```
)
res
```

 eval_timeROC

Time-Dependent AUC and Integrated AUC

Description

Evaluates the cumulative/dynamic time-dependent AUC and integrated AUC (iAUC) using inverse probability of censoring weighting (IPCW).

Usage

```
eval_timeROC(time, event, S_mat, times)
```

Arguments

time	Numeric vector of observed follow-up times.
event	Numeric vector of event indicators (1 = event, 0 = censored).
S_mat	A numeric matrix of predicted survival probabilities.
times	Numeric vector of evaluation times matching the columns of S_mat.

Value

A list containing the AUC_curve at each time point, the times, and the integrated AUC iAUC.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:40, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:10, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.coxph(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
  obsWeights = rep(1, nrow(dat)),
  id = NULL
)

eval_timeROC(
```

```

    time = dat$duration[1:10],
    event = dat$event[1:10],
    S_mat = fit$pred,
    times = times
  )

```

explain_kernel	<i>Explain Predictions with Global SHAP (Kernel SHAP)</i>
----------------	---

Description

Explain Predictions with Global SHAP (Kernel SHAP)

Usage

```

explain_kernel(
  model,
  X_explain,
  X_background,
  nsim = 20,
  only_best = FALSE,
  verbose = FALSE
)

```

Arguments

model	A fitted SuperSurv object OR a single wrapper output.
X_explain	The dataset you want to explain (e.g., X_test[1:10,]).
X_background	The reference dataset for fastshap (e.g., X_train[1:100,]).
nsim	Number of simulations. Defaults to 20.
only_best	Logical. If TRUE and model is SuperSurv, only explains the highest-weighted base learner.
verbose	Logical; if TRUE, progress messages are shown.

Value

A data.frame of class c("explain", "data.frame") containing the calculated SHAP values. The columns correspond to the covariates in X_explain.

Examples

```

if (requireNamespace("fastshap", quietly = TRUE) &&
    requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:80, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
}

```

```

new.times <- seq(20, 120, by = 20)

fit <- SuperSurv(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = X,
  new.times = new.times,
  event.library = c("surv.coxph", "surv.ridge"),
  cens.library = c("surv.coxph"),
  control = list(saveFitLibrary = TRUE)
)

shap_values <- explain_kernel(
  model = fit,
  X_explain = X[1:10, , drop = FALSE],
  X_background = X[11:40, , drop = FALSE],
  nsim = 5
)

dim(shap_values)
}

```

explain_survex

Create a Time-Dependent Survex Explainer

Description

Bridges a fitted SuperSurv ensemble or a single base learner to the survex package for Time-Dependent SHAP and Model Parts.

Usage

```
explain_survex(model, data, y, times, label = NULL)
```

Arguments

model	A fitted SuperSurv object OR a single wrapper output.
data	Covariate data for explanation (data.frame).
y	The survival object (Surv(time, event)).
times	The time grid for evaluation.
label	Optional character string to name the explainer.

Value

An explainer object of class `survex_explainer` created by [explain_survival](#), which can be passed to DALEX and survex functions for further model diagnostics and plotting.

Examples

```

if (requireNamespace("survex", quietly = TRUE) &&
    requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:80, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  times <- seq(20, 120, by = 20)
  y <- survival::Surv(dat$duration, dat$event)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,
    new.times = times,
    event.library = c("surv.coxph", "surv.ridge"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )

  explainer <- explain_survex(
    model = fit,
    data = X,
    y = y,
    times = times,
    label = "SuperSurv_demo"
  )

  class(explainer)
}

```

list_wrappers

List Available Wrappers and Screeners in SuperSurv

Description

This function prints all built-in prediction algorithms and feature screening algorithms available in the SuperSurv package.

Usage

```
list_wrappers(what = "both")
```

Arguments

what Character string. If "both" (default), lists both prediction and screening functions. If "surv", lists only prediction models. If "screen", lists only screening algorithms. Otherwise, lists all exports.

Value

An invisible character vector containing the requested function names.

Examples

```
list_wrappers()
```

metabric

METABRIC Breast Cancer Dataset

Description

A subset of the Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) dataset to demonstrate the SuperSurv package. 9 Covariates: 4 gene indicators (MKI67, EGFR, PGR, and ERBB2) and 5 clinical features (age at diagnosis, and indicators for hormone treatment, radiotherapy, and chemotherapy)

Usage

```
metabric
```

Format

A data frame with the clinical and genomic variables:

duration Survival time.

event Event indicator (1 = event, 0 = censored).

x0 Feature x0.

x1 Feature x1.

x2 Feature x2.

x3 Feature x3.

x4 Feature x4.

x5 Feature x5.

x6 Feature x6.

x7 Feature x7.

x8 Feature x8.

plot_beeswarm	<i>Beeswarm Summary Plot for SuperSurv SHAP</i>
---------------	---

Description

Beeswarm Summary Plot for SuperSurv SHAP

Usage

```
plot_beeswarm(shap_values, data, top_n = 10)
```

Arguments

shap_values	The output from explain_kernel().
data	The covariate data used (X_explain)
top_n	Number of features to display

Value

A ggplot object visualizing the SHAP values.

Examples

```
if (requireNamespace("fastshap", quietly = TRUE) &&
    requireNamespace("ggforce", quietly = TRUE) &&
    requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:80, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  new.times <- seq(20, 120, by = 20)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,
    new.times = new.times,
    event.library = c("surv.coxph", "surv.ridge"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )

  shap_values <- explain_kernel(
    model = fit,
    X_explain = X[1:20, , drop = FALSE],
    X_background = X[21:50, , drop = FALSE],
    nsim = 5
  )
}
```

```

plot_beeswarm(
  shap_values = shap_values,
  data = X[1:20, , drop = FALSE],
  top_n = 5
)
}

```

plot_benchmark

Plot Longitudinal Benchmark Metrics

Description

Generates time-dependent performance curves comparing the SuperSurv ensemble against its base learners, or evaluates a single standalone learner.

Usage

```

plot_benchmark(
  object,
  newdata,
  time,
  event,
  eval_times,
  metrics = c("brier", "auc", "cindex"),
  verbose = FALSE
)

```

Arguments

object	A fitted SuperSurv object OR a fitted standalone learner.
newdata	A data.frame of test covariates.
time	Numeric vector of observed follow-up times for the test set.
event	Numeric vector of event indicators for the test set.
eval_times	Numeric vector of times at which to evaluate predictions.
metrics	Character vector specifying which plots to return. Options: "brier", "auc", "cindex". Defaults to all three.
verbose	Logical; if TRUE, progress messages are shown. Defaults to FALSE.

Value

A combined patchwork ggplot object, or a single ggplot if only one metric is selected.

Examples

```

if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:120, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  eval_times <- seq(20, 120, by = 20)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,
    new.times = eval_times,
    event.library = c("surv.coxph", "surv.ranger"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )

  plot_benchmark(
    object = fit,
    newdata = X,
    time = dat$duration,
    event = dat$event,
    eval_times = eval_times,
    metrics = c("brier")
  )
}

```

plot_calibration *Plot Survival Calibration Curve*

Description

Plot Survival Calibration Curve

Usage

