Package ‘glmnetr’

December 14, 2022

Title Relaxed Lasso Model for Data Which Might Have Long Run Times
Using 'glmnet'

Version 0.1-1

Date 2022-12-10

Depends R (>= 3.4.0)

Suggests R.rsp

VignetteBuilder R.rsp

Imports glmnet, survival, Matrix

ByteCompile Yes

Author Walter K Kremers [aut, cre] (<https://orcid.org/0000-0001-5714-3473>)

Maintainer Walter K Kremers <kremers.walter@mayo.edu>

Description For some datasets, for example when the design matrix is not of full rank, 'glmnet' may have very long run times when fitting the relaxed lasso model, in particular when fitting a Cox based model, making it difficult to get solutions either from glmnet() or cv.glmnet(). In this package, 'glmnetr', we provide a workaround and solve for the non penalized relaxed model where gamma=0 for model structures analogue to R functions like glm() or coxph() of the survival package. If you are not fitting relaxed lasso models, or if you are able to get convergence using 'glmnet', then this package may not be of much benefit to you. Note, while this package may allow one to fit relaxed lasso models that have difficulties converging using 'glmnet', this package does not afford the full function and versatility of 'glmnet'.
In addition to fitting the relaxed lasso model this package also includes the function cv.glmnetr() to perform a cross validation to identify hyper-parameters for a lasso fit, much like the cv.glmnet() function of the 'glmnet' package. Additionally, the package includes the function nested.glmnetr() to perform a nested cross validation to assess the fit of a cross validated derived lasso model fit. If though you are fitting not a relaxed lasso model but an elastic-net model, then the R-packages 'nestedcv' <https://cran.r-project.org/package=nestedcv>, 'glmnetSE' <https://cran.r-project.org/package=glmnetSE> or others may provide greater functionality when performing a nested CV.
As with the 'glmnet' package, this package passes most relevant output to the output object and tabular and graphical summaries can be generated using the summary and plot functions. Use of the 'glmnetr' has many similarities to the 'glmnet' package and it is recom-

mended that the user of 'glmnetr' first become familiar with the 'glmnet' package <https://cran.r-project.org/package=glmnet>, with the ``An Introduction to 'glmnet''' and ``The Relaxed Lasso'' being especially helpful in this regard.

License GPL-3

NeedsCompilation no

Copyright Mayo Foundation for Medical Education and Research

RoxygenNote 7.2.2

Encoding UTF-8

Repository CRAN

Date/Publication 2022-12-14 12:10:02 UTC

R topics documented:

`aicreg` *Identify model based upon AIC criteria from a stepreg() putput*

Description

Identify model based upon AIC criteria from a stepreg() putput

Usage

```
aicreg(
  xs,
  start,
  y_,
  event,
  steps_n = steps_n,
  family = family,
  object = NULL,
  time = 0
)
```

Arguments

`xs`	predictor input - an n by p matrix, where n (rows) is sample size, and p (columns) the number of predictors. Must be in matrix form for complete data, no NA's, no Inf's, etc., and not a data frame.
`start`	start time, Cox model only - class numeric of length same as number of patients (n)
`y_`	output vector: time, or stop time for Cox model, Y_ 0 or 1 for binomal (logistic), numeric for gaussian. Must be a vector of length same as number of sample size.
`event`	event indicator, 1 for event, 0 for census, Cox model only. Must be a numeric vector of length same as sample size.
`steps_n`	number of steps done in stepwise regression fitting
`family`	model family, "cox", "binomial" or "gaussian"
`object`	A stepreg() output. If NULL it will be derived.
`time`	Indicate whether or not to update progress in the console. Default of 0 suppresses these updates. The option of 1 provides these updates. In fitting clinical data with non full rank design matrix we have found some R-packages to take a vary long time or seemingly be caught in infinite loops. Therefore we allow the user to track the package and judge whether things are moving forward or if the process should be stopped.

Value

The identified model in form of a glm() or coxph() output object, and an entry of the stepreg() output object.

Examples

```
set.seed(18306296)
sim.data=glmnetr.simdata(nrows=100, ncols=100, beta=c(0,1,1))
# this gives a more intersting case but takes longer to run
xs=sim.data$xs
# this will work numerically
xs=sim.data$xs[,c(2,3,50:55)]
y_=sim.data$yt
event=sim.data$event
cox.aic.fit = aicreg(xs, NULL, y_, event, family="cox", steps_n=40)
summary(cox.aic.fit)

y_=sim.data$yt
norm.aic.fit = aicreg(xs, NULL, y_, NULL, family="gaussian", steps_n=40)
summary(norm.aic.fit)
```

