Package: penAFT (via r-universe)

October 28, 2026

Type Package

Title Fit the Semiparametric Accelerated Failure Time Model with Elastic Net and Sparse Group Lasso Penalties

Version 0.3.2

Description The semiparametric accelerated failure time (AFT) model is an attractive alternative to the Cox proportional hazards model. This package provides a suite of functions for fitting one popular rank-based estimator of the semiparametric AFT model, the regularized Gehan estimator. Specifically, we provide functions for cross-validation, prediction, coefficient extraction, and visualizing both trace plots and cross-validation curves. For further details, please see Suder, P. M. and Molstad, A. J., (2022) Scalable algorithms for semiparametric accelerated failure time models in high dimensions, Statistics in Medicine <doi:10.1002/sim.9264>.

License GPL (>= 2)

URL https://ajmolstad.github.io/research/

BugReports https://github.com/ajmolstad/penAFT/issues

Imports Rcpp, Matrix, ggplot2, irlba

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Author Aaron J. Molstad [aut, cre] (ORCID: <https://orcid.org/0000-0003-0645-5105>), Piotr M. Suder [aut]

Maintainer Aaron J. Molstad <amolstad@ufl.edu>

Repository https://cran.r-universe.dev

Date/Publication 2025-06-16 16:49:49 UTC

RemoteUrl https://github.com/cran/penAFT

RemoteRef HEAD

RemoteSha 398fe26d19d8b6d440c945f591e26654902c20f9

Contents

penAFT-package . . . 2
genSurvData . . . 3
penAFT . . . 4
penAFT.coef . . . 7
penAFT.cv . . . 9
penAFT.plot . . . 12
penAFT.predict . . . 13
penAFT.trace . . . 15

Index **17**

`penAFT-package` *Fit the semiparametric accelerated failure time model in high dimensions by minimizing a rank-based estimation criterion plus weighted elastic net or weighted sparse group-lasso penalty.*

Description

This package contains numerous functions related to a penalized rank-based estimator of the semiparametric AFT, the so-called penalized Gehan estimator. In particular, the main functions are for solution path computation, cross-validation, prediction, and coefficient extraction.

Details

The primary functions are `penAFT` and `penAFT.cv`, the latter of which performs cross-validation. In general, both functions fit the penalized Gehan estimator—the minimizer of a rank-based loss function. Given $(\log(y_1), x_1, \delta_1), \ldots, (\log(y_n), x_n, \delta_n)$ where y_i is the minimum of the survival time and censoring time, x_i is a p-dimensional predictor, and δ_i is the indicator of censoring, `penAFT` fits the solution path for the argument minimizing

$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\delta_i\{\log(y_i) - \log(y_j) - (x_i - x_j)'\beta\}^- + \lambda g(\beta)$$

where $\{a\}^- := \max(-a, 0)$, $\lambda > 0$, and g is either the weighted elastic net penalty or weighted sparse group lasso penalty. The weighted elastic net penalty is defined as

$$\alpha\|w \circ \beta\|_1 + \frac{(1-\alpha)}{2}\|\beta\|_2^2$$

where w is a set of non-negative weights (which can be specified in the `weight.set` argument). The weighted sparse group-lasso penalty we consider is

$$\alpha\|w \circ \beta\|_1 + (1-\alpha)\sum_{l=1}^{G} v_l\|\beta_{\mathcal{G}_l}\|_2$$

where again, w is a set of non-negative weights and v_l are weights applied to each of the G (user-specified) groups.

For a comprehensive description of the algorithm, and more details about rank-based estimation in general, please refer to the referenced manuscript.

Author(s)

Aaron J. Molstad and Piotr M. Suder Maintainer: Aaron J. Molstad <amolstad@ufl.edu>

`genSurvData` *Generate a survival dataset from the log-logistic accelerated failure time model.*

Description

This is a function for generating synthetic datasets from the log-logistic accelerated failure time model. The purpose of this function is to provide structured data for the examples of the other functions' usage.

Usage

```
genSurvData(n, p, s, mag, cens.quant = 0.6)
```

Arguments

`n`	The numer of subjects to be included in the dataset.
`p`	Dimension of the predictor. Note that the function computes the square-root of a $p \times p$ covariance matrix, so setting `p` large may be time-consuming.
`s`	The number of nonzero regression coefficients in β.
`mag`	The magnitude of the `s` nonzero regression coefficients. Signs of coefficients are assigned at random.
`cens.quant`	The quantile of true survival times used to set the mean of the exponential distribution from which censoring times are drawn. Default is 0.6.

