Package 'pcoxtime'

October 14, 2022

Type Package

Title Penalized Cox Proportional Hazard Model for Time-Dependent Covariates

Version 1.0.4

Description
Fits penalized models for both time-independent and time-dependent survival data. It fully implements elastic net and uses proximal gradient descent to solve the optimization problem. The package is an implementation of Steve Cygu and Benjamin M. Bolker. (2021) <arXiv:2102.02297>.

License GPL (>= 2)

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.5), survival, doParallel, parallel, foreach, ggplot2, prodlim, riskRegression, PermAlgo, pec

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.2

Encoding UTF-8

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/CYGUBICKO/pcoxtime-pkg

NeedsCompilation yes

Author Bicko Cygu [aut, cre] (<https://orcid.org/0000-0002-9284-8863>), Ben Bolker [aut] (<https://orcid.org/0000-0002-2127-0443>), Trevor Hastie [cph] (getmin function implementation copied from package glmnet)

Maintainer Bicko Cygu <cygubicko@gmail.com>

Repository CRAN

Date/Publication 2022-05-13 00:10:02 UTC

R topics documented:

coef.pcoxtime . . . 2
coef.pcoxtimecv . . . 3
concordScore.pcoxtime . . . 4
extractoptimal.pcoxtimecv . . . 5
pcoxsurvfit.pcoxtime . . . 6
pcoxtheme . . . 7
pcoxtime . . . 8
pcoxtimecv . . . 11
plot.pcoxsurvfit . . . 15
plot.pcoxtimecv . . . 16
plot.Score . . . 18
plot.varimp . . . 20
predict.pcoxtime . . . 20
predictRisk.pcoxtime . . . 22
predictSurvProb.pcoxtime . . . 23
print.pcoxbasehaz . . . 24
print.pcoxsurvfit . . . 25
print.pcoxtime . . . 25
print.pcoxtimecv . . . 26
varimp.pcoxtime . . . 27

Index **29**

`coef.pcoxtime` *Extract coefficient estimates of pcoxtime object*

Description

This function extracts the estimates for all the coefficients.

Usage

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

## S3 method for class 'pcoxtime'
coefficients(object, ...)
```

Arguments

`object`	fitted `pcoxtime` model object
`...`	for future implementations

Details

The call that produced `pcoxtime` is printed, followed by coefficient estimates.

Value

A vector of coefficient estimates.

A vector of coefficient estimates.

`coef.pcoxtimecv` *Extract coefficient estimates of pcoxtimecv object*

Description

This function extracts cross-validation estimates for a particular lambda.

Usage

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

## S3 method for class 'pcoxtimecv'
coefficients(object, lambda, ...)
```

Arguments

`object`	`pcoxtimecv` object
`lambda`	the value of lambda for which to return the coefficient estimates. It can be any of the character string, "min", "optimal" or "best" for optimal lambda; "1se" for 1 standard error lambda; or any numeric value for lambda. See details.
`...`	for future implementations

Details

Extract the coefficient estimates for optimal lambda-alpha pair or based on specified the value of lambda for an optimal alpha. If the value of lambda specified is not exact (not in lambdas), the nearest value is used, based on `nearest <- function(values, value){values[which(abs(values-value)==min(abs(value`
It requires that `pcoxtimecv` is run with `refit = TRUE`.

Value

A data frame of coefficient estimates.

A vector of coefficient estimates.

concordScore.pcoxtime *Compute the concordance statistic for the pcoxtime model*

Description

The function computes the agreement between the observed response and the predictor.

Usage

```
## S3 method for class 'pcoxtime'
concordScore(fit, newdata = NULL, stats = FALSE, reverse = TRUE, ...)
```

Arguments

fit	fitted pcoxtime.
newdata	optional data frame containing the variables appearing on the right hand side of pcoxtime formula.
stats	logical. If TRUE all the related concordance statistics are returned.
reverse	if TRUE (default) then assume that larger x values predict smaller response values y; a proportional hazards model is the common example of this.
...	additional arguments passed to concordance.

Details

Computes Harrell's C index for predictions for pcoxtime object and takes into account censoring. See concordance.

Value

an object containing the concordance, followed by the number of pairs that agree, disagree, are tied, and are not comparable.

Examples

```
if (packageVersion("survival")>="3.2.9") {
   data(cancer, package="survival")
} else {
   data(veteran, package="survival")
}
# Penalized
lam <- 0.1
alp <- 0.5
pfit1 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, lambda = lam
, alpha = alp
```

```
)
c1 <- concordScore(pfit1)
c1

# Unpenalized
lam <- 0
alp <- 1
pfit2 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, lambda = lam
, alpha = alp
)
c2 <- concordScore(pfit2)
c2
```

`extractoptimal.pcoxtimecv` *Extract optimal parameter values*

Description

Extract cross-validation summaries and data frames.

