Package 'glmnetr'

January 17, 2025

Title Nested Cross Validation for the Relaxed Lasso and Other Machine Learning Models

Version 0.5-5

Date 2024-12-28

Depends R (>= 3.4.0)

Suggests R.rsp

VignetteBuilder R.rsp

Imports glmnet, survival, Matrix, xgboost, smoof, mlrMBO, ParamHelpers, randomForestSRC, rpart, torch, aorsf, DiceKriging, rgenoud

ByteCompile Yes

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

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

Description Cross validation informed Relaxed LASSO, Artificial Neural Network (ANN), gradient boosting machine ('xgboost'), Random Forest ('RandomForestSRC'), Oblique Random Forest ('aorsf'), Recursive Partitioning ('RPART') or step wise regression models are fit. Cross validation leave out samples (leading to nested cross validation) or bootstrap out-of-bag samples are used to evaluate and compare performances between these models with results presented in tabular or graphical means. Calibration plots can also be generated, again based upon (outer nested) cross validation or bootstrap leave out (out of bag) samples. 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, from our experience when fitting Cox models on data with many predictors and many patients, making it difficult to get solutions from either glmnet() or cv.glmnet(). This may be remedied by using the 'path=TRUE' option when calling glmnet() and cv.glmnet(). Within the glmnetr package the approach of path=TRUE is taken by default.
When 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.

Use of the 'glmnetr' has many similarities to the 'glmnet' package and it is recommended that the user of 'glmnetr' also 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 useful in this regard.

License GPL-3

NeedsCompilation no

RoxygenNote 7.3.1

Encoding UTF-8

Repository CRAN

Date/Publication 2025-01-17 05:20:02 UTC

Contents

`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,
  track = 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 binomial (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	maximum 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.
track	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 very long time or possibly get 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, with an entry of the stepreg() output object.

See Also

stepreg , cv.stepreg , nested.glmnetr

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)
```

ann_tab_cv *Fit an Artificial Neural Network model on "tabular" provided as a matrix, optionally allowing for an offset term*

Description

Fit an Artificial Neural Network model for analysis of "tabular" data. The model has two hidden layers where the number of terms in each layer is configurable by the user. The activation function can also be switched between relu() (default) gelu() or sigmoid(). Optionally an offset term may be included. Model "family" may be "cox" to fit a generalization of the Cox proportional hazards model, "binomial" to fit a generalization of the logistic regression model and "gaussian" to fit a generalization of linear regression model for a quantitative response. See the corresponding vignette for examples.

Usage

```
ann_tab_cv(
  myxs,
  mystart = NULL,
  myy,
  myevent = NULL,
  myoffset = NULL,
  family = "binomial",
  fold_n = 5,
  epochs = 200,
  eppr = 40,
  lenz1 = 16,
  lenz2 = 8,
  actv = 1,
  drpot = 0,
  mylr = 0.005,
  wd = 0,
  l1 = 0,
  lasso = 0,
  lscale = 5,
  scale = 1,
  resetlw = 1,
  minloss = 1,
  gotoend = 0,
  seed = NULL,
  foldid = NULL
)
```

Arguments

myxs	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.

mystart	an optional vector of start times in case of a Cox model. Class numeric of length same as number of patients (n)
myy	dependent variable as a vector: time, or stop time for Cox model, Y_ 0 or 1 for binomial (logistic), numeric for gaussian. Must be a vector of length same as number of sample size.
myevent	event indicator, 1 for event, 0 for census, Cox model only. Must be a numeric vector of length same as sample size.
myoffset	an offset term to be used when fitting the ANN. Not yet implemented in its pure form. Functionally an offset can be included in the first column of the predictor or feature matrix myxs and indicated as such using the lasso option.
family	model family, "cox", "binomial" or "gaussian" (default)
fold_n	number of folds for each level of cross validation
epochs	number of epochs to run when tuning on number of epochs for fitting final model number of epochs informed by cross validation
eppr	for EPoch PRint. print summary info every eppr epochs. 0 will print first and last epochs, 0 for first and last epoch, -1 for minimal and -2 for none.
lenz1	length of the first hidden layer in the neural network, default 16
lenz2	length of the second hidden layer in the neural network, default 16
actv	for ACTiVation function. Activation function between layers, 1 for relu, 2 for gelu, 3 for sigmoid.
drpot	fraction of weights to randomly zero out. NOT YET implemented.
mylr	learning rate for the optimization step in the neural network model fit
wd	a possible weight decay for the model fit, default 0 for not considered
l1	a possible L1 penalty weight for the model fit, default 0 for not considered
lasso	1 to indicate the first column of the input matrix is an offset term, often derived from a lasso model, else 0 (default)
lscale	Scale used to allow ReLU to exend +/- lscale before capping the inputted linear estimated
scale	Scale used to transform the inital random paramter assingments by dividing by scale
resetlw	1 as default to re-adjust weights to account for the offset every epoch. This is only used in case lasso is set to 1.
minloss	default of 1 for minimizing loss, else maximizing agreement (concordance for Cox and Binomial, R-square for Gaussian), as function of epochs by cross validaition
gotoend	fit to the end of epochs. Good for plotting and exploration
seed	an optional a numerical/integer vector of length 2, for R and torch random generators, default NULL to generate these. Integers should be positive and not more than 2147483647.
foldid	a vector of integers to associate each record to a fold. Should be integers from 1 and fold_n.

Value

an artificial neural network model fit

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

ann_tab_cv_best , predict_ann_tab, nested.glmnetr

---

ann_tab_cv_best *Fit multiple Artificial Neural Network models on "tabular" provided as a matrix, and keep the best one.*

Description

Fit an multiple Artificial Neural Network models for analysis of "tabular" data using ann_tab_cv() and select the best fitting model according to cross validaiton.

Usage