Details

This function generates predictors to follow a p-dimensional multivariate normal distribution whose covariance has an AR(1) structure with lag 0.7. Then, log survival times are generated as

$$\log(T) = X\beta + \epsilon$$

where ϵ has independent components drawn from a logistic distribution with location parmeter zero and scale parameter two. Then censoring times are drawn from an exponential distribution with mean equal to the quantile `cens.quant` of T.

Value

`beta`	The true data generating regression coefficient vector.
`logY`	The observed failure times or censoring times on the log scale.
`status`	Indicator of censoring; a value of 1 indicates the corresponding component of logY is an observed log failure time and a value of 0 indicates a log censoring time.
`Xn`	The $n \times p$ matrix of predictors.

Examples

```
# ------------------------------------
# Generate data
# ------------------------------------
set.seed(1)
genData <- penAFT::genSurvData(n = 50, p = 100, s = 10, mag = 1, cens.quant = 0.6)
X <- genData$X
logY <- genData$logY
delta <- genData$status
str(X)
head(logY)
head(delta)
```

---

penAFT *Fit the solution path for the penalized semiparametric accelerated failure time model with weighted elastic net or weighted sparse group lasso penalties.*

Description

A function to fit the solution path for the regularized semiparametric accelerated failure time model estimator.

Usage

```
penAFT(X, logY, delta, nlambda = 50,
  lambda.ratio.min = 0.1, lambda = NULL,
  penalty = NULL, alpha = 1, weight.set = NULL,
  groups = NULL, tol.abs = 1e-8, tol.rel = 2.5e-4,
  gamma = 0,  standardize = TRUE,
  admm.max.iter = 1e4, quiet=TRUE)
```

Arguments

X	An $n \times p$ matrix of predictors. Observations should be organized by row.
logY	An n-dimensional vector of log-survival or log-censoring times.
delta	An n-dimensional binary vector indicating whether the jth component of logY is an observed log-survival time ($\delta_j = 1$) or a log-censoring time ($\delta_j = 0$) for $j = 1, \ldots, n$.
nlambda	The number of candidate tuning parameters to consider.
lambda.ratio.min	The ratio of minimum to maximum candidate tuning parameter value. As a default, we suggest 0.1, but standard model selection procedures should be applied to select λ. Must belong to (0,1).
lambda	An optional (not recommended) prespecified vector of candidate tuning parameters. Should be in descending order.

penalty	Either "EN" or "SG" for elastic net or sparse group lasso penalties.
alpha	The tuning parameter α. See documentation.
weight.set	A list of weights. For both penalties, w is an n-dimensional vector of nonnegative weights. For "SG" penalty, can also include v – a non-negative vector the length of the number of groups. See documentation for usage example.
groups	When using penalty "SG", a p-dimensional vector of integers corresponding the to group assignment of each predictor (i.e., column of X).
tol.abs	Absolute convergence tolerance.
tol.rel	Relative convergence tolerance.
gamma	A non-negative optimization parameter which can improve convergence speed in certain settings. It is highly recommended to set equal to zero.
standardize	Should predictors be standardized (i.e., column-wise average zero and scaled to have unit variance) for model fitting?
admm.max.iter	Maximum number of ADMM iterations.
quiet	TRUE or FALSE variable indicating whether progress should be printed.

Details

Given $(\log y_1, x_1, \delta_1), \ldots, (\log y_n, x_n, \delta_n)$ where y_i is the minimum of the survival time and censoring time, x_i is a p-dimensional predictor, and δ_i is the indicator of censoring, penAFT fits the solution path for the argument minimizing

$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\delta_i\{\log y_i - \log y_j - (x_i - x_j)'\beta\}^- + \lambda g(\beta)$$

where $\{a\}^- := \max(-a, 0)$, $\lambda > 0$, and g is either the weighted elastic net penalty (penalty = "EN") or weighted sparse group lasso penalty (penalty = "SG"). The weighted elastic net penalty is defined as

$$\alpha\|w \circ \beta\|_1 + \frac{(1-\alpha)}{2}\|\beta\|_2^2$$

where w is a set of non-negative weights (which can be specified in the weight.set argument). The weighted sparse group-lasso penalty we consider is

$$\alpha\|w \circ \beta\|_1 + (1-\alpha)\sum_{l=1}^{G} v_l\|\beta_{\mathcal{G}_l}\|_2$$

where again, w is a set of non-negative weights and v_l are weights applied to each of the G groups.