Usage

```
## S3 method for class 'pcoxtimecv'
extractoptimal(object, what = c("optimal", "cvm", "coefs"), ...)
```

Arguments

`object`	`pcoxtimecv` object
`what`	the summary or data frame to extract. Currently, three options are available: `what = "optimal"` extracts a data frame showing optimal parameter values, `what = "cvm"` extracts the data frame containing the mean cross-validation error for various lambda-alpha combination, and `what = "coef"`, requires `refit = TRUE` in `pcoxtimecv`, extracts a data frame containing the coefficient estimates for various lambda-alpha combination.
`...`	for future implementations

Details

Extract cross-validation summaries based on the optimal parameters or data frames containing all the summaries for all the parameter values.

Value

A data frame depending on the specification described above.

`pcoxsurvfit.pcoxtime` *Compute survival curve and cumulative hazard from a pcoxtime model*

Description

Compute the predicted survivor and cumulative hazard function for a penalized Cox proportional hazard model.

Usage

```
## S3 method for class 'pcoxtime'
pcoxsurvfit(fit, newdata, ...)

## S3 method for class 'pcoxtime'
pcoxbasehaz(fit, centered = TRUE)
```

Arguments

`fit`	fitted `pcoxtime` object
`newdata`	a data frame containing the variables appearing on the right hand side of `pcoxtime` formula.
`...`	for future implementations
`centered`	if `TRUE` (default), return data from a predicted survival function at the mean values of the predictors, if `FALSE` returns prediction for all predictors equal to zero (baseline hazard).

Details

`pcoxsurvfit` and `pcoxbasehaz` functions produce survival curves and estimated cumulative hazard, respectively, for the fitted `pcoxtime` model. They both return the estimated survival probability and the estimated cumulative hazard, which are both Breslow estimate.

The `pcoxbasehaz` is an alias for `pcoxsurvfit` which simply computed the predicted survival estimates (baseline).

If the `newdata` argument is missing, the "average" survival or cumulative hazard estimates are produced with the predictor values equal to means of the data set. See `survfit.coxph` for warning against this. If the `newdata` is specified, then the returned object will contain a matrix of both survival and cumulative hazard estimates with each column for each row in the `newdata`.

Value

`pcoxsurvfit` and `pcoxbasehaz` return S3 objects of class `pcoxsurvfit.pcoxtime` and `pcoxbasehaz.pcoxtime`, respectively:

`n`	number of observations used in the fit.
`events`	total number of events of interest in the fit.
`time`	time points defined by the risk set.

n.risk	the number of individuals at risk at time t.
n.event	the number of events that occur at time t.
n.censor	the number of subjects who exit the risk set, without an event, at time t.
surv	a vector or a matrix of estimated survival function.
cumhaz, hazard	a vector or a matrix of estimated cumulative hazard.
call	the call that produced the object.

See Also

pcoxtime, plot.pcoxsurvfit

Examples

```
data(heart, package="survival")
lam <- 0.1
alp <- 0.8
pfit <- pcoxtime(Surv(start, stop, event) ~ age + year + surgery + transplant
, data = heart
, lambda = lam
, alpha = alp
)

# Survival estimate
psurv <- pcoxsurvfit(pfit)
print(psurv)

# Baseline survival estimate
bsurv <- pcoxbasehaz(pfit, centered = FALSE)
```

---

pcoxtheme *Set theme for pcoxtime plots*

Description

Sets a theme for pcoxtime and other ggplot objects

Usage

```
pcoxtheme()
```

Value

No return value, called for side effects (setting pcotime plotting theme).

Examples

```
library(ggplot2)
pcoxtheme()
data(heart, package="survival")
lam <- 0.02
alp <- 1
pfit <- pcoxtime(Surv(start, stop, event) ~ age + year + surgery + transplant
, data = heart
, lambda = lam
, alpha = alp
)

# Plot survival curves
psurv <- pcoxsurvfit(pfit)
plot(psurv)
```

---

pcoxtime *Fit penalized Cox model*

Description

Fits a Cox model with either lasso, ridge or elasticnet penalty for both time-independent and time-dependent (varying) covariates survival data.

Usage

```
pcoxtime(
  formula,
  data,
  alpha = 1,
  lambda = 1,
  maxiter = 1e+05,
  tol = 1e-08,
  quietly = FALSE,
  lambmax = FALSE,
  origin_scale = TRUE,
  contrasts.arg = NULL,
  xlevs = NULL,
  na.action = na.omit,
  ...
)
```

Arguments

formula	object of class formula describing the model. The response is specified similar to Surv function from package **survival**. The terms (predictors) are specified on the right of "~" in the formula.
data	optional data frame containing variables specified in the formula.