```
plot_calibration(object, newdata, time, event, eval_time, bins = 5)
```

Arguments

object	A fitted SuperSurv object OR a standalone base learner.
newdata	A data.frame of test covariates.
time	Numeric vector of observed follow-up times for the test set.
event	Numeric vector of event indicators for the test set.
eval_time	Numeric. A single time point at which to assess calibration.
bins	Integer. Defaults to 5.

Value

A ggplot object.

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:120, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  eval_times <- seq(20, 120, by = 20)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,
    new.times = eval_times,
    event.library = c("surv.coxph", "surv.ridge"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )

  plot_calibration(
    object = fit,
    newdata = X,
    time = dat$duration,
    event = dat$event,
    eval_time = 100,
    bins = 4
  )
}
```

plot_dependence

Plot SHAP Dependence for SuperSurv

Description

Plot SHAP Dependence for SuperSurv

Usage

```
plot_dependence(shap_values, data, feature_name, title = NULL)
```

Arguments

shap_values	The output from explain_kernel().
data	The original covariate data used for the explanation (X_explain)
feature_name	String name of the column to plot
title	Optional custom title.

Value

A ggplot object visualizing the SHAP values.

Examples

```

if (requireNamespace("fastshap", quietly = TRUE) &&
    requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:80, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  new.times <- seq(20, 120, by = 20)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,
    new.times = new.times,
    event.library = c("surv.coxph", "surv.ridge"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )

  shap_values <- explain_kernel(
    model = fit,
    X_explain = X[1:20, , drop = FALSE],
    X_background = X[21:50, , drop = FALSE],
    nsim = 5
  )

  plot_dependence(
    shap_values = shap_values,
    data = X[1:20, , drop = FALSE],
    feature_name = colnames(X)[1]
  )
}

```

plot_global_importance

Plot Global Feature Importance for SuperSurv

Description

Plot Global Feature Importance for SuperSurv

Usage

```
plot_global_importance(
  shap_values,
  title = "SuperSurv: Ensemble Feature Importance",
  top_n = 10
)
```

Arguments

shap_values	The output from explain_kernel().
title	Plot title.
top_n	Number of features to show (default 10)

Value

A ggplot object visualizing the SHAP values.

Examples

```
if (requireNamespace("fastshap", quietly = TRUE) &&
    requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:80, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  new.times <- seq(20, 120, by = 20)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,
    new.times = new.times,
    event.library = c("surv.coxph", "surv.ridge"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )

  shap_values <- explain_kernel(
    model = fit,
    X_explain = X[1:10, , drop = FALSE],
    X_background = X[11:40, , drop = FALSE],
    nsim = 5
  )

  plot_global_importance(shap_values, top_n = 5)
}
```

plot_marginal_rmst_curve

Plot Causal RMST Difference Over Time

Description

Generates a curve showing how the causal Restricted Mean Survival Time (RMST) difference between treatment groups evolves across a sequence of different truncation times.

Usage

```
plot_marginal_rmst_curve(fit, data, trt_col, times, tau_seq)
```

Arguments

fit	A fitted SuperSurv ensemble object.
data	A data.frame containing the patient covariates and the treatment assignment.
trt_col	Character string. The exact name of the binary treatment indicator column in data.
times	Numeric vector of time points matching the prediction grid.
tau_seq	Numeric vector. A sequence of truncation times (τ) to evaluate and plot.

Value

A ggplot object visualizing the causal RMST difference curve.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:80, ]
x_cols <- grep("^x", names(dat))[1:5]
X <- dat[, x_cols, drop = FALSE]
new.times <- seq(20, 120, by = 20)

fit <- SuperSurv(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = X,
  new.times = new.times,
  event.library = c("surv.coxph", "surv.glmnet"),
  cens.library = c("surv.coxph"),
  control = list(saveFitLibrary = TRUE)
)

tau_grid <- seq(40, 120, by = 20)
plot_marginal_rmst_curve(
  fit = fit,
```

```

data = dat,
trt_col = "x4",
times = new.times,
tau_seq = tau_grid
)

```

```
plot_patient_waterfall
```

Waterfall Plot for an Individual Patient

Description

Waterfall Plot for an Individual Patient

Usage

```
plot_patient_waterfall(shap_values, patient_index = 1, top_n = 10)
```

Arguments

shap_values	The output from explain_kernel().
patient_index	The row index of the patient to explain
top_n	Number of features to show (default 10)

Value

A ggplot object visualizing the SHAP values.

Examples

```

if (requireNamespace("fastshap", quietly = TRUE) &&
    requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:80, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  new.times <- seq(20, 120, by = 20)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,
    new.times = new.times,
    event.library = c("surv.coxph", "surv.ridge"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )
}

```

```

shap_values <- explain_kernel(
  model = fit,
  X_explain = X[1:10, , drop = FALSE],
  X_background = X[11:40, , drop = FALSE],
  nsim = 5
)

plot_patient_waterfall(
  shap_values = shap_values,
  patient_index = 1,
  top_n = 5
)
}

```

plot_predict

Plot Predicted Survival Curves

Description

Plot Predicted Survival Curves

Usage

```
plot_predict(preds, eval_times, patient_idx = 1)
```

Arguments

preds	A list containing SuperSurv predictions OR a raw prediction matrix.
eval_times	Numeric vector of times at which predictions were evaluated.
patient_idx	Integer vector. Defaults to 1.

Value

A ggplot object.

Examples

```

if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:120, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  eval_times <- seq(20, 120, by = 20)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,

```

```

    new.times = eval_times,
    event.library = c("surv.coxph", "surv.ridge"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )

  preds <- predict(fit, newdata = X, new.times = eval_times)

  plot_predict(
    preds = preds,
    eval_times = eval_times,
    patient_idx = c(1, 2)
  )
}

```

plot_rmst_vs_obs

Plot Predicted RMST vs. Observed Survival Times

Description

Evaluates the calibration of the causal RMST estimator by plotting the model's predicted RMST for each patient against their actual observed follow-up time.

Usage

```
plot_rmst_vs_obs(fit, data, time_col, event_col, times, tau)
```

Arguments

fit	A fitted SuperSurv ensemble object.
data	A data.frame containing the patient covariates, times, and events.
time_col	Character string. The exact name of the observed follow-up time column in data.
event_col	Character string. The exact name of the event indicator column in data (e.g., 1 for event, 0 for censored).
times	Numeric vector of time points matching the prediction grid.
tau	Numeric. A single truncation time limit up to which the RMST is calculated.

Value

A ggplot object comparing predicted RMST to observed outcomes.

Examples

```

data("metabric", package = "SuperSurv")
dat <- metabric[1:80, ]
x_cols <- grep("^x", names(dat))[1:5]
X <- dat[, x_cols, drop = FALSE]
new.times <- seq(20, 120, by = 20)

fit <- SuperSurv(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = X,
  new.times = new.times,
  event.library = c("surv.coxph", "surv.glmnet"),
  cens.library = c("surv.coxph"),
  control = list(saveFitLibrary = TRUE)
)

plot_rmst_vs_obs(
  fit = fit,
  data = dat,
  time_col = "duration",
  event_col = "event",
  times = new.times,
  tau = 350
)

```

plot_survival_heatmap *Survival Probability Heatmap*

Description

Survival Probability Heatmap

Usage

```
plot_survival_heatmap(object, newdata, times)
```

Arguments

object	A fitted SuperSurv object
newdata	Test covariates (e.g., X_te[1:50,])
times	The time grid to visualize

Value

A ggplot object visualizing the SHAP values.

Examples

```

if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:80, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  times <- seq(20, 120, by = 20)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,
    new.times = times,
    event.library = c("surv.coxph", "surv.ridge"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )

  plot_survival_heatmap(
    object = fit,
    newdata = X[1:20, , drop = FALSE],
    times = times
  )
}

```

predict.SuperSurv *Predict method for SuperSurv fits*

Description

Obtains predicted survival probabilities from a fitted SuperSurv ensemble.

Usage

```

## S3 method for class 'SuperSurv'
predict(object, newdata, new.times, onlySL = FALSE, threshold = 1e-04, ...)

```

Arguments

object	A fitted object of class SuperSurv.
newdata	A data.frame of new covariate values.
new.times	A numeric vector of times at which to predict survival.
onlySL	Logical. If TRUE, only uses models with weights > threshold.
threshold	Numeric. The weight threshold for onlySL.
...	Additional ignored arguments.

Value

A list containing:

- `event.predict`: A numeric matrix (rows = observations, columns = times) of the final predicted survival probabilities from the ensemble.
- `event.library.predict`: A 3D numeric array (observations x times x models) containing the individual survival predictions from each base learner.
- `cens.predict`: A numeric matrix of the predicted censoring probabilities.
- `cens.library.predict`: A 3D numeric array of the individual censoring predictions.

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:80, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:10, , drop = FALSE]
  new.times <- seq(20, 120, by = 20)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,
    new.times = new.times,
    event.library = c("surv.coxph", "surv.ridge"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )

  preds <- predict(
    object = fit,
    newdata = newX,
    new.times = new.times
  )

  dim(preds$event.predict)
}
```

predict.surv.aorsf *Prediction function for AORSF*

Description

Obtains predicted survivals from a fitted `surv.aorsf` object. Uses the native `aorsf` prediction engine to calculate survival directly at requested times.

Usage

```
## S3 method for class 'surv.aorsf'
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.aorsf object.
newdata	New covariate data.frame.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
if (requireNamespace("aorsf", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.aorsf(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    n_tree = 10,
    leaf_min_events = 2
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
  dim(pred)
}
```

predict.surv.bart

Prediction function for BART

Description

Obtains predicted survivals from a fitted surv.bart object. Manually expands the time grid to bypass the 10-column expectation.

Usage

```
## S3 method for class 'surv.bart'  
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.bart object.
newdata	New covariate data.frame to predict on.
new.times	Times to predict.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
if (.Platform$OS.type != "windows" &&  
    requireNamespace("BART", quietly = TRUE)) {  
  data("metabric", package = "SuperSurv")  
  dat <- metabric[1:20, ]  
  x_cols <- grep("^x", names(dat))[1:3]  
  X <- dat[, x_cols, drop = FALSE]  
  newX <- X[1:5, , drop = FALSE]  
  times <- seq(50, 150, by = 50)  
  
  fit <- surv.bart(  
    time = dat$duration,  
    event = dat$event,  
    X = X,  
    newdata = newX,  
    new.times = times,  
    ntree = 3,  
    ndpost = 5,  
    nskip = 5  
  )  
  
  pred <- fit$pred  
  dim(pred)  
}
```

predict.surv.coxboost *Prediction function for CoxBoost wrapper*

Description

Obtains predicted survivals from a fitted surv.coxboost object.

Usage

```
## S3 method for class 'surv.coxboost'
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.coxboost object.
newdata	New covariate data.frame.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
if (requireNamespace("CoxBoost", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.coxboost(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    stepno = 10,
    penalty = 50
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
  dim(pred)
}
```

predict.surv.coxph *Prediction function for Cox regression wrapper*

Description

Obtains predicted survivals from a fitted surv.coxph object.

Usage

```
## S3 method for class 'surv.coxph'
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.coxph object.
newdata	New covariate data.frame for which to obtain predictions.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:30, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:5, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.coxph(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
  obsWeights = rep(1, nrow(dat)),
  id = NULL
)

pred <- predict(fit$fit, newdata = newX, new.times = times)
dim(pred)
```

predict.surv.gam

Prediction function for GAM wrapper

Description

Obtains predicted survivals from a fitted surv.gam object.

Usage

```
## S3 method for class 'surv.gam'
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.gam object.
newdata	New covariate data.frame.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
if (requireNamespace("mgcv", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.gam(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    cts.num = 5
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
  dim(pred)
}
```

predict.surv.gbm

Prediction function for GBM wrapper

Description

Obtains predicted survivals from a fitted surv.gbm object. Uses a step function to align the Breslow baseline hazard with the requested times.

Usage

```
## S3 method for class 'surv.gbm'
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.gbm object.
newdata	New covariate data.frame.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
if (requireNamespace("gbm", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.gbm(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    n.trees = 20,
    interaction.depth = 1,
    shrinkage = 0.05,
    cv.folds = 0,
    n.minobsinnode = 3
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
  dim(pred)
}
```

predict.surv.glmnet *Prediction function for GLMNET wrapper*

Description

Obtains predicted survivals from a fitted surv.glmnet object.

Usage

```
## S3 method for class 'surv.glmnet'
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.glmnet object.
newdata	New covariate data.frame.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.glmnet(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    alpha = 1,
    nfold = 3
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
  dim(pred)
}
```

predict.surv.km

Predict Method for Kaplan-Meier Wrapper

Description

Obtains predicted survivals from a fitted surv.km object.

Usage

```
## S3 method for class 'surv.km'
predict(object, newdata, new.times, ...)
```

Arguments

object	A fitted object of class <code>surv.km</code> .
newdata	New covariate data.frame for which to obtain predictions (Ignored).
new.times	Numeric vector of times at which to predict survival.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in `newdata` and columns correspond to the evaluation times.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:30, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:5, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.km(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
  obsWeights = rep(1, nrow(dat)),
  id = NULL
)

pred <- predict(fit$fit, newdata = newX, new.times = times)
dim(pred)
```

predict.surv.parametric

Prediction function for Universal Parametric Wrapper

Description

Obtains predicted survivals from a fitted `surv.parametric` object using exact closed-form equations.

Usage

```
## S3 method for class 'surv.parametric'  
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.parametric object.
newdata	New covariate data.frame.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
data("metabric", package = "SuperSurv")  
dat <- metabric[1:30, ]  
x_cols <- grep("^x", names(dat))[1:3]  
X <- dat[, x_cols, drop = FALSE]  
newX <- X[1:5, , drop = FALSE]  
times <- seq(50, 150, by = 50)  
  
fit <- surv.parametric(  
  time = dat$duration,  
  event = dat$event,  
  X = X,  
  newdata = newX,  
  new.times = times,  
  obsWeights = rep(1, nrow(dat)),  
  id = NULL,  
  dist = "weibull"  
)  
  
pred <- predict(fit$fit, newdata = newX, new.times = times)  
dim(pred)
```

predict.surv.ranger *Prediction function for Ranger wrapper*

Description

Obtains predicted survivals from a fitted surv.ranger object.

Usage

```
## S3 method for class 'surv.ranger'
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.ranger object.
newdata	New covariate data.frame.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
if (requireNamespace("ranger", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.ranger(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    num.trees = 10,
    min.node.size = 3
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
  dim(pred)
}
```

predict.surv.rfsrc *Prediction function for survival random forest (RFSRC)*

Description

Obtains predicted survivals from a fitted surv.rfsrc object.

Usage

```
## S3 method for class 'surv.rfsrc'
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.rfsrc object.
newdata	New covariate data.frame.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
if (requireNamespace("randomForestSRC", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.rfsrc(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    ntree = 10,
    nodesize = 3
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
  dim(pred)
}
```

predict.surv.ridge *Prediction function for Ridge wrapper*

Description

Obtains predicted survivals from a fitted surv.ridge object.

Usage

```
## S3 method for class 'surv.ridge'
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted <code>surv.ridge</code> object.
newdata	New covariate data.frame.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in `newdata` and columns correspond to the evaluation times in `new.times`.

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.ridge(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    nfolds = 3
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
  dim(pred)
}
```

predict.surv.rpart *Prediction function for rpart wrapper*

Description

Obtains predicted survivals from a fitted `surv.rpart` object.

Usage

```
## S3 method for class 'surv.rpart'
predict(object, newdata, new.times, ...)
```

Arguments

object	Fitted surv.rpart object.
newdata	New covariate data.frame.
new.times	Times at which to obtain predicted survivals.
...	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in newdata and columns correspond to the evaluation times in new.times.

Examples

```
if (requireNamespace("rpart", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.rpart(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    cp = 0.01,
    minsplit = 5,
    maxdepth = 3
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
  dim(pred)
}
```

predict.surv.svm *Prediction function for SVM wrapper*

Description

Obtains predicted survivals from a fitted `surv.svm` object. Applies the learned Cox calibration to the new raw utility scores.

Usage

```
## S3 method for class 'surv.svm'
predict(object, newdata, new.times, ...)
```

Arguments

<code>object</code>	Fitted <code>surv.svm</code> object.
<code>newdata</code>	New covariate data.frame.
<code>new.times</code>	Times at which to obtain predicted survivals.
<code>...</code>	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in `newdata` and columns correspond to the evaluation times in `new.times`.

Examples

```
if (requireNamespace("survivalsvm", quietly = TRUE) &&
    requireNamespace("quadprog", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:25, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.svm(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
  dim(pred)
}
```

predict.surv.xgboost *Prediction function for XGBoost wrapper*

Description

Obtains predicted survivals from a fitted `surv.xgboost` object.

Usage

```
## S3 method for class 'surv.xgboost'
predict(object, newdata, new.times, ...)
```

Arguments

<code>object</code>	Fitted <code>surv.xgboost</code> object.
<code>newdata</code>	New covariate data.frame.
<code>new.times</code>	Times at which to obtain predicted survivals.
<code>...</code>	Additional ignored arguments.

Value

A numeric matrix of predicted survival probabilities, where rows correspond to the observations in `newdata` and columns correspond to the evaluation times in `new.times`.

Examples

```
if (requireNamespace("xgboost", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.xgboost(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    nrounds = 5,
    early_stopping_rounds = 2,
    max_depth = 1
  )

  pred <- predict(fit$fit, newdata = newX, new.times = times)
```

```
    dim(pred)
  }
```

screen.all *Keep All Variables Screener*

Description

Keep All Variables Screener

Usage

```
screen.all(X, ...)
```

Arguments

X Training covariate data.frame.
... Additional ignored arguments.

Value

A logical vector of the same length as the number of columns in X, indicating which variables passed the screening algorithm (TRUE to keep, FALSE to drop).

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:20, ]
x_cols <- grep("^x", names(dat))[1:5]
X <- dat[, x_cols, drop = FALSE]

screen.all(X)
```

screen.elasticnet *Elastic Net Screening Algorithm*

Description

This screening algorithm uses [cv.glmnet](#) to select covariates. Unlike LASSO ($\alpha = 1$), which drops correlated features, Elastic Net ($\alpha = 0.5$ by default) shrinks correlated groups of features together, making it ideal for selecting entire biological pathways.

Usage

```
screen.elasticnet(
  time,
  event,
  X,
  obsWeights = NULL,
  alpha = 0.5,
  minscreen = 2,
  nfolds = 10,
  nlambda = 100,
  ...
)
```

Arguments

time	Numeric vector of observed follow-up times.
event	Numeric vector of event indicators (1 = event, 0 = censored).
X	Training covariate data.frame or matrix.
obsWeights	Numeric vector of observation weights.
alpha	Numeric penalty exponent for glmnet. Defaults to 0.5 (Elastic Net).
minscreen	Integer. Minimum number of covariates to return. Defaults to 2.
nfolds	Integer. Number of folds for cross-validation. Defaults to 10.
nlambda	Integer. Number of penalty parameters to search over. Defaults to 100.
...	Additional arguments passed to screen.glmnet.

Value

A logical vector of the same length as the number of columns in X, indicating which variables passed the screening algorithm (TRUE to keep, FALSE to drop).

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:40, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]

  screen.elasticnet(
    time = dat$duration,
    event = dat$event,
    X = X,
    alpha = 0.5,
    minscreen = 2,
    nfolds = 3,
    nlambda = 20
  )
}
```

screen.glmnet	<i>GLMNET (Lasso) Screening</i>
---------------	---------------------------------

Description

GLMNET (Lasso) Screening

Usage

```
screen.glmnet(
  time,
  event,
  X,
  obsWeights = NULL,
  alpha = 1,
  minscreen = 2,
  nfolds = 10,
  nlambda = 100,
  ...
)
```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
obsWeights	Observation weights.
alpha	Penalty exponent (1 = lasso).
minscreen	Minimum number of covariates to return. Defaults to 2.
nfolds	Number of CV folds.
nlambda	Number of penalty parameters.
...	Additional ignored arguments.

Value

A logical vector of the same length as the number of columns in X, indicating which variables passed the screening algorithm (TRUE to keep, FALSE to drop).

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:40, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
```

```

screen.glmnet(
  time = dat$duration,
  event = dat$event,
  X = X,
  alpha = 1,
  minscreen = 2,
  nolds = 3,
  nlambdas = 20
)
}

```

screen.marg

Marginal Cox Regression Screening

Description

Marginal Cox Regression Screening

Usage

```
screen.marg(time, event, X, obsWeights = NULL, minscreen = 2, min.p = 0.1, ...)
```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
obsWeights	Observation weights.
minscreen	Minimum number of covariates to return. Defaults to 2.
min.p	Threshold p-value. Defaults to 0.1.
...	Additional ignored arguments.

Value

A logical vector of the same length as the number of columns in X, indicating which variables passed the screening algorithm (TRUE to keep, FALSE to drop).

Examples

```

data("metabric", package = "SuperSurv")
dat <- metabric[1:40, ]
x_cols <- grep("^x", names(dat))[1:5]
X <- dat[, x_cols, drop = FALSE]

screen.marg(
  time = dat$duration,

```

```

event = dat$event,
X = X,
minscreen = 2,
min.p = 0.2
)

```

screen.rfsrc

Random Survival Forest Screening Algorithm

Description

This screening algorithm uses the randomForestSRC package to select covariates based on their Variable Importance (VIMP). It grows a fast forest and retains features with a VIMP greater than zero.

Usage

```

screen.rfsrc(
  time,
  event,
  X,
  obsWeights = NULL,
  minscreen = 2,
  ntree = 100,
  ...
)

```

Arguments

time	Numeric vector of observed follow-up times.
event	Numeric vector of event indicators (1 = event, 0 = censored).
X	Training covariate data.frame or matrix.
obsWeights	Numeric vector of observation weights.
minscreen	Integer. Minimum number of covariates to return. Defaults to 2.
ntree	Integer. Number of trees to grow. Defaults to 100 for fast screening.
...	Additional arguments passed to rfsrc .

Value

A logical vector of the same length as the number of columns in X, indicating which variables passed the screening algorithm (TRUE to keep, FALSE to drop).

Examples

```

if (requireNamespace("randomForestSRC", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:40, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]

  screen.rfsrc(
    time = dat$duration,
    event = dat$event,
    X = X,
    minscreen = 2,
    ntree = 10
  )
}

```

screen.var

*High Variance Screening Algorithm (Unsupervised)***Description**

An unsupervised screening algorithm that filters out low-variance features. This is particularly useful for high-dimensional genomic or transcriptomic data where many features remain relatively constant across all observations.

Usage

```

screen.var(
  time,
  event,
  X,
  obsWeights = NULL,
  keep_fraction = 0.5,
  minscreen = 2,
  ...
)

```

Arguments

time	Numeric vector of observed follow-up times (Ignored internally).
event	Numeric vector of event indicators (Ignored internally).
X	Training covariate data.frame or matrix.
obsWeights	Numeric vector of observation weights (Ignored internally).
keep_fraction	Numeric value between 0 and 1. The fraction of highest-variance features to retain. Defaults to 0.5 (keeps the top 50%).
minscreen	Integer. Minimum number of covariates to return. Defaults to 2.
...	Additional ignored arguments.

Value

A logical vector of the same length as the number of columns in X , indicating which variables passed the screening algorithm (TRUE to keep, FALSE to drop).

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:40, ]
x_cols <- grep("^x", names(dat))[1:6]
X <- dat[, x_cols, drop = FALSE]

screen.var(
  time = dat$duration,
  event = dat$event,
  X = X,
  keep_fraction = 0.5,
  minscreen = 2
)
```

SuperSurv

Super Learner for conditional survival functions

Description

Orchestrates the cross-validation, metalearner optimization, and prediction for an ensemble of survival base learners.

Usage

```
SuperSurv(
  time,
  event,
  X,
  newdata = NULL,
  new.times,
  event.library,
  cens.library,
  id = NULL,
  verbose = FALSE,
  control = list(),
  cvControl = list(),
  obsWeights = NULL,
  metalearner = "brier",
  selection = "ensemble",
  nFolds = 10,
  parallel = FALSE
)
```

Arguments

<code>time</code>	Observed follow-up time.
<code>event</code>	Observed event indicator.
<code>X</code>	Training covariate data.frame.
<code>newdata</code>	Test covariate data.frame for prediction (defaults to X).
<code>new.times</code>	Times at which to obtain predicted survivals.
<code>event.library</code>	Character vector of prediction algorithms for the event.
<code>cens.library</code>	Character vector of prediction algorithms for censoring.
<code>id</code>	Cluster identification variable.
<code>verbose</code>	Logical. If TRUE, prints progress messages.
<code>control</code>	List of control parameters for the Super Learner.
<code>cvControl</code>	List of control parameters for cross-validation.
<code>obsWeights</code>	Observation weights.
<code>metalearner</code>	Character string specifying the optimizer (e.g., "least_squares").
<code>selection</code>	Character. Specifies how the meta-learner combines the base models. Use "ensemble" (default) to calculate a weighted average (convex combination) of the base learners. Use "best" to act as a Discrete Super Learner, which assigns a weight of 1.0 to the single model with the lowest cross-validated risk.
<code>nFolds</code>	Number of cross-validation folds (default: 10).
<code>parallel</code>	Logical. If TRUE, uses future.apply for parallel execution.

Value

A list of class SuperSurv containing:

- `call`: The matched function call.
- `event.predict`: Matrix of in-sample cross-validated survival predictions.
- `cens.predict`: Matrix of in-sample cross-validated censoring predictions.
- `event.coef`: Numeric vector of optimized ensemble weights for the event.
- `cens.coef`: Numeric vector of optimized ensemble weights for censoring.
- `event.library.predict`: 3D array of cross-validated predictions from individual event learners.
- `event.libraryNames`: Data frame detailing the algorithms and screeners used.
- `event.fitLibrary`: List of the fitted base learner models (if `saveFitLibrary = TRUE`).
- `times`: The time grid used for evaluation.

Examples

```

if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:80, ]
  x_cols <- grep("^x", names(dat))[1:5]
  X <- dat[, x_cols, drop = FALSE]
  new.times <- seq(20, 120, by = 20)

  fit <- SuperSurv(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = X,
    new.times = new.times,
    event.library = c("surv.coxph", "surv.ridge"),
    cens.library = c("surv.coxph"),
    control = list(saveFitLibrary = TRUE)
  )

  fit$event.library.predict
}

```

surv.aorsf

Wrapper for AORSF (Oblique Random Survival Forest)

Description

Final Production Wrapper for AORSF (Tunable & Robust).

Usage

```

surv.aorsf(
  time,
  event,
  X,
  newdata,
  new.times,
  obsWeights,
  id,
  n_tree = 500,
  leaf_min_events = 5,
  mtry = NULL,
  ...
)

```

Arguments

time Observed follow-up time; i.e. minimum of the event and censoring times.

event	Observed event indicator; i.e, whether the follow-up time corresponds to an event or censoring.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction. Should have the same variable names and structure as X.
new.times	Times at which to obtain the predicted survivals.
obsWeights	Observation weights.
id	Optional cluster/individual ID indicator.
n_tree	Number of trees to grow (default: 500).
leaf_min_events	Minimum number of events in a leaf node (default: 5).
mtry	Number of predictors evaluated at each node.
...	Additional arguments passed to <code>orsf</code> .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```
if (requireNamespace("aorsf", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.aorsf(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    n_tree = 10,
    leaf_min_events = 2
  )

  dim(fit$pred)
}
```

surv.bart

*Wrapper for BART (Bayesian Additive Regression Trees)***Description**

Final Production Wrapper for BART (Tunable & Robust). Uses the `mc.surv.bart` function. Automatically reshapes the flat output vector into a survival matrix and interpolates the predictions to the requested `new.times`.

Usage

```
surv.bart(
  time,
  event,
  X,
  newdata = NULL,
  new.times,
  obsWeights = NULL,
  id = NULL,
  ntree = 10,
  ndpost = 30,
  nskip = 10,
  ...
)
```

Arguments

<code>time</code>	Observed follow-up time.
<code>event</code>	Observed event indicator.
<code>X</code>	Training covariate data.frame.
<code>newdata</code>	Test covariate data.frame to use for prediction.
<code>new.times</code>	Times at which to obtain predicted survivals.
<code>obsWeights</code>	Observation weights (Note: BART does not natively support weights).
<code>id</code>	Optional cluster/individual ID indicator.
<code>ntree</code>	Number of trees (default: 50).
<code>ndpost</code>	Number of posterior draws (default: 1000).
<code>nskip</code>	Number of burn-in draws (default: 250).
<code>...</code>	Additional arguments passed to <code>mc.surv.bart</code> .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.

- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```
if (.Platform$OS.type != "windows" &&
    requireNamespace("BART", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:20, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.bart(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    ntree = 3,
    ndpost = 5,
    nskip = 5
  )

  dim(fit$pred)
}
```

surv.coxboost

Wrapper function for Component-Wise Boosting (CoxBoost)

Description

Final Production Wrapper for CoxBoost (Tunable & Robust). Estimates a Cox model via component-wise likelihood based boosting.

Usage

```
surv.coxboost(
  time,
  event,
  X,
  newdata,
  new.times,
  obsWeights,
  id,
  stepno = 100,
  penalty = 100,
  ...
)
```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain the predicted survivals.
obsWeights	Observation weights (Note: CoxBoost does not natively support weights, so these are ignored).
id	Optional cluster/individual ID indicator.
stepno	Number of boosting steps (default: 100).
penalty	Penalty value for the update (default: 100).
...	Additional arguments passed to CoxBoost .

Value

A list containing:

- fit: The fitted model object (e.g., the raw coxph or xgb.Booster object). If the model fails to fit, this may be an object of class try-error.
- pred: A numeric matrix of cross-validated survival predictions evaluated at the specified new.times grid.

Examples

```
if (requireNamespace("CoxBoost", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.coxboost(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    stepno = 10,
    penalty = 50
  )

  dim(fit$pred)
}
```

 surv.coxph

Wrapper for standard Cox Proportional Hazards

Description

Final Production Wrapper for CoxPH. Uses partial maximum likelihood and the Breslow estimator.

Usage

```
surv.coxph(time, event, X, newdata, new.times, obsWeights, id, ...)
```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain predicted survivals.
obsWeights	Observation weights.
id	Optional cluster/individual ID indicator.
...	Additional arguments passed to coxph .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:30, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:5, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.coxph(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
```

```

      obsWeights = rep(1, nrow(dat)),
      id = NULL
    )

    dim(fit$pred)

```

surv.exponential *Parametric Survival Prediction Wrapper (Exponential)*

Description

Parametric Survival Prediction Wrapper (Exponential)

Usage

```
surv.exponential(time, event, X, newdata, new.times, obsWeights, id, ...)
```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain the predicted survivals.
obsWeights	Observation weights.
id	Cluster identification variable.
...	Additional ignored arguments.

Value

A list containing the fitted model and predictions.

Examples

```

data("metabric", package = "SuperSurv")
dat <- metabric[1:30, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:5, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.exponential(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,

```

```

  obsWeights = rep(1, nrow(dat)),
  id = NULL
)

dim(fit$pred)

```

surv.gam

Wrapper for Generalized Additive Cox Regression (GAM)

Description

Final Production Wrapper for GAM (Tunable & Robust). Uses [gam](#) to fit an additive combination of smooth and linear functions.

Usage

```
surv.gam(time, event, X, newdata, new.times, obsWeights, id, cts.num = 5, ...)
```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain predicted survivals.
obsWeights	Observation weights (Note: Ignored, as mgcv uses weights for the event indicator).
id	Optional cluster/individual ID indicator.
cts.num	Cutoff of unique values at which a numeric covariate receives a smooth term (s).
...	Additional arguments passed to gam .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```
if (requireNamespace("mgcv", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.gam(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    cts.num = 5
  )

  dim(fit$pred)
}
```

surv.gbm

Wrapper function for Gradient Boosting (GBM) prediction algorithm

Description

Final Production Wrapper for GBM (Tunable & Robust). Estimates a Cox proportional hazards model via gradient boosting. Uses the Breslow estimator with a step-function approach for the baseline hazard. Includes internal safeguards against C++ crashes and small cross-validation folds.

Usage

```
surv.gbm(
  time,
  event,
  X,
  newdata,
  new.times,
  obsWeights,
  id,
  n.trees = 1000,
  interaction.depth = 2,
  shrinkage = 0.01,
  cv.folds = 5,
  n.minobsinnode = 10,
  ...
)
```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain the predicted survivals.
obsWeights	Observation weights.
id	Optional cluster/individual ID indicator.
n.trees	Integer specifying the total number of trees to fit (default: 1000).
interaction.depth	Maximum depth of variable interactions (default: 2).
shrinkage	A shrinkage parameter applied to each tree (default: 0.01).
cv.folds	Number of cross-validation folds to perform internally for optimal tree selection (default: 5).
n.minobsinnode	Minimum number of observations in the trees terminal nodes (default: 10).
...	Additional arguments passed to gbm .

Value

A list containing:

- fit: The fitted model object (e.g., the raw coxph or xgb.Booster object). If the model fails to fit, this may be an object of class try-error.
- pred: A numeric matrix of cross-validated survival predictions evaluated at the specified new.times grid.

Examples

```
if (requireNamespace("gbm", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.gbm(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    n.trees = 20,
    interaction.depth = 1,
```

```

    shrinkage = 0.05,
    cv.folds = 0,
    n.minobsinnode = 3
  )

  dim(fit$pred)
}

```

surv.glmnet

Wrapper function for Penalized Cox Regression (GLMNET)

Description

Final Production Wrapper for GLMNET (Tunable & Robust). Estimates a penalized Cox model (Lasso, Ridge, or Elastic Net) with automatic lambda selection. Uses the Breslow estimator with a step-function approach for the baseline hazard.

Usage

```

surv.glmnet(
  time,
  event,
  X,
  newdata,
  new.times,
  obsWeights,
  id,
  alpha = 1,
  nfolds = 10,
  ...
)

```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain the predicted survivals.
obsWeights	Observation weights.
id	Optional cluster/individual ID indicator.
alpha	The elasticnet mixing parameter (0 = Ridge, 1 = Lasso). Default is 1.
nfolds	Number of folds for internal cross-validation to select lambda. Default is 10.
...	Additional arguments passed to cv.glmnet .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.glmnet(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    alpha = 1,
    nfolds = 3
  )

  dim(fit$pred)
}
```

surv.km

Kaplan-Meier Prediction Algorithm

Description

This prediction algorithm ignores all covariates and computes the marginal Kaplan-Meier survival estimator using the [survfit](#) function.

Usage

```
surv.km(time, event, X, newdata, new.times, obsWeights, id, ...)
```

Arguments

time	Numeric vector of observed follow-up times.
event	Numeric vector of event indicators (1 = event, 0 = censored).
X	Training covariate data.frame (Ignored by KM).
newdata	Test covariate data.frame to use for prediction.
new.times	Numeric vector of times at which to predict survival.
obsWeights	Numeric vector of observation weights.
id	Optional vector indicating subject/cluster identities.
...	Additional ignored arguments.

Value

A list containing:

- fit: A list containing the fitted `survfit` object.
- pred: A numeric matrix of cross-validated survival predictions evaluated at `new.times`.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:30, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:5, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.km(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
  obsWeights = rep(1, nrow(dat)),
  id = NULL
)

dim(fit$pred)
```

surv.loglogistic

Parametric Survival Prediction Wrapper (Log-Logistic)

Description

Parametric Survival Prediction Wrapper (Log-Logistic)

Usage

```
surv.loglogistic(time, event, X, newdata, new.times, obsWeights, id, ...)
```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain the predicted survivals.
obsWeights	Observation weights.
id	Cluster identification variable.
...	Additional ignored arguments.

Value

A list containing the fitted model and predictions.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:30, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:5, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.loglogistic(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
  obsWeights = rep(1, nrow(dat)),
  id = NULL
)

dim(fit$pred)
```

surv.lognormal

Parametric Survival Prediction Wrapper (Log-Normal)

Description

Parametric Survival Prediction Wrapper (Log-Normal)

Usage

```
surv.lognormal(time, event, X, newdata, new.times, obsWeights, id, ...)
```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain the predicted survivals.
obsWeights	Observation weights.
id	Cluster identification variable.
...	Additional ignored arguments.

Value

A list containing the fitted model and predictions.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:30, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:5, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.lognormal(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
  obsWeights = rep(1, nrow(dat)),
  id = NULL
)

dim(fit$pred)
```

surv.parametric

Universal Parametric Survival Wrapper

Description

Final Production Wrapper for AFT Models (Weibull, Exponential, LogNormal, LogLogistic). Replaces individual wrappers with one robust, vectorized function.

Usage

```
surv.parametric(
  time,
  event,
  X,
  newdata,
  new.times,
  obsWeights,
  id,
  dist = "weibull",
  ...
)
```

Arguments

<code>time</code>	Observed follow-up time.
<code>event</code>	Observed event indicator.
<code>X</code>	Training covariate data.frame.
<code>newdata</code>	Test covariate data.frame to use for prediction.
<code>new.times</code>	Times at which to obtain predicted survivals.
<code>obsWeights</code>	Observation weights.
<code>id</code>	Optional cluster/individual ID indicator.
<code>dist</code>	Distribution for the AFT model (default: "weibull").
<code>...</code>	Additional arguments passed to survreg .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:30, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:5, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.parametric(
  time = dat$duration,
  event = dat$event,
  X = X,
```

```

    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    dist = "weibull"
  )

  dim(fit$pred)

```

surv.ranger

Wrapper function for Ranger Random Survival Forest

Description

Final Production Wrapper for Ranger (Tunable & Fast). Uses the [ranger](#) C++ implementation to estimate survival curves.

Usage

```

surv.ranger(
  time,
  event,
  X,
  newdata,
  new.times,
  obsWeights,
  id,
  num.trees = 500,
  mtry = NULL,
  min.node.size = NULL,
  ...
)

```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain predicted survivals.
obsWeights	Observation weights.
id	Optional cluster/individual ID indicator.
num.trees	Number of trees (default: 500).
mtry	Number of variables to split at each node. Defaults to \sqrt{p} .
min.node.size	Minimum node size (default: 15 for survival).
...	Additional arguments passed to ranger .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```
if (requireNamespace("ranger", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.ranger(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,
    num.trees = 10,
    min.node.size = 3
  )

  dim(fit$pred)
}
```

surv.rfsrc

Wrapper function for Random Survival Forests (RFSRC)

Description

Final Production Wrapper for RFSRC (Tunable & Robust). Estimates a survival random forest using [rfsrc](#).

Usage

```
surv.rfsrc(
  time,
  event,
  X,
  newdata = NULL,
```

```

    new.times,
    obsWeights = NULL,
    id = NULL,
    ntree = 1000,
    nodesize = 15,
    mtry = NULL,
    ...
  )

```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain the predicted survivals.
obsWeights	Observation weights.
id	Currently ignored.
ntree	Number of trees to grow (default: 1000).
nodesize	Minimum number of deaths in terminal nodes (default: 15).
mtry	Number of variables randomly selected as candidates for splitting a node.
...	Additional arguments passed to rfsrc .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```

if (requireNamespace("randomForestSRC", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.rfsrc(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,

```

```

    new.times = times,
    ntree = 10,
    nodesize = 3
  )

  dim(fit$pred)
}

```

surv.ridge

Wrapper for Ridge Regression (Penalized Cox)

Description

Final Production Wrapper for Ridge Regression (Tunable & Robust). Estimates a penalized Cox model using a pure Ridge penalty ($\alpha = 0$).

Usage

```

surv.ridge(
  time,
  event,
  X,
  newdata,
  new.times,
  obsWeights = NULL,
  id = NULL,
  nfolds = 10,
  ...
)

```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain the predicted survivals.
obsWeights	Observation weights.
id	Optional cluster/individual ID indicator.
nfolds	Number of folds for internal cross-validation to select lambda. Default is 10.
...	Additional arguments passed to cv.glmnet .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.ridge(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    nfolds = 3
  )

  dim(fit$pred)
}
```

surv.rpart

Wrapper for Survival Regression Trees (rpart)

Description

Final Production Wrapper for single decision trees. Uses `rpart` with `method="exp"` and calculates survival probabilities using the Breslow estimator.

Usage

```
surv.rpart(
  time,
  event,
  X,
  newdata,
  new.times,
  obsWeights,
  id,
```

```

    cp = 0.01,
    minsplit = 20,
    maxdepth = 30,
    ...
)

```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain predicted survivals.
obsWeights	Observation weights.
id	Optional cluster/individual ID indicator.
cp	Complexity parameter (default: 0.01).
minsplit	Minimum number of observations to attempt a split (default: 20).
maxdepth	Maximum depth of any node of the final tree (default: 30).
...	Additional arguments passed to rpart.control .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```

if (requireNamespace("rpart", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.rpart(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times,
    obsWeights = rep(1, nrow(dat)),
    id = NULL,

```

```

      cp = 0.01,
      minsplit = 5,
      maxdepth = 3
    )

    dim(fit$pred)
  }

```

surv.svm

Wrapper for Survival Support Vector Machine (survivalsvm)

Description

Final Production Wrapper for SVM (Tunable & Robust). Estimates a survival SVM and calibrates the raw utility scores into survival probabilities using a univariate Cox proportional hazards model.

Usage

```

surv.svm(
  time,
  event,
  X,
  newdata,
  new.times,
  obsWeights,
  id,
  gamma.mu = 0.1,
  type = "vanbelle2",
  kernel = "lin_kernel",
  opt.meth = "quadprog",
  ...
)

```

Arguments

time	Observed follow-up time.
event	Observed event indicator.
X	Training covariate data.frame.
newdata	Test covariate data.frame to use for prediction.
new.times	Times at which to obtain the predicted survivals.
obsWeights	Observation weights.
id	Optional cluster/individual ID indicator.
gamma.mu	Regularization parameter for the SVM (default: 0.1).
type	Type of SVM implementation (default: "vanbelle2").
kernel	Kernel type for the SVM (default: "lin_kernel").
opt.meth	Optimization method (default: "quadprog").
...	Additional arguments passed to survivalsvm .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```
if (requireNamespace("survivalsvm", quietly = TRUE) &&
    requireNamespace("quadprog", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:25, ]
  x_cols <- grep("^x", names(dat))[1:3]
  X <- dat[, x_cols, drop = FALSE]
  newX <- X[1:5, , drop = FALSE]
  times <- seq(50, 150, by = 50)

  fit <- surv.svm(
    time = dat$duration,
    event = dat$event,
    X = X,
    newdata = newX,
    new.times = times
  )

  dim(fit$pred)
}
```

surv.weibull

Parametric Survival Prediction Wrapper (Weibull)

Description

Parametric Survival Prediction Wrapper (Weibull)

Usage

```
surv.weibull(time, event, X, newdata, new.times, obsWeights, id, ...)
```

Arguments

<code>time</code>	Observed follow-up time.
<code>event</code>	Observed event indicator.
<code>X</code>	Training covariate data.frame.
<code>newdata</code>	Test covariate data.frame to use for prediction.

<code>new.times</code>	Times at which to obtain the predicted survivals.
<code>obsWeights</code>	Observation weights.
<code>id</code>	Cluster identification variable.
<code>...</code>	Additional ignored arguments.

Value

A list containing the fitted model and predictions.

Examples

```
data("metabric", package = "SuperSurv")
dat <- metabric[1:30, ]
x_cols <- grep("^x", names(dat))[1:3]
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:5, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.weibull(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
  obsWeights = rep(1, nrow(dat)),
  id = NULL
)

dim(fit$pred)
```

surv.xgboost

Wrapper for XGBoost (Robust CV-Tuned + Safe Prediction)

Description

Estimates a Cox proportional hazards model via XGBoost. Incorporates safe Breslow hazard calculation and matrix alignment to prevent C++ crashes.

Usage

```
surv.xgboost(
  time,
  event,
  X,
  newdata = NULL,
  new.times,
  obsWeights,
  id,
```

```

nrounds = 1000,
early_stopping_rounds = 10,
eta = 0.05,
max_depth = 2,
min_child_weight = 5,
lambda = 10,
subsample = 0.7,
...
)

```

Arguments

<code>time</code>	Observed follow-up time.
<code>event</code>	Observed event indicator.
<code>X</code>	Training covariate data.frame.
<code>newdata</code>	Test covariate data.frame to use for prediction.
<code>new.times</code>	Times at which to obtain the predicted survivals.
<code>obsWeights</code>	Observation weights.
<code>id</code>	Optional cluster/individual ID indicator.
<code>nrounds</code>	Max number of boosting iterations (default: 1000).
<code>early_stopping_rounds</code>	Rounds with no improvement to trigger early stopping (default: 10).
<code>eta</code>	Learning rate (default: 0.05).
<code>max_depth</code>	Maximum tree depth (default: 2).
<code>min_child_weight</code>	Minimum sum of instance weight in a child (default: 5).
<code>lambda</code>	L2 regularization term on weights (default: 10).
<code>subsample</code>	Subsample ratio of the training instances (default: 0.7).
<code>...</code>	Additional arguments passed to <code>xgb.train</code> .

Value

A list containing:

- `fit`: The fitted model object (e.g., the raw `coxph` or `xgb.Booster` object). If the model fails to fit, this may be an object of class `try-error`.
- `pred`: A numeric matrix of cross-validated survival predictions evaluated at the specified `new.times` grid.

Examples

```

if (requireNamespace("xgboost", quietly = TRUE)) {
  data("metabric", package = "SuperSurv")
  dat <- metabric[1:30, ]
  x_cols <- grep("^x", names(dat))[1:3]

```

```
X <- dat[, x_cols, drop = FALSE]
newX <- X[1:5, , drop = FALSE]
times <- seq(50, 150, by = 50)

fit <- surv.xgboost(
  time = dat$duration,
  event = dat$event,
  X = X,
  newdata = newX,
  new.times = times,
  obsWeights = rep(1, nrow(dat)),
  id = NULL,
  nrounds = 5,
  early_stopping_rounds = 2,
  max_depth = 1
)

dim(fit$pred)
}
```

Index

* datasets

metabric, 12

CoxBoost, 53

coxph, 54

cv.glmnet, 41, 59, 68

estimate_marginal_rmst, 3

eval_brier, 4

eval_cindex, 5

eval_summary, 6

eval_timeROC, 8

explain_kernel, 9

explain_survex, 10

explain_survival, 10

gam, 56

gbm, 58

list_wrappers, 11

mc.surv.bart, 51

metabric, 12

orsf, 50

plot_beeswarm, 13

plot_benchmark, 14

plot_calibration, 15

plot_dependence, 16

plot_global_importance, 17

plot_marginal_rmst_curve, 19

plot_patient_waterfall, 20

plot_predict, 21

plot_rmst_vs_obs, 22

plot_survival_heatmap, 23

predict.SuperSurv, 24

predict.surv.aorsf, 25

predict.surv.bart, 26

predict.surv.coxboost, 27

predict.surv.coxph, 28

predict.surv.gam, 29

predict.surv.gbm, 30

predict.surv.glmnet, 31

predict.surv.km, 32

predict.surv.parametric, 33

predict.surv.ranger, 34

predict.surv.rfsrc, 35

predict.surv.ridge, 36

predict.surv.rpart, 37

predict.surv.svm, 39

predict.surv.xgboost, 40

ranger, 65

rfsrc, 45, 66, 67

rpart, 69

rpart.control, 70

screen.all, 41

screen.elasticnet, 41

screen.glmnet, 43

screen.marg, 44

screen.rfsrc, 45

screen.var, 46

SuperSurv, 47

surv.aorsf, 49

surv.bart, 51

surv.coxboost, 52

surv.coxph, 54

surv.exponential, 55

surv.gam, 56

surv.gbm, 57

surv.glmnet, 59

surv.km, 60

surv.loglogistic, 61

surv.lognormal, 62

surv.parametric, 63

surv.ranger, 65

surv.rfsrc, 66

surv.ridge, 68

surv.rpart, 69

surv.svm, [71](#)
surv.weibull, [72](#)
surv.xgboost, [73](#)
survfit, [60](#), [61](#)
survivalsvm, [71](#)
survreg, [64](#)

xgb.train, [74](#)