```
ann_tab_cv_best(
  myxs,
  mystart = NULL,
  myy,
  myevent = NULL,
  myoffset = NULL,
  family = "binomial",
  fold_n = 5,
  epochs = 200,
  eppr = 40,
  lenz1 = 32,
  lenz2 = 8,
  actv = 1,
  drpot = 0,
  mylr = 0.005,
  wd = 0,
  l1 = 0,
  lasso = 0,
  lscale = 5,
  scale = 1,
  resetlw = 1,
  minloss = 1,
  gotoend = 0,
  bestof = 10,
  seed = NULL,
```

```
  foldid = NULL
)
```

Arguments

myxs	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.
mystart	an optional vector of start times in case of a Cox model. Class numeric of length same as number of patients (n)
myy	dependent variable as a vector: time, or stop time for Cox model, Y_ 0 or 1 for binomial (logistic), numeric for gaussian. Must be a vector of length same as number of sample size.
myevent	event indicator, 1 for event, 0 for census, Cox model only. Must be a numeric vector of length same as sample size.
myoffset	an offset term to be ues when fitting the ANN. Not yet implemented.
family	model family, "cox", "binomial" or "gaussian" (default)
fold_n	number of folds for each level of cross validation
epochs	number of epochs to run when tuning on number of epochs for fitting final model number of epochs informed by cross validation
eppr	for EPoch PRint. print summry info every eppr epochs. 0 will print first and last epochs, -1 nothing.
lenz1	length of the first hidden layer in the neural network, default 16
lenz2	length of the second hidden layer in the neural network, default 16
actv	for ACTiVation function. Activation function between layers, 1 for relu, 2 for gelu, 3 for sigmoid.
drpot	fraction of weights to randomly zero out. NOT YET implemented.
mylr	learning rate for the optimization step in teh neural network model fit
wd	weight decay for the model fit.
l1	a possible L1 penalty weight for the model fit, default 0 for not considered
lasso	1 to indicate the first column of the input matrix is an offset term, often derived from a lasso model
lscale	Scale used to allow ReLU to extend +/- lscale before capping the inputted linear estimated
scale	Scale used to transform the initial random parameter assingments by dividing by scale
resetlw	1 as default to re-adjust weights to account for the offset every epoch. This is only used in case lasso is set to 1
minloss	default of 1 for minimizing loss, else maximizing agreement (concordance for Cox and Binomial, R-square for Gaussian), as function of epochs by cross validation
gotoend	fit to the end of epochs. Good for plotting and exploration

bestof	how many models to run, from which the best fitting model will be selected.
seed	an optional a numerical/integer vector of length 2, for R and torch random generators, default NULL to generate these. Integers should be positive and not more than 2147483647.
foldid	a vector of integers to associate each record to a fold. Should be integers from 1 and fold_n.

Value

an artificial neural network model fit

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

ann_tab_cv , predict_ann_tab, nested.glmnetr

---

best.preds *Get the best models for the steps of a stepreg() fit*

Description

Get the best models for the steps of a stepreg() fit

Usage

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

Arguments

modsum	model summmary
risklist	riskset list

Value

best predictors at each step of a stepwise regression

See Also

stepreg , cv.stepreg , nested.glmnetr

boot.factor.foldid — *Generate foldid's by 0/1 factor for bootstrap like samples where unique option between 0 and 1*

Description

Generate foldid's by 0/1 factor for bootstrap like samples where unique option between 0 and 1

Usage

```
boot.factor.foldid(event, fraction)
```

Arguments

event	the outcome variable in a vector identifying the different potential levels of the outcome
fraction	the fraction of the whole sample included in the bootstratp sample

Value

foldid's in a vector the same length as event

See Also

get.foldid , nested.glmnetr

calceloss — *calculate cross-entry for multinomial outcomes*

Description

calculate cross-entry for multinomial outcomes

Usage

