Package 'pch'

February 1, 2021

Type Package

Title Piecewise Constant Hazard Models for Censored and Truncated Data

Version 2.0

Date 2021-01-29

Author Paolo Frumento

Maintainer Paolo Frumento <`paolo.frumento@unipi.it`>

Description Piecewise constant hazard models for survival data.
The package allows for right-censored, left-truncated, and interval-censored data.

Depends survival, Hmisc

Imports stats

License GPL-2

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-02-01 15:30:03 UTC

R topics documented:

`pchreg` *Piecewise Constant Hazard Models*

Description

This function estimates piecewise exponential models on right-censored, left-truncated, or interval-censored data. The function is mainly intended for prediction and, unlike the `phreg` function available in the **eha** package, it allows the effect of covariates, and not just the baseline hazard, to depend on time.

Usage

```
pchreg(formula, breaks, data, weights, splinex = NULL)
```

Arguments

`formula`	an object of class "`formula`": a symbolic description of the regression model. The response must be a `Surv` object as returned by `Surv` (see 'Details').
`breaks`	either a numeric vector of two or more unique cut points, or a single number giving the number of intervals. If missing, the number and position of the `breaks` are determined automatically.
`data`	an optional data frame containing the variables in the model.
`weights`	an optional vector of weights to be used in the fitting process. The weights will be normalized to sum to the sample size. This implies that, for example, using double weights will not halve the standard errors.
`splinex`	either `NULL`, or an object created with `splinex` (see 'Details').

Details

The left side of the `formula` must be specified as `Surv(time, event)`, for right-censored data; `Surv(time0, time, event)`, for right-censored and left-truncated data (`time0 < time`, `time0` can be `-Inf`); and `Surv(time1, time2, type = "interval2")` for interval-censored data (use `time1 = time2` for exact observations, `time1 = -Inf` or `NA` for left-censored, and `time2 = Inf` or `NA` for right-censored). Using `Surv(time)` is also allowed and indicates that the data are neither censored nor truncated. Note that the response variable (and thus the `breaks`) can be negative.

To fit the model, the time interval is first divided in sub-intervals as defined by `breaks`. When the location of `breaks` is not specified, the empirical quantiles are used as cut points. A different costant hazard (exponential) model is then fitted in each sub-interval, modeling the log-hazard as a linear function of covariates. The special function `splinex` can be used to build flexible models.

This type of model can be utilized to obtain a nonparametric maximum likelihood estimator of a conditional distribution, achieving the flexibility of nonparametric estimators while keeping the model parametric in practice. Users unfamiliar with this approach are recommended to read Geman and Hwang (1982) for an overview, and the paper by Ackerberg, Chen and Hahn (2012) describing how this approach can be applied to simplify inference in two-step semiparametric models.

Value

An object of class "`pch`", which is a list with the following items:

`call`	the matched call.
`beta`	a matrix of regression coefficients. Rows correspond to covariates, while columns correspond to different time intervals.
`breaks`	the used cut points, with attributes `'h'` indicating the length of each interval, and `'k'` denoting the number of intervals.
`covar`	the estimated asymptotic covariance matrix.
`logLik`	the value of the maximized log-likelihood, with attribute "`df`" indicating the number of free model parameters.
`lambda`	the fitted hazard values in each interval.
`Lambda`	the fitted cumulative hazard values at the end of each interval.
`mf`	the model frame used.
`x`	the model matrix.
`conv.status`	a code indicating the convergence status. It takes value `0` if the algorithm has converged successfully; `1` if convergence has not been achieved; and `2` if, although convergence has been achieved, more than 1% of observations have an associated survival numerically equal to zero, indicating that the solution may not be well-behaved or the model is misspecified.

The accessor functions `summary`, `coef`, `predict`, `nobs`, `logLik`, `AIC`, `BIC` can be used to extract information from the fitted model. This function is mainly intended for prediction and simulation: see `predict.pch`.

Note

NOTE1. Right-censoring is a special case of interval censoring, in which exact events are identified by `time2 = time1`, while censored observations have `time2 = Inf`. Note, however, that `pchreg` will *not* use the same routines for right-censored and interval-censored data, implying that `pchreg(Surv(time1,time2,type = "interval2") ~ x)` may not be identical to `pchreg(Surv(time = time1,event = (time2 < Inf)) ~ x)`. The latter is usually faster and slightly more accurate.

NOTE2. Within each interval, the risk of the event may be zero at some covariate values. For each covariate `x`, the algorithm will try to identify a threshold `c` such that all events (in any given interval) occur when `x < c` (`x > c`). A zero risk will be automatically fitted above (below) the threshold, using an offset of `-100` on the log-hazard.

Author(s)

Paolo Frumento <paolo.frumento@unipi.it>

References

Ackerberg, D., Chen, X., and Hahn, J. (2012). A Practical Asymptotic Variance Estimator for Two-Step Semiparametric Estimators. The Review of Economics and Statistics, 94(2), 481-498.

Friedman, M. (1982). Piecewise Exponential Models for Survival Data with Covariates. The Annals of Statistics, 10(1), pp. 101-113.