`alpha`	elasticnet mixing parameter, with $0 \leq \alpha \leq 1$. See details
`lambda`	tuning parameter for the lasso penalization, with $\lambda \geq 0$. $\lambda = 0$ fits unpenalized Cox model. See details
`maxiter`	maximum number of iterations to convergence. Default is $1e4$. Consider increasing it if the model does not converge.
`tol`	convergence threshold for proximal gradient gradient descent. Each proximal update continues until the relative change in all the coefficients (i.e. $\sqrt{\sum(\beta_{k+1} - \beta_k)^2}$/stepsize) is less than tol. The default value is $1e-8$.
`quietly`	logical. If TRUE, iteration progress printed.
`lambmax`	logical. Sufficiently large, $\lambda_{\max}$, that sets $\beta = 0$ for regularization path. If TRUE, $\lambda_{\max}$ is returned.
`origin_scale`	logical. If TRUE (default), the estimated coefficients are returned on the original covariate scale. Otherwise, FALSE, coefficients are standardized.
`contrasts.arg`	an optional list. See the contrasts.arg of `model.matrix.default`.
`xlevs`	a named list of character vectors giving the full set of levels to be assumed for each factor. See `model.frame`.
`na.action`	a function which indicates what should happen when the data contain NAs. See `model.frame`.
`...`	additional arguments not implemented.

Details

The algorithm estimates the coefficients based on observed survival data, with either time-independent or time-dependent covariates, through penalized partial log-likelihood

$$\text{pen } \ell(\beta)_{\alpha,\lambda} = -\ell(\beta) + P_{\alpha,\lambda}(\beta)$$

using elasticnet (which combines both lasso and ridge) penalty

$$\lambda \left(\alpha \sum_{i=1}^{p} |\beta_i| + 0.5(1-\alpha) \sum_{i=1}^{p} \beta_i^2 \right)$$

.

`alpha = 1` (α) is the lasso penalty, and `alpha = 0` is the ridge penalty. `lambda = 0` fits the standard Cox proportional hazard model.

User can provide a particular lambda. Typical usage is to use the `pcoxtimecv` to select the optimal lambda first.

The routine to handle time-dependent covariates is similar to that implemented in `coxph`: if there are tied event times, Breslow approximation is used.

Value

An S3 object of class pcoxtime:

coef	a named vector of coefficients. If any of the coefficients violates KKT conditions, the model will print a warning but still return coefficient estimates.
min.nloglik	estimated log-likelihood at convergence.
min.dev	the deviation satisfying the tol stopping creteria.
iter.dev	deviations between previous and current coefficient estimate at each iteration.
convergence	convergence message containing the number of iterations
n	the number of observations used in the fit.
n.risk	the number of individuals at risk at time t.
n.event	the number of events that occur at time t.
n.censor	the number of subjects who exit the risk set, without an event, at time t.
time	time points defined by the risk set.
Y	Surv object defining the event times and event status.
data	data frame used.
timevarlabel, eventvarlabel	time and event variables, respectively.
predictors	a vector of predictors/covariates in the model.
lambda, alpha	lambda and alpha used, respectively.
formula	model formula used in the fit.
means	vector of column means of the X matrix. Subsequent survival curves are adjusted to this value.
assign, xlevels, terms	See model.frame for assign, xlevels, contrasts and terms.

See Also

coxph, pcoxtimecv

Examples

```
# Time-independent covariates
if (packageVersion("survival")>="3.2.9") {
data(cancer, package="survival")
} else {
data(veteran, package="survival")
}
## Fit unpenalized Cox using pcoxtime
lam <- 0 # Should fit unpenalized Cox model
pfit1 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, lambda = lam
, alpha = 1
```

```
)
print(pfit1)

## fit survival::coxph
cfit1 <- coxph(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, method = 'breslow'
, ties = "breslow"
)
print(cfit1)

## Penalized Cox model (pcoxtime)
lam <- 0.1
alp <- 0.5
pfit2 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, lambda = lam
, alpha = alp
)
print(pfit2)

# Time-varying covariates
data(heart, package="survival")
lam <- 0.1
alp <- 0.8
pfit2 <- pcoxtime(Surv(start, stop, event) ~ age + year + surgery + transplant
, data = heart
, lambda = lam
, alpha = alp
)
print(pfit2)
```

`pcoxtimecv` *Cross-validation for pcoxtime*

Description

Performs k-fold cross-validation for pcoxtime, plots solution path plots, and returns optimal value of lambda (and optimal alpha if more than one is given).

