Package 'ahaz'

November 4, 2024

Type Package

Title Regularization for Semiparametric Additive Hazards Regression

Version 1.15.1

Date 2022-05-12

Depends R (>= 2.10), survival, Matrix, methods

Description Computationally efficient procedures for regularized estimation with the semiparametric additive hazards regression model.

License GPL-2

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-11-04 18:35:24 UTC

Author Anders Gorst-Rasmussen [aut, cre]

Maintainer Anders Gorst-Rasmussen <`agorstras@gmail.com`>

Contents

`ahaz` *Fit semiparametric additive hazards model*

Description

Fit a semiparametric additive hazards regression model. Right-censored and left-truncated survival data are supported.

Usage

```
ahaz(surv, X, weights, univariate=FALSE, robust=FALSE)
```

Arguments

`surv`	Response in the form of a survival object, as returned by the function `Surv()` in the package **survival**. Right-censoring and left-truncation is supported. Tied survival times are not supported.
`X`	Design matrix. Missing values are not supported.
`weights`	Optional vector of observation weights. Default is 1 for each observation.
`univariate`	Fit all univariate models instead of the joint model. Default is `univar = FALSE`.
`robust`	Robust calculation of variance. Default is `robust = FALSE`.

Details

The semiparametric additive hazards model specifies a hazard function of the form:

$$h(t) = h_0(t) + \beta' Z_i$$

for $i = 1, \ldots, n$ where Z_i is the vector of covariates, β the vector of regression coefficients and h_0 is an unspecified baseline hazard. The semiparametric additive hazards model can be viewed as an additive analogue of the well-known Cox proportional hazards regression model.

Estimation is based on the estimating equations of Lin & Ying (1994).

The option `univariate` is intended for screening purposes in data sets with a large number of covariates. It is substantially faster than the standard approach of combining `ahaz` with `apply`, see the examples.

Value

An object with S3 class "ahaz".

call	The call that produced this object.
nobs	Number of observations.
nvars	Number of covariates.
D	A nvars x nvars matrix (or vector of length nvars if univar = TRUE).
d	A vector of length nvars; the regression coefficients equal solve(D,d).
B	An nvars x nvars matrix such that $D^{-1}BD^{-1}$ estimates the covariance matrix of the regression coefficients. If robust=FALSE then B is estimated using an asymptotic approximation; if robust=TRUE then B is estimated from residuals, see residuals.
univariate	Is univariate=TRUE?
data	Formatted version of original data (for internal use).
robust	Is robust=TRUE?

References

Lin, D.Y. & Ying, Z. (1994). *Semiparametric analysis of the additive risk model.* Biometrika; **81**:61-71.

See Also

summary.ahaz, predict.ahaz, plot.ahaz. The functions coef, vcov, residuals.

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,15:24])

# Fit additive hazards model
fit1 <- ahaz(surv, X)
summary(fit1)

# Univariate models
X <- as.matrix(sorlie[,3:ncol(sorlie)])
fit2 <- ahaz(surv, X, univariate = TRUE)
# Equivalent to the following (slower) solution
beta <- apply(X,2,function(x){coef(ahaz(surv,x))})
plot(beta,coef(fit2))
```

`ahaz.adjust` *Adjusted univariate association measures from ahaz*

Description

Fast calculation of univariate association measures in the semiparametric additive risk model, adjusted for user-specified covariates

Usage

```
ahaz.adjust(surv, X, weights, idx, method=c("coef", "z", "crit"))
```

Arguments

`surv`	Response in the form of a survival object, as returned by the function `Surv()` in the package **survival**. Right-censored and counting process format (left-truncation) is supported. Tied survival times are not supported.
`X`	Design matrix. Missing values are not supported.
`weights`	Optional vector of observation weights. Default is 1 for each observation.
`idx`	Vector specifying the indices of the covariates to adjust for.
`method`	The type of adjusted association measure to calculate. See details.

Details

The function is intended mainly for **programming use** and screening purposes, when a very large number of covariates are considered and direct application of `ahaz` is unfeasible.

Running this function is equivalent to running `ahaz` with design matrix `cbind(X[,i],X[,idx])` for each column `X[,i]` in `X`. By utilizing basic matrix identities, `ahaz.adjust` runs many times faster.

The following univariate association measures are currently implemented:

- `method="z"`, Z-statistics, obtained from a fitted `ahaz` model.
- `method="coef"`, regression coefficients, obtained from a fitted `ahaz` model.
- `method="crit"`, the increase in the natural loss function of the semiparametric additive hazards model when the covariate is included in the model.

Value

A list containing the following elements:

`call`	The call that produced this object.
`idx`	A copy of the argument `idx`.
`adj`	Adjusted association statistic, as specified by `method`. Entries with index in `idx` are set to `NA`.

See Also

ahaz, ahaz.partial, ahazisis.

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,3:ncol(sorlie)])

# Adjust for first 10 covariates
idx <- 1:10
a <- ahaz.adjust(surv,X,idx=idx)

# Compare with (slower) solution
b <- apply(X[,-idx],2,function(x){coef(ahaz(surv,cbind(x,X[,idx])))[1]})
plot(b,a$adj[-idx])
```

`ahaz.partial` *Partial calculation of estimating quantities used by ahaz*

Description

Partial calculation of the quantities used in the estimating equations for ahaz.

Usage

```
ahaz.partial(surv, X, weights, idx)
```

Arguments

`surv`	Response in the form of a survival object, as returned by the function `Surv()` in the package **survival**. Right-censored and counting process format (left-truncation) is supported. Tied survival times are not supported.
`X`	Design matrix. Missing values are not supported.
`weights`	Optional vector of observation weights. Default is 1 for each observation.
`idx`	Vector of indices of covariates to use in the calculations.

Details

The function is intended mainly for **programming use** when a very large number of covariates are considered and direct application of `ahaz` is unfeasible.