```
calceloss(xx, yy)
```

Arguments

xx	the sigmoid of the link, i.e, the estimated probabilities, i.e. xx = 1/(1+exp(-xb))
yy	the observed data as 0's and 1's

Value

the cross-entropy on a per observation basis

calplot *Construct calibration plots for a nested.glmnetr output object*

Description

Using k-fold cross validation this function constructs calibration plots for a nested.glmnetr output object. Each hold out subset of the k-fold cross validation is regressed on the x*beta predicteds based upon the model fit using the non-hold out data using splines. This yields k spline functions for evaluating model performance. These k spline functions are averaged to provide an overall model calibration. Standard deviations of the k spline fits are also calculated as a function of the predicted X*beta, and these are used to derive and plot approximate 95 (mean +/- 2 * SD/sqrt(k)). Note, standard errors calculated in this manner may underestimate (or overestimate?) the true standard error, the displayed confidence intervals might be too narrow for a 95 probability and should be interpreted with caution. See the package vignettes for discussion and references. Further, because regression equations can be unreliable when extrapolating beyond the data range used in model derivation, we display this overall calibration fit and CIs with solid lines only for the region which lies within the ranges of the predicted x*betas for all the k leave out sets. The spline fits are made using the same framework as in the original machine learning model fits, i.e. one of "cox", "binomial" or "gaussian"family. For the "cox" framework the pspline() funciton is used, and for the "binomial" and "gaussian" frameworks the ns() function is used. Predicted X*betas beyond the range of any of the hold out sets are displayed by dashed lines to reflect the lessor certainty when extrapolating even for a single hold out set.

Usage

```
calplot(
  object,
  wbeta = NULL,
  df = 3,
  resample = NULL,
  oob = 1,
  bootci = 0,
  plot = 1,
  plotfold = 0,
  plot_full = 0,
  plothr = 0,
  knottype = 1,
  trim = 0,
  vref = 0,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  col.term = 1,
  col.se = 2,
  rug = 1,
  seed = NULL,
```

```
  cv = NULL,
  fold = NULL,
  ...
)
```

Arguments

object	A nested.glmnetr() output object for calibration
wbeta	Which Beta should be plotted, an integer. This will depend on which machine learning models were run when creating the output object. If unsure the user can run the function without specifying wbeta and a legend will be directed to the console.
df	The degrees of freedom for the spline function
resample	1 to base the splines on the leave out X*Beta's ($xbetas.cv or $xbetas.boot.oob), or 0 to use the naive X*Beta's ($xbetas). This can be done to see biases associated with the naive approach.
oob	1 (default) to construct calibration plots using the out-of-bag data points, 0 to use in bag (including resampled data points) data points. This option only applies when bootstrap is used instead of k-fold cross validation, and when resample is set to 1. For cross validation evaluations out-of-bag samples (folds) are always used for evaluation. The purpose of oob = 0 is to allow evaluation of the variability of bootstrap calibrations ignoring bias like done in Riley et al., 2023, doi: 10.1186/s12916-023-03212-y and Austin and Steyerberg 2013, doi: 10.1002/sim.5941
bootci	1 to calculate bootstrap confidence intervals for calibration curves adjusting for bias, 0 (default) to simply plot the calibration curves based upon the inbag data. This is for exploration only, and only when bootstrap samples were used for model performance evaluation. The applicability of bootstrap confidence intervals for these calibration curves is questionable. If bootci is set to 1 then oob is set to 0.
plot	1 by default to produce plots, 0 to output data for plots only, 2 to plot and output data.
plotfold	0 by default to not plot the individual fold calibrations, 1 to overlay the k leave out spline calibration fits in a single figure and 2 to produce separate plots for each of the k hold out calibration curves.
plot_full	plot full data
plothr	a power > 1 determining the spacing of the values on the axes, e.g. 2, exp(1), sqrt(10) or 10. The default of 0 plots the X*Beta. This only applies fore "cox" survival data models.
knottype	1 (default) to use XBeta used for the spline fit to choose knots in ns() for gaussian and binomial families, 2 to use the XBeta from all re-samples to determine the knots.
trim	the percent of top and bottom of the data to be trimmed away when producing plots. The original data are still used used calcualting the curves for plotting.
vref	Similar to trim but instead of trimming the spline lines, plots vertical refence lines aht the top vref and bottom vref percent of the model X*Betas's

xlim	xlim for the plots. This does not effect the curves within the plotted region. Caution, for the "cox" framework the xlim are specified in terms of the X*beta and not the HR, even when HR is described on the axes.
ylim	ylim for the plots, which will usually only be specified in a second run of for the same data. This does not effect the curves within the plotted region. Caution, for the "cox" framework the ylim are specified in terms of the X*beta and not the HR, even when HR is described on the axes.
xlab	a user specified label for the x axis
ylab	a user specified label for the y axis
col.term	a number for the line depicting the overall calibration estimates
col.se	a number for the line depicting the +/- 2 * standard error lines for the overall calibration estimates
rug	1 to plot a rug for the model x*betas, 0 (default) to not.
seed	an integer seed used to random select the multiple of X*Betas to be used in the rug when using bootstraping for model evaluation as sample elements may be included multiple times as test (Out Of Bag) data.
cv	Deprecated. Use resample option instead.
fold	Deprecated. This term is now ignored.
...	allowance to pass terms to the invoked plot function

Details

Optionally, for comparison, the program can fit a spline based upon the predicted x*betas ignoring the cross validation structure, or one can fit a spline using the x*betas calculated using the model based upon all data.

Value

Calibration plots are returned by default, and optionally data for plots are output to a list.

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

plot.nested.glmnetr , summary.nested.glmnetr , nested.glmnetr

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

Description

Calculate 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*B)). For the simple case without ties the saturated log-likelihood is 0 as the contribution to the log-likelihood at each event time point can be made arbitrarily close to 1 by assigning a much larger weight to the record with 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 assign 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 at each event time 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.

See Also

nested.glmnetr

cv.glmnetr *Get a cross validation informed relaxed lasso model fit.*

Description

Derive a relaxed lasso model and identifies hyperparameters, i.e. lambda and gamma, which give the best bit using cross validation. It is analogous to the cv.glmnet() function of the 'glmnet' package, but handles cases where glmnet() may run slowly when using the relaxed=TRUE option.

Usage

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

Arguments

xs	predictor matrix
start	vector of start times or the Cox model. Should be NULL for other models.
y_	outcome vector
event	event vector in case of the Cox model. May be NULL for other models.
family	model family, "cox", "binomial" or "gaussian" (default)
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 generally recommended is 10.
limit	limit the small values for lambda after the initial fit. This will eliminate calculations that have small or 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 validation many times to more finely tune the hyperparameters. See the 'glmnet' package documentation.
track	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 program progress and judge whether things are moving forward or if the process should be stopped.
seed	a seed for set.seed() so one can reproduce the model fit. If NULL the program will generate a random seed. Whether specified or NULL, the seed is stored

	in the output object for future reference. Note, for the default this randomly generated seed depends on the seed in memory at that time so will depend on any calls of set.seed prior to the call of this function.
foldid	a vector of integers to associate each record to a fold. The integers should be between 1 and folds_n.
ties	method for handling ties in Cox model for relaxed model component. Default is "efron", optionally "breslow". For penalized fits "breslow" is always used as in the 'glmnet' package.
stratified	folds are to be constructed stratified on an indicator outcome 1 (default) for yes, 0 for no. Pertains to event variable for "cox" and y_ for "binomial" family.
time	track progress by printing to console elapsed and split times. Suggested to use track option instead as time options will be eliminated.
...	Additional arguments that can be passed to glmnet()

Details

This is the 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 factors (categorical variables) one should first construct corresponding numerical variables to represent the factor levels. To take advantage of the lasso model, one can use one hot coding assigning an indicator for each level of each categorical variable, or creating as well other contrasts variables suggested by the subject matter.

Value

A cross validation informed relaxed lasso model fit.

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

summary.cv.glmnetr , predict.cv.glmnetr , 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)
plot(cv.glmnetr.fit)
plot(cv.glmnetr.fit, coefs=1)
summary(cv.glmnetr.fit)
```