`best.preds` *Title Get the best models for the steps of a stepwise fit*

Description

Title Get the best models for the steps of a stepwise fit

Usage

```
best.preds(modsum, risklist)
```

Arguments

`modsum`	Model summmary
`risklist`	Riskset list

Value

best predictors at each step of a stepwise regerssion

`cox.sat.dev` *Calculate the CoxPH saturated log-likelihood*

Description

This function calculates the saturated log-likelihood for the Cox model using both the Efron and Breslow approximations for the case where all ties at a common event time have the same weights (exp(X without ties the saturated log-likelihood is 0 as the contribution to the log-likelihood at that time point can be made arbitrarily close to 1 by assigning a large weight to the record corresponding to an event. Similarly, in the case of ties one can assign a much larger weight to be associated with one of the event times such that the associated record contributes a 1 to the likelihood. Next one can assigns a very large weight to a second tie, but smaller than the first tie considered, and this too will contribute a 1 to the likelihood. Continuing in this way for this and all time points with ties, the partial log-likelihood is 0, just like for the no-ties case. Note, this is the same argument with which we derive the log-likelihood of 0 for the no ties case. Still, to be consistent with others we derive the saturated log-likelihood with ties under the constraint that all ties carry the same weights.

Usage

```
cox.sat.dev(y_, e_)
```

Arguments

`y_`	Time variable for a survival analysis, whether or not there is a start time
`e_`	Event indicator with 1 for event 0 otherwise.

Value

Saturated log likelihood for the Efron and Breslow approximations.

`cv.glmnetr` *Get a cross validated tuned relaxed lasso model fit.*

Description

This function derives a relaxed lasso model and derives hyperparmaters using cross validaiton. It is analogous to the glmnet() function of the _glmnet_ package, but handles caes where glmnet() may run slowly when using the relaxed-=TRUE option.

Usage

```
cv.glmnetr(
  xs,
  start,
  y_,
  event,
  family = "cox",
  lambda = NULL,
  gamma = c(0, 0.25, 0.5, 0.75, 1),
  folds_n = 10,
  limit = 2,
  fine = 0,
  time = 0,
  seed = NULL,
  foldid = NULL
)
```

Arguments

`xs`	Predictor matrix
`start`	Vector of start times or the Cox model. May be NULL.
`y_`	outcome vector
`event`	event vector in case of the Cox model. May be NULL for other models.
`family`	Model family, one of "cox", "gaussian" or "binomial".
`lambda`	The lambda vector. May be NULL.
`gamma`	The gamma vector. Default is c(0,0.25,0.50,0.75,1).
`folds_n`	Number of folds for cross validation. Default and recommended is 10.
`limit`	limit the small values for lambda after the initial fit. This will calcualtions that have minimal impact on the cross validation. Default is 2 for moderate limitation, 1 for less limitation, 0 for none.
`fine`	Use a finer step in determining lambda. Of little value unless one repeats the cross valiaiton many times to more finely tune the hyper parameters. See the _glmnet_ documentation.
`time`	Indicate whether or not to update progress in the console. Default of 0 suppresses these updates. The option of 1 provides these updates. In fitting clinical data with non full rank design matrix we have found some R-packages to take a vary long time or seemingly be caught in infinite loops. Therefore we allow the user to track the package and judge whether things are moving forward or if the process should be stopped.
`seed`	A seed for set.seed to assure one can get the same results twice. If NULL the program will generate a random seed. Whether specified or NULL, the seed is stored in the output object for future reference.
`foldid`	A vector of integers to associate each record to a fold. Should be integers between 1 and folds_n.

Details

#' This is main program for model derivation. As currently implemented the package requires the data to be input as #' vectors and matrices with no missing values (NA). All data vectors and matrices must be numerical. For categorical variables one should first construct corresponding numerical variables to represent these categories. To take advantage of the lasso model, one can use one hot coding assigning an indicator for each level of each categorical varaible, or creating as well other contrasts variables suggested by the subject matter.

Value

A cross validated relaxed lasso model fit.

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

`glmnetr` , `nested.glmnetr` , `glmnetr.simdata`

Examples

```
# set seed for random numbers, optionally, to get reproducible results
set.seed(82545037)
sim.data=glmnetr.simdata(nrows=100, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$y_
event=sim.data$event
# for this example we use a small number for folds_n to shorten run time
cv.glmnetr.fit = cv.glmnetr(xs, NULL, y_, NULL, family="gaussian", folds_n=3, limit=2)
plot(cv.glmnetr.fit)
plot(cv.glmnetr.fit, coefs=1)
summary(cv.glmnetr.fit)
```

`cv.stepreg` *Title Cross validated tuned stepwise regression.*

Description

Title Cross validated tuned stepwise regression.