Usage

```
pcoxtimecv(
  formula,
  data,
  alphas = 1,
  lambdas = NULL,
  nlambdas = 100,
  lammin_fract = NULL,
```

```
  lamfract = 0.6,
  nfolds = 10,
  foldids = NULL,
  devtype = "vv",
  refit = FALSE,
  maxiter = 1e+05,
  tol = 1e-08,
  quietly = FALSE,
  seed = NULL,
  nclusters = 1,
  na.action = na.omit,
  ...
)
```

Arguments

formula	object of class formula describing the model. The response is specified similar to Surv function from package **survival**. The terms (predictors) are specified on the right of "~" in the formula.
data	optional data frame containing variables specified in the formula.
alphas	elasticnet mixing parameter, with 0 <= alphas <= 1. If a vector of alphas is supplied, cross-validation will be performed for each of the alphas and optimal value returned. The default is 1.
lambdas	optional user-supplied sequence. If lambdas = NULL (default – highly recommended), the algorithm chooses its own sequence.
nlambdas	the default number of lambdas values. Default is 100.
lammin_fract	smallest value of lambda, as fraction of maximum lambda. If NULL, default, it depends on the number of observations (n) relative to the number of variables (p). If n > p, the default is 0.0001, otherwise 0.01. Increasing this value may lead to faster convergence.
lamfract	proportion of regularization path to consider. If lamfract = 1, complete regularization path is considered. However, if 0.5 <= lamfract <1, only a proportion of the nlambdas considered. Choosing a smaller lamfract reduces computational time and potentially stable estimates for model with large number of predictors. See details.
nfolds	number of folds. Default is 10. The smallest allowable is nfolds = 3.
foldids	an optional sequence of values between 1 and nfolds specifying what fold each observation is in. This is important when comparing performance across models. If specified, nfolds can be missing.
devtype	loss to use for cross-validation. Currently, two options are available but versions will implement concordScore.pcoxtime loss too. The two are, default (devtype = "vv") Verweij Van Houwelingen partial-likelihood deviance and basic cross-validated parial likelihood devtype = "basic". See Dai, B., and Breheny, P. (2019) for details.
refit	logical. Whether to return solution path based on optimal lambda and alpha picked by the model. Default is refit = FALSE.

maxiter	maximum number of iterations to convergence. Default is $1e5$. Consider increasing it if the model does not converge.
tol	convergence threshold for proximal gradient gradient descent. Each proximal update continues until the relative change in all the coefficients (i.e. $\sqrt{\sum(\beta_{k+1} - \beta_k)^2}$/stepsize) is less than tol. The default value is $1e - 8$.
quietly	logical. If TRUE, refit progress is printed.
seed	random seed. Default is NULL, which generated the seed internally.
nclusters	number of cores to use to run the cross-validation in parallel. Default is nclusters = 1 which runs serial.
na.action	a function which indicates what should happen when the data contain NAs.
...	additional arguments not implemented.

Details

The function fits pcoxtime folds + 1 (if refit = FALSE) or folds + 2 times (if refit = FALSE). In the former case, the solution path to display using plot.pcoxtimecv is randomly picked from all the cross-validation runs. However, in the later case, the solution path plot is based on the model refitted using the optimal parameters. In both cases, the function first runs plot.pcoxtimecv to compute the lambda sequence and then perform cross-validation on nfolds.

If more than one alphas is specified, say code(0.2, 0.5, 1), the pcoxtimecv will search (experimental) for optimal values for alpha with respect to the corresponding lambda values. In this case, optimal alpha and lambda sequence will be returned, i.e., the (alphas, lambdas) pair that corresponds to the lowest predicted cross-validated error (likelihood deviance).

For data sets with a very large number of predictors, it is recommended to only calculate partial paths by lowering the value of lamfract. In other words, for p > n problems, the near lambda = 0 solution is poorly behaved and this may account for over 99% of the function's runtime. We therefore recommend always specifying lamfract < 1 and increase if the optimal lambda suggests lower values.

Value

An S3 object of class pcoxtimecv:

lambda.min	the value of lambda that gives minimum cross-validated error.
lambda.1se	largest value of lambda such that error is within 1 standard error of the minimum.
alpha.optimal	optimal alpha corresponding to lambda.min.
lambdas.optimal	the sequence of lambdas containing lambda.min.
foldids	the fold assignment used.
dfs	list of data frames containing mean cross-validated error summaries and estimated coefficients in each fold.
fit	if refit = TRUE, summaries corresponding to the optimal alpha and lambdas. This is used to plot solution path

.

References

Dai, B., and Breheny, P. (2019). *Cross validation approaches for penalized Cox regression. arXiv preprint arXiv:1905.10432*.

Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2011) *Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5) 1-13* doi:10.18637/jss.v039.i05.