`cv.stepreg` *Cross validation informed stepwise regression model fit.*

Description

Cross validation informed stepwise regression model fit.

Usage

```
cv.stepreg(
  xs_cv,
  start_cv = NULL,
  y_cv,
  event_cv,
  family = "cox",
  steps_n = 0,
  folds_n = 10,
  method = "loglik",
  seed = NULL,
  foldid = NULL,
  stratified = 1,
  track = 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.
`family`	model family, "cox", "binomial" or "gaussian"
`steps_n`	Maximun number of steps done in stepwise regression fitting. If 0, then takes the value rank(xs_cv).
`folds_n`	number of folds for cross validation
`method`	method for choosing model in stepwise procedure, "loglik" or "concordance". Other procedures use the "loglik".
`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. The integers should be between 1 and folds_n.
stratified	folds are to be constructed stratified on an indicator outcome 1 (default) for yes, 0 for no. Pertains to event variable for "cox" and y_ for "binomial" family.
track	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 very long time. Therefore we allow the user to track the program progress and judge whether things are moving forward or if the process should be stopped.

Value

cross validation infomred stepwise regression model fit tuned by number of model terms or p-value for inclusion.

See Also

predict.cv.stepreg , summary.cv.stepreg, stepreg , aicreg , nested.glmnetr

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
# for this example we use small numbers for steps_n and folds_n to shorten run time
cv.stepreg.fit = cv.stepreg(xs, NULL, y_, event, steps_n=10, folds_n=3, track=0)
summary(cv.stepreg.fit)
```

---

devrat_ *Calculate deviance ratios for CV based*

Description

Calculate deviance ratios for individual folds and collectively. Calculations are based upon the average -2 Log Likelihoods calculated on each leave out test fold data for the models trained on the other (K-1) folds.

Usage

```
devrat_(m2.ll.mod, m2.ll.null, m2.ll.sat, n__)
```

Arguments

m2.ll.mod	-2 Log Likelihoods calculated on the test data
m2.ll.null	-2 Log Likelihoods for the null models
m2.ll.sat	-2 Log Likelihoods for teh saturated models
n__	sample zize for the indivual foles, or number of events for the Cox model

Value

a list with devrat.cv for the deviance ratios for the indivual folds, and devrat, a single collective deviance ratio

See Also

nested.glmnetr

---

diff_time *Output to console the elapsed and split times*

Description

Output to console the elapsed and split times

Usage

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

Arguments

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

Value

Time of program invocation

See Also

diff_time , nested.glmnetr

Examples

```
time_start = diff_time()
time_last = diff_time(time_start)
time_last = diff_time(time_start,time_last)
time_last = diff_time(time_start,time_last)
```

diff_time1 *Get elapsed time in c(hour, minute, secs)*

Description

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

Usage

```
diff_time1(time1, time2)
```

Arguments

time1	start time
time2	stop time

Value

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

See Also

diff_time

factor.foldid *Generate foldid's by factor levels*

Description

Generate foldid's by factor levels

Usage

```
factor.foldid(event, fold_n = 10)
```

Arguments

event	the outcome variable in a vector identifying the different potential levels of the outcome
fold_n	the numbe of folds to be constructed