Value

beta	A $p\times$ nlambda sparse matrix consisting of the estimates of β for the candidate values of λ. It is recommended to use penAFT.coef to extract coefficients.
lambda	The candidate tuning parameter values.
standardize	Were predictors standardized to have unit variance for model fitting?
X.mean	The mean of the predictors.
X.sd	The standard deviation of the predictors.
alpha	The tuning parameter α. See documentation.

Examples

```
# ------------------------------------
# Generate data
# ------------------------------------
set.seed(1)
genData <- genSurvData(n = 50, p = 50, s = 10, mag = 2, cens.quant = 0.6)
X <- genData$X
logY <- genData$logY
delta <- genData$status


# ------------------------------------------------
# Fit elastic net penalized estimator
# ------------------------------------------------
fit.en <- penAFT(X = X, logY = logY, delta = delta,
                  nlambda = 50, lambda.ratio.min = 0.01,
                  penalty = "EN",
                  alpha = 1)


coef.en.10 <- penAFT.coef(fit.en, lambda = fit.en$lambda[10])


# ------------------------------------------------
# Fit weighted elastic net penalized estimator
# ------------------------------------------------
weight.set <- list("w" = c(0, 0, rep(1, 48)))
fit.weighted.en <- penAFT(X = X, logY = logY, delta = delta,
                  nlambda = 50, weight.set = weight.set,
                  penalty = "EN",
                  alpha = 1)
coef.wighted.en.10 <- penAFT.coef(fit.weighted.en, lambda = fit.weighted.en$lambda[10])


# ------------------------------------------------
# Fit ridge penalized estimator with user-specified lambda
# ------------------------------------------------
fit.ridge <- penAFT(X = X, logY = logY, delta = delta,
                  lambda = 10^seq(-4, 4, length=50),
                  penalty = "EN",
                  alpha = 0)


# ------------------------------------------------
# Fit sparse group penalized estimator
# ------------------------------------------------
groups <- rep(1:5, each = 10)
fit.sg <- penAFT(X = X, logY = logY, delta = delta,
                  nlambda = 50, lambda.ratio.min = 0.01,
                  penalty = "SG", groups = groups,
                  alpha = 0.5)
```

```
# ------------------------------------------------
# Fit weighted sparse group penalized estimator
# ------------------------------------------------
groups <- rep(1:5, each = 10)
weight.set <- list("w" = c(0, 0, rep(1, 48)),
      "v" = 1:5)
fit.weighted.sg <- penAFT(X = X, logY = logY, delta = delta,
                   nlambda = 100,
                   weight.set = weight.set,
                   penalty = "SG", groups = groups,
                   alpha = 0.5)

coef.weighted.sg.20 <- penAFT.coef(fit.weighted.sg, lambda = fit.weighted.sg$lambda[20])
```

`penAFT.coef` *Extract regression coefficients from fitted model object*

Description

A function to extract coefficients along the solution path for the regularized semiparametric acceleratred failure time model estimator.

Usage

```
penAFT.coef(fit, lambda = NULL)
```

Arguments

`fit`	A fitted model from `penAFT` or `penAFT.cv`.
`lambda`	The tuning parameter value at which to extract coefficients. If `NULL` and `fit` is a `penAFT.cv` object, will use the tuning parameter value with minimum cross-validation linear predictor score.

Details

The regression coefficients stored in the fitted model objects coming from `penAFT` or `penAFT.cv` will (i) be on the scale of standardized predictors if standardization was used (which is the default) and (ii) are stored as a specific sparse matrix so that coefficient extraction is cumbersome. This function returns the regression coefficient estimates on the original scale of the predictors for a particular tuning parmaeter value. It is important to note that this method does not return an estimate of the intercept: the intercept is absored into the error term as the Gehan loss function is invariant to translation of its inputs logY.

Value

`beta`	The coefficient estimates

Examples

```
# ------------------------------------
# Generate data
# ------------------------------------
set.seed(1)
genData <- genSurvData(n = 100, p = 50, s = 10, mag = 1, cens.quant = 0.6)
X <- genData$X
logY <- genData$logY
delta <- genData$status


# ------------------------------------
# Fit elastic net penalized estimator without CV
# ------------------------------------
fit <- penAFT(X = X, logY = logY, delta = delta,
                  nlambda = 50,
                  penalty = "EN",
                  alpha = 1)

coef.10 <- penAFT.coef(fit, lambda = fit$lambda[10])
coef.20 <- penAFT.coef(fit, lambda = fit$lambda[20])

# Cannot obtain fit at lambda not in fit$lambda
## Not run: coef.error <- penAFT.coef(fit, lambda = 10) # throws error


  # ----------------------------------------
  # Fit elastic net penalized estimator with CV
  # ----------------------------------------
  fit.cv <- penAFT.cv(X = X, logY = logY, delta = delta,
                  nlambda = 50,
                  penalty = "EN",
                  alpha = 1, nfolds = 5)

  ## --- coefficients at lambda minimizing cross-validation error
  coef.cv <- penAFT.coef(fit.cv)

  ## ---- coefficients at 10th considered lambda
  coef.cv10 <- penAFT.coef(fit.cv, lambda = fit.cv$full.fit$lambda[10])

  # ----------------------------------------
  # Repeat with sparse group lasso without CV
  # ----------------------------------------
  groups <- rep(1:10, each = 5)
  fit.sg <- penAFT(X = X, logY = logY, delta = delta,
                  nlambda = 50, groups = groups,
                  penalty = "SG",
                  alpha = 0.5)

  coef.sg.10 <- penAFT.coef(fit.sg, lambda = fit.sg$lambda[10])
  coef.sg.20 <- penAFT.coef(fit.sg, lambda = fit.sg$lambda[20])
```

```
# ------------------------------------------
# Finally, fit sparse group lasso with CV
# ------------------------------------------
groups <- rep(1:10, each = 5)
fit.sg.cv <- penAFT.cv(X = X, logY = logY, delta = delta,
                nlambda = 50, groups = groups,
                penalty = "SG",
                alpha = 0.5, nfolds = 5)

coef.sg.cv <- penAFT.coef(fit.sg.cv)
coef.sg.cv10 <- penAFT.coef(fit.sg.cv, lambda = fit.sg$full.fit$lambda[20])
```

`penAFT.cv` *Cross-validation function for fitting a regularized semiparametric accelerated failure time model*

Description

A function to perform cross-validation and compute the solution path for the regularized semiparametric accelerated failure time model estimator.

Usage

```
penAFT.cv(X, logY, delta, nlambda = 50,
  lambda.ratio.min = 0.1, lambda = NULL,
  penalty = NULL, alpha = 1,weight.set = NULL,
  groups = NULL, tol.abs = 1e-8, tol.rel = 2.5e-4,
  standardize = TRUE, nfolds = 5, cv.index = NULL,
  admm.max.iter = 1e4,quiet = TRUE)
```

Arguments

`X`	An $n \times p$ matrix of predictors. Observations should be organized by row.
`logY`	An n-dimensional vector of log-survival or log-censoring times.
`delta`	An n-dimensional binary vector indicating whether the jth component of `logY` is an observed log-survival time ($\delta_j = 1$) or a log-censoring time ($\delta_j = 0$) for $j = 1, \ldots, n$.
`nlambda`	The number of candidate tuning parameters to consider.
`lambda.ratio.min`	The ratio of minimum to maximum candidate tuning parameter value. As a default, we suggest 0.1, but standard model selection procedures should be applied to select λ. Must belong to (0,1).
`lambda`	An optional (not recommended) prespecified vector of candidate tuning parameters. Should be in descending order.

penalty	Either "EN" or "SG" for elastic net or sparse group lasso penalties.
alpha	The tuning parameter α. See documentation.
weight.set	A list of weights. For both penalties, w is an n-dimensional vector of nonnegative weights. For "SG" penalty, can also include v – a non-negative vector the length of the number of groups. See documentation for usage example.
groups	When using penalty "SG", a p-dimensional vector of integers corresponding the to group assignment of each predictor (i.e., column of X).
tol.abs	Absolute convergence tolerance.
tol.rel	Relative convergence tolerance.
standardize	Should predictors be standardized (i.e., scaled to have unit variance) for model fitting?
nfolds	The number of folds to be used for cross-validation. Default is five. Ten is recommended when sample size is especially small.
cv.index	A list of length nfolds of indices to be used for cross-validation. This is to be used if trying to perform cross-validation for both α and λ. Use with extreme caution: this overwrites nfolds.
admm.max.iter	Maximum number of ADMM iterations.
quiet	TRUE or FALSE variable indicating whether progress should be printed.

Details

Given $(\log y_1, x_1, \delta_1), \ldots, (\log y_n, x_n, \delta_n)$ where for subject i ($i = 1, \ldots, n$), y_i is the minimum of the survival time and censoring time, x_i is a p-dimensional predictor, and δ_i is the indicator of censoring, penAFT.cv performs nfolds cross-validation for selecting the tuning parameter to be used in the argument minimizing

$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\delta_i\{\log y_i - \log y_j - (x_i - x_j)'\beta\}^- + \lambda g(\beta)$$

where $\{a\}^- := \max(-a, 0)$, $\lambda > 0$, and g is either the weighted elastic net penalty (penalty = "EN") or weighted sparse group lasso penalty (penalty = "SG"). The weighted elastic net penalty is defined as

$$\alpha\|w \circ \beta\|_1 + \frac{(1-\alpha)}{2}\|\beta\|_2^2$$

where w is a set of non-negative weights (which can be specified in the weight.set argument). The weighted sparse group-lasso penalty we consider is

$$\alpha\|w \circ \beta\|_1 + (1-\alpha)\sum_{l=1}^{G} v_l\|\beta_{\mathcal{G}_l}\|_2$$

where again, w is a set of non-negative weights and v_l are weights applied to each of the G groups.

Next, we define the cross-validation errors. Let $\mathcal{V}_1, \ldots, \mathcal{V}_K$ be a random nfolds = K element partition of $[n]$ (the subjects) with the cardinality of each $\mathcal{V}_k$ (the "kth fold"") approximately equal for $k = 1, \ldots, K$. Let $\hat{\beta}_{\lambda(-\mathcal{V}_k)}$ be the solution with tuning parameter λ using only data indexed by

$[n] \setminus \{\mathcal{V}_k\}$ (i.e., outside the kth fold). Then, defining $e_i(\beta) := \log y_i - \beta' x_i$ for $i = 1, \ldots, n$, we call

$$\sum_{k=1}^{K} \left[\frac{1}{|\mathcal{V}_k|^2} \sum_{i \in \mathcal{V}_k} \sum_{j \in \mathcal{V}_k} \delta_i \{ e_i(\hat{\beta}_{\lambda(-\mathcal{V}_k)}) - e_j(\hat{\beta}_{\lambda(-\mathcal{V}_k)}) \}^- \right],$$

the cross-validated Gehan loss at λ in the kth fold, and refer to the sum over all `nfolds` = K folds as the cross-validated Gehan loss. Similarly, letting letting

$$\tilde{e}_i(\hat{\beta}_\lambda) = \sum_{k=1}^{K} (\log y_i - x_i' \hat{\beta}_{\lambda(-\mathcal{V}_k)}) \mathbf{1}(i \in \mathcal{V}_k)$$

for each $i \in [n]$, we call

$$\left[\sum_{i=1}^{n} \sum_{j=1}^{n} \delta_i \{ \tilde{e}_i(\hat{\beta}_\lambda) - \tilde{e}_j(\hat{\beta}_\lambda) \}^- \right]$$

the cross-validated linear predictor score at λ.

Value

`full.fit`	A model fit with the same output as a model fit using `penAFT`. See documentation for `penAFT` for more.
`cv.err.linPred`	A `nlambda`-dimensional vector of cross-validated linear predictor scores.
`cv.err.obj`	A `nfolds` $\times$`nlambda` matrix of cross-validated Gehan losses.
`cv.index`	A list of length `nfolds`. Each element contains the indices for subjects belonging to that particular fold.

Examples

```
 # -------------------------------------
# Generate data
# -------------------------------------
set.seed(1)
genData <- genSurvData(n = 50, p = 50, s = 10, mag = 2,  cens.quant = 0.6)
X <- genData$X
logY <- genData$logY
delta <- genData$status
p <- dim(X)[2]

# -----------------------------------------------
# Fit elastic net penalized estimator
# -----------------------------------------------
fit.en <- penAFT.cv(X = X, logY = logY, delta = delta,
                    nlambda = 10, lambda.ratio.min = 0.1,
                    penalty = "EN", nfolds = 5,
                    alpha = 1)
# ---- coefficients at tuning parameter minimizing cross-valdiation error
coef.en <- penAFT.coef(fit.en)

# ---- predict at 8th tuning parameter from full fit
```

```
Xnew <- matrix(rnorm(10*p), nrow=10)
predict.en <- penAFT.predict(fit.en, Xnew = Xnew, lambda = fit.en$full.fit$lambda[8])


  # ------------------------------------------------
  # Fit sparse group penalized estimator
  # ------------------------------------------------
  groups <- rep(1:5, each = 10)
  fit.sg <- penAFT.cv(X = X, logY = logY, delta = delta,
                    nlambda = 50, lambda.ratio.min = 0.01,
                    penalty = "SG", groups = groups, nfolds = 5,
                    alpha = 0.5)

  # ------------------------------------------------
  # Pass fold indices
  # ------------------------------------------------
  groups <- rep(1:5, each = 10)
  cv.index <- list()
  for(k in 1:5){
    cv.index[[k]] <- which(rep(1:5, length=50) == k)
  }
  fit.sg.cvIndex <- penAFT.cv(X = X, logY = logY, delta = delta,
                    nlambda = 50, lambda.ratio.min = 0.01,
                    penalty = "SG", groups = groups,
                    cv.index = cv.index,
                    alpha = 0.5)
  # --- compare cv indices
  ## Not run: fit.sg.cvIndex$cv.index  == cv.index
```

`penAFT.plot` *Plot cross-validation curves*

Description

A function for plotting the cross-validation curves for the regularized semiparametric accelerated failure time model estimator.

Usage

```
penAFT.plot(fit)
```

Arguments

`fit`	A fitted model from `penAFT.cv`.

Details

This function returns a plot with the cross-validation curves for the regularized Gehan estimator. The vertical blue line indicates the tuning parameter which minimized cross-validated linear predictor scores and the vertical black line indicates the tuning parameter which minimized the cross-validated Gehan loss according to the one-standard-error rule. The vertical axis (and blue line) denotes the cross-validated linear predictor scores whereas the right vertical axis (and black line) denotes cross-validated Gehan loss and standard errors). To make matters simple, we do not allow for customization of the plot: please refer to the source code if extensive customization is desired.

Value

No return value; prints a plot of cross-validation curves as described in Details.

Examples

```
# ------------------------------------
# Generate data
# ------------------------------------
set.seed(1)
genData <- genSurvData(n = 50, p = 100, s = 10, mag = 2, cens.quant = 0.6)
X <- genData$X
logY <- genData$logY
delta <- genData$status


# ----------------------------------------
# Fit elastic net penalized estimator with CV
# ----------------------------------------
fit.cv <- penAFT.cv(X = X, logY = logY, delta = delta,
                    nlambda = 50,
                    penalty = "EN", tol.rel = 1e-5,
                    alpha = 1, nfolds = 10)
penAFT.plot(fit.cv)
```

---

penAFT.predict — *Obtain linear predictor for new subjects using fitted model from* penAFT *or* penAFT.cv

---

Description

A function for prediction along the solution path of the regularized semiparametric accelerated failure time model estimator.

Usage

```
penAFT.predict(fit, Xnew, lambda = NULL)
```

Arguments

fit	A fitted model from penAFT or penAFT.cv.
Xnew	A matrix of dimension $n_{\text{new}} \times p$. Must be a matrix, even if $n_{\text{new}} = 1$.
lambda	The value of λ used to estimate β. If NULL and fit was obtained using nfolds non-NULL, the function will use the tuning parameter which minimized cross-validation linear predictor scores.

Details

It is important to note that the output of this function should not be treated as an estimate of the log-survival time. Because the Gehan loss function is location invariant, the intercept is absored into the error. If predictors were standardized for model fitting, this function returns $\tilde{X}_{\text{new}}\hat{\beta}$ where $\tilde{X}_{\text{new}}$ is the version of input Xnew which has been centered and standardized according to the design matrix used to fit the penAFT or penAFT.cv object. If predictors were not standardized, this function returns $X_{\text{new}}\hat{\beta}$.

We recommend input Xnew as a matrix, although if a p-dimensional vector is input, the function will detect this.

Value

preds	The matrix of linear predictors: rows correspond to rows of Xnew.

Examples

```
# ------------------------------------
# Generate data
# ------------------------------------
set.seed(1)
genData <- genSurvData(n = 50, p = 50, s = 10, mag = 2, cens.quant = 0.6)
X <- genData$X
logY <- genData$logY
delta <- genData$status

# --- generate data for two new subjects
p <- dim(X)[2]
Xnew <- rbind(rnorm(p), rnorm(p))

# ------------------------------------------------
# Fit elastic net penalized estimator without CV
# ------------------------------------------------
fit <- penAFT(X = X, logY = logY, delta = delta,
                  nlambda = 10, lambda.ratio.min = 0.1,
                  penalty = "EN",
                  alpha = 1)

# predict at 10th candidate tuning parameter
linPred.10 <- penAFT.predict(fit, Xnew = Xnew, lambda = fit$lambda[10])


  # ----------------------------------------
```

```
# Fit elastic net penalized estimator with CV
# ------------------------------------------
fit.cv <- penAFT.cv(X = X, logY = logY, delta = delta,
                    nlambda = 50,
                    penalty = "EN",
                     alpha = 1, nfolds = 5)

# --- return linear predictor at lambda minimizing cross-validation error
linPred.cv <- penAFT.predict(fit.cv, Xnew = Xnew)

# --- predict at 10th candidate tuning parameter
linPred.cv10 <- penAFT.predict(fit.cv, Xnew = Xnew, lambda = fit.cv$full.fit$lambda[10])


# ------------------------------------------
# Fit penAFT with cross-validation
# ------------------------------------------
groups <- rep(1:5, each = 10)
fit.sg.cv <- penAFT.cv(X = X, logY = logY, delta = delta,
                    nlambda = 50, groups = groups,
                    penalty = "SG",
                    alpha = 0.5, nfolds = 5)

# ---- return linear predictor at lambda minimizing cross-validation error
linPred.sg.cv <- penAFT.predict(fit.sg.cv, Xnew = Xnew)

# --- predict at 10th candidate tuning parameter
linPred.sg.cv10 <- penAFT.predict(fit.sg.cv, Xnew = Xnew, lambda = fit.sg.cv$full.fit$lambda[10])
```

`penAFT.trace` *Print trace plot for the semiparametric AFT fit using* `penAFT` *or* `penAFT.cv`

Description

Print the trace plot for the regularized Gehan estimator.

Usage

```
penAFT.trace(fit, groupNames=NULL)
```

Arguments

`fit`	A fitted model from `penAFT` or `penAFT.cv`.
`groupNames`	A list of group names to be printed when `fit` used penalty "SG".

Details

The function penAFT.trace returns a trace plot for a fitted model obtained from either penAFT or penAFT.cv. If the model is fit using the sparse group-lasso penalty, you may provide names for the groups (in order of the integer values specifying the groups). This feature may not be desired if there are a large number of groups, however. The vertical blue line indicates the tuning parameter which minimized cross-validated linear predictor scores and the vertical black line indicates the tuning parameter minimizing the cross-validated Gehan loss according to the one-standard error rule.

Value

No return value; prints a trace plot as described in Details.

Examples

```
# ------------------------------------
# Generate data
# ------------------------------------
set.seed(1)
genData <- genSurvData(n = 50, p = 50, s = 10, mag = 2, cens.quant = 0.6)
X <- genData$X
logY <- genData$logY
delta <- genData$status


# ----------------------------------------
# Fit elastic net penalized estimator with CV
# ----------------------------------------
fit.cv <- penAFT.cv(X = X, logY = logY, delta = delta,
                    nlambda = 10,
                    penalty = "EN",
                    alpha = 1, nfolds = 5)

# -- print plot
penAFT.trace(fit.cv)


  # ----------------------------------------
  # Fit sparse group-lasso estimator with CV
  # ----------------------------------------
  groups <- rep(1:5, length=10)
  fit.sg.cv <- penAFT.cv(X = X, logY = logY, delta = delta,
                     nlambda = 50, groups = groups,
                     penalty = "SG", tol.rel= 1e-5,
                     alpha = 0, nfolds = 5)

  penAFT.trace(fit.sg.cv, groupNames = paste("Group", 1:5, sep="-"))
```

Index

∗ **package**
penAFT-package, 2

genSurvData, 3

penAFT, 4
penAFT-package, 2
penAFT.coef, 7
penAFT.cv, 9
penAFT.plot, 12
penAFT.predict, 13
penAFT.trace, 15