Geman, S., and Hwang, C.R. (1982). Nonparametric Maximum Likelihood Estimation by the Method of Sieves. The Annals of Statistics,10(2), 401-414.

See Also

`predict.pch`, `splinex`

Examples

```
# Simulate right-censored data

n <- 1000
x <- runif(n) # a covariate
time <- rexp(n, exp(1 + x)) # time-to-event
cens <- runif(n,0,2) # censoring event
y <- pmin(time,cens) # observed variable
d <- (time <= cens) # indicator of the event
model <- pchreg(Surv(y,d) ~ x, breaks = 10)




# Simulate right-censored, left-truncated data

n <- 1000
x <- runif(n) # a covariate
time0 <- rexp(n, 10) # time at enrollment
time <- rexp(n, exp(1 + x)) # time-to-event
cens <- runif(n,0,2) # censoring event

# y,d,x are only observed if (y > time0)
y <- pmin(time,cens)
d <- (time <= cens)
u <- (y > time0)
y <- y[u]
d <- d[u]
x <- x[u]
z <- time0[u]
model <- pchreg(Surv(z,y,d) ~ x, breaks = 10)




# Simulate interval-censored data

n <- 1000
x <- runif(n) # a covariate
time <- 10*rexp(n, exp(1 + x)) # time-to-event
time1 <- floor(time)
```

```
time2 <- ceiling(time)
# Individuals are observed at discrete times
# I observe (time1,time2) such that time1 <= time <= time2
model <- pchreg(Surv(time1,time2, type = "interval2") ~ x, breaks = 10)




# Try summary(model), predict(model)
# See the documentation of predict.pch for more examples
```

`predict.pch` *Prediction from Fitted Piecewise Constant Hazard Models*

Description

This function returns predictions for an object of class "pch", usually the result of a call to `pchreg`.

Usage

```
## S3 method for class 'pch'
predict(object, type = c("distr", "quantile", "sim"),
   newdata, p, sim.method = c("quantile", "sample"), ...)
```

Arguments

`object`	a "pch" object.
`type`	a character string (just the first letter can be used) indicating the type of prediction. See 'Details'.
`newdata`	optional data frame in which to look for variables with which to predict. It must include all the covariates that enter the model and, if `type = 'distr'`, also the time variable (see 'Details' for additional information of interval-censored data). If `newdata` is omitted, the original data will be used.
`p`	vector of quantiles, to be specified if `type = "quantile"`.
`sim.method`	a character string (just the first letter can be used) indicating the simulation method if `type = "sim"`. Only `sim.method = 'quantile'` is valid with interval-censored data.
`...`	for future methods.

Details

If `type = "distr"` (the default), this function returns a data frame with columns `(haz,Haz,Surv,f)` containing the fitted values of the hazard function, the cumulative hazard, the survival function, and the probability density function, respectively.

If `type = "quantile"`, a data frame with the fitted quantiles (corresponding to the supplied values of `p`) is returned.

If `type = "sim"`, new data are simulated from the fitted model. Two methods are available: with `sim.method = "quantile"`, data are simulated by applying the estimated quantile function to a vector of random uniform numbers; if `sim.method = "sample"`, the quantile function is only used to identify the time interval, and the data are resampled from the observed values in the interval. The second method only works properly if there is a large number of breaks. However, it is less sensitive to model misspecification and facilitates sampling from distributions with a probability mass or non compact support. This method is not applicable to interval-censored data.

Predictions are computed at `newdata`, if supplied. Note that `newdata` must include all the variables that are needed for the prediction, and that if `type = "distr"`, new values of the response variable are also required. If the data are interval-censored between `time1` and `time2`, these will *not* be used as time-to-events and `newdata` must include a variable `'time'` at which to compute predictions.

Value

If `type = "distr"`, a 4-columns data frame with columns `(haz,Haz,Surv,f)`. If `type = "quantile"`, a named data frame with a column for each value of `p`. If `type = "sim"`, a vector of simulated data.

The presence of missing values in the response or the covariates will always cause the prediction to be `NA`.

Note

If the data are right-censored, some high quantiles may not be estimated: beyond the last observable quantile, all types of predictions (including `type = "sim"` with `sim.method = "sample"`) are computed assuming that the hazard remains constant after the last interval.

Author(s)

Paolo Frumento <paolo.frumento@unipi.it>

See Also

`pchreg`

Examples

```
# using simulated data

##### EXAMPLE 1 - Continuous distribution ############################

n <- 1000
x <- runif(n)
time <- rnorm(n, 1 + x, 1 + x) # time-to-event
cens <- rnorm(n,2,2) # censoring variable
y <- pmin(time,cens) # observed variable
d <- (time <= cens) # indicator of the event
model <- pchreg(Surv(y,d) ~ x, breaks = 20)

# predicting hazard, cumulative hazard, survival, density
```

```
pred <- predict(model, type = "distr")
plot(pred$Surv, 1 - pnorm(y, 1 + x, 1 + x)); abline(0,1)
# true vs fitted survival


# predicting quartiles

predQ <- predict(model, type = "quantile", p = c(0.25,0.5,0.75))
plot(x,time)
points(x, qnorm(0.5, 1 + x, 1 + x), col = "red") # true median
points(x, predQ$p0.5, col = "green")             # fitted median


# simulating new data

tsim1 <- predict(model, type = "sim", sim.method = "quantile")
tsim2 <- predict(model, type = "sim", sim.method = "sample")

qt <- quantile(time, (1:9)/10)  # deciles of t
q1 <- quantile(tsim1, (1:9)/10) # deciles of tsim1
q2 <- quantile(tsim2, (1:9)/10) # deciles of tsim2

par(mfrow = c(1,2))
plot(qt,q1, main = "sim.method = 'quantile'"); abline(0,1)
plot(qt,q2, main = "sim.method = 'sample'"); abline(0,1)

# prediction with newdata

predict(model, type = "distr", newdata = data.frame(y = 0, x = 0.5)) # need y!
predict(model, type = "quantile", p = 0.5, newdata = data.frame(x = 0.5))
predict(model, type = "sim", sim.method = "sample", newdata = data.frame(x = c(0,1)))

##### EXAMPLE 2 - non-compact support ############################
# to simulate, sim.method = "sample" is recommended ##############

n <- 1000
t <- c(rnorm(n,-5), rnorm(n,5))
model <- pchreg(Surv(t) ~ 1, breaks = 30)

tsim1 <- predict(model, type = "sim", sim.method = "quantile")
tsim2 <- predict(model, type = "sim", sim.method = "sample")

par(mfrow = c(1,3))
hist(t, main = "true distribution")
hist(tsim1, main = "sim.method = 'quantile'") # the empty spaces are 'filled'
hist(tsim2, main = "sim.method = 'sample'")   # perfect!
```

`splinex` *Including Splines in Piecewise Constant Hazard Regression*

Description

This function can be used within a call to `pchreg` to automatically include spline functions in the linear predictor of the model.

Usage

```
splinex(method = c("ns", "bs"), df = 2, degree = 2, v = 0.98, ...)
```

Arguments

`method`	a character string indicating whether natural splines (`ns`) or B-splines (`bs`) should be used. Default is "`ns`".
`df`	the degrees of freedom of the spline basis.
`degree`	the degree of the polynomial (only for `bs`).
`v`	a value between `0` and `1` determining how many principal components of the design matrix must be used in model fitting (see "Details").
`...`	for future arguments.

Details

The piecewise constant hazard model implemented by `pchreg` can be used as a nonparametric maximum likelihood estimator, in which the number of parameters is allowed to increase with the sample size in order to achieve any desired flexibility. Modeling the effect of covariates is as important as setting a sufficiently large number of `breaks`.

By letting `splinex = splinex(...)`, each column of the original design matrix is automatically replaced by the corresponding spline basis, defined by `method`, `df`, and `degree`.

This modeling approach has the drawback of generating a potentially large design matrix. To reduce its dimension, select `v < 1`. With this option, the original design matrix will be converted into principal components, and only the PCs explaining at least a proportion `v` of the variance will be used to fit the model (see "Examples").

Value

The function returns its arguments, to be passed to an internal function `build.splinex` that actually computes the design matrix.

Note

A multidimensional spline can be created by including a tensor product of splines, e.g., `ns(x1,df)*ns(x2,df)`. This is not supported by `splinex`, as it may generate a very large design matrix.

Author(s)

Paolo Frumento <paolo.frumento@unipi.it>

See Also

`pchreg`

Examples

```
require(splines)
n <- 1000
x1 <- runif(n,-2,2)
x2 <- runif(n,-2,2)
t <- rexp(n, exp(-abs(x1 - x2)))

# a simple model
model1 <- pchreg(Surv(t) ~ x1 + x2)

# using splinex: the same as ~ ns(x1, df = 2) + ns(x2, df = 2)
model2 <- pchreg(Surv(t) ~ x1 + x2, splinex = splinex("ns", v = 1))

# include interaction: ~ ns(x1, df = 2) + ns(x2, df = 2) + ns(x1*x2, df = 2)
model3 <- pchreg(Surv(t) ~ x1 * x2, splinex = splinex("ns", v = 1))

# the same as model 3, only keep the PCs explaining at least 95 percent of the variance
model4 <- pchreg(Surv(t) ~ x1 * x2, splinex = splinex("ns", v = 0.95))

# true CDF vs fitted

trueF <- pexp(t, exp(-abs(x1 - x2)))
par(mfrow = c(2,2))
plot(trueF, 1 - predict(model1)$Surv); abline(0,1, col = "red", lwd = 2) # does not fit
plot(trueF, 1 - predict(model2)$Surv); abline(0,1, col = "red", lwd = 2) # neither
plot(trueF, 1 - predict(model3)$Surv); abline(0,1, col = "red", lwd = 2) # great!
plot(trueF, 1 - predict(model4)$Surv); abline(0,1, col = "red", lwd = 2) # almost as good
```

Index