Value

foldid's in a vector the same length as event

See Also

get.foldid , nested.glmnetr

`get.foldid` *Get foldid's with branching for cox, binomial and gaussian models*

Description

Get foldid's with branching for cox, binomial and gaussian models

Usage

```
get.foldid(y_, event, family, folds_n, stratified = 1)
```

Arguments

`y_`	see help for cv.glmnetr() or nested.glmnetr()
`event`	see help for cv.glmnetr() or nested.glmnetr()
`family`	see help for cv.glmnetr() or nested.glmnetr()
`folds_n`	see help for cv.glmnetr() or nested.glmnetr()
`stratified`	see help for cv.glmnetr() or nested.glmnetr()

Value

A numeric vector with foldid's for use in a cross validation

See Also

`factor.foldid` , `nested.glmnetr`

`get.id.foldid` *Get foldid's when id variable is used to identify groups of dependent sampling units. With branching for cox, binomial and gaussian models*

Description

Get foldid's when id variable is used to identify groups of dependent sampling units. With branching for cox, binomial and gaussian models

Usage

```
get.id.foldid(y_, event, id, family, folds_n, stratified)
```

Arguments

y_	see help for cv.glmnetr() or nested.glmnetr()
event	see help for cv.glmnetr() or nested.glmnetr()
id	see help for nested.glmnetr()
family	see help for cv.glmnetr() or nested.glmnetr()
folds_n	see help for cv.glmnetr() or nested.glmnetr()
stratified	see help for cv.glmnetr() or nested.glmnetr()

Value

A numeric vector with foldid's for use in a cross validation

See Also

factor.foldid , nested.glmnetr

---

glmnetr *Fit relaxed part of lasso model*

Description

Derive 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,
  track = 0,
  ties = "efron",
  time = NULL,
  ...
)
```

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 variable in case of Cox model
family	model family, "cox", "binomial" or "gaussian" (default)
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 lasso models, i.e. gamma=1. Default is NULL in which case these are derived within the function.
track	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 possibly get 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.
ties	method for handling ties in Cox model for relaxed model component. Default is "efron", optionally "breslow". For penalized fits "breslow" is always used as in the 'glmnet' package.
time	track progress by printing to console elapsed and split times. Suggested to use track option instead as time options will be eliminated.
...	Additional arguments that can be passed to glmnet()

Value

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

See Also

predict.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")
plot(glmnetr.fit)
```

glmnetr.cis *A redirect to nested.cis()*

Description

See nested.cis(), glmnetr.cis() is depricated

Usage

```
glmnetr.cis(object, type = "devrat", pow = 1, digits = 4, returnd = 0)
```

Arguments

object	A nested.glmnetr output object.
type	determines what type of nested cross validation performance measures are compared. Possible values are "devrat" to compare the deviance ratios, i.e. the fractional reduction in deviance relative to the null model deviance, "agree" to compare agreement, "lincal" to compare the linear calibration slope coefficients, "intcal" to compare the linear calibration intercept coefficients, from the nested cross validation.
pow	the power to which the average of correlations is to be raised. Only applies to the "gaussian" model. Default is 2 to yield R-square but can be on to show correlations. pow is ignored for the family of "cox" and "binomial". When pow = 2, calculations are made using correlations and the final estimates and confidence intervals are raised to the power of 2. A negative sign before an R-square estimate or confidence limit indicates the estimate or confidence limit was negative before being raised to the power of 2.
digits	digits for printing of z-scores, p-values, etc. with default of 4
returnd	1 to return the deviance ratios in a list, 0 to not return. The deviances are stored in the nested.glmnetr() output object but not the deviance ratios. This function provides a simple mechanism to obtain the cross validated deviance ratios.

Value

A printout to the R console

glmnetr.compcv *A redirect to nested.compare*

Description

See nested.compare(), as glmnetr() is depricated

See nested.compare(), as glmnetr() is depricated

See nested.compare(), as glmnetr.compcv() is depricated

See nested.compare(), as glmnetr.compcv() is depricated

Usage

```
glmnetr.compcv(object, digits = 4, type = "devrat", pow = 1)

glmnetr.compcv(object, digits = 4, type = "devrat", pow = 1)

glmnetr.compcv(object, digits = 4, type = "devrat", pow = 1)

glmnetr.compcv(object, digits = 4, type = "devrat", pow = 1)
```

Arguments

object	A nested.glmnetr output object.
digits	digits for printing of z-scores, p-values, etc. with default of 4
type	determines what type of nested cross validation performance measures are compared. Possible values are "devrat" to compare the deviance ratios, i.e. the fractional reduction in deviance relative to the null model deviance, "agree" to compare agreement, "lincal" to compare the linear calibration slope coefficients, "intcal" to compare the linear calibration intercept coefficients, from the nested cross validation.
pow	the power to which the average of correlations is to be raised.

Value

A printout to the R console.

A printout to the R console.

A printout to the R console.

A printout to the R console.