Usage

```
cv.stepreg(
  xs_cv,
  start_cv = NULL,
  y_cv,
  event_cv,
  steps_n = 0,
  folds_n = 10,
  method = "loglik",
  family = "cox",
  foldid = NULL,
  time = 0
)
```

Arguments

`xs_cv`	predictor input - an n by p matrix, where n (rows) is sample size, and p (columns) the number of predictors. Must be in matrix form for complete data, no NA's, no Inf's, etc., and not a data frame.
`start_cv`	start time, Cox model only - class numeric of length same as number of patients (n)
`y_cv`	output vector: time, or stop time for Cox model, Y_ 0 or 1 for binomal (logistic), numeric for gaussian. #' Must be a vector of length same as number of sample size.
`event_cv`	event indicator, 1 for event, 0 for census, Cox model only. Must be a numeric vector of length same as sample size.
`steps_n`	number of steps done in stepwise regression fitting
`folds_n`	number of folds for each level of cross validation
`method`	method for choosing model in stepwise procedure, "loglik" or "concordance". Other procedures use the "loglik".
`family`	model family, "cox", "binomial" or "gaussian"
`foldid`	A vector of integers to associate each record to a fold. Should be integers between 1 and folds_n.
`time`	1 to update fit progress in the console, 0 (default) to suppress. In fitting clinical data with non full rank design matrix we have found some R-packages to take a vary long time or seemingly be caught in infinite loops. Therefore we allow the user to track the package and judge whether things are moving forward or if the process should be stopped.

Value

cross validated stepwise regression tuned by number of model terms or p

Examples

```
set.seed(955702213)
sim.data=glmnetr.simdata(nrows=1000, ncols=100, beta=c(0,1,1))
# this gives a more interesting case but takes longer to run
xs=sim.data$xs
# this will work numerically as an example
xs=sim.data$xs[,c(2,3,50:55)]
dim(xs)
```

```
y_=sim.data$yt
event=sim.data$event
cv.stepreg.fit = cv.stepreg(xs, NULL, y_, event, steps_n=10, folds_n=3, time=0)
summary(cv.stepreg.fit)
```

`difftime1` *Get elapsed time in c(hour, minute, secs)*

Description

Get elapsed time in c(hour, minute, secs)

Usage

```
difftime1(time1, time2)
```

Arguments

`time1`	start time
`time2`	stop time

Value

Returns a vector of elapsed time in (hour, minute, secs)

`difftime2` *Output elapsed time and split split*

Description

Output elapsed time and split split

Usage

```
difftime2(time_start = NULL, time_last = NULL)
```

Arguments

`time_start`	beginning time for printing elapsed time
`time_last`	last time for printing time split

Value

Time of call for input, as well as time-start and time-last

Examples

```
time_start = difftime2()
time_last = difftime2(time_start)
time_last = difftime2(time_start,time_last)
time_last = difftime2(time_start,time_last)
```

---

getlamgam *get lam and gam*

Description

get lam and gam

Usage

```
getlamgam(object, lam, gam, comment)
```

Arguments

`object`	- glmnetr object as input
`lam`	value for lam, may be NULL
`gam`	value for gam, may be MULL
`comment`	Default of TRUE to write to console information on lam and gam selected for output. FALSE will suppress this write to console.

Value

numerical values for lam and gam for useage in plot and predict

---

glmnetr *Fit relaxed part of lasso model*

Description

This function derives the relaxed lasso fits and optionally calls glmnet() to derive the fully penalized lasso fit.

Usage

```
glmnetr(
  xs_tmp,
  start_tmp,
  y_tmp,
  event_tmp,
  family = "cox",
  lambda = NULL,
  gamma = c(0, 0.25, 0.5, 0.75, 1),
  object = NULL,
  time = 0
)
```

Arguments

`xs_tmp`	predictor (X) matrix
`start_tmp`	start time in case Cox model and (Start, Stop) time for use in model
`y_tmp`	outcome (Y) variable, in case of Cox model (stop) time
`event_tmp`	event variablee in case of Cox model
`family`	one of "cox", "gaussian" or "binomial"
`lambda`	lambda vector, as in _glmnet_, default is NULL
`gamma`	gamma vector, as with _glmnet_, default c(0,0.25,0.50,0.75,1)
`object`	an output object from _glmnet_ using relax=FALSE with the model fits for the fully penalized lass fits, i.e. gamma=1. Defualt is NULL in which case these are derived within the function.
`time`	Indicate whether or not to update progress in the console. Default of 0 suppresses these updates. The option of 1 provides these updates. In fitting clinical data with non full rank design matrix we have found some R-packages to take a vary long time or seemingly be caught in infinite loops. Therefore we allow the user to track the package and judge whether things are moving forward or if the process should be stopped.

Value

A list with two matrices, one for the model coefficients with gamma=1 and the other with gamma=0.