The estimating equations for the semiparametric additive hazards model are of the form $D\beta = d$ with D a quadratic matrix with number of columns equal to the number of covariates. The present function returns `d[idx]`, `D[idx,]`, and `B[idx,]`; the latter a matrix such that $D^{-1}BD^{-1}$ estimates the covariance matrix of the regression coefficients.

Value

A list containing the following elements:

`call`	The call that produced this object.
`idx`	A copy of the argument `idx`.
`nobs`	Number of observations.
`nvars`	Number of covariates.
`d`	Vector of length `length(idx)`.
`D`	Matrix of size `length(idx)` x `nvars`.
`B`	Matrix of size `length(idx)` x `nvars`.

See Also

`ahaz`, `ahaz.adjust`.

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,3:ncol(sorlie)])

# Get D for the first 10 covariates only
a<-ahaz.partial(surv,X,idx=1:10)
pD1 <- a$D

# Equivalent to the (slower) solution
b <- ahaz(surv,X)
pD2 <- b$D[1:10,]
max(abs(pD1-pD2))
```

ahaz.tune.control *Tuning controls for regularization*

Description

Define the type of tuning method used for regularization. Currently only used by `tune.ahazpen`.

Usage

```
# Cross-validation
cv.control(nfolds=5, reps=1, foldid=NULL, trace=FALSE)

# BIC-inspired
bic.control(factor = function(nobs){log(nobs)})
```

Arguments

`nfolds`	Number of folds for cross-validation. Default is `nfolds=5`. Each fold must have size > 1, i.e. `nfolds` must be less than half the sample size.
`reps`	Number of repetitions of cross-validation with `nfolds` folds. Default is `rep=1`. A `rep` larger than 1 can be useful to reduce variance of cross-validation scores.
`foldid`	An optional vector of values between 1 and `nfolds` identifying the fold to which each observation belongs. Supercedes `nfolds` and `rep` if supplied.
`trace`	Print progress of cross-validation. Default is `trace=FALSE`.
`factor`	Defines how strongly the number of nonzero penalty parameters penalizes the score in a BIC-type criterion; see the details.

Details

For examples of usage, see `tune.ahazpen`.

The regression coefficients of the semiparametric additive hazards model are estimated by solving a linear system of estimating equations of the form $D\beta = d$ with respect to β. The natural loss function for such a linear function is of the least-squares type

$$L(\beta) = \beta' D\beta - 2d'\beta.$$

This loss function is used for cross-validation as described by Martinussen & Scheike (2008).

Penalty parameter selection via a BIC-inspired approach was described by Gorst-Rasmussen & Scheike (2011). With *df* is the degrees of freedom and n the number of observations, we consider a BIC inspired criterion of the form

$$BIC = \kappa L(\beta) + df \cdot factor(n)$$

where κ is a scaling constant included to remove dependency on the time scale and better mimick the behavior of a 'real' (likelihood) BIC. The default `factor=function(n){log(n)}` has desirable theoretical properties but may be conservative in practice.

Value

An object with S3 class `"ahaz.tune.control"`.

`type`	Type of penalty.
`factor`	Function specified by `factor`, if applicable
`getfolds`	A function specifying how folds are calculated, if applicable.
`rep`	How many repetitions of cross-validation, if applicable.
`trace`	Print out progress?

References

Gorst-Rasmussen, A. & Scheike, T. H. (2011). *Independent screening for single-index hazard rate models with ultra-high dimensional features.* Technical report R-2011-06, Department of Mathematical Sciences, Aalborg University.

Martinussen, T. & Scheike, T. H. (2008). *Covariate selection for the semiparametric additive risk model.* Scandinavian Journal of Statistics; **36**:602-619.

See Also

`tune.ahazpen`

`ahazisis` *Independent screening for the semiparametric additive hazards model*

Description

Fast and scalable model selection for the semiparametric additive hazards model via univariate screening combined with penalized regression.

Usage

```
ahazisis(surv, X, weights, standardize=TRUE,
       nsis=floor(nobs/1.5/log(nobs)), do.isis=TRUE,
       maxloop=5, penalty=sscad.control(), tune=cv.control(),
       rank=c("FAST","coef","z","crit"))
```

Arguments

`surv`	Response in the form of a survival object, as returned by the function `Surv()` in the package **survival**. Right-censored and counting process format (left-truncation) is supported. Tied survival times are not supported.
`X`	Design matrix. Missing values are not supported.
`weights`	Optional vector of observation weights. Default is 1 for each observation.
`standardize`	Logical flag for variable standardization, prior to model fitting. Estimates are always returned on the original scale. Default is `standardize=TRUE`.

`nsis`	Number of covariates to recruit initially. If `do.isis=TRUE`, then this is also the maximal number of variables that the algorithm will recruit. Default is `nsis=floor(nobs/log(nobs)/1.5)`

.

`do.isis`	Perform iterated independent screening?
`maxloop`	Maximal number of iterations of the algorithm if `do.isis=TRUE`.
`rank`	Method to use for (re)recruitment of variables. See details.
`penalty`	A description of the penalty function to be used for the variable selection part. This can be a character string naming a penalty function (currently `"lasso"` or stepwise SCAD, `"sscad"`) or a call to the penalty function. Default is `penalty=sscad.control()`. See `ahazpen` and `ahazpen.pen.control` for more options and examples.
`tune`	A description of the tuning method to be used for the variable selection part. This can be a character string naming a tuning control function (currently `"cv"` or `"bic"`) or a call to the tuning control function. Default is `tune=cv.control()`. See `ahaz.tune.control` for options and examples.

Details

The function is a basic implementation of the iterated sure independent screening method described in Gorst-Rasmussen & Scheike (2011). Briefly, the algorithm does the following:

1. Recruits the `nsis` most relevant covariates by ranking them according to the univariate ranking method described by `rank`.
2. Selects, using `ahazpen` with penalty function described in `penalty`, a model among the top two thirds of the `nsis` most relevant covariates. Call the size of this model m.
3. Recruits '`nsis` minus m' new covariates among the non-selected covariates by ranking their relevance according to the univariate ranking method described in `rank`, adjusted for the already selected variables (using an unpenalized semiparametric additive hazards model).

Steps 2-3 are iterated for `maxloop` times, or until `nsis` covariates has been recruited, or until the set of selected covariate is stable between two iterations; whichever comes first.

The following choices of ranking method exist:

- `rank="FAST"` corresponds to ranking, in the initial recruitment step only, by the basic FAST-statistic described in Gorst-Rasmussen & Scheike (2011). If `do.isis=TRUE` then the algorithm sets `rank="z"` for subsequent rankings.
- `rank="coef"` corresponds to ranking by absolute value of (univariate) regression coefficients, obtained via `ahaz`
- `rank="z"` corresponds to ranking by the $|Z|$-statistic of the (univariate) regression coefficients, obtained via `ahaz`
- `rank="crit"` corresponds to ranking by the size of the decrease in the (univariate) natural loss function used for estimation by `ahaz`.

Value

An object with S3 class "ahazisis".

call	The call that produced this object.
initRANKorder	The initial ranking order.
detail.pickind	List (of length at most maxloop) listing the covariates selected in each recruitment step.
detail.ISISind	List (of length at most maxloop) listing the covariates selected in each variable selection step.
detail.ISIScoef	List (of length at most maxloop) listing the estimated penalized regression coefficients corresponding to the indices in detail.ISISind.
SISind	Indices of covariates selected in the initial recruitment step.
ISISind	Indices of the final set of covariates selected by the iterated algorithm.
ISIScoef	Vector of the penalized regression coefficients of the covariates in ISISind.
nsis	The argument nsis.
do.isis	The argument do.isis.
maxloop	The argument maxloop.

References

Gorst-Rasmussen, A. & Scheike, T. H. (2011). *Independent screening for single-index hazard rate models with ultra-high dimensional features.* Technical report R-2011-06, Department of Mathematical Sciences, Aalborg University.

See Also

print.ahazisis, ahazpen, ahaz.adjust

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,3:ncol(sorlie)])

# Basic ISIS/SIS with a single step
set.seed(10101)
m1 <- ahazisis(surv,X,maxloop=1,rank="coef")
m1
# Indices of the variables from the initial recruitment step
m1$SISind
```

```
# Indices of selected variables
m1$ISISind
# Check fit
score <- X[,m1$ISISind]%*%m1$ISIScoef
plot(survfit(surv~I(score>median(score))))
```

---

ahazpen *Fit penalized semiparametric additive hazards model*

Description

Fit a semiparametric additive hazards model via penalized estimating equations using, for example, the lasso penalty. The complete regularization path is computed at a grid of values for the penalty parameter lambda via the method of cyclic coordinate descent.

Usage

```
ahazpen(surv, X, weights,  standardize=TRUE,  penalty=lasso.control(),
        nlambda=100, dfmax=nvars, pmax=min(nvars, 2*dfmax),
        lambda.minf=ifelse(nobs < nvars,0.05, 1e-4), lambda,
        penalty.wgt=NULL, keep=NULL, control=list())
```

Arguments

`surv`	Response in the form of a survival object, as returned by the function `Surv()` in the package **survival**. Right-censored and counting process format (left-truncation) is supported. Tied survival times are not supported.
`X`	Design matrix. Missing values are not supported.
`weights`	Optional vector of observation weights. Default is 1 for each observation.
`standardize`	Logical flag for variable standardization, prior to model fitting. Estimates are always returned on the original scale. Default is `standardize=TRUE`.
`penalty`	A description of the penalty function to be used for model fitting. This can be a character string naming a penalty function (currently `"lasso"` or stepwise SCAD, `"sscad"`) or a call to the desired penalty function. See `ahazpen.pen.control` for the available penalty functions and advanced options; see also the examples.
`nlambda`	The number of `lambda` values. Default is `nlambda=100`.
`dfmax`	Limit the maximum number of variables in the model. Unless a complete regularization path is needed, it is highly recommended to initially choose a relatively smaller value of `dfmax` to substantially reduce computation time.
`pmax`	Limit the maximum number of variables to ever be considered by the coordinate descent algorithm.

`lambda.minf`	Smallest value of `lambda`, as a fraction of `lambda.max`, the (data-derived) smallest value of `lambda` for which all regression coefficients are zero. The default depends on the sample size `nobs` relative to the number of variables `nvars`. If `nobs` >= `nvars`, the default is `0.0001`, close to zero. When `nobs` < `nvars`, the default is `0.05`.
`lambda`	An optional user supplied sequence of penalty parameters. Typical usage is to have the program compute its own `lambda` sequence based on `nlambda` and `lambda.minf`. A user-specified lambda sequence overrides `dfmax` but not `pmax`.
`penalty.wgt`	A vector of nonnegative penalty weights for each regression coefficient. This is a number that multiplies `lambda` to allow differential penalization. Can be 0 for some variables, which implies no penalization so that the variable is always included in the model; or `Inf` which implies that the variable is never included in the model. Default is 1 for all variables.
`keep`	A vector of indices of variables which should always be included in the model (no penalization). Equivalent to specifying a `penalty.wgt` of 0.
`control`	A list of parameters for controlling the model fitting algorithm. The list is passed to `ahazpen.fit.control`.

Details

Fits the sequence of models implied by the penalty function `penalty`, the sequence of penalty parameters `lambda` by using the very efficient method of cyclic coordinate descent.

For data sets with a very large number of covariates, it is recommended to only calculate partial paths by specifying a smallish value of `dmax`.

The sequence `lambda` is computed automatically by the algorithm but can also be set (semi)manually by specifying `nlambda` or `lambda`. The stability and efficiency of the algorithm is highly dependent on the grid `lambda` values being reasonably dense, and `lambda` (and `nlambda`) should be specified accordingly. In particular, it is not recommended to specify a single or a few lambda values. Instead, a partial regularization path should be calculated and the functions `predict.ahazpen` or `coef.ahazpen` should be used to extract coefficient estimates at specific lambda values.

Value

An object with S3 class `"ahazpen"`.

`call`	The call that produced this object
`beta`	An `nvars` x `length(lambda)` matrix (in sparse column format, class `dgCMatrix`) of penalized regression coefficients.
`lambda`	The sequence of actual `lambda` values used.
`df`	The number of nonzero coefficients for each value of `lambda`.
`nobs`	Number of observations.
`nvars`	Number of covariates.
`surv`	A copy of the argument `survival`.
`npasses`	Total number of passes by the fitting algorithm over the data, for all lambda values.

`penalty.wgt`	The actually used `penalty.wgt`.
`penalty`	An object of class `ahaz.pen.control`, as specified by `penalty`.
`dfmax`	A copy of `dfmax`.
`penalty`	A copy of `pmax`.

References

Gorst-Rasmussen A., Scheike T. H. (2012). *Coordinate Descent Methods for the Penalized Semiparametric Additive Hazards Model*. Journal of Statistical Software, **47**(9):1-17. https://www.jstatsoft.org/v47/i09/

Gorst-Rasmussen, A. & Scheike, T. H. (2011). *Independent screening for single-index hazard rate models with ultra-high dimensional features.* Technical report R-2011-06, Department of Mathematical Sciences, Aalborg University.

Leng, C. & Ma, S. (2007). *Path consistent model selection in additive risk model via Lasso*. Statistics in Medicine; **26**:3753-3770.

Martinussen, T. & Scheike, T. H. (2008). *Covariate selection for the semiparametric additive risk model.* Scandinavian Journal of Statistics; **36**:602-619.

Zou, H. & Li, R. (2008). *One-step sparse estimates in nonconcave penalized likelihood models*, Annals of Statistics; **36**:1509-1533.

See Also

`print.ahazpen`, `predict.ahazpen`, `coef.ahazpen`, `plot.ahazpen`, `tune.ahazpen`.

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,3:ncol(sorlie)])

# Fit additive hazards regression model
fit1 <- ahazpen(surv, X,penalty="lasso", dfmax=30)
fit1
plot(fit1)

# Extend the grid to contain exactly 100 lambda values
lrange <- range(fit1$lambda)
fit2 <- ahazpen(surv, X,penalty="lasso", lambda.minf=lrange[1]/lrange[2])
plot(fit2)

# User-specified lambda sequence
lambda <- exp(seq(log(0.30), log(0.1), length = 100))
fit2 <- ahazpen(surv, X, penalty="lasso", lambda = lambda)
```

```
plot(fit2)

# Advanced usage - specify details of the penalty function
fit4 <- ahazpen(surv, X,penalty=sscad.control(nsteps=2))
fit4
fit5 <- ahazpen(surv, X,penalty=lasso.control(alpha=0.1))
plot(fit5)
```

`ahazpen.fit.control` *Controls for ahazpen fitting algorithm*

Description

Controls the numerical algorithm for fitting the penalized semiparametric additive hazards model. This is typically only used in a call to `ahazpen`.

Usage

```
ahazpen.fit.control(thresh=1e-5, maxit=100000, ...)
```

Arguments

`thresh`	Declare convergence when the maximal relative change from the last iteration is less than `thresh`. Default is `thresh=1e-5`.
`maxit`	Maximal number passes by the algorithm over the data for all values of the regularization parameter lambda. Default is `maxit=100000`.
`...`	For future methods.

Value

A list with elements named as the arguments.

See Also

`ahazpen`

`ahazpen.pen.control` *Penalty controls for ahazpen*

Description

Describe the penalty function to be used in the penalized semiparametric additive hazards model. Typically only used in a call to `ahazpen` or `tune.ahazpen`.

Usage

```
# (Adaptive) lasso/elasticnet
lasso.control(alpha=1, ada.wgt=NULL)

# Stepwise SCAD
sscad.control(a=3.7, nsteps=1, init.sol=NULL, c=NULL)
```

Arguments

`alpha`	Elasticnet penalty parameter with default `alpha=1` corresponding to the standard lasso; see details.
`ada.wgt`	Optional covariate weights used for fitting the adaptive lasso. Default is *not* to use weights, i.e. fit the standard lasso. A user-specified `init.sol` can be a nonnegative vector of length corresponding to the number of covariates in the model. For advanced use it may also be specified as a function with arguments `surv`, `X` and `weights` precisely; see the details.
`a`	Parameter of the stepwise SCAD penalty, see details. Default is `a=3.7`
`nsteps`	Number of steps in stepwise SCAD. Default is `nsteps=1`.
`init.sol`	Optional initial solution for stepwise SCAD consisting of a numerical vector of length corresponding to the number of covariates in the model. Default is a vector of regression coefficients obtained from `ahaz` if there are more observations than covariates, zero otherwise. For advanced use, `initsol` it can also be specified as a **function** with arguments `surv`, `X` and `weights` precisely; see the details.
`c`	Optional scaling factor for stepwise SCAD. Usually it is not necessary to change supply this; see the details.

Details

The lasso/elasticnet penalty function takes the form

$$p_\lambda(\beta) = \lambda((1-\alpha)\|\beta\|_2 + \alpha\|\beta\|_1)$$

where $0 < \alpha \leq 1$. Choosing $\alpha < 1$ encourages joint selection of correlated covariates and may improve the fit when there are substantial correlations among covariates.

The stepwise SCAD penalty function takes the form

$$p_\lambda(\beta) = w_\lambda(c|b_1|)|\beta_1| + \ldots + w_\lambda(c|b_{nvars}|)|\beta_{nvars}|$$

where b is some initial estimate, c is a scaling factor, and for I the indicator function

$$w_\lambda(x) = \lambda I(x \leq \lambda) + \frac{(a\lambda - x)_+}{a-1} I(x > \lambda)$$

The scaling factor c controls how ‘different’ the stepwise SCAD penalty is from the standard lasso penalty (and is also used to remove dependency of the penalty on the scaling of the time axis).

The one-step SCAD method of Zou & Li (2008) corresponds to taking b equal to the estimator derived from `ahaz`. See Gorst-Rasmussen & Scheike (2011) for details. By iterating such one-step SCAD and updating the initial solution b accordingly, the algorithm approximates the solution obtained using full SCAD. Note that calculating the full SCAD solution generally leads to a non-convex optimization problem: multiple solutions and erratic behavior of solution paths can be an issue.

The arguments `ada.wgt` and `init.sol` can be specified as functions of the observations. This is convenient, for example, when using cross-validation for tuning parameter selection. Such a function **must** be specified precisely with the arguments `surv`, `X` and `weights` and **must** output a numeric vector of length corresponding to the number of covariates. `ahazpen` will take care of scaling so the function should produce output on the original scale. See the examples here as well as the examples for `tune.ahazpen` for usage of this feature in practice.

Value

An object with S3 class `"ahaz.pen.control"`.

`type`	Type of penalty.
`init.sol`	Function specifying the initial solution, if applicable.
`alpha`	Value of `alpha`, if applicable.
`nsteps`	Number of steps for stepwise SCAD penalty, if applicable.
`a`	Parameter for stepwise SCAD penalty, if applicable.
`c`	Scaling factor for stepwise SCAD penalty, if applicable.
`ada.wgt`	Function specifying the weights for the adaptive lasso penalty, if applicable.

References

Gorst-Rasmussen, A. & Scheike, T. H. (2011). *Independent screening for single-index hazard rate models with ultra-high dimensional features.* Technical report R-2011-06, Department of Mathematical Sciences, Aalborg University.

Leng, C. & Ma, S. (2007). *Path consistent model selection in additive risk model via Lasso*. Statistics in Medicine; **26**:3753-3770.

Martinussen, T. & Scheike, T. H. (2008). *Covariate selection for the semiparametric additive risk model.* Scandinavian Journal of Statistics; **36**:602-619.

Zou, H. & Li, R. (2008). *One-step sparse estimates in nonconcave penalized likelihood models*, Annals of Statistics; **36**:1509-1533.

See Also

ahazpen, tune.ahazpen

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,3:ncol(sorlie)])

# Fit additive hazards regression model with elasticnet penalty
model <- ahazpen(surv,X,penalty=lasso.control(alpha=0.1),dfmax=30)
plot(model)

# Adaptive lasso with weights 1/|beta_i|^0.5. Note that, although
# we do not use 'weights', it MUST be included as an argument
adafun <- function(surv,X,weights)
 return(1/abs(coef(ahaz(surv,X)))^.5)
model <- ahazpen(surv,X[,1:50],penalty=lasso.control(ada.wgt=adafun))
plot(model)

# One-step SCAD with initial solution derived from univariate regressions
scadfun <- function(surv,X,weights){
 fit <- ahaz(surv,X,univariate=TRUE)
 return(coef(fit))
}
set.seed(10101)
model.ssc <- tune.ahazpen(surv,X,dfmax=30,penalty=sscad.control(init.sol=scadfun))
plot(model.ssc)
```

---

plot.ahaz *Plot an ahaz object*

Description

Plot method for a fitted semiparametric additive hazards model; plots the Breslow estimate of underlying cumulative hazard function.

Usage

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

Arguments

`x`	The result of an `ahaz` fit.
`...`	Additional graphical arguments passed to the `plot` function.

Details

Calling `plot.ahaz` is equivalent to first calling `ahaz`, then calling `predict` with `type="cumhaz"`, and finally calling `plot`.

See Also

`ahaz`, `predict.ahaz`, `plot.cumahaz`

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,15:24])

# Fit additive hazards model
fit <- ahaz(surv, X)
plot(fit)
```

`plot.ahazpen` *Plot an ahazpen object*

Description

Plots regularization paths for fitted penalized semiparametric additive hazards model.

Usage

```
## S3 method for class 'ahazpen'
plot(x, xvar=c("norm","lambda"), labels=FALSE, df=TRUE,
                ylab="Regression coefficients", xlab=xname,...)
```

Arguments

`x`	The result of an `ahazpen` fit.
`xvar`	Scaling for first axis. Options are the L^1 norm of the vector of regression coefficients (`"norm"`) or the penalty parameter on a log scale (`"lambda"`).

`labels`	Try to display indices for the regression coefficients in the right-hand margin. Default is `labels=FALSE`.
`df`	Display number of nonzero parameters in top margin. Default is `df=TRUE`.
`ylab`	Label for y-axis.
`xlab`	Label for x-axis. The default is either "L1 norm" or λ, depending on `xvar`.
`...`	Additional graphical arguments passed to the `plot` function.

See Also

`ahazpen`, `print.ahazpen`, `predict.ahazpen`, `coef.ahazpen`.

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,3:ncol(sorlie)])

# Fit additive hazards regression model
fit <- ahazpen(surv, X, dfmax=50)
par(mfrow=c(1,2)); plot(fit); plot(fit,xvar="lambda")

# With labels only
plot(fit,labels=TRUE,df=FALSE)
```

`plot.cumahaz` *Plot a cumahaz object*

Description

Plots the Breslow estimate of cumulative hazard function, as obtained from the `predict.ahaz`

Usage

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

Arguments

`x`	Result of a call to the `predict.ahaz` function with option `type="cumhaz"`.
`...`	Additional graphical arguments passed to the `plot` function.

See Also

predict.ahaz, predict.ahazpen

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,15:24])

# Fit additive hazards regression model
fit <- ahaz(surv, X)

# Cumulative hazard
cumhaz <- predict(fit, type="cumhaz")
plot(cumhaz)
```

---

plot.tune.ahazpen *Plot a tune.ahazpen object*

Description

Plot, as a function of the penalty parameter, the curve of tuning scores produced when tuning a penalized semiparametric additive hazards model.

Usage

```
## S3 method for class 'tune.ahazpen'
plot(x, df = TRUE, ...)
```

Arguments

x	The result of a call to `tune.ahazpen`.
df	Display number of nonzero parameters in top margin. Default is `df=TRUE`.
...	Additional graphical arguments passed to the `plot` function.

Details

A plot is produced displaying the tuning score for each value of penalty parameter (alongside upper and lower standard deviation curves, if cross-validation has been used). The value of `lambda` which minimizes the estimated tuning score is indicated with a dashed vertical line.

See Also

ahazpen, tune.ahazpen, print.tune.ahazpen.

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,3:ncol(sorlie)])

# Do 10 fold cross-validation
set.seed(10101)
tune.fit <- tune.ahazpen(surv, X, penalty="lasso",
              dfmax=50, tune = cv.control(nfolds=10))
plot(tune.fit)
```

---

predict.ahaz *Prediction methods for ahaz*

Description

Compute regression coefficients, linear predictor, cumulative hazard function, or integrated martingale residuals for a fitted semiparametric additive hazards model.

Usage

```
## S3 method for class 'ahaz'
predict(object, newX, type=c("coef", "lp",
        "residuals", "cumhaz"), beta=NULL, ...)
## S3 method for class 'ahaz'
coef(object, ...)
## S3 method for class 'ahaz'
vcov(object, ...)
## S3 method for class 'ahaz'
residuals(object, ...)
```

Arguments

`object`	The result of an ahaz fit.
`newX`	Optional new matrix of covariates at which to do predictions. Currently only supported for `type="lp"`.

`type`	Type of prediction. Options are the regression coefficients (`"coef"`), the linear predictor (`"lp"`), the martingale residuals (`"residuals"`), or the cumulative hazard (`"cumhaz"`). See the details.
`beta`	Optional vector of regression coefficients. If unspecified, the regression coefficients derived from `object` are used.
`...`	For future methods.

Details

The Breslow estimator of the baseline cumulative hazard is described in Lin & Ying (1994).

The regression coefficients β_0 in the semiparametric additive hazards model are obtained as the solution $\hat{\beta}$ to a quadratic system of linear equations $D\beta = d$. The (integrated) martingale residuals ϵ_i for $i = 1, ..., n$ are vectors, of length corresponding to the number of covariates, so that

$$D(\hat{\beta} - \beta_0) - d \approx \epsilon_1 + \cdots + \epsilon_n$$

The residuals estimate integrated martingales and are asymptotically distributed as mean-zero IID multivariate Gaussian. They can be used to derive a sandwich-type variance estimator for regression coefficients (implemented in `summary.ahaz` when `robust=TRUE` is specified). They can moreover be used for implementing consistent standard error estimation under clustering; or for implementing resampling-based inferential methods.

See Martinussen & Scheike (2006), Chapter 5.4 for details.

Value

For `type="coef"` and `type="lp"`, a vector of predictions.

For `type="coef"`, a matrix of (integrated) martingale residuals, with number of columns corresponding to the number of covariates.

For `type="cumhaz"`, an object with S3 class `"cumahaz"` consisting of:

`time`	Jump times for the cumulative hazard estimate.
`cumhaz`	The cumulative hazard estimate.
`event`	Status at jump times (1 corresponds to death, 0 corresponds to entry/exit).

References

Martinussen, T. & Scheike, T. H. & (2006). *Dynamic Regression Models for Survival Data.* Springer.

See Also

`ahaz`, `summary.ahaz`, `plot.cumahaz`.

Examples

```
data(sorlie)

set.seed(10101)

# Break ties
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,15:24])

# Fit additive hazards regression model
fit <- ahaz(surv, X)

# Parameter estimates
coef(fit)

# Linear predictor, equivalent to X%*%coef(fit)
predict(fit,type="lp")

# Cumulative baseline hazard
cumahaz <- predict(fit, type="cumhaz")

# Residuals - model fit
resid <- predict(fit, type = "residuals")
# Decorrelate, standardize, and check QQ-plots
stdres <- apply(princomp(resid)$scores,2,function(x){x/sd(x)})
par(mfrow = c(2,2))
for(i in 1:4){
  qqnorm(stdres[,i])
  abline(c(0,1))
}

# Residuals - alternative variance estimation
resid <- residuals(fit)
cov1 <- summary(fit)$coef[,2]
invD <- solve(fit$D)
Best<-t(resid)%*%resid
cov2 <- invD %*% Best %*% invD
# Compare with (nonrobust) SEs from 'summary.ahaz'
plot(cov1, sqrt(diag(cov2)),xlab="Nonrobust",ylab="Robust")
abline(c(0,1))
```

`predict.ahazpen` *Prediction methods for ahazpen*

Description

Compute regression coefficient estimates, linear predictor, cumulative hazard function, or integrated martingale residuals for a fitted penalized semiparametric additive hazards model.

Usage

```
## S3 method for class 'ahazpen'
predict(object, newX, type=c("coef","lp","residuals","cumhaz"),
        lambda=NULL, ...)
## S3 method for class 'ahazpen'
coef(object, ...)
```

Arguments

`object`	The result of an `ahazpen` fit.
`newX`	New matrix of covariates at which to do predictions. Required unless `type="coef"`.
`lambda`	Value of lambda for at which predictions are to be made. This argument is required for `type="residuals"` and `type="cumhaz"`. Since predictions rely on interpolations between lambda values, it is recommended not to use a lambda-value smaller than the minimum of `object$lambda`.
`type`	The type of prediction. Options are the regression coefficients ("`coef`"), the linear predictors ("`lp`"), the (integrated) martingale residuals ("`residuals`"), or the cumulative hazard ("`cumhaz`")
`...`	For future methods.

Details

See the details in `predict.ahaz` for information on the different types of predictions.

Value

For `type="coef"` and `type="lp"`, a matrix of regression coefficients, respectively linear predictions for each value of the penalty parameter.

For `type="residuals"`, a matrix of (integrated) martingale residuals associated with the nonzero penalized regression coefficients for a regularization parameter equal to $lambda$.

For `type="cumhaz"`, an object with S3 class `"cumahaz"` based on the regression coefficients estimated for a regularization parameter equal to $lambda$, the object containing:

`time`	Jump times for the cumulative hazard estimate.
`cumhaz`	The cumulative hazard estimate.
`event`	Status at jump times (1 corresponds to death, 0 corresponds to entry/exit).

See Also

`ahazpen`, `print.ahazpen`, `plot.ahazpen`, `predict.ahaz`, `plot.cumahaz`.

Examples

```
data(sorlie)

set.seed(10101)

# Break ties
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,3:ncol(sorlie)])

# Fit additive hazards regression model w/lasso penalty
fit <- ahazpen(surv, X, dfmax=100)

# Coefficients
beta <- predict(fit,X,lambda=0.08,type="coef")
barplot(as.numeric(beta))

# Linear predictions
linpred <- predict(fit,X,lambda=0.1,type="lp")
riskgrp <- factor(linpred < median(linpred))
plot(survfit(surv~riskgrp))

# Residuals
resid <- predict(fit, X, lambda=0.1, type = "residuals")
par(mfrow = c(1,2))
hist(resid[,1],main=colnames(resid)[1])
hist(resid[,2],main=colnames(resid)[2])

# Cumulative hazard
cumhaz <- predict(fit,X,lambda=0.1,type="cumhaz")
plot(cumhaz)
```

`predict.tune.ahazpen` *Prediction methods for tune.ahazpen*

Description

Compute regression coefficient estimates, linear predictor, cumulative hazard function, or integrated martingale residuals for a fitted and tuned penalized semiparametric additive hazards model.

Usage

```
## S3 method for class 'tune.ahazpen'
predict(object, newX,  lambda="lambda.min", ...)
## S3 method for class 'tune.ahazpen'
coef(object, ...)
```

Arguments

`object`	The result of an `ahazpen` fit.
`newX`	New matrix of covariates at which to do predictions. Required for some types of predictions, see `predict.ahazpen`.
`lambda`	Value of lambda at which predictions are to be made. Required for some types of predictions, see `predict.ahazpen`. Default is the optimal lambda value saved in `object`.
`...`	Additional arguments to be passed to `predict.ahazpen` (usually the type of prediction required).

Details

See the details in `predict.ahazpen` for information on the available types of predictions.

Value

The obejct returned depends on the details in the argument `...` passed to `predict.ahazpen`.

See Also

`predict.ahazpen`, `ahazpen`, `print.ahazpen`, `plot.ahazpen`, `predict.ahaz`, `plot.cumahaz`.

Examples

```
data(sorlie)

set.seed(10101)

# Break ties
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,3:ncol(sorlie)])

# Fit additive hazards regression model w/lasso penalty
cv.fit <- tune.ahazpen(surv, X, dfmax=100, tune="cv")

# Predict coefficients at cv.fit$lambda.min
coef(cv.fit)

# Predict risk score at cv.fit$lambda.min
predict(cv.fit,newX=X,type="lp")
```

`print.ahazisis` *Print an ahazisis object*

Description

Print method for sure independence screening based on the semiparametric additive hazards model.

Usage

```
## S3 method for class 'ahazisis'
print(x, digits=max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	Fitted `ahazisis` object.
`digits`	Significant digits to print.
`...`	For future methods.

Details

The call that produced `x` is printed, alongside the number of covariates initially recruited, the number of covariates finally recruited (if applicable) and the number of iterations (if applicable).

See Also

`ahazisis`

`print.ahazpen` *Print an ahazpen object*

Description

Print method for fitted penalized semiparametric additive hazards model.

Usage

```
## S3 method for class 'ahazpen'
print(x, digits=max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	Fitted `ahazpen` object.
`digits`	Significant digits to print.
`...`	For future methods.

Details

The call that produced x is printed, alongside the number of observations, the number of covariates, and details on the sequence of penalty parameters.

See Also

ahazpen, predict.ahazpen, coef.ahazpen.

---

print.summary.ahaz *Print a summary.ahaz object*

Description

Produces a printed summary of a fitted semiparametric additive hazards model.

Usage

```
## S3 method for class 'summary.ahaz'
print(x, digits=max(getOption("digits") - 3, 3),
      signif.stars=getOption("show.signif.stars"), ...)
```

Arguments

x	The result of a call to summary.ahaz.
digits	Significant digits to print.
signif.stars	Show stars to highlight small p-values.
...	For future methods.

See Also

summary.ahaz, ahaz, plot.ahaz.

---

print.tune.ahazpen *Print a tune.ahazpen object*

Description

Print method for tune.ahazpen objects.

Usage

```
## S3 method for class 'tune.ahazpen'
print(x, digits=max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	The result of a call to `tune.ahazpen`.
`digits`	Significant digits in printout.
`...`	Additional print arguments.

Details

The call that produced `x` is printed, alongside the number of penalty parameters used, the value of the optimal penalty and the number of non-zero regression coefficients at the optimal penalty parameter.

See Also

`ahazpen`, `tune.ahazpen`, `plot.tune.ahazpen`.

`sorlie` *Sorlie gene expressions*

Description

Dataset containing 549 gene expression measurement, exit time and exit status in a study of breast cancer among 115 women.

Usage

```
data(sorlie)
```

Format

time Time to exit.

status Status at exit (censoring = 0, event = 1).

X1,...,X549 Gene expression measurements.

References

Soerlie T., et al. (2003). *Repeated observation of breast tumor subtypes in independent gene expression data sets*. Proc Natl Acad Sci **100**:8418-8423

Examples

```
data(sorlie)
```

`summary.ahaz` *Summarize an ahaz object*

Description

Produces a summary of a fitted semiparametric additive hazards model.

Usage

```
## S3 method for class 'ahaz'
summary(object, ...)
```

Arguments

`object`	The result of an `ahaz` fit.
`...`	For future methods.

Value

An object with S3 class `"summary.ahaz"`.

`call`	The call that produced this object.
`coefficients`	Vector of regression coefficients.
`cov`	Estimated covariance matrix of regression coefficients.
`nobs`	Number of observations.
`nvars`	Number of covariates
`waldtest`	Vector of quantities from a Wald test.
`univar`	Logical: summarizing univariate regressions (option `univariate` in `ahaz`)?

See Also

`ahaz`, `plot.ahaz`

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,15:25])

# Fit additive hazards model
fit1 <- ahaz(surv, X)
summary(fit1)
```

tune.ahazpen *Choice of penalty parameter in ahazpen*

Description

Tuning of penalty parameters for the penalized semiparametric additive hazards model via cross-validation - or via non-stochastic procedures, akin to BIC for likelihood-based models.

Usage

```
tune.ahazpen(surv, X, weights, standardize=TRUE, penalty=lasso.control(),
             tune=cv.control(), dfmax=nvars, lambda, ...)
```

Arguments

`surv`	Response in the form of a survival object, as returned by the function `Surv()` in the package **survival**. Right-censored and counting process format (left-truncation) is supported. Tied survival times are not supported.
`X`	Design matrix. Missing values are not supported.
`weights`	Optional vector of observation weights. Default is 1 for each observation.
`standardize`	Logical flag for variable standardization, prior to model fitting. Parameter estimates are always returned on the original scale. Default is `standardize=TRUE`.
`penalty`	A description of the penalty function to be used for model fitting. This can be a character string naming a penalty function (currently `"lasso"` or stepwise SCAD, `"sscad"`) or it can be a call to the penalty function. Default is `penalty=lasso.control()`. See `ahazpen.pen.control` for the available penalty functions and advanced options; see also the examples.
`dfmax`	Limit the maximum number of covariates included in the model. Default is `nvars=nobs-1`. Unless a complete regularization path is needed, it is **highly** recommended to initially choose a relatively smaller value of `dfmax` to reduce computation time and memory usage.
`lambda`	An optional user supplied sequence of penalty parameters. Typical usage is to have the program compute its own `lambda` sequence based on `nlambda` and `lambda.min`.
`tune`	A description of the tuning method to be used. This can be a character string naming a tuning control function (currently `"cv"` or `"bic"`) or a call to the tuning control function. Default is 5-fold cross-validation, `tune=cv.control()`, see `ahaz.tune.control` for more options. See also the examples.
`...`	Additional arguments to be passed to `ahazpen`, see `ahazpen` for options.

Details

The function performs an initial penalized fit based on the penalty supplied in `penalty` to obtain a sequence of penalty parameters. Subsequently, it selects among these an optimal penalty parameter based on the tuning control function described in `tune`, see `ahaz.tune.control`.

Value

An object with S3 class "`tune.ahazpen`".

`call`	The call that produced this object.
`lambda`	The actual sequence of `lambda` values used.
`tunem`	The tuning score for each value of `lambda` (mean cross-validated error, if `tune=cv.control()`).
`tunesd`	Estimate of the cross-validated standard error, if `tune=cv.control()`.
`tunelo`	Lower curve = `tunem-tunemsd`, if `tune=cv.control()`.
`tuneup`	Upper curve = `tunem+tunemsd`, if `tune=cv.control()`.
`lambda.min`	Value of `lambda` for which `tunem` is minimized.
`df`	Number of non-zero coefficients at each value of `lambda`.
`tune`	The selected `tune` of S3 class "`ahaz.tune.control`".
`penalty`	The selected `penalty` of S3 class "`ahazpen.pen.control`".
`foldsused`	Folds actually used, if `tune=cv.control()`.

References

Gorst-Rasmussen, A. & Scheike, T. H. (2011). *Independent screening for single-index hazard rate models with ultra-high dimensional features.* Technical report R-2011-06, Department of Mathematical Sciences, Aalborg University.

See Also

`ahaz.tune.control`, `plot.tune.ahazpen`, `ahazpen`.

Examples

```
data(sorlie)

# Break ties
set.seed(10101)
time <- sorlie$time+runif(nrow(sorlie))*1e-2

# Survival data + covariates
surv <- Surv(time,sorlie$status)
X <- as.matrix(sorlie[,3:ncol(sorlie)])

# Training/test data
set.seed(20202)
train <- sample(1:nrow(sorlie),76)
test <- setdiff(1:nrow(sorlie),train)

# Run cross validation on training data
set.seed(10101)
cv.las <- tune.ahazpen(surv[train,], X[train,],dfmax=30)
plot(cv.las)

# Check fit on the test data
```

```
testrisk <- predict(cv.las,X[test,],type="lp")
plot(survfit(surv[test,]~I(testrisk<median(testrisk))),main="Low versus high risk")

# Advanced example, cross-validation of one-step SCAD
# with initial solution derived from univariate models.
# Since init.sol is specified as a function, it is
# automatically cross-validated as well
scadfun<-function(surv,X,weights){coef(ahaz(surv,X,univariate=TRUE))}
set.seed(10101)
cv.ssc<-tune.ahazpen(surv[train,],X[train,],
                     penalty=sscad.control(init.sol=scadfun),
                     tune=cv.control(rep=5),dfmax=30)
# Check fit on test data
testrisk <- predict(cv.ssc,X[test,],type="lp")
plot(survfit(surv[test,]~I(testrisk<median(testrisk))),main="Low versus high risk")
```

Index