See Also

nested.compare

nested.compare

---

glmnetr.simdata *Generate example data*

Description

Generate an example data set with specified number of observations, and predictors. The first column in the design 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 another character variable. Columns 12 to 17 are for 3 binomial predictors, again over parameterized. Such over parameterization can cause difficulties with the glmnet() of the 'glmnet' package.

Usage

```
glmnetr.simdata(
  nrows = 1000,
  ncols = 100,
  beta = NULL,
  intr = NULL,
  nid = 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.
intr	either NULL for no interactions or a vector of length 3 to impose a product effect as decribed by intr[1]*xs[,3]*xs[,8] + intr[2]*xs[,4]*xs[,16] + intr[3]*xs[,18]*xs[,19] + intr[4]*xs[,21]*xs[,22]
nid	number of id levels where each level is associated with a random effect, of variance 1 for normal data.

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

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
```

glmnetr_seed *Get seeds to store, facilitating replicable results*

Description

Get seeds to store, facilitating replicable results

Usage

```
glmnetr_seed(seed, folds_n = 10, folds_ann_n = NULL)
```

Arguments

seed	The input seed as a start, NULL, a vector of length 1 or 2, or a list with vectors of length 1 or the number of folds, $seedr for most models and $seedt for the ANN fits
folds_n	The number of folds in general
folds_ann_n	The number of folds for the ANN fits

Value

seed(s) in a list format for input to subsequent runs

See Also

nested.glmnetr

nested.cis *Calculate performance measure "nominal" CI's and p's*

Description

Calculate overall estimates and "nominal" confidence intervals for performance measures based upon stored cross validation performance measures in a nested.glmnetr() output object. The simple standard errors derived here from cross-validation are questionable and the actual coverage probabilities of these CIs and the p's, may be differ meaningfully. See the Vignette references.

Usage

```
nested.cis(object, type = "devrat", pow = 1, digits = 4, returnd = 0)
```

Arguments

object	A nested.glmnetr output object.
type	determines what type of nested cross validation performance measures are compared. Possible values are "devrat" to compare the deviance ratios, i.e. the fractional reduction in deviance relative to the null model deviance, "agree" to compare agreement, "lincal" to compare the linear calibration slope coefficients, "intcal" to compare the linear calibration intercept coefficients, from the nested cross validation.
pow	the power to which the average of correlations is to be raised. Only applies to the "gaussian" model. Default is 2 to yield R-square but can be on to show correlations. pow is ignored for the family of "cox" and "binomial". When pow = 2, calculations are made using correlations and the final estimates and confidence intervals are raised to the power of 2. A negative sign before an R-square estimate or confidence limit indicates the estimate or confidence limit was negative before being raised to the power of 2.
digits	digits for printing of z-scores, p-values, etc. with default of 4
returnd	1 to return the deviance ratios in a list, 0 to not return. The deviances are stored in the nested.glmnetr() output object but not the deviance ratios. This function provides a simple mechanism to obtain the cross validated deviance ratios.

Value

A printout to the R console

See Also

nested.compare , summary.nested.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)
nested.cis(fit3)
```

---

nested.compare *Compare cross validation fit performances from a nested.glmnetr output.*

Description

Compare cross-validation model fits in terms of average performances from the nested cross validation fits. In general the standard deviations for the performance measures evaluated on the leave-out samples may be biased. While the standard deviations of the paired within fold differences of performances intuitively might be less biased this has not been shown. See the package vignettes for more discussion.

Usage

```
nested.compare(object, type = "devrat", digits = 4, pow = 1)
```

Arguments

object	A nested.glmnetr output object.
type	determines what type of nested cross validation performance measures are compared. Possible values are "devrat" to compare the deviance ratios, i.e. the fractional reduction in deviance relative to the null model deviance, "agree" to compare agreement, "lincal" to compare the linear calibration slope coefficients, "intcal" to compare the linear calibration intercept coefficients, from the nested cross validation.
digits	digits for printing of z-scores, p-values, etc. with default of 4
pow	the power to which the average of correlations is to be raised. Only applies to the "gaussian" model. Default is 2 to yield R-square but can be on to show correlations. pow is ignored for the family of "cox" and "binomial".

Value

A printout to the R console.

See Also

nested.cis , summary.nested.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)
nested.compare(fit3)
```

nested.compare_0_5_1 *Compare cross validation fit performances from a nested.glmnetr output.*

Description

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

Usage

```
nested.compare_0_5_1(object, digits = 4, type = "devrat", pow = 1)
```

Arguments

object	A nested.glmnetr output object.
digits	digits for printing of z-scores, p-values, etc. with default of 4
type	determines what type of nested cross validation performance measures are compared. Possible values are "devrat" to compare the deviance ratios, i.e. the fractional reduction in deviance relative to the null model deviance, "agree" to compare agreement, "lincal" to compare the linear calibration slope coefficients, "intcal" to compare the linear calibration intercept coefficients, from the nested cross validation.
pow	the power to which the average of correlations is to be raised. Only applies to the "gaussian" model. Default is 2 to yield R-square but can be on to show correlations. pow is ignored for the family of "cox" and "binomial".

Value

A printout to the R console.