See Also

`cv.glmnetr` , `nested.glmnetr` , `glmnetr.simdata`

Examples

```
set.seed(82545037)
sim.data=glmnetr.simdata(nrows=200, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$yt
event=sim.data$event
```

```
glmnetr.fit = glmnetr( xs, NULL, y_, event, family="cox")
plot(glmnetr.fit)
```

glmnetr.compcv *Compare cross validation fits from a nested.glmnetr output.*

Description

Compare cross-validation model fits in terms of average concordance from the nested cross validaiton fits.

Usage

```
glmnetr.compcv(object)
```

Arguments

`object`	A nested.glmnetr output object.

Value

A printout to the R console.

See Also

`summary.nested.glmnetr`

Examples

```
sim.data=glmnetr.simdata(nrows=1000, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$yt
event=sim.data$event
# for this example we use a small number for folds_n to shorten run time
fit3 = nested.glmnetr(xs, NULL, y_, event, family="cox", folds_n=3)
glmnetr.compcv(fit3)
```

`glmnetr.compcv0` *A glmnetr specifc paired t-test*

Description

Perform a paired t-test as called from glmnetr.compcv.

Usage

```
glmnetr.compcv0(a, b)
```

Arguments

`a`	One term
`b`	A second term

Value

A t-test

`glmnetr.simdata` *Generate example data*

Description

This function generates an example data set with specified number of observations, and predictors. The first column in teh dsing matrix is identically equal to 1 for an intercept. Columns 2 to 5 are for the 4 levels of a character variable, 6 to 11 for the 6 levels of a character variable. Columns 12 to 17 are for 3 binomial predictors, again over paramtereized. Such over paramtereization can cause difficulties with the _glmnet()_ of the glmnet package.

Usage

```
glmnetr.simdata(nrows = 1000, ncols = 100, beta = NULL)
```

Arguments

`nrows`	Sample size (>=100) for simulated data, default=1000.
`ncols`	Number of columns (>=17) in design matrix, i.e. predictors, default=100.
`beta`	Vector of length <= ncols for "left most" coefficients. If beta has length < ncols, then the values at length(beta)+1 to ncols are set to 0. Default=NULL, where a beta of length 25 is assigned standard normal values.

Value

A list with elements xs for desing matrix, y_ for a quantitative outcome, yt for a survival time, event for an indicator of event (1) or censoring (0), in the Cox proportional hazards survival model setting, yb for yes/no (binomial) outcome data, and beta the beta used in random number generation.

See Also

glmnetr , cv.glmnetr , nested.glmnetr

Examples

```
sim.data=glmnetr.simdata(nrows=1000, ncols=100, beta=NULL)
# for Cox PH survial model data
xs=sim.data$xs
y_=sim.data$yt
event=sim.data$event
# for linear regression model data
xs=sim.data$xs
y_=sim.data$y_
# for logistic regression model data
xs=sim.data$xs
y_=sim.data$yb
```

---

glmnetrll_1fold *Evaluation fit of leave out fold*

Description

This function derives the log likelihood for a leave out based upon the fit of the input object.

Usage

```
glmnetrll_1fold(
  object,
  xs_new,
  start_new,
  y_new,
  event_new,
  family = "cox",
  lambda_n = NULL,
  gamma = c(0, 0.25, 0.5, 0.75, 1)
)
```

Arguments

`object`	an output object from _cv.glmnetr_
`xs_new`	A new predictor matrix
`start_new`	A new vector of start times or the Cox model. May be NULL.
`y_new`	a new outcome vector.
`event_new`	event vector in case of the Cox model. May be NULL for other models.
`family`	Model family, one of "cox", "gaussian" or "binomial".
`lambda_n`	length of the lambda vector.
`gamma`	The gamma vector.

Value

Returns the log likelihood of object fit using new data.

`glmnetr_devratio` *Get Deviance ratio.*

Description

fit models to derive the devaince ratios.

Usage

```
glmnetr_devratio(object, object2, xs_new, start_new, y_new, event_new, family)
```

Arguments

`object`	- A _glmnet_ output object with relax=FALSE, i.e model fit for gamma=1.
`object2`	- A _glmnetr_ output object with relaxed fits, i.e model fit for gamma=0.
`xs_new`	- predictor matrix
`start_new`	- start times in case of usage in Cox model. May be NULL.
`y_new`	- outcome vector.
`event_new`	- event indicator in case of Cox model.
`family`	- Model family, one of "cox", "gaussian" or "binomial".

Value

- Deviance ratios.

nested.glmnetr — *Using nested cross validation, describe the fit of a cross validated tuned relaxed lasso model fit.*

Description

Performs a nested cross validation for a cross validated informed relaxed lasso model.

Usage

```
nested.glmnetr(
  xs,
  start = NULL,
  y_,
  event,
  family = NULL,
  steps_n = 0,
  folds_n = 10,
  dolasso = 1,
  doaic = 0,
  dostep = 0,
  method = "loglik",
  lambda = NULL,
  gamma = NULL,
  relax = TRUE,
  limit = 1,
  fine = 0,
  time = 0,
  seed = NULL,
  foldid = NULL
)
```

Arguments

`xs`	- predictor input - an n by p matrix, where n (rows) is sample size, and p (columns) the number of predictors. Must be in matrix form for complete data, no NA's, no Inf's, etc., and not a data frame.
`start`	- start time, Cox model only - class numeric of length same as number of patients (n)
`y_`	- output vector: time, or stop time for Cox model, Y_ 0 or 1 for binomal (logistic), numeric for gaussian. Must be a vector of length same as number of sample size.
`event`	- event indicator, 1 for event, 0 for census, Cox model only. Must be a numeric vector of length same as sample size.
`family`	- model family, "cox", "binomial" or "gaussian"

`steps_n`	- number of steps done in stepwise regression fitting
`folds_n`	- number of folds for each level of cross validation
`dolasso`	- fit and do cross validition for lasso model, 0 or 1
`doaic`	- fit and do cross validation for AIC fit, 0 or 1. This is provided for reference only and is not recommended.
`dostep`	- fit and do cross validation for stepwise regression fit, 0 or 1, as discussed in James, Witten, Hastie and Tibshirani, 2nd edition.
`method`	- method for choosing model in stepwise procedure, "loglik" or "concordance". Other procedures use the "loglik".
`lambda`	- lambda vector for teh lasso fit
`gamma`	- gamma vector for the relaxed lasso fit, default is c(0,0.25,0.5,0.75,1).
`relax`	- Fit the relaxed lasso model when fitting a lasso model.
`limit`	- limit the small values for lambda after the initial fit. This will calcualtions that have minimal impact on the cross validation. Default is 2 for moderate limitation, 1 for less limitation, 0 for none.
`fine`	- Use a finer step in determining lambda. Of little value unless one repeats the cross valiaiton many times to more finely tune the hyper paramters. See the _glmnet_ documentation.
`time`	- print out the time splits.
`seed`	A seed for set.seed() to assure one can get the same results twice. If NULL the program will generate a random seed. Whether specified or NULL, the seed is stored in the output object for future reference.
`foldid`	A vector of integers to associate each record to a fold. Should be integers between 1 and folds_n. These will only be used in the outer folds.

Value

- The fit of a cross validated tuned relaxed lasso model fit, obtained by nested cross validation.

See Also

`glmnetr` , `cv.glmnetr` , `glmnetr.simdata` , `summary.nested.glmnetr` , , `plot.nested.glmnetr`

Examples

```
sim.data=glmnetr.simdata(nrows=1000, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$y_
# for this example we use a small number for folds_n to shorten run time
nested.glmnetr.fit = nested.glmnetr( xs, NULL, y_, NULL, family="gaussian", folds_n=3)
plot(nested.glmnetr.fit)
plot(nested.glmnetr.fit, coefs=TRUE)
summary(nested.glmnetr.fit)
summary(nested.glmnetr.fit, cvfit=TRUE)
```

`plot.cv.glmnetr` *Plot cross-validation deviances, or model coefficients.*

Description

By default, with coefs=FALSE, plots the average deviances as function of lam (lambda) and gam (gamma), and also indicates the gam and lam which minimize diviance based upon either a cv.glmnetr() or nested.glmneter() output object. Optionally, with coefs=TRUE, plots the relaxes lasso coefficients.

Usage

```
## S3 method for class 'cv.glmnetr'
plot(
  x,
  gam = NULL,
  lambda.lo = NULL,
  plup = 0,
  title = NULL,
  coefs = FALSE,
  comment = TRUE,
  ...
)
```

Arguments

`x`	a cv.glmnetr or nested.glmnetr output object.
`gam`	a specific level of gamma for plotting. By default gamma.min will be used.
`lambda.lo`	a lower limit of lambda when plotting.
`plup`	An indicator to plot the upper 95 percent two-sided confidence limits.
`title`	A title for the plot.
`coefs`	Default of FALSE plots deviances, option of TRUE plots coefficients.
`comment`	Default of TRUE to write to console information on lam and gam selected for output. FALSE will suppress this write to console.
`...`	Additional arguments passed to the plot function.

Value

This program returns a plot to the graphics window, and may provide some numerical information to the R Console. If gam is not specified, then then the gamma.min from the deviance minimizing (lambda.min, gamma.min) pair will be used, and the corresponding lambda.min will be indicated by a vertical line, and the lambda minimizing deviance under the restricted set of models where gamma=0 will indicated by a second vertical line.

See Also

plot.glmnetr , plot.nested.glmnetr , cv.glmnetr

Examples

```
# set seed for random numbers, optionally, to get reproducible results
set.seed(82545037)
sim.data=glmnetr.simdata(nrows=100, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$y_
event=sim.data$event
# for this example we use a small number for folds_n to shorten run time
cv.glmnetr.fit = cv.glmnetr(xs, NULL, y_, NULL, family="gaussian", folds_n=3, limit=2)
plot(cv.glmnetr.fit)
plot(cv.glmnetr.fit, coefs=1)
```

plot.glmnetr *Plot the relaxed lasso coefficients.*

Description

Plot the relaxed lasso coefficients from either a glmnetr(), cv.glmnetr() or nested.glmnetr() output object. One may specify gam, single value for gamma. If gam is unspecified (NULL), then cv.glmnetr and nested.glmnetr() will use the gam which minimizes loss, and glmentr() will use gam=1.

Usage

```
## S3 method for class 'glmnetr'
plot(x, gam = NULL, lambda.lo = NULL, title = NULL, comment = TRUE, ...)
```

Arguments

`x`	Either a glmnetr, cv.glmnetr or a nested.glmnetr output object.
`gam`	A specific level of gamma for plotting. By default gamma.min from the deviance minimizing (lambda.min, gamma.min) pair will be used.
`lambda.lo`	A lower limit of lambda for plotting.
`title`	A title for the plot
`comment`	Default of TRUE to write to console information on lam and gam selected for output. FALSE will suppress this write to console.
`...`	Additional arguments passed to the plot function.

Value

This program returns a plot to the graphics window, and may provide some numerical information to the R Console. If the input object is from a nested.glmnetr or cv.glmnetr object, and gamma is not specified, then the gamma.min from the deviance minimizing (lambda.min, gamma.min) pair will be used, and the minimizing lambda.min will be indicated by a vertical line. Also, if one specifies gam=0, the lambda which minimizes deviance for the restricted set of models where gamma=0 will indicated by a vertical line.

See Also

`plot.cv.glmnetr` , `plot.nested.glmnetr` , `glmnetr`

Examples

```
set.seed(82545037)
sim.data=glmnetr.simdata(nrows=200, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$yt
event=sim.data$event
glmnetr.fit = glmnetr( xs, NULL, y_, event, family="cox")
plot(glmnetr.fit)
```

---

plot.nested.glmnetr — *Plot the cross validated relaxed lasso deviances or coefficients from a nested.glmnetr call.*

Description

Plot the cross validated relaxed lasso deviances or coefficients from a nested.glmnetr call.

Usage

```
## S3 method for class 'nested.glmnetr'
plot(
  x,
  gam = NULL,
  lambda.lo = NULL,
  title = NULL,
  plup = 0,
  coefs = FALSE,
  comment = TRUE,
  ...
)
```

Arguments

`x`	A nested.glmnetr output object
`gam`	A specific level of gamma for plotting. By default gamma.min will be used.
`lambda.lo`	A lower limit of lambda when plotting.
`title`	A title
`plup`	Plot upper 95 percent two-sided confidence intervals for the deviance plots.
`coefs`	Default is FALSE to plot deviances. Option of TRUE to plot coefficients.
`comment`	Default of TRUE to write to console information on lam and gam selected for output. FALSE will suppress this write to console.
`...`	Additional arguments passed to the plot function.

Value

This program returns a plot to the graphics window, and may provide some numerical information to the R Console.

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

`plot.glmnetr` , `plot.cv.glmnetr` , `nested.glmnetr`

Examples

```
sim.data=glmnetr.simdata(nrows=1000, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$yt
event=sim.data$event
# for this example we use a small number for folds_n to shorten run time
fit3 = nested.glmnetr(xs, NULL, y_, event, family="cox", folds_n=3)
plot(fit3)
plot(fit3, coefs=TRUE)
```

`predict.cv.glmnetr` *Give predicteds based upon the _glmnetr_ output object contained in the cv.glmnetr output object.*

Description

Give predicteds based upon the _glmnetr_ output object contained in the cv.glmnetr output object.

Usage

```
## S3 method for class 'cv.glmnetr'
predict(object, xs_new = NULL, lam = NULL, gam = NULL, comment = TRUE, ...)
```

Arguments

`object`	A cv.glmnetr or nested.glmnetr output object.
`xs_new`	The predictor matrix. If NULL, then betas are provided.
`lam`	The lambda value for choice of beta. If NULL, then lambda.min is used from the cross validated tuned relaxed model. We use the term lam instead of lambda as lambda usually denotes a vector in the package.
`gam`	The gamma value for choice of beta. If NULL, then gamma.min is used from the cross validated tuned relaxed model. We use the term gam instead of gamma as gamma usually denotes a vector in the package.
`comment`	Default of TRUE to write to console information on lam and gam selected for output. FALSE will suppress this write to console.
`...`	Additional arguments passed to the predict function.

Value

Either predicteds (XS*beta estimates based upon the predictor matrix XS) or model coefficients, based upon a cv.glmnetr output object. When outputting coefficients (beta), creates a list with the first element, beta_, including 0 and non-0 terms and the second element, beta, including only non 0 terms.

See Also

`predict.glmnetr` , `cv.glmnetr` , `nested.glmnetr`

Examples

```
# set seed for random numbers, optionally, to get reproducible results
set.seed(82545037)
sim.data=glmnetr.simdata(nrows=200, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$y_
event=sim.data$event
# for this example we use a small number for folds_n to shorten run time
cv.glmnetr.fit = cv.glmnetr(xs, NULL, y_, NULL, family="gaussian", folds_n=3, limit=2)
predict(cv.glmnetr.fit)
```

`predict.glmnetr` *Get coefficients or predictions using a glmnetr output object*

Description

Get coefficients or predictions using a glmnetr output object

Usage

```
## S3 method for class 'glmnetr'
predict(object, xs_new = NULL, lam = NULL, gam = NULL, ...)
```

Arguments

`object`	A glmnetr output object
`xs_new`	A desing matrix for predictions
`lam`	The value for lambda for determining the lasso fit
`gam`	The value for gamma for determining the lasso fit
`...`	Additional arguments passed to the predict function.

Value

Coefficients or predictions using a glmnetr output object. When outputting coefficients (beta), creates a list with the first element, beta_, including 0 and non-0 terms and the second element, beta, including only non 0 terms.

See Also

`glmnetr`, `cv.glmnetr`, `nested.glmnetr`

Examples

```
set.seed(82545037)
sim.data=glmnetr.simdata(nrows=200, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$yt
event=sim.data$event
glmnetr.fit = glmnetr( xs, NULL, y_, event, family="cox")
betas = predict(glmnetr.fit,NULL,exp(-2),0.5 )
betas$beta
```

predict.nested.glmnetr

Give predicteds based upon the cv.glmnet output object contained in the nested.glmnetr output object.

Description

This is essentially a redirect to the summary.cv.glmnetr function for nested.glmnetr output objects, based uopn the cv.glmnetr output object contained in the nested.glmnetr output object.

Usage

```
## S3 method for class 'nested.glmnetr'
predict(object, xs_new = NULL, lam = NULL, gam = NULL, comment = TRUE, ...)
```

Arguments

`object`	A nested.glmnetr output object.
`xs_new`	The predictor matrix. If NULL, then betas are provided.
`lam`	The lambda value for choice of beta. If NULL, then lambda.min is used from the cross validated tuned relaxed model. We use the term lam instead of lambda as lambda usually denotes a vector in the package.
`gam`	The gamma value for choice of beta. If NULL, then gamma.min is used from the cross validated tuned relaxed model. We use the term gam instead of gamma as gamma usually denotes a vector in the package.
`comment`	Default of TRUE to write to console information on lam and gam selected for output. FALSE will suppress this write to console.
`...`	Additional arguments passed to the predict function.

Value

Either the XS*Beta estimates based upon the predictor matrix, or model coefficients.

See Also

`predict.cv.glmnetr`

Examples

```
sim.data=glmnetr.simdata(nrows=1000, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$yt
event=sim.data$event
# for this example we use a small number for folds_n to shorten run time
fit3 = nested.glmnetr(xs, NULL, y_, event, family="cox", folds_n=3)
betas = predict(fit3)
```

```
betas$beta
```

preds_1 *preds_1 function*

Description

preds_1 function

Usage

```
preds_1(modsumbest, k_, risklist, risklistl)
```

Arguments

modsumbest	matrix with best predictors based upon umber of model terms
k_	Value for number of predictors in model
risklist	Riskset list
risklistl	Number of terms (length) in the riskset

Value

input to best.preds()

stepreg *Title Fit the steps of a stepwise regression.*

Description

Title Fit the steps of a stepwise regression.

Usage

```
stepreg(
  xs_st,
  start_time_st = NULL,
  y_st,
  event_st,
  steps_n = 0,
  method = "loglik",
  family = NULL,
  time = 0
)
```

Arguments

`xs_st`	predictor input - an n by p matrix, where n (rows) is sample size, and p (columns) the number of predictors. Must be in matrix form for complete data, no NA's, no Inf's, etc., and not a data frame.
`start_time_st`	start time, Cox model only - class numeric of length same as number of patients (n)
`y_st`	output vector: time, or stop time for Cox model, y_st 0 or 1 for binomal (logistic), numeric for gaussian. Must be a vector of length same as number of sample size.
`event_st`	event_st indicator, 1 for event, 0 for census, Cox model only. Must be a numeric vector of length same as sample size.
`steps_n`	number of steps done in stepwise regression fitting
`method`	method for choosing model in stepwise procedure, "loglik" or "concordance". Other procedures use the "loglik".
`family`	model family, "cox", "binomial" or "gaussian"
`time`	1 to output stepwise fit program, 0 (default) to suppress

Value

does a stepwise regression of depth specified by steps_n

Examples

```
set.seed(18306296)
sim.data=glmnetr.simdata(nrows=100, ncols=100, beta=c(0,1,1))
# this gives a more intersting case but takes longer to run
xs=sim.data$xs
# this will work numerically
xs=sim.data$xs[,c(2,3,50:55)]
y_=sim.data$yt
event=sim.data$event
# for a Cox model
cox.step.fit = stepreg(xs, NULL, y_, event, family="cox", steps_n=40)
# ... and for a linear model
y_=sim.data$yt
norm.step.fit = stepreg(xs, NULL, y_, NULL, family="gaussian", steps_n=40)
```

`summary.cv.glmnetr` *Output summary of a _cv.glmnetr_ output object.*

Description

Summarize the cross-validated model fit to the R console. The fully penalized (gamma=1) beta estimate will not be given by default but can too be output using printg1=TRUE.

Usage

```
## S3 method for class 'cv.glmnetr'
summary(object, printg1 = "FALSE", orderall = FALSE, ...)
```

Arguments

`object`	A _cv.glmnetr_ output object.
`printg1`	TRUE to also print out the fully penalized lasso beta, else to suppress.
`orderall`	By default (orderall=FALSE) the order terms enter into the lasso model is given for the number of terms that enter in lasso minimizing loss model. If orderall=TRUE then all terms that are included in any lasso fit are described.
`...`	Additional arguments passed to the summary function.

Value

Coefficient estimates (beta)

See Also

`cv.glmnetr` , `nested.glmnetr`

Examples

```
# set seed for random numbers, optionally, to get reproducible results
set.seed(82545037)
sim.data=glmnetr.simdata(nrows=100, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$y_
event=sim.data$event
# for this example we use a small number for folds_n to shorten run time
cv.glmnetr.fit = cv.glmnetr(xs, NULL, y_, NULL, family="gaussian", folds_n=3, limit=2)
summary(cv.glmnetr.fit)
```

`summary.cv.stepreg` *Summarize some results from a cv.stepwise() output object.*

Description

Summarize some results from a cv.stepwise() output object.

Usage

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

Arguments

`object`	A cv.stepwise() output object
`...`	Additional arguments passed to the summary function.

Value

Summary of a stepwise regression

Examples

```
set.seed(955702213)
sim.data=glmnetr.simdata(nrows=1000, ncols=100, beta=c(0,1,1))
# this gives a more interesting case but takes longer to run
xs=sim.data$xs
# this will work numerically as an example
xs=sim.data$xs[,c(2,3,50:55)]
dim(xs)
y_=sim.data$yt
event=sim.data$event
cv.stepreg.fit = cv.stepreg(xs, NULL, y_, event, steps_n=10, folds_n=3, time=0)
summary(cv.stepreg.fit)

#'
```

---

summary.nested.glmnetr

Summary of the fit of a cross validated tuned relaxed lasso model fit, inferred by nested cross validation. .

Description

Summarize the model fit from a nested.glmnetr output object, i.e. a nested, cross-validated relaxed lasso model. Else summarize the cross-validated model fit.

Usage

```
## S3 method for class 'nested.glmnetr'
summary(object, cvfit = FALSE, printg1 = FALSE, ...)
```

Arguments

`object`	A nested.glmnetr output object.
`cvfit`	Default is FALSE to describe the evaluation of the cross validated relaxed lasso model. Option of TRUE will describe the cross validation tuned relaxed lasso model itself.
`printg1`	TRUE to also print out the fully penalized lasso beta, else to suppress. Only applies to cvfit=TRUE.
`...`	Additional arguments passed to the summary function.

Value

- A fit summary, or a model summary.

See Also

`glmnetr.compcv` , `summary.cv.stepreg` , `nested.glmnetr`

Examples

```
sim.data=glmnetr.simdata(nrows=1000, ncols=100, beta=NULL)
xs=sim.data$xs
y_=sim.data$yt
event=sim.data$event
# for this example we use a small number for folds_n to shorten run time
fit3 = nested.glmnetr(xs, NULL, y_, event, family="cox", folds_n=3)
summary(fit3)
```

`summary.stepreg` *Title Give a brief summary of the steps in a stepwise regression fit*

Description

Title Give a brief summary of the steps in a stepwise regression fit

Usage

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

Arguments

`object`	A stepreg() output object
`...`	Additional arguments passed to the summary function.

Value

Summarize a stepreg object

Index