See Also

plot.pcoxtimecv, pcoxtime

Examples

```
# Time-independent covariates
if (packageVersion("survival")>="3.2.9") {
   data(cancer, package="survival")
} else {
   data(veteran, package="survival")
}

cv1 <- pcoxtimecv(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, alphas = 1
, refit = FALSE
, lamfract = 0.6
)
print(cv1)

# Train model using optimal alpha and lambda
fit1 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, alpha = cv1$alpha.optimal
, lambda = cv1$lambda.min
)
print(fit1)
# Time-varying covariates
data(heart, package="survival")
cv2 <- pcoxtimecv(Surv(start, stop, event) ~ age + year + surgery + transplant
, data = heart
, alphas = 1
, refit = FALSE
, lamfract = 0.6
)
print(cv2)

# Train model
fit2 <- pcoxtime(Surv(start, stop, event) ~ age + year + surgery + transplant
, data = heart
, alpha = cv2$alpha.optimal
, lambda = cv2$lambda.min
```

```
)
print(fit2)
```

`plot.pcoxsurvfit` *Plot survival and cumulative hazard curves*

Description

Plot estimated survival and cumulative hazard curves for `pcoxtime` model.

Usage

```
## S3 method for class 'pcoxsurvfit'
plot(
  x,
  ...,
  type = c("surv", "cumhaz"),
  lsize = 0.3,
  lcol = "black",
  compare = FALSE
)
```

Arguments

`x`	a `pcoxsurvfit.pcoxtime` or `pcoxbasehaz.pcoxtime` object.
`...`	for future implementations
`type`	type of curve to generate. Either `type = "surv"` for survival curves or `type = "cumhaz"` for cumulative hazard curve.
`lsize`	line size for the curves.
`lcol`	colour for the curves.
`compare`	logical. Whether to return plot with labels to add additional `geom` object for comparison. Default is `FALSE`.

Details

Depending on the specification in `pcoxsurvfit.pcoxtime`, this function plots either average or individual survival or cumulative hazard curves. The plot is a `ggplot` object, hence can be be customized further, see example below.

Value

a `ggplot` object.

Examples

```
library(ggplot2)
data(heart, package="survival")
lam <- 0.02
alp <- 1
pfit <- pcoxtime(Surv(start, stop, event) ~ age + year + surgery + transplant
, data = heart
, lambda = lam
, alpha = alp
)

# Plot survival curves
psurv <- pcoxsurvfit(pfit)
plot(psurv)

# Baseline survival curve
bsurv <- pcoxbasehaz(pfit, centered = FALSE)
plot(bsurv)

# Compare overall and baseline cumulative hazard
p1 <- plot(psurv, type = "cumhaz", compare = TRUE)
df2 <- data.frame(time = bsurv$time, cumhaz = bsurv$hazard)
p2 <- (p1
+ geom_step(data = df2, aes(x = time, y = cumhaz, group = 1, col = "baseline"))
+ scale_colour_manual(name = "C. hazard"
, values = c("#E41A1C", "#000000")
, labels = c("baseline", "overall")
)
)
print(p2)
```

`plot.pcoxtimecv` *Plot solution path for pcoxtimecv*

Description

Plots the cross-validation curve, and upper and lower standard deviation curves, as a function of the optimal lambdas. Also, plots the solution path as a function of optimal lambdas (or randomly picked fold, if `refit` = `FALSE`) or `l1`-norm.

Usage

```
## S3 method for class 'pcoxtimecv'
plot(
  x,
  ...,
  type = c("cve", "fit"),
```

```
  xvar = c("lambda", "l1"),
  show_nzero = FALSE,
  seed = 1234,
  geom = c("point", "line"),
  g.size = 0.2,
  g.col = "red",
  bar.col = g.col,
  scales = "free_x",
  show_min_cve = TRUE
)
```

Arguments

x	fitted pcoxtimecv object.
...	for future implementations
type	which plot to return. type = "cve" (default) return a cross-validation curve and type = "fit" returns coefficient profiles (solution path). See details.
xvar	only if type = "fit". Plot coefficients a function of either lambda (xvar = "lambda") or l1-norm (xvar = "l1").
show_nzero	logical. Whether to show number of nonzero coefficients on the plot. Default is show_nzero = FALSE. Still experimental for type = "cve".
seed	random number generator. Important if refit = FALSE in pcoxtimecv.
geom	geom ("point" or "line") for partial likelihood
g.size	size specification for points/lines
g.col	colour specification for points/lines
bar.col	colour specification for error bars
scales	should scales be "fixed", "free", "free_x" or "free_y"?
show_min_cve	whether or not to show the alpha which gives minimum cross-validation error. Ignored if a single alpha is specified. This replaced "Optimal" in the version 1.01.1 and below.

Details

To plot solution path corresponding to optimal alpha and lambda, set refit = TRUE in pcoxtimecv. The plot is a ggplot object, hence can be be customized further.

Value

a ggplot object.

Examples

```
library(ggplot2)
# Time-varying covariates
## Not run:
```

```
data(heart, package="survival")
# Using a vector of alphas = (0.8, 1)
cv1 <- pcoxtimecv(Surv(start, stop, event) ~ age + year + surgery + transplant
, data = heart
, alphas = c(0.8, 1)
, refit = TRUE
, lamfract = 0.6
, seed = 1234
)
# Plot cross-validation curves
plot(cv1, type = "cve")

# Plot
plot(cv1, type = "fit")

## End(Not run)
```

`plot.Score` *Prediction performance*

Description

Plots predictive performance of `pcoxtime` in comparison to other models. It uses risk scoring from `Score`. `pcoxtime` also supports performance measure scoring by R package `pec`. See examples.

Usage

```
## S3 method for class 'Score'
plot(x, ..., type = c("roc", "auc", "brier"), pos = 0.3)
```

Arguments

`x`	`Score` object. See examples.
`...`	for future implementations.
`type`	metric to return. Choices are `"roc"`, `"auc"`, `"brier"`.
`pos`	spacing between the lines.

Details

Implements plot method for `Score` for time-dependent Brier score, AUC and ROC. However, currently, no support for time-dependent covariate models.

Value

a `ggplot` object.

Examples

```
if (packageVersion("survival")>="3.2.9") {
   data(cancer, package="survival")
} else {
   data(veteran, package="survival")
}
# pcoxtime
lam <- 0.1
alp <- 1
pfit1 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, lambda = lam
, alpha = alp
)

# coxph
cfit1 <- coxph(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, method = "breslow"
, x = TRUE
, y = TRUE
)

# Evaluate model performance at 90, 180, 365 time points
score_obj <- Score(list("coxph" = cfit1, "pcox" = pfit1)
, Surv(time, status) ~ 1
, data = veteran
, plots = "roc"
, metrics = c("auc", "brier")
, B = 10
, times = c(90, 180, 365)
)

# Plot AUC
plot(score_obj, type = "auc")
# Plot ROC
plot(score_obj, type = "roc")
# Plot brier
plot(score_obj, type = "brier")

# Prediction error using pec package
## Not run:
if (require("pec")) {
pec_fit <- pec(list("coxph" = cfit1, "pcox" = pfit1)
, Surv(time, status) ~ 1
, data = veteran
, splitMethod = "Boot632plus"
, keep.matrix = TRUE
)
plot(pec_fit)
}
```

```
## End(Not run)
```

`plot.varimp` *Generic method for plotting variable importance*

Description

Plots variable importance for `pcoxtime` fit.

Usage

```
## S3 method for class 'varimp'
plot(x, ..., pos = 0.5, drop_zero = TRUE)
```

Arguments

`x`	a `varimp` object.
`...`	for future implementations.
`pos`	spacing between labels.
`drop_zero`	if `TRUE` only nonzero estimates are shown.

See Also

`varimp`

`predict.pcoxtime` *Prediction for pcoxtime model*

Description

Compute fitted values and model terms for the pcoxtime model.

Usage

```
## S3 method for class 'pcoxtime'
predict(
  object,
  ...,
  newdata = NULL,
  type = c("lp", "risk", "expected", "terms", "survival"),
  terms = object$predictors,
  na.action = na.pass
)
```

Arguments

object	fitted pcoxtime object
...	for future implementations.
newdata	optional data frame containing the variables appearing on the right hand side of pcoxtime formula. If absent, the predictions are for the data frame used in the original fit.
type	the type of predicted value. Either linear predictor ("lp"), the risk score ("risk" equivalently exp(lp)), the expected number of events given the covariates and follow-up time ("expected"), the terms of linear predictor ("terms") and the survival probability for each individual ("survival").
terms	if type = "terms", this argument can be used to specify which terms to be return. Default is all.
na.action	defines the missing value action for the newdata. If newdata is absent, then the behavior of missing is dictated by the na.action option of the original fit.

Details

The computation of these predictions similar to those in predict.coxph. Our current implementation does not incorporate stratification.

Value

a vector of predictions, depending on the type.

Examples

```
data(heart, package="survival")
lam <- 0.1
alp <- 0.8
pfit <- pcoxtime(Surv(start, stop, event) ~ age + year + surgery + transplant
, data = heart
, lambda = lam
, alpha = alp
)

predict(pfit, type = "lp")
predict(pfit, type = "expected")
predict(pfit, type = "risk")
predict(pfit, type = "survival")
predict(pfit, type = "terms")
```

`predictRisk.pcoxtime` *Extract predictions from pcoxtime model*

Description

Extract event probabilities from the fitted model.

Usage

```
## S3 method for class 'pcoxtime'
predictRisk(object, newdata, times, ...)
```

Arguments

`object`	fitted `pcoxtime`.
`newdata`	a data frame containing the variables appearing on the right hand side of `pcoxtime` formula.
`times`	a vector of times in the range of the response, at which to return the survival probabilities.
`...`	for future implementations.

Details

For survival outcome, the function predicts the risk, $1 - S(t|x)$, where $S(t|x)$ is the survival chance of an individual characterized by x.

Value

a matrix of probabilities with as many rows as the rows of the `newdata` and as many columns as number of time points (`times`).

Examples

```
if (packageVersion("survival")>="3.2.9") {
   data(cancer, package="survival")
} else {
   data(veteran, package="survival")
}
# Penalized
lam <- 0.1
alp <- 0.5
pfit1 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, lambda = lam
, alpha = alp
)
r1 <- predictRisk(pfit1, newdata = veteran[1:80,], times = 10)
```

```
# Unpenalized
lam <- 0
alp <- 1
pfit2 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, lambda = lam
, alpha = alp
)
r2 <- predictRisk(pfit2, newdata = veteran[1:80,], times = 10)
plot(r1, r2, xlim=c(0,1), ylim=c(0,1)
, xlab = "Penalized predicted survival chance at 10"
, ylab="Unpenalized predicted survival chance at 10"
)
```

`predictSurvProb.pcoxtime` *Predict survival probabilities at various time points*

Description

The function extracts the survival probability predictions from a `pcoxtime` model.

Usage

```
## S3 method for class 'pcoxtime'
predictSurvProb(object, newdata, times, ...)
```

Arguments

`object`	fitted `pcoxtime`.
`newdata`	a data frame containing the variables appearing on the right hand side of `pcoxtime` formula.
`times`	a vector of times in the range of the response, at which to return the survival probabilities.
`...`	for future implementations.

Value

a matrix of probabilities with as many rows as the rows of the `newdata` and as many columns as number of time points (`times`).

Examples

```
if (packageVersion("survival")>="3.2.9") {
   data(cancer, package="survival")
} else {
   data(veteran, package="survival")
}
# Penalized
lam <- 0.1
alp <- 0.5
pfit1 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, lambda = lam
, alpha = alp
)
p1 <- predictSurvProb(pfit1, newdata = veteran[1:80,], times = 10)

# Unpenalized
lam <- 0
alp <- 1
pfit2 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, lambda = lam
, alpha = alp
)
p2 <- predictSurvProb(pfit2, newdata = veteran[1:80,], times = 10)
plot(p1, p2, xlim=c(0,1), ylim=c(0,1)
, xlab = "Penalized predicted survival chance at 10"
, ylab="Unpenalized predicted survival chance at 10"
)
```

---

print.pcoxbasehaz *Print baseline hazard function data frame*

Description

Print the head of baseline hazard function data frame.

Usage

```
## S3 method for class 'pcoxbasehaz'
print(x, n = 5, ...)
```

Arguments

x	the result of a call to the pcoxbasehaz.pcoxtime function.
n	number of rows to print. Default is 5.
...	for future implementations

Details

Provide a summary of pcoxbasehaz.pcoxtime object.

Value

The call to the pcoxbasehaz.pcoxtime and the head of baseline hazard function data frame.

---

print.pcoxsurvfit *Print a short summary of survival function*

Description

Print the number of observations and number of events.

Usage

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

Arguments

x	the result of a call to the pcoxsurvfit.pcoxtime function.
...	for future implementations

Details

Provide a summary of pcoxsurvfit.pcoxtime object.

Value

The call to the pcoxsurvfit.pcoxtime and the summary of the survival function.

---

print.pcoxtime *Print coefficients from a pcoxtime object*

Description

This function prints a summary of the pcoxtime object.

Usage

```
## S3 method for class 'pcoxtime'
print(x, ..., nprint = 10)
```

Arguments

x	fitted pcoxtime model object
...	for future implementations
nprint	number of coefficients to print out

Details

The call that produced pcoxtime is printed, followed by coefficient estimates with their corresponding exponentiated values. Depending on the number of coefficients, nprint can be used to specify the number of coefficients to print out.

Value

A two column output, the first column is the coefficient estimate and the second column is the exponent of the coefficient estimate. Additonal summary about the number of nonzero coefficents, the number of observations and the number of event of interest are also printed.

---

print.pcoxtimecv *Print cross-validated pcoxtime object*

Description

Print the summary of the result of cross-validation for a pcoxtime object.

Usage

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

Arguments

x	pcoxtimecv object
...	for future implementations

Details

A summary of optimal lambda and alpha for training pcoxtime model.

Value

The call to the pcoxtimecv and the summary of the optimal alpha and lambdas.

`varimp.pcoxtime` *Compute variable or coefficient importance score*

Description

Compute variable or coefficient importance score

Usage

```
## S3 method for class 'pcoxtime'
varimp(
  object,
  newdata,
  type = c("coef", "perm", "model"),
  relative = TRUE,
  nrep = 50,
  parallelize = FALSE,
  nclusters = 1,
  estimate = c("mean", "quantile"),
  probs = c(0.025, 0.5, 0.975),
  seed = NULL,
  ...
)
```

Arguments

`object`	fitted `pcoxtime`.
`newdata`	data frame containing the variables appearing on the right hand side of `pcoxtime` formula.
`type`	if `type = "coef"` or `type = "model"` absolute value of estimated coefficients is computed. If `type = "perm"` variable level importance is computed using permutation.
`relative`	logical. If `TRUE` the scores are divided by the absolute sum of the coefficients.
`nrep`	number of replicates for permutations. Default is `nrep = 50`.
`parallelize`	whether to run in parallel. Default is `FALSE`.
`nclusters`	number of cores to use if `parallelize = TRUE`.
`estimate`	character string specify which summary statistic to use for the estimates. Default is `"mean"`.
`probs`	numeric vector of probabilities with values in `[0,1]`.
`seed`	a single value for for random number generation.
`...`	for future implementation.

Details

Absolute value of the coefficients (parameters) corresponding the pcoxtime object (type = "coef"). Otherwise, variable level importance is computed using permutation (type = "perm"). In the case of permutation: given predictors x_1, x_2, ..., x_n used to predict the survival outcome, y. Suppose, for example, x_1 has low predictive power for the response. Then, if we randomly permute the observed values for x_1, then the prediction for y will not change much. Conversely, if any of the predictors highly predicts the response, the permutation of that specific predictor will lead to a considerable change in the predictive measure of the model. In this case, we conclude that this predictor is important. In our implementation, Harrel's concordance index is used to measure the prediction accuracy.

Value

a named vector of variable scores (estimate = "mean") or a data frame (estimate = "quantile").

Examples

```
if (packageVersion("survival")>="3.2.9") {
   data(cancer, package="survival")
} else {
   data(veteran, package="survival")
}
# Penalized
lam <- 0.1
alp <- 0.5
pfit1 <- pcoxtime(Surv(time, status) ~ factor(trt) + karno + diagtime + age + prior
, data = veteran
, lambda = lam
, alpha = alp
)
imp1 <- varimp(pfit1, veteran)
plot(imp1)
```

Index

coef.pcoxtime, 2
coef.pcoxtimecv, 3
coefficients.pcoxtime *(*coef.pcoxtime*)*, 2
coefficients.pcoxtimecv
*(*coef.pcoxtimecv*)*, 3
concordance, *4*
concordScore *(*concordScore.pcoxtime*)*, 4
concordScore.pcoxtime, 4, *12*
coxph, *9, 10*

extractoptimal
*(*extractoptimal.pcoxtimecv*)*, 5
extractoptimal.pcoxtimecv, 5

ggplot, *15*, *17, 18*

model.frame, *9, 10*
model.matrix.default, *9*

pcoxbasehaz *(*pcoxsurvfit.pcoxtime*)*, 6
pcoxbasehaz.pcoxtime, *6*, *15*, *24, 25*
pcoxsurvfit *(*pcoxsurvfit.pcoxtime*)*, 6
pcoxsurvfit.pcoxtime, *6*, 6, *15*, *25*
pcoxtheme, 7
pcoxtime, *2*, *4*, *6, 7*, 8, *10*, *13, 14*, *20–23*,
26–28
pcoxtimecv, *3*, *5*, *9, 10*, 11, *13*, *17*, *26*
plot.pcoxsurvfit, *7*, 15
plot.pcoxtimecv, *13, 14*, 16
plot.Score, 18
plot.varimp, 20
predict.coxph, *21*
predict.pcoxtime, 20
predictRisk *(*predictRisk.pcoxtime*)*, 22
predictRisk.pcoxtime, 22
predictSurvProb
*(*predictSurvProb.pcoxtime*)*, 23
predictSurvProb.pcoxtime, 23
print.pcoxbasehaz, 24
print.pcoxsurvfit, 25
print.pcoxtime, 25
print.pcoxtimecv, 26

Score, *18*
Surv, *8*, *12*
survfit.coxph, *6*

varimp, *20*
varimp *(*varimp.pcoxtime*)*, 27
varimp.pcoxtime, 27