See Also

nested.cis , summary.nested.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)
nested.compare(fit3)
```

`nested.glmnetr` *Using (nested) cross validation, describe and compare some machine learning model performances*

Description

Performs a nested cross validation or bootstrap validation for cross validation informed relaxed lasso, Gradient Boosting Machine (GBM), Random Forest (RF), (artificial) Neural Network (ANN) with two hidden layers, Recursive Partitioning (RPART) and step wise regression. That is hyper parameters for all these models are informed by cross validation (CV) (or in the case of RF by out-of-bag calculations), and a second layer of resampling is used to evaluate the performance of these CV informed model fits. For step wise regression CV is used to inform either a p-value for entry or degrees of freedom (df) for the final model choice. For input we require predictors (features) to be in numeric matrix format with no missing values. This is similar to how the glmnet package expects predictors. For survival data we allow input of start time as an option, and require stop time, and an event indicator, 1 for event and 0 for censoring, as separate terms. This may seem unorthodox as it might seem simpler to accept a Surv() object as input. However, multiple packages we use for model fitting models require data in various formats and this choice was the most straight forward for constructing the data formats required. As an example, the XGBoost routines require a data format specific to the XGBoost package, not a matrix, not a data frame. Note, for XGBoost and survival models, only a "stop time" variable, taking a positive value to indicate being associated with an event, and the negative of the time when associated with a censoring, is passed to the input data object for analysis.

Usage

```
nested.glmnetr(
  xs,
  start = NULL,
  y_,
  event = NULL,
  family = "gaussian",
  resample = NULL,
  folds_n = 10,
  stratified = NULL,
  dolasso = 1,
  doxgb = 0,
  dorf = 0,
  doorf = 0,
  doann = 0,
  dorpart = 0,
  dostep = 0,
  doaic = 0,
  ensemble = 0,
  method = "loglik",
  lambda = NULL,
  gamma = NULL,
```

```
  relax = TRUE,
  steps_n = 0,
  seed = NULL,
  foldid = NULL,
  limit = 1,
  fine = 0,
  ties = "efron",
  keepdata = 0,
  keepxbetas = 1,
  bootstrap = 0,
  unique = 0,
  id = NULL,
  track = 0,
  do_ncv = NULL,
  int_file = 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 (numeric) matrix form for complete data, no NA's, no Inf's, etc., and not a data frame.
start	optional start times in case of a Cox model. A numeric (vector) of length same as number of patients (n). Optionally start may be specified as a column matrix in which case the colname value is used when outputting summaries. Only the lasso, stepwise, and AIC models allow for (start,stop) time data as input.
y_	dependent variable as a numeric vector: time, or stop time for Cox model, 0 or 1 for binomial (logistic), numeric for gaussian. Must be a vector of length same as number of sample size. Optionally y_ may be specified as a column matrix in which case the colname value is used when outputting summaries.
event	event indicator, 1 for event, 0 for census, Cox model only. Must be a numeric vector of length same as sample size. Optionally event may be specified as a column matrix in which case the colname value is used when outputing summaries.
family	model family, "cox", "binomial" or "gaussian" (default)
resample	1 by default to do the Nested Cross Validation or bootstrap resampling calculations to assess model performance (see bootstrap option), or 0 to only fit the various models without doing resampling. In this case the nested.glmnetr() function will only derive the models based upon the full data set. This may be useful when exploring various models without having to do the timely resampling to assess model performance, for example, when wanting to examine extreme gradient boosting models (GBM) or Artificial Neural Network (ANN) models which can take a long time.
folds_n	the number of folds for the outer loop of the nested cross validation, and if not overridden by the individual model specifications, also the number of folds for the inner loop of the nested cross validation, i.e. the number of folds used in model derivation.

`stratified` 1 to generate fold IDs stratified on outcome or event indicators for the binomial or Cox model, 0 to generate foldid's without regard to outcome. Default is 1 for nested CV (i.e. bootstrap=0), and 0 for bootstrap>=1.

`dolasso` fit and do cross validation for lasso model, 0 or 1

`doxgb` fit and evaluate a cross validation informed XGBoost (GBM) model. 1 for yes, 0 for no (default). By default the number of folds used when training the GBM model will be the same as the number of folds used in the outer loop of the nested cross validation, and the maximum number of rounds when training the GBM model is set to 1000. To control these values one may specify a list for the doxgb argument. The list can have elements $nfold, $nrounds, and $early_stopping_rounds, each numerical values of length 1, $folds, a list as used by xgb.cv() do identify folds for cross validation, and $eta, $gamma, $max_depth, $min_child_weight, $colsample_bytree, $lambda, $alpha and $subsample, each a numeric of length 2 giving the lower and upper values for the respective tuning parameter. Here we deviate from nomenclature used elsewhere in the package to be able to use terms those used in the 'xgboost' (and mlrMBO) package, in particular as used in xgb.train(), e.g. nfold instead of folds_n and folds instead of foldid. If not provided defaults will be used. Defaults can be seen from the output object$doxgb element, again a list. In case not NULL, the seed and folds option values override the $seed and $folds values.

If to shorten run time the user sets nfold to a value other than folds_n we recommend that nfold = folds_n/2 or folds_n/3. Then the folds will be formed by collapsing the folds_n folds allowing a better comparisons of model performances between the different machine learning models. Typically one would want to keep the full data model but the GBM models can cause the output object to require large amounts of storage space so optionally one can choose to not keep the final model when the goal is basically only to assess model performance for the GBM. In that case the tuning parameters for the final tuned model ae retained facilitating recalculation of the final model, this will also require the original training data.

`dorf` fit and evaluate a random forest (RF) model. 1 for yes, 0 for no (default). Also, if dorf is specified by a list, then RF models will be fit. The randomForestSRC package is used. This list can have three elements. One is the vector mtryc, and contains values for mtry. The program searches over the different values to find a better fir for the final model. If not specified mtryc is set to round(sqrt(dim(xs)[2]) * c(0.67 , 1, 1.5, 2.25, 3.375)). The second list element the vector ntreec. The first item (ntreec[1]) specifies the number of trees to fit in evaluating the models specified by the different mtry values. The second item (ntreec[2]) specifies the number of trees to fit in the final model. The default is ntreec = c(25,250). The third element in the list is the numeric variable keep, with the value 1 (default) to store the model fit on all data in the output object, or the value 0 to not store the full data model fit. Typically one would want to keep the full data model but the RF models can cause the output object to require large amounts of storage space so optionally one can choose to not keep the final model when the goal is basically only to assess model performance for the RF. Random forests use the out-of-bag (OOB) data elements for assessing model fit and hyperparameter tuning and so cross validation is not used for tuning. Still, because of the number of trees in the forest random forest can take long to run.

doorf — fit and evaluate an Oblique random forest (RF) model. 1 for yes, 0 for no (default). While the nomenclature used by orrsf() is slightly different than that used by rfsrc() nomenclature for this object follows that of dorf.

doann — fit and evaluate a cross validation informed Artificial Neural Network (ANN) model with two hidden levels. 1 for yes, 0 for no (default). By default the number of folds used when training the ANN model will be the same as the number of folds used in the outer loop of the nested cross validation. To override this, for example to shrtn run time, one may specify a list for the doann argument where the element $folds_ann_n gives the number of folds used when training the ANN. To shorten run we recommend folds_ann_n = folds_n/2 or folds_n/3, and at least 3. Then the folds will be formed by collapsing the folds_n folds using in fitting other models allowing a better comparisons of model performances between the different machine learning models. The list can also have elements $epochs, $epochs2, $myler, $myler2, $eppr, $eppr2, $lenv1, $lenz2, $actv, $drpot, $wd, wd2, l1, l12, $lscale, $scale, $minloss and $gotoend. These arguments are then passed to the ann_tab_cv_best() function, with the meanings described in the help for that function, with some exception. When there are two similar values like $epoch and $epoch2 the first applies to the ANN models trained without transfer learning and the second to the models trained with transfer learning from the lasso model. Elements of this list unspecified will take default values. The user may also specify the element $bestof (a positive integer) to fit bestof models with different random starting weights and biases while taking the best performing of the different fits based upon CV as the final model. The default value for bestof is 1.

dorpart — fit and do a nested cross validation for an RPART model. As rpart() does its own approximation for cross validation there is no new functions for cross validation.

dostep — fit and do cross validation for stepwise regression fit, 0 or 1, as discussed in James, Witten, Hastie and Tibshirani, 2nd edition.

doaic — fit and do cross validation for AIC fit, 0 or 1. This is provided primarily as a reference.

ensemble — This is a vector 8 characters long and specifies a set of ensemble like model to be fit based upon the predicteds form a relaxed lasso model fit, by either inlcuding the predicteds as an additional term (feature) in the machine learning model, or including the predicteds similar to an offset. For XGBoost, the offset is specified in the model with the "base_margin" in the XGBoost call. For the Artificial Neural Network models fit using the ann_tab_cv_best() function, one can initialize model weights (parameters) to account for the predicteds in prediction and either let these weights by modified each epoch or update and maintain these weights during the fitting process. For ensemble[1] = 1 a model is fit ignoring these predicteds, ensemble[2]=1 a model is fit including the predicteds as an additional feature. For ensemble[3]=1 a model is fit using the predicteds as an offset when running the xgboost model, or a model is fit including the predicteds with initial weights corresponding to an offset, but then weights are allowed to be tuned over the epochs. For i >= 4 ensemble[i] only applies to the neural network models. For ensemble[4]=1 a model is fit like for ensemble[3]=1 but the weights are reassigned to correspond to an offset after each epoch. For i in (5,6,7,8) ensemble[i] is similar to ensemble[i-4] except the

original predictor (feature) set is replaced by the set of non-zero terms in the relaxed lasso model fit. If ensemble is specified as 0 or NULL, then ensemble is assigned c(1,0,0,0, 0,0,0,0). If ensemble is specified as 1, then ensemble is assigned c(1,0,0,0, 0,1,0,1).

`method` method for choosing model in stepwise procedure, "loglik" or "concordance". Other procedures use the "loglik".

`lambda` lambda vector for the 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

`steps_n` number of steps done in stepwise regression fitting

`seed` optional, either NULL, or a numerical/integer vector of length 2, for R and torch random generators, or a list with two vectors, each of length folds_n+1, for generation of random folds for the full data model as well as the the outer cross validation loop, and the remaining folds_n terms for the random generation of the folds or the bootstrap samples for the model fