Package 'survkl'

May 9, 2026

Title Estimate Survival Data with Data Integration

Version 1.0.0

URL https://um-kevinhe.github.io/survkl/

Description Provides flexible and efficient tools for integrating external risk scores into Cox proportional hazards models while accounting for population heterogeneity. Enables robust estimation, improved predictive accuracy, and user-friendly workflows for modern survival analysis. For more information, see Wang et al. (2023) <doi:10.48550/arXiv.2302.11123>.

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

License GPL-3

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Imports Rcpp, ggplot2, stats, cowplot, Matrix, rlang

Depends R (>= 4.0)

Suggests knitr, rmarkdown, survival

VignetteBuilder knitr

SystemRequirements GNU make

NeedsCompilation yes

Author Yubo Shao [aut, cre],
Lingfeng Luo [aut],
Xiaohan Liu [aut],
Junyi Qiu [aut],
Di Wang [aut],
Kevin He [aut]

Maintainer Yubo Shao <ybshao@umich.edu>

Repository CRAN

Date/Publication 2026-04-21 18:00:02 UTC

Contents

`cal_surv_prob` *Calculate Survival Probabilities*

Description

Computes individual survival probabilities from a fitted linear predictor `z%*%beta` using a stratified Breslow-type baseline hazard estimate.

Usage

```
cal_surv_prob(z, delta, time, beta, stratum)
```

Arguments

`z`	A numeric matrix (or data frame coercible to matrix) of covariates. Each row is an observation and each column a predictor.
`delta`	A numeric vector of event indicators (1 = event, 0 = censored).
`time`	A numeric vector of observed times (event or censoring).

beta	A numeric vector of regression coefficients with length equal to the number of columns in z.
stratum	An optional vector specifying the stratum for each observation. If missing, a single-stratum model is assumed.

Details

Inputs are internally sorted by stratum and time. Within each stratum, a baseline hazard increment is computed as delta/S0, where S0 is the risk set sum returned by ddloglik_S0. The stratified baseline cumulative hazard Lambda0 is then formed by a cumulative sum within stratum, and individual survival curves are computed as S(t) = exp(-Lambda0(t) * exp(z %*% beta)).

Value

A numeric matrix of survival probabilities with nrow(z) rows and length(time) columns. Rows correspond to observations; columns are in the internal sorted order of (stratum, time) (i.e., not collapsed to unique event times). Entry S[i, j] is the estimated survival probability for subject i evaluated at the j-th sorted time point.

---

coef.coxkl *Extract Coefficients from a* coxkl *Object*

Description

Extracts the estimated regression coefficients (beta) from a fitted coxkl object. Optionally, a value (or vector) of eta can be supplied. If the requested eta values are not in the fitted sequence, linear interpolation is performed between the nearest neighboring eta values; out-of-range requests error.

Usage

```
## S3 method for class 'coxkl'
coef(object, eta = NULL, ...)
```

Arguments

object	An object of class "coxkl", typically the result of coxkl.
eta	Optional numeric value or vector specifying the η values for which to extract (or interpolate) coefficients. If NULL, all estimated coefficients are returned.
...	Additional arguments (currently ignored).

Value

A numeric matrix of regression coefficients. Each column corresponds to one value of eta, sorted in ascending order.

Examples

```
data(ExampleData_lowdim)

train_dat_lowdim <- ExampleData_lowdim$train
beta_external_good_lowdim <- ExampleData_lowdim$beta_external_good
eta_list <- generate_eta(method = "exponential", n = 5, max_eta = 5)

model <- coxkl(z = train_dat_lowdim$z,
               delta = train_dat_lowdim$status,
               time = train_dat_lowdim$time,
               stratum = train_dat_lowdim$stratum,
               beta = beta_external_good_lowdim,
               etas = eta_list)
coef(model)
```

---

coef.coxkl_enet *Extract Coefficients from a* coxkl_enet *Object*

Description

Extracts the estimated regression coefficients (beta) from a fitted coxkl_enet object. Optionally, one or more lambda values can be supplied. If requested lambda values are not in the fitted sequence, linear interpolation is performed between nearest neighbors; out-of-range requests error.

Usage

```
## S3 method for class 'coxkl_enet'
coef(object, lambda = NULL, ...)
```

Arguments

object	An object of class "coxkl_enet", typically the result of coxkl_enet.
lambda	Optional numeric value or vector specifying the regularization parameter(s) for which to extract (or interpolate) coefficients. If NULL, all estimated coefficients are returned.
...	Additional arguments (currently ignored).

Value

A numeric matrix of regression coefficients; each column corresponds to one value of lambda, sorted in *descending* order.

Examples

```
data(ExampleData_highdim)

train_dat_highdim <- ExampleData_highdim$train
beta_external_highdim <- ExampleData_highdim$beta_external


enet_model <- coxkl_enet(z = train_dat_highdim$z,
                         delta = train_dat_highdim$status,
                         time = train_dat_highdim$time,
                         beta = beta_external_highdim,
                         eta = 1,
                         alpha = 1.0)
coef(enet_model)[1:5, 1:10]
```

`coef.coxkl_ridge` *Extract Coefficients from a* `coxkl_ridge` *Object*

Description

Extracts the estimated regression coefficients (`beta`) from a fitted `coxkl_ridge` object. Optionally, one or more `lambda` values can be supplied. If requested `lambda` values are not in the fitted sequence, linear interpolation is performed between nearest neighbors; out-of-range requests error.

Usage

```
## S3 method for class 'coxkl_ridge'
coef(object, lambda = NULL, ...)
```

Arguments

`object`	An object of class `"coxkl_ridge"`, typically the result of `coxkl_ridge`.
`lambda`	Optional numeric value or vector specifying the regularization parameter(s) for which to extract (or interpolate) coefficients. If `NULL`, all estimated coefficients are returned.
`...`	Additional arguments (currently ignored).

Value

A numeric matrix of regression coefficients. Each column corresponds to one value of `lambda`, sorted in *descending* order.

Examples

```
data(ExampleData_highdim)

train_dat_highdim <- ExampleData_highdim$train
beta_external_highdim <- ExampleData_highdim$beta_external

model_ridge <- coxkl_ridge(z = train_dat_highdim$z,
                           delta = train_dat_highdim$status,
                           time = train_dat_highdim$time,
                           beta = beta_external_highdim,
                           eta = 1)
coef(model_ridge)[1:5, 1:10]
```

`coxkl` *Cox Proportional Hazards Model with KL Divergence for Data Integration*

Description

Fits a Cox proportional hazards model that incorporates external information via a Kullback–Leibler (KL) divergence penalty. External information can be supplied either as external risk scores (`RS`) or as external coefficients (`beta`). The tuning parameter(s) `etas` control the strength of integration.

Usage

```
coxkl(
  z,
  delta,
  time,
  stratum = NULL,
  RS = NULL,
  beta = NULL,
  etas,
  tol = 1e-04,
  Mstop = 100,
  backtrack = FALSE,
  message = FALSE,
  data_sorted = FALSE,
  beta_initial = NULL
)
```

Arguments

`z`	Numeric matrix of covariates with rows representing observations and columns representing predictor variables. All covariates must be numeric.
`delta`	Numeric vector of event indicators (1 = event, 0 = censored).

time	Numeric vector of observed event or censoring times. No sorting required.
stratum	Optional numeric or factor vector defining strata.
RS	Optional numeric vector or matrix of external risk scores. Length (or number of rows) must equal the number of observations. If not supplied, beta must be provided.
beta	Optional numeric vector of external coefficients (e.g., from prior studies). Length must equal the number of columns in z. Use zeros to represent covariates without external information. If not supplied, RS must be provided.
etas	Numeric vector of tuning parameters controlling the reliance on external information. Larger values place more weight on the external source.
tol	Convergence tolerance for the optimization algorithm. Default is 1e-4.
Mstop	Maximum number of iterations for the optimization algorithm. Default is 100.
backtrack	Logical; if TRUE, backtracking line search is applied during optimization. Default is FALSE.
message	Logical; if TRUE, progress messages are printed during model fitting. Default is FALSE.
data_sorted	Logical; if TRUE, input data are assumed to be already sorted by stratum and time. Default is FALSE.
beta_initial	Optional numeric vector of length p giving the starting value for the first eta. If NULL, a zero vector is used.

Details

If beta is supplied (length ncol(z)), external risk scores are computed internally as RS = z %*% beta. If RS is supplied, it is used directly. Data are optionally sorted by stratum (or a single stratum if NULL) and increasing time when data_sorted = FALSE. Estimation proceeds over the sorted data, and the returned linear.predictors are mapped back to the original order. Optimization uses warm starts across the (ascending) etas grid and supports backtracking line search when backtrack = TRUE.

Internally, the routine computes a stratum-wise adjusted event indicator (delta_tilde) and maximizes a KL-regularized partial likelihood. The current implementation fixes lambda = 0 in the low-level optimizer and exposes etas as the primary tuning control.

Value

An object of class "coxkl" containing:

- eta: the fitted η sequence.
- beta: estimated coefficient matrix ($p \times |\eta|$).
- linear.predictors: matrix of linear predictors.
- likelihood: vector of partial likelihoods.
- data: a list containing the input data used in fitting (z, time, delta, stratum, data_sorted).

Examples

```
data(ExampleData_lowdim)

train_dat_lowdim <- ExampleData_lowdim$train
beta_external_good_lowdim <- ExampleData_lowdim$beta_external_good
eta_list <- generate_eta(method = "exponential", n = 10, max_eta = 5)

model <- coxkl(z = train_dat_lowdim$z,
               delta = train_dat_lowdim$status,
               time = train_dat_lowdim$time,
               stratum = train_dat_lowdim$stratum,
               beta = beta_external_good_lowdim,
               etas = eta_list)
```

`coxkl_enet` *Cox Proportional Hazards Model with KL Divergence for Data Integration and Lasso & Elastic Net Penalty*

Description

Fits a Cox proportional hazards model that incorporates external information using Kullback–Leibler (KL) divergence, with an optional L1 (Lasso) or elastic net penalty on the coefficients. External information can be supplied either as precomputed external risk scores (`RS`) or as externally derived coefficients (`beta`). The integration strength is controlled by the tuning parameter `eta`.

Usage

```
coxkl_enet(
  z,
  delta,
  time,
  stratum = NULL,
  RS = NULL,
  beta = NULL,
  eta = NULL,
  alpha = NULL,
  lambda = NULL,
  nlambda = 100,
  lambda.min.ratio = ifelse(n < p, 0.05, 0.001),
  lambda.early.stop = FALSE,
  tol = 1e-04,
  Mstop = 1000,
  max.total.iter = (Mstop * nlambda),
  group = 1:ncol(z),
  group.multiplier = NULL,
  standardize = TRUE,
```

```
  nvar.max = ncol(z),
  group.max = length(unique(group)),
  stop.loss.ratio = 0.001,
  actSet = TRUE,
  actIter = Mstop,
  actGroupNum = sum(unique(group) != 0),
  actSetRemove = FALSE,
  returnX = FALSE,
  trace.lambda = FALSE,
  message = FALSE,
  data_sorted = FALSE,
  ...
)
```

Arguments

z	Numeric matrix of covariates with rows representing observations and columns representing predictor variables. All covariates must be numeric.
delta	Numeric vector of event indicators (1 = event, 0 = censored).
time	Numeric vector of observed event or censoring times. No sorting required.
stratum	Optional numeric or factor vector defining strata.
RS	Optional numeric vector or matrix of external risk scores. Length (or number of rows) must equal the number of observations. If not supplied, beta must be provided.
beta	Optional numeric vector of external coefficients (e.g., from prior studies). Length must equal the number of columns in z. Use zeros to represent covariates without external information. If not supplied, RS must be provided.
eta	Numeric tuning parameter controlling the reliance on external information. Larger values place more weight on the external source.
alpha	Elastic-net mixing parameter in $(0, 1]$. When $\alpha = 1$ the penalty is lasso.
lambda	Optional nonnegative penalty parameter(s). If a numeric vector is supplied, the path is taken as-is. If NULL, a sequence is generated using nlambda and lambda.min.ratio.
nlambda	Integer number of lambda values to generate when lambda is NULL. Default 100.
lambda.min.ratio	Ratio of the smallest to the largest lambda when generating a sequence (when lambda is NULL). Default 1e-3.
lambda.early.stop	Logical; if TRUE, stop traversing the lambda path early based on convergence or screening criteria. Default FALSE.
tol	Convergence tolerance for the optimization algorithm. Default is 1e-3.
Mstop	Maximum number of iterations for the inner optimization at a given lambda. Default is 1000.
max.total.iter	Maximum total iterations across the entire lambda path. Default is (Mstop * nlambda).

group	Integer vector of group indices defining group membership of predictors for grouped penalties; use 0 to indicate unpenalized variables.
group.multiplier	A vector of values representing multiplicative factors by which each covariate's penalty is to be multiplied. Default is a vector of 1's.
standardize	Logical; if TRUE, columns of z are standardized prior to fitting, with coefficients re-scaled on output. Default TRUE.
nvar.max	Integer cap on the number of active variables allowed during fitting. Default number of predictors.
group.max	Integer cap on the number of active groups allowed during fitting. Default total number of groups.
stop.loss.ratio	Relative improvement threshold for early stopping along the path; optimization may stop if objective gain falls below this value. Default 1e-3.
actSet	Logical; if TRUE, use an active-set strategy. Default TRUE.
actIter	Maximum number of active-set refinement iterations per lambda. Default Mstop.
actGroupNum	Maximum number of active groups allowed under the active-set scheme.
actSetRemove	Logical; if TRUE, allow dropping variables/groups from the active set during iterations. Default FALSE.
returnX	Logical; if TRUE, return standardized design and related internals in result$returnX. Default FALSE.
trace.lambda	Logical; if TRUE, record path-wise traces across the lambda sequence. Default FALSE.
message	Logical; if TRUE, progress messages are printed during model fitting. Default is FALSE.
data_sorted	Logical; if TRUE, input is assumed already sorted by stratum then time. Default FALSE.
...	Additional arguments.

Details

Setting lambda = 0 reduces to the unpenalized coxkl model.

When lambda > 0, the model fits a KL-regularized Cox objective with an elastic-net penalty:

$$\ell_{\mathrm{KL}}(\beta;\eta) \;-\; \lambda\Big\{\alpha\|\beta\|_1 \;+\; (1-\alpha)\tfrac{1}{2}\|\beta\|_2^2\Big\},$$

where $\alpha = 1$ gives lasso and $0 < \alpha < 1$ gives elastic net. Grouped penalties are supported via group (use 0 for unpenalized variables), with optional per-group scaling through group.multiplier. If lambda is NULL, a decreasing path of length nlambda is generated using lambda.min.ratio; early stopping can prune the path (lambda.early.stop, stop.loss.ratio). When standardize = TRUE, predictors are standardized for fitting and coefficients are rescaled on output. If data_sorted = FALSE, data are sorted by stratum then time for optimization and predictions are returned in the original order (reported via W = exp(linear predictors)). An active-set scheme (actSet, actIter, nvar.max, group.max, actGroupNum, actSetRemove) is used to accelerate the solution along the lambda path.

Value

An object of class "coxkl_enet", a list with components:

beta Coefficient estimates (vector or matrix across the path).

group A factor of the original group assignments.

lambda The lambda value(s) used or generated.

alpha The elastic-net mixing parameter used.

likelihood Vector of log-partial likelihoods for each lambda.

n Number of observations.

df Effective degrees of freedom (e.g., number of nonzero coefficients or group-adjusted count) along the path.

iter Number of iterations taken (per lambda and/or total).

W Exponentiated linear predictors on the original scale.

group.multiplier Group-specific penalty multipliers used.

returnX Only when returnX = TRUE: a list with elements XX (standardization/orthogonalization info from std.Z), time, delta, stratum, and RS.

See Also

coxkl

Examples

```
data(ExampleData_highdim)

train_dat_highdim <- ExampleData_highdim$train
beta_external_highdim <- ExampleData_highdim$beta_external

model_enet <- coxkl_enet(z = train_dat_highdim$z,
                         delta = train_dat_highdim$status,
                         time = train_dat_highdim$time,
                         beta = beta_external_highdim,
                         eta = 0,
                         alpha = 1.0)
```

---

coxkl_ridge *Cox Proportional Hazards Model with Ridge Penalty and External Information*

Description

Fits a Cox proportional hazards model using a ridge-type penalty (L2) on all covariates. The model can integrate external information either as precomputed risk scores (RS) or externally supplied coefficients (beta). A tuning parameter eta controls the relative weight of the external information. If lambda is not provided, a lambda sequence is automatically generated.

Usage

```
coxkl_ridge(
  z,
  delta,
  time,
  stratum = NULL,
  RS = NULL,
  beta = NULL,
  eta = NULL,
  lambda = NULL,
  nlambda = 100,
  lambda.min.ratio = ifelse(n_obs < n_vars, 0.01, 1e-04),
  penalty.factor = 0.999,
  tol = 1e-04,
  Mstop = 50,
  backtrack = FALSE,
  message = FALSE,
  data_sorted = FALSE,
  beta_initial = NULL,
  ...
)
```

Arguments

z	Numeric matrix of covariates (observations in rows, predictors in columns).
delta	Numeric vector of event indicators (1 = event, 0 = censored).
time	Numeric vector of observed times.
stratum	Optional numeric or factor vector specifying strata.
RS	Optional numeric vector or matrix of external risk scores.
beta	Optional numeric vector of externally derived coefficients.
eta	Non-negative scalar controlling the strength of external information.
lambda	Optional numeric scalar or vector of penalty parameters. If NULL, a sequence is generated automatically.
nlambda	Number of lambda values to generate if lambda is NULL.
lambda.min.ratio	Ratio defining the minimum lambda relative to lambda.max.
penalty.factor	Numeric scalar in [0, 1). Controls the overall strength of the penalty when generating the ridge regression lambda sequence. Smaller values correspond to stronger penalization. Only used when lambda = NULL.
tol	Convergence tolerance for the iterative estimation algorithm.
Mstop	Maximum number of iterations for estimation.
backtrack	Logical; if TRUE, uses backtracking line search.
message	Logical; if TRUE, progress messages are printed during model fitting. Default is FALSE.

data_sorted	Logical; if TRUE, assumes input data is already sorted by strata and time.
beta_initial	Optional; default NULL. When NULL, the algorithm initializes beta_initial to a zero vector as a warm start
...	Additional arguments.

Details

The estimator maximizes a KL-regularized Cox partial log-likelihood with a ridge (L2) penalty on all coefficients. External information is incorporated via a KL term weighted by eta: if beta is supplied (length ncol(z)), external risk scores are computed internally as RS = z %*% beta; otherwise RS must be provided. If lambda is NULL, a decreasing lambda path of length nlambda is generated using lambda.min.ratio (its overall scale is influenced by penalty.factor). Optimization proceeds along the lambda path with warm starts (re-using the previous solution as beta_initial); when beta_initial = NULL, the first step uses zeros. If data_sorted = FALSE, data are sorted by stratum and time for fitting and the returned linear predictors are mapped back to the original observation order. tol, Mstop, and backtrack control convergence and line search.

Value

An object of class "coxkl_ridge" containing:

- lambda: The lambda sequence used for estimation.
- beta: Matrix of estimated coefficients for each lambda.
- linear.predictors: Matrix of linear predictors.
- likelihood: Vector of log-partial likelihoods.
- data: A list containing the input data used in fitting (z, time, delta, stratum, data_sorted).

Examples

```
data(ExampleData_highdim)

train_dat_highdim <- ExampleData_highdim$train
beta_external_highdim <- ExampleData_highdim$beta_external

model_ridge <- coxkl_ridge(z = train_dat_highdim$z,
                           delta = train_dat_highdim$status,
                           time = train_dat_highdim$time,
                           beta = beta_external_highdim)
```

---

cv.coxkl *Cross-Validated Selection of Integration Parameter (eta) for the Cox–KL Model*

Description

Performs K-fold cross-validation to select the integration parameter `eta` for the Cox–KL model. Each fold fits the model on a training split and evaluates on the held-out split using the specified performance criterion.

Usage

```
cv.coxkl(
  z,
  delta,
  time,
  stratum = NULL,
  RS = NULL,
  beta = NULL,
  etas = NULL,
  tol = 1e-04,
  Mstop = 100,
  backtrack = FALSE,
  nfolds = 5,
  criteria = c("V&VH", "LinPred", "CIndex_pooled", "CIndex_foldaverage"),
  c_index_stratum = NULL,
  message = FALSE,
  seed = NULL,
  ...
)
```

Arguments

`z`	Numeric matrix of covariates (rows = observations, columns = variables).
`delta`	Numeric vector of event indicators (1 = event, 0 = censored).
`time`	Numeric vector of observed event or censoring times.
`stratum`	Optional numeric or factor vector defining strata. If `NULL`, all observations are treated as a single stratum.
`RS`	Optional numeric vector or matrix of external risk scores. If omitted, `beta` must be supplied.
`beta`	Optional numeric vector of external coefficients. If omitted, `RS` must be supplied.
`etas`	Numeric vector of candidate tuning values to be cross-validated. (required). Values are internally sorted in ascending order.
`tol`	Convergence tolerance for the optimizer used inside `coxkl`. Default `1e-4`.
`Mstop`	Maximum number of Newton iterations used inside `coxkl`. Default `100`.
`backtrack`	Logical; if `TRUE`, backtracking line search is applied during optimization. Default is `FALSE`.
`nfolds`	Number of cross-validation folds. Default `5`.

criteria	Character string specifying the performance criterion. Choices are "V&VH" (default), "LinPred", "CIndex_pooled", or "CIndex_foldaverage".
c_index_stratum	Optional stratum vector. Only required when criteria is set to "CIndex_pooled" or "CIndex_foldaverage", and a stratified C-index is desired while the fitted model is non-stratified. Default NULL.
message	Logical; if TRUE, prints progress messages and per-fold progress bars. Default FALSE.
seed	Optional integer seed for reproducible fold assignment. Default NULL.
...	Additional arguments passed to coxkl.

Details

External information is required: supply either RS or beta (if beta is given, RS is computed as z %*% beta). Folds are created with stratification by stratum and censoring status. Within each fold and each candidate eta, the function fits coxkl on the training split with warm-starts initialized to zero and evaluates on the test split:

- "V&VH": uses the difference of partial log-likelihoods between full and training fits; reported as -2 times the aggregated quantity.
- "LinPred": aggregates the test-split linear predictors across folds and evaluates -2 times the partial log-likelihood on the full data.
- "CIndex_pooled": pools pairwise comparable counts across folds (numerator/denominator).
- "CIndex_foldaverage": averages the per-fold stratified C-index.

The function also computes an external baseline statistic from RS using the same criterion for comparison.

Value

An object of class "cv.coxkl" with components:

internal_stat A data.frame with one row per eta containing eta and the cross-validated measure named according to criteria (one of VVH_Loss, LinPred_Loss, CIndex_pooled, CIndex_foldaverage).

external_stat Scalar baseline statistic computed from RS under the same criteria.

criteria The evaluation criterion used.

nfolds Number of folds.

Examples

```
data(ExampleData_lowdim)

train_dat_lowdim <- ExampleData_lowdim$train
beta_external_good_lowdim <- ExampleData_lowdim$beta_external_good

etas <- generate_eta(method = "exponential", n = 10, max_eta = 5)

cv_res <- cv.coxkl(z = train_dat_lowdim$z,
```

```
delta = train_dat_lowdim$status,
time = train_dat_lowdim$time,
beta = beta_external_good_lowdim,
etas = etas)
```

`cv.coxkl_enet` *Cross-Validation for CoxKL Model with elastic net & lasso penalty*

Description

This function performs cross-validation on the high-dimensional Cox model with Kullback–Leibler (KL) penalty. It tunes the parameter `eta` (external information weight) using user-specified cross-validation criteria, while also evaluating a `lambda` path (either provided or generated) and selecting the best `lambda` per `eta`.

Usage

```
cv.coxkl_enet(
  z,
  delta,
  time,
  stratum = NULL,
  RS = NULL,
  beta = NULL,
  etas,
  alpha = 1,
  lambda = NULL,
  nlambda = 100,
  lambda.min.ratio = ifelse(n < p, 0.05, 0.001),
  nfolds = 5,
  cv.criteria = c("V&VH", "LinPred", "CIndex_pooled", "CIndex_foldaverage"),
  c_index_stratum = NULL,
  message = FALSE,
  seed = NULL,
  ...
)
```

Arguments

`z`	Numeric matrix of covariates with rows representing individuals and columns representing predictors.
`delta`	Numeric vector of event indicators (1 = event, 0 = censored).
`time`	Numeric vector of observed times (event or censoring).
`stratum`	Optional factor or numeric vector indicating strata.
`RS`	Optional numeric vector or matrix of external risk scores. If not provided, `beta` must be supplied.

beta	Optional numeric vector of external coefficients (length equal to ncol(z)). If not provided, RS must be supplied.
etas	Numeric vector of candidate eta values to be evaluated.
alpha	Elastic-net mixing parameter in $(0, 1]$. Default = 1 (lasso penalty).
lambda	Optional numeric scalar or vector of penalty parameters. If NULL, a decreasing path is generated using nlambda and lambda.min.ratio.
nlambda	Integer number of lambda values to generate when lambda is NULL. Default 100.
lambda.min.ratio	Ratio of the smallest to the largest lambda when generating a sequence (when lambda is NULL). Default 0.05 when n < p, otherwise 1e-3.
nfolds	Integer; number of cross-validation folds. Default = 5.
cv.criteria	Character string specifying the cross-validation criterion. Choices are: • "V&VH" (default): "V&VH" loss. • "LinPred": loss based on cross-validated linear predictors. • "CIndex_pooled": pool all held-out predictions and compute one overall C-index. • "CIndex_foldaverage": average C-index across folds.
c_index_stratum	Optional stratum vector. Used only when cv.criteria is "CIndex_pooled" or "CIndex_foldaverage" to compute a stratified C-index while the fitted model is non-stratified; if supplied, it must be identical to stratum. Default NULL.
message	Logical; whether to print progress messages. Default = FALSE.
seed	Optional integer random seed for fold assignment.
...	Additional arguments passed to coxkl_enet.

Details

Data are sorted by stratum and time. External info must be from RS or beta (if beta given with length ncol(z), RS = z %*% beta); alpha $\in (0, 1]$.

For each candidate eta, a decreasing lambda path is used (generated from nlambda/lambda.min.ratio if lambda = NULL); CV folds are created by get_fold. Each fold fits coxkl_enet on the training split (full lambda path) and evaluates the chosen criterion on the test split.

Aggregation follows the code paths for "V&VH", "LinPred", "CIndex_pooled", or "CIndex_foldaverage":

- "V&VH": sums pl(full) - pl(train) across folds (reported as loss via Loss = -2 * score).
- "LinPred": aggregates test-fold linear predictors and evaluates partial log-likelihood on full data (reported as Loss = -2 * score).
- "CIndex_pooled": pools comparable-pair numerators/denominators across folds to compute one C-index.
- "CIndex_foldaverage": averages the per-fold stratified C-index.

The best lambda is selected per eta (min loss / max C-index), and the function returns full results, the per-eta optimum, corresponding coefficients, and an external baseline from RS.

Value

An object of class "cv.coxkl_enet":

integrated_stat.full_results Data frame with columns eta, lambda, and the aggregated CV score for each lambda under the chosen cv.criteria. For loss criteria, an additional column with the transformed loss (Loss = -2 * score); for C-index criteria, a column named CIndex_pooled or CIndex_foldaverage.

integrated_stat.best_per_eta Data frame with the best lambda (per eta) according to the chosen cv.criteria (minimizing loss or maximizing C-index).

integrated_stat.betahat_best Matrix of coefficient vectors (columns) corresponding to the best lambda for each eta.

external_stat Scalar baseline statistic computed from the external risk score RS under the same cv.criteria.

criteria The evaluation criterion used (as provided in cv.criteria).

alpha The elastic-net mixing parameter used.

nfolds Number of folds.

Examples

```
data(ExampleData_highdim)

train_dat_highdim <- ExampleData_highdim$train
beta_external_highdim <- ExampleData_highdim$beta_external

etas <- generate_eta(method = "exponential", n = 10, max_eta = 100)

cv_res <- cv.coxkl_enet(z = train_dat_highdim$z,
                        delta = train_dat_highdim$status,
                        time = train_dat_highdim$time,
                        stratum = NULL,
                        RS = NULL,
                        beta = beta_external_highdim,
                        etas = etas,
                        alpha = 1.0)
```

---

cv.coxkl_ridge *Cross-Validation for CoxKL Ridge Model (eta tuning)*

Description

This function performs cross-validation on the Cox model with Kullback–Leibler (KL) penalty and ridge (L2) regularization. It tunes the parameter eta (external information weight) using user-specified cross-validation criteria, while internally evaluating a lambda path (provided or generated) and selecting the best lambda per eta

Usage

```
cv.coxkl_ridge(
  z,
  delta,
  time,
  stratum = NULL,
  RS = NULL,
  beta = NULL,
  etas,
  lambda = NULL,
  nlambda = 100,
  lambda.min.ratio = ifelse(n_obs < n_vars, 0.01, 1e-04),
  nfolds = 5,
  cv.criteria = c("V&VH", "LinPred", "CIndex_pooled", "CIndex_foldaverage"),
  c_index_stratum = NULL,
  message = FALSE,
  seed = NULL,
  ...
)
```

Arguments

`z`	Numeric matrix of covariates with rows representing individuals and columns representing predictors.
`delta`	Numeric vector of event indicators (1 = event, 0 = censored).
`time`	Numeric vector of observed times (event or censoring).
`stratum`	Optional factor or numeric vector indicating strata.
`RS`	Optional numeric vector or matrix of external risk scores. If not provided, `beta` must be supplied.
`beta`	Optional numeric vector of external coefficients (length equal to `ncol(z)`). If not provided, `RS` must be supplied.
`etas`	Numeric vector of candidate `eta` values to be evaluated.
`lambda`	Optional numeric scalar or vector of penalty parameters. If `NULL`, a sequence is generated automatically.
`nlambda`	Integer number of lambda values to generate when `lambda` is `NULL`. Default `100`.
`lambda.min.ratio`	Ratio of the smallest to the largest lambda when generating a sequence (when `lambda` is `NULL`). Default `0.01` when `n < p`, otherwise `1e-4`.
`nfolds`	Integer; number of cross-validation folds. Default `5`.
`cv.criteria`	Character string specifying the cross-validation criterion. Choices are: • `"V&VH"` (default): `"V&VH"` loss. • `"LinPred"`: loss based on cross-validated linear predictors. • `"CIndex_pooled"`: pool all held-out predictions and compute one overall C-index.

- "CIndex_foldaverage": average C-index across folds.

c_index_stratum
: Optional stratum vector. Used only when cv.criteria is "CIndex_pooled" or "CIndex_foldaverage" to compute a stratified C-index for a non-stratified fit; if supplied, it must be identical to stratum. Default NULL.

message
: Logical; whether to print progress messages. Default FALSE.

seed
: Optional integer random seed for fold assignment.

...
: Additional arguments passed to coxkl_ridge.

Details

Data are sorted by stratum and time. External information must be given via RS or beta (if beta has length ncol(z), the function computes RS = z %*% beta). For each candidate eta, a lambda path is determined (generated if lambda = NULL, otherwise the supplied lambda values are sorted decreasingly). Cross-validation folds are created by get_fold. In each fold, coxkl_ridge is fit on the training split across the full lambda path with data_sorted = TRUE, and the chosen criterion is evaluated on the test split and aggregated:

- "V&VH": sums pl(full) - pl(train) across folds (reported as loss via Loss = -2 * score).
- "LinPred": aggregates test-fold linear predictors and evaluates partial log-likelihood on full data (reported as Loss = -2 * score).
- "CIndex_pooled": pools comparable-pair numerators/denominators across folds to compute one C-index.
- "CIndex_foldaverage": averages the per-fold stratified C-index.

The best lambda is chosen per eta (minimizing loss or maximizing C-index). The function also computes an external baseline statistic from RS under the same criterion.

Value

An object of class "cv.coxkl_ridge":

integrated_stat.full_results
: Data frame with columns eta, lambda, and the aggregated CV score per lambda; for loss criteria an additional column Loss = -2 * score; for C-index criteria a column named CIndex_pooled or CIndex_foldaverage.

integrated_stat.best_per_eta
: Data frame with the best lambda (per eta) according to the chosen criterion.

external_stat
: Scalar baseline statistic computed from RS under the same cv.criteria.

criteria
: The evaluation criterion used.

nfolds
: Number of folds.

Examples

```
data(ExampleData_highdim)

train_dat_highdim <- ExampleData_highdim$train
beta_external_highdim <- ExampleData_highdim$beta_external
```

```
etas <- generate_eta(method = "exponential", n = 10, max_eta = 100)
cv_res <- cv.coxkl_ridge(z = train_dat_highdim$z,
                         delta = train_dat_highdim$status,
                         time = train_dat_highdim$time,
                         beta = beta_external_highdim,
                         etas = etas)
```

`cv.plot` *Plot Cross-Validation Results vs Eta*

Description

Plots cross-validated performance across `eta` for `cv.coxkl`, `cv.coxkl_ridge`, or `cv.coxkl_enet` results. The main CV curve is drawn as a solid purple line; a green dotted horizontal reference line is placed at the value corresponding to `eta = 0` (or the closest available `eta`), with a solid green point marking that reference level.

Usage

```
cv.plot(object, line_color = "#7570B3", baseline_color = "#1B9E77", ...)
```

Arguments

`object`	A fitted cross-validation result of class `"cv.coxkl"`, `"cv.coxkl_ridge"`, or `"cv.coxkl_enet"`.
`line_color`	Color for the CV performance curve. Default `"#7570B3"`.
`baseline_color`	Color for the horizontal reference line and point. Default `"#1B9E77"`.
`...`	Additional arguments (currently ignored).

Details

The function reads the performance metric from the object:

- For `"cv.coxkl"`: uses `object$internal_stat` (one row per `eta`).
- For `"cv.coxkl_ridge"` and `"cv.coxkl_enet"`: uses `object$integrated_stat.best_per_eta` (best `lambda` per `eta`).

The y-axis label is set to "Loss" if `criteria` in the object is "V&VH" or "LinPred"; otherwise it is "C Index". The horizontal reference ("baseline") is taken from the plotted series at `eta = 0` (or the nearest `eta` present in the results).

Value

A `ggplot` object showing cross-validation performance versus `eta`.

See Also

cv.coxkl, cv.coxkl_ridge, cv.coxkl_enet

Examples

```
data(Exampledata_lowdim)

train_dat_lowdim <- ExampleData_lowdim$train
beta_external_good_lowdim <- ExampleData_lowdim$beta_external_good

etas <- generate_eta(method = "exponential", n = 100, max_eta = 30)
cv_res <- cv.coxkl(z = train_dat_lowdim$z,
                   delta = train_dat_lowdim$status,
                   time = train_dat_lowdim$time,
                   stratrum = train_dat_lowdim$stratum,
                   beta = beta_external_good_lowdim,
                   etas = etas,
                   nfolds = 5,
                   criteria = c("V&VH"),
                   seed = 1)
cv.plot(cv_res)
```

ExampleData_highdim *Example high-dimensional survival data*

Description

A simulated survival dataset in a high-dimensional linear setting with 50 covariates (6 signals + 44 AR(1) noise), Weibull baseline hazard, and controlled censoring. Includes internal train/test sets, and an external-data–estimated coefficient vector.

Usage

```
data(ExampleData_highdim)
```

Format

A list containing the following elements:

train A list with components:

- **z** Data frame of size $n_{\text{train}} \times 50$ with covariates Z1–Z50.
- **status** Vector of event indicators (1=event, 0=censored).
- **time** Numeric vector of observed times $\min(T, C)$.
- **stratum** Vector of stratum labels (here all 1).

test A list with the same structure as train, with size $n_{\text{test}} \times 50$ for z.

beta_external Numeric vector (length 50, named Z1–Z50) of Cox coefficients estimated on an external dataset using only Z1–Z6 and expanded to length 50 (zeros for Z7–Z50).

Details

Data-generating mechanism:

- Covariates: 50 variables with signals `Z1`–`Z6` and noise `Z7`–`Z50`.
  - `Z1`, `Z2` ~ bivariate normal with AR(1) correlation $\rho = 0.5$.
  - `Z3`, `Z4` ~ independent Bernoulli(0.5).
  - `Z5` ~ $N(2, 1)$, `Z6` ~ $N(-2, 1)$ (group indicator fixed at 1).
  - `Z7`–`Z50` ~ multivariate normal with AR(1) correlation $\rho = 0.5$.
- True coefficients: $\beta = (0.3, -0.3, 0.3, -0.3, 0.3, -0.3, 0, \ldots, 0)$ (length 50).
- Event times: Weibull baseline hazard $h_0(t) = \lambda\nu\, t^{\nu-1}$ with $\lambda = 1$, $\nu = 2$. Given linear predictor $\eta = Z^\top\beta$, draw $U \sim \mathrm{Unif}(0, 1)$ and set

$$T = \left(\frac{-\log U}{\lambda\, e^{\eta}}\right)^{1/\nu}.$$

- Censoring: $C \sim \mathrm{Unif}(0, \mathrm{ub})$ with ub tuned iteratively to achieve the target censoring rate (internal: `0.70`; external: `0.50`). Observed time is $\min(T, C)$, status is $\mathbf{1}\{T \le C\}$.
- External coefficients: Fit a Cox model `Surv(time, status) ~ Z1 + ... + Z6` on the external data (Breslow ties), then place the estimated coefficients into a length-50 vector (zeros elsewhere).

Examples

```
data(ExampleData_highdim)

head(ExampleData_highdim$train$z)
table(ExampleData_highdim$train$status)
summary(ExampleData_highdim$train$time)

head(ExampleData_highdim$test$z)
table(ExampleData_highdim$test$status)
summary(ExampleData_highdim$test$time)
```

ExampleData_lowdim *Example low-dimensional survival data*

Description

A simulated survival dataset in a low-dimensional linear setting with 6 covariates (2 correlated continuous, 2 binary, 2 mean-shifted normals), Weibull baseline hazard, and controlled censoring. Includes internal train/test sets, and three external-quality coefficient vectors.

Usage

```
data(ExampleData_lowdim)
```

Format

A list containing the following elements:

train A list with components:

z Data frame of size $n_{\text{train}} \times 6$ with covariates Z1–Z6.

status Vector of event indicators (1=event, 0=censored).

time Numeric vector of observed times $\min(T, C)$.

stratum Vector of stratum labels (here all 1).

test A list with the same structure as train, with size $n_{\text{test}} \times 6$ for z.

beta_external_good Numeric vector (length 6; named Z1–Z6) of Cox coefficients estimated on a "Good" external dataset using all Z1–Z6.

beta_external_fair Numeric vector (length 6; names Z1–Z6) of Cox coefficients estimated on a "Fair" external dataset using a reduced subset Z1, Z3, Z5, Z6; coefficients for variables not used are 0.

beta_external_poor Numeric vector (length 6; names Z1–Z6) of Cox coefficients estimated on a "Poor" external dataset using Z1 and Z5 only; remaining entries are 0.

Details

Data-generating mechanism:

- Covariates: 6 variables Z1–Z6.
  - Z1, Z2 ~ bivariate normal with AR(1) correlation $\rho = 0.5$.
  - Z3, Z4 ~ independent Bernoulli(0.5).
  - Z5 ~ $N(2, 1)$, Z6 ~ $N(-2, 1)$ (group indicator fixed at 1 for internal train/test).
- True coefficients: $\beta = (0.3, -0.3, 0.3, -0.3, 0.3, -0.3)$ (length 6).
- Event times: Weibull baseline hazard $h_0(t) = \lambda\nu\, t^{\nu-1}$ with $\lambda = 1$, $\nu = 2$. Given linear predictor $\eta = Z^\top\beta$, draw $U \sim \text{Unif}(0, 1)$ and set

$$T = \left(\frac{-\log U}{\lambda\, e^{\eta}}\right)^{1/\nu}.$$

- Censoring: $C \sim \text{Unif}(0, \text{ub})$ with ub tuned iteratively to achieve the target censoring rate (internal: 0.70; external: 0.50). Observed time is $\min(T, C)$, status is $\mathbf{1}\{T \le C\}$.
- External coefficients: For each quality level ("Good", "Fair", "Poor"), fit a Cox model Surv(time, status) ~ Z1 + ... on the corresponding external data (Breslow ties) using the specified covariate subset; place estimates into a length-6 vector named Z1–Z6 with zeros for variables not included.

Examples

```
data(ExampleData_lowdim)

head(ExampleData_lowdim$train$z)
table(ExampleData_lowdim$train$status)
summary(ExampleData_lowdim$train$time)
```

```
head(ExampleData_lowdim$test$z)
table(ExampleData_lowdim$test$status)
summary(ExampleData_lowdim$test$time)
```

`generate_eta` *Generate a Sequence of Tuning Parameters (eta)*

Description

Produces a numeric vector of `eta` values to be used in Cox–KL model.

Usage

```
generate_eta(method = "exponential", n = 10, max_eta = 5, min_eta = 0)
```

Arguments

`method`	Character string selecting how to generate `eta`: "linear" or "exponential". Default is "exponential". for an exponentially spaced sequence scaled to `max_eta`. Default is `"exponential"`.
`n`	Integer, the number of `eta` values to generate. Default is 10.
`max_eta`	Numeric, the maximum value of `eta` in the sequence. Default is 5.
`min_eta`	Numeric, the minimum value of `eta` in the sequence. Default is 0.

Details

- *Exponential*: values are formed by exponentiating a grid from `log(1)` to `log(100)`, then linearly rescaling to the interval `[0, max_eta]`. Thus the smallest value equals `0` and the largest equals `max_eta`.
- *Linear*: the current implementation calls `seq(min_eta, max_eta, length.out = n)` and therefore assumes a numeric object `min_eta` exists in the calling environment.

Only the exact strings "linear" and "exponential" are supported; other values for `method` will result in an error because `eta_values` is never created.

Value

Numeric vector of length `n` containing the generated `eta` values.

Examples

```
# Generate 10 exponentially spaced eta values up to 5
generate_eta(method = "exponential", n = 10, max_eta = 5)

# Generate 5 linearly spaced eta values up to 3
generate_eta(method = "linear", n = 5, max_eta = 3)
```

loss_fn *Calculate the Log-Partial Likelihood for a Stratified Cox Model*

Description

Computes the stratified Cox partial log-likelihood for given covariates, event indicators, times, and coefficients.

Usage

```
loss_fn(z, delta, time, stratum, beta)
```

Arguments

z	A numeric matrix (or data frame coercible to matrix) of covariates. Each row is an observation and each column a predictor.
delta	A numeric vector of event indicators (1 = event, 0 = censored).
time	A numeric vector of observed times (event or censoring).
stratum	An optional vector specifying the stratum for each observation (factor/character/numeric). If missing, a single-stratum model is assumed.
beta	A numeric vector of regression coefficients with length equal to the number of columns in z.

Details

Inputs are internally sorted by stratum and time. The function evaluates the stratified Cox partial log-likelihood using the supplied z, delta, beta, and the stratum sizes.

Value

A single numeric value giving the stratified Cox partial log-likelihood.

plot.coxkl *Plot Model Performance vs Eta for* coxkl

Description

Plots model performance across the eta sequence. Performance is either loss (-2 times partial log-likelihood) or concordance index (C-index). If no test data are provided, the curve is computed on the training data stored in x$data.

Usage

```
## S3 method for class 'coxkl'
plot(
  x,
  test_z = NULL,
  test_time = NULL,
  test_delta = NULL,
  test_stratum = NULL,
  criteria = c("loss", "CIndex"),
  ...
)
```

Arguments

x	A fitted model object of class "coxkl".
test_z	Optional numeric matrix of test covariates.
test_time	Optional numeric vector of test survival times.
test_delta	Optional numeric vector of test event indicators.
test_stratum	Optional vector of test stratum membership.
criteria	Character string: "loss" or "CIndex".
...	Additional arguments (ignored).

Details

When criteria = "loss" and no test data are supplied, the plotted values are (-2 * x$likelihood) / n, where n is the number of rows in the (training) data. When test data are provided, performance is computed via test_eval(..., criteria = "loss") and divided by the test sample size. For criteria = "CIndex", performance is computed via test_eval(..., criteria = "CIndex") on the chosen dataset. The plot adds a dotted horizontal reference line at the value corresponding to eta = 0 (closest point on the eta grid).

Value

A ggplot object showing the performance curve.

Examples

```
data(ExampleData_lowdim)

train_dat_lowdim  <- ExampleData_lowdim$train
test_dat_lowdim   <- ExampleData_lowdim$test
beta_external_good_lowdim <- ExampleData_lowdim$beta_external_good
eta_grid <- generate_eta(method = "exponential", n = 100, max_eta = 30)

model <- coxkl(z = train_dat_lowdim$z,
               delta = train_dat_lowdim$status,
               time = train_dat_lowdim$time,
               stratum = train_dat_lowdim$stratum,
```

```
               beta = beta_external_good_lowdim,
               etas = eta_grid)
plot(model,
     test_z = test_dat_lowdim$z,
     test_time = test_dat_lowdim$time,
     test_delta = test_dat_lowdim$status,
     test_stratum = test_dat_lowdim$stratum,
     criteria = "loss")
```

`plot.coxkl_enet` *Plot Model Performance vs Lambda for* `coxkl_enet`

Description

Plots model performance across the `lambda` sequence. Performance is loss (`-2` times partial log-likelihood) or concordance index (C-index). If no test data are provided, the curve uses the training data stored in `x$data`.

Usage

```
## S3 method for class 'coxkl_enet'
plot(
  x,
  test_z = NULL,
  test_time = NULL,
  test_delta = NULL,
  test_stratum = NULL,
  criteria = c("loss", "CIndex"),
  ...
)
```

Arguments

`x`	A fitted model object of class `"coxkl_enet"`.
`test_z`	Optional numeric matrix of test covariates.
`test_time`	Optional numeric vector of test survival times.
`test_delta`	Optional numeric vector of test event indicators.
`test_stratum`	Optional vector of test stratum membership.
`criteria`	Character string: `"loss"` or `"CIndex"`.
`...`	Additional arguments (ignored).

Details

When `criteria = "loss"` and no test data are supplied, the plotted values are `-2 * x$likelihood` (no normalization). When test data are provided, performance is computed via `test_eval(..., criteria)`. The x-axis is shown in decreasing `lambda` with a reversed log10 scale.

Value

A ggplot object showing the performance curve.

Examples

```
data(ExampleData_highdim)

train_dat_highdim <- ExampleData_highdim$train
test_dat_highdim <- ExampleData_highdim$test
beta_external_highdim <- ExampleData_highdim$beta_external

model_enet <- coxkl_enet(z = train_dat_highdim$z,
                         delta = train_dat_highdim$status,
                         time = train_dat_highdim$time,
                         beta = beta_external_highdim,
                         eta = 1,
                         alpha = 1.0)
plot(model_enet,
     test_z = test_dat_highdim$z,
     test_time = test_dat_highdim$time,
     test_delta = test_dat_highdim$status,
     test_stratum = test_dat_highdim$stratum,
     criteria = "loss")
```

---

plot.coxkl_ridge *Plot Model Performance vs Lambda for* coxkl_ridge

---

Description

Plots model performance across the lambda sequence. Performance is loss (-2 times partial log-likelihood) or concordance index (C-index). If no test data are provided, the curve uses the training data stored in x$data.

Usage

```
## S3 method for class 'coxkl_ridge'
plot(
  x,
  test_z = NULL,
  test_time = NULL,
  test_delta = NULL,
  test_stratum = NULL,
  criteria = c("loss", "CIndex"),
  ...
)
```

Arguments

x	A fitted model object of class "coxkl_ridge".
test_z	Optional numeric matrix of test covariates.
test_time	Optional numeric vector of test survival times.
test_delta	Optional numeric vector of test event indicators.
test_stratum	Optional vector of test stratum membership.
criteria	Character string: "loss" or "CIndex".
...	Additional arguments (ignored).

Details

When criteria = "loss" and no test data are supplied, the plotted values are -2 * x$likelihood (no normalization). When test data are provided, performance is computed via test_eval(..., criteria). The x-axis is shown in decreasing lambda with a reversed log10 scale.

Value

A ggplot object showing the performance curve.

Examples

```
data(ExampleData_highdim)

train_dat_highdim <- ExampleData_highdim$train
test_dat_highdim <- ExampleData_highdim$test
beta_external_highdim <- ExampleData_highdim$beta_external


model_ridge <- coxkl_ridge(z = train_dat_highdim$z,
                           delta = train_dat_highdim$status,
                           time = train_dat_highdim$time,
                           beta = beta_external_highdim,
                           eta = 1)

plot(
  model_ridge,
  test_z       = test_dat_highdim$z,
  test_time    = test_dat_highdim$time,
  test_delta   = test_dat_highdim$status,
  test_stratum = test_dat_highdim$stratum,
  criteria     = "CIndex"
)
```

predict.coxkl *Predict Linear Predictors from a* coxkl *Object*

Description

Computes linear predictors for new data based on a fitted coxkl model. If eta is supplied, predictions are returned for those eta values; otherwise predictions are returned for all fitted etas. Linear interpolation is applied if an intermediate eta value is requested.

Usage

```
## S3 method for class 'coxkl'
predict(object, newz, eta = NULL, ...)
```

Arguments

object	A fitted model object of class "coxkl".
newz	A numeric matrix or data frame of new covariates (must match the dimension of the training design matrix used to fit the model).
eta	Optional numeric vector of eta value(s) for which to predict. If NULL, predictions for all fitted eta values are returned.
...	Additional arguments.

Details

The linear predictors are computed as as.matrix(newz) %*% beta.

Value

A numeric matrix of linear predictors with one column per eta (sorted ascending).

See Also

coef.coxkl

predict.coxkl_enet *Predict Linear Predictors from a coxkl_enet Object*

Description

Computes linear predictors for new data using a fitted coxkl_enet model. If lambda is supplied, predictions are returned for those lambda values; otherwise predictions are returned for all fitted lambdas. When a requested lambda lies between fitted values, coefficients are linearly interpolated.

Usage

```
## S3 method for class 'coxkl_enet'
predict(object, newz, lambda = NULL, ...)
```

Arguments

object	A fitted model object of class "coxkl_enet".
newz	A numeric matrix or data frame of new covariates (same columns as in training data).
lambda	Optional numeric value(s) specifying the regularization parameter(s) for which to predict. If NULL, predictions for all fitted lambda values are returned.
...	Additional arguments.

Details

The linear predictors are computed as as.matrix(newz) %*% beta.

Value

A numeric matrix of linear predictors. Each column corresponds to one lambda, sorted in descending order.

See Also

coef.coxkl_enet

---

predict.coxkl_ridge *Predict Linear Predictors from a coxkl_ridge Object*

Description

Computes linear predictors for new data using a fitted coxkl_ridge model. If lambda is supplied, predictions are returned for those lambda values; otherwise predictions are returned for all fitted lambdas. When a requested lambda lies between fitted values, coefficients are linearly interpolated.

Usage

```
## S3 method for class 'coxkl_ridge'
predict(object, newz, lambda = NULL, ...)
```

Arguments

object	A fitted model object of class "coxkl_ridge".
newz	A numeric matrix or data frame of new covariates (same columns as in training data).
lambda	Optional numeric value(s) specifying the regularization parameter(s) for which to predict. If NULL, predictions for all fitted lambda values are returned.
...	Additional arguments.

Details

The linear predictors are computed as `as.matrix(newz) %*% beta`.

Value

A numeric matrix of linear predictors. Each column corresponds to one `lambda`, sorted in descending order.

See Also

`coef.coxkl_ridge`

`support` *Study to Understand Prognoses Preferences Outcomes and Risks of Treatment*

Description

The `support` dataset tracks five response variables: hospital death, severe functional disability, hospital costs, and time until death and death itself. The patients are followed for up to 5.56 years. See Bhatnagar et al. (2020) for details.

Usage

```
data(support)
```

Format

A data frame with 9,104 observations and 34 variables after imputation and the removal of response variables like hospital charges, patient ratio of costs to charges and micro-costs following Bhatnagar et al. (2020). Ordinal variables, namely functional disability and income, were also removed. Finally, Surrogate activities of daily living were removed due to sparsity. There were 6 other model scores in the data-set and they were removed; only aps and sps were kept.

age stores a double representing age.

death death at any time up to NDI (National Death Index) date: 12/31/1994.

sex 0=female, 1=male.

slos days from study entry to discharge.

d.time days of follow-up.

dzgroup each level of dzgroup: ARF/MOSF w/Sepsis, COPD, CHF, Cirrhosis, Coma, Colon Cancer, Lung Cancer, MOSF with malignancy.

dzclass ARF/MOSF, COPD/CHF/Cirrhosis, Coma and cancer disease classes.

num.co the number of comorbidities.

edu years of education of patients.

scoma the SUPPORT coma score based on Glasgow D3.

avtisst average TISS, days 3-25.

race indicates race: White, Black, Asian, Hispanic or other.

hday day in Hospital at Study Admit.

diabetes diabetes (Com27-28, Dx 73).

dementia dementia (Comorbidity 6).

ca cancer state.

meanbp mean arterial blood pressure day 3.

wblc white blood cell count on day 3.

hrt heart rate day 3.

resp respiration rate day 3.

temp temperature, in Celsius, on day 3.

pafi PaO2/(0.01*FiO2) day 3.

alb serum albumin day 3.

bili bilirubin day 3.

crea serum creatinine day 3.

sod serum sodium day 3.

ph serum pH (in arteries) day 3.

glucose serum glucose day 3.

bun bun day 3.

urine urine output day 3.

adlp adl patient day 3.

adlsc imputed adl calibrated to surrogate, if a surrogate was used for a follow up.

sps SUPPORT physiology score.

aps apache III physiology score.

Details

Some of the original data was missing. Before imputation, there were a total of 9,104 individuals and 47 variables. Following Bhatnagar et al. (2020), a few variables were removed. Three response variables were removed: hospital charges, patient ratio of costs to charges and patient micro-costs. Hospital death was also removed as it was directly informative of the event of interest, namely death. Additionally, functional disability and income were removed as they are ordinal covariates. Finally, 8 covariates were removed related to the results of previous findings: SUPPORT day 3 physiology score (`sps`), APACHE III day 3 physiology score (`aps`), SUPPORT model 2-month survival estimate, SUPPORT model 6-month survival estimate, Physician's 2-month survival estimate for pt., Physician's 6-month survival estimate for pt., Patient had Do Not Resuscitate (DNR) order, and Day of DNR order (<0 if before study). Of these, `sps` and `aps` were added on after imputation, as they were missing only 1 observation. First the imputation is done manually using the normal values for physiological measures recommended by Knaus et al. (1995). Next, a single dataset was imputed using **mice** with default settings. After imputation, the covariate for surrogate activities of daily living was not imputed. This is due to collinearity between the other two covariates for activities of daily living. Therefore, surrogate activities of daily living were removed. See details in the R package (casebase) by Bhatnagar et al. (2020).

Source

Available at the following website: https://archive.ics.uci.edu/dataset/880/support2.

References

Bhatnagar, S., Turgeon, M., Islam, J., Hanley, J. A., and Saarela, O. (2020) casebase: Fitting Flexible Smooth-in-Time Hazards and Risk Functions via Logistic and Multinomial Regression. *R package version 0.9.0*, https://CRAN.R-project.org/package=casebase.

Knaus, W. A., Harrell, F. E., Lynn, J., Goldman, L., Phillips, R. S., Connors, A. F., et al. (1995) The SUPPORT prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of Internal Medicine*, **122(3)**: 191-203.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  data(support)
  set.seed(123)

  support <- support[support$ca %in% "metastatic", ]
  time <- support$d.time
  death <- support$death
  diabetes <- model.matrix(~ factor(support$diabetes))[, -1]
  # sex: female as the reference group
  sex <- model.matrix(~ support$sex)[, -1]
  # age: continuous variable
  age <- support$age
  age[support$age <= 50] <- "<50"
  age[support$age > 50 & support$age <= 60] <- "50-59"
  age[support$age > 60 & support$age < 70] <- "60-69"
  age[support$age >= 70] <- "70+"
  age <- factor(age, levels = c("60-69", "<50", "50-59", "70+"))
  z_age <- model.matrix(~ age)[, -1]
  z <- data.frame(z_age, sex, diabetes)
  colnames(z) <- c("age_50", "age_50_59", "age_70", "diabetes", "male")
  dat <- data.frame(time, death, z)

  n <- nrow(dat)
  n_ext  <- floor(0.87 * n)
  n_int  <- floor(0.03 * n)
  n_test <- n - n_ext - n_int
  idx <- sample(seq_len(n))
  idx_ext  <- idx[1:n_ext]
  idx_int  <- idx[(n_ext + 1):(n_ext + n_int)]
  idx_test <- idx[(n_ext + n_int + 1):n]

  external_data <- dat[idx_ext, ]
  internal_data <- dat[idx_int, ]
  test_data <- dat[idx_test, ]

  ext_cox <- survival::coxph(
```

```
  survival::Surv(time, death) ~ age_50 + age_50_59 + age_70 + diabetes + male,
  data = external_data
)
beta_external <- coef(ext_cox)

result1 <- cv.coxkl(
  z = internal_data[, c("age_50", "age_50_59", "age_70", "diabetes", "male")],
  delta = internal_data$death,
  time = internal_data$time,
  beta = beta_external,
  stratum = NULL,
  etas = generate_eta(method = "exponential", n = 50, max_eta = 50)
)
cv.plot(result1)
}
```

`test_eval` *Evaluate model performance on test data*

Description

Evaluates model performance on a test dataset using either the log-partial-likelihood loss or the concordance index (C-index).

This function accepts either:

- `test_z` and `betahat`, which will be multiplied to obtain risk scores; or
- `test_RS`, a pre-computed numeric vector of risk scores.

Usage

```
test_eval(
  test_z = NULL,
  test_RS = NULL,
  test_delta,
  test_time,
  test_stratum = NULL,
  betahat = NULL,
  criteria = c("loss", "CIndex")
)
```

Arguments

`test_z`	Optional numeric matrix or data frame of covariates for the test dataset. Required if `test_RS` is not provided.
`test_RS`	Optional numeric vector of pre-computed risk scores (e.g., linear predictors). If provided, `test_z` and `betahat` are ignored.
`test_delta`	Numeric vector of event indicators (1 = event, 0 = censored).

test_time	Numeric vector of survival times for the test dataset.
test_stratum	Optional vector indicating stratum membership for each test observation. If NULL, all observations are assumed to belong to a single stratum.
betahat	Optional numeric vector of estimated regression coefficients. Required if test_RS is not provided.
criteria	Character string specifying the evaluation criterion; one of: • "loss": negative twice the log–partial-likelihood. • "CIndex": concordance index.

Details

Prior to evaluation, observations are sorted by *(stratum, time)* to ensure correct risk-set construction. For stratified C-index computation, the provided test_stratum is used; otherwise all test data are treated as a single stratum.

You may supply either covariates and coefficients (test_z with betahat) or a precomputed risk score vector (test_RS). When test_RS is provided, test_z and betahat are ignored.

Value

A numeric value representing either:

- if criteria = "loss": the negative twice log–partial-likelihood on the test data.
- if criteria = "CIndex": the concordance index on the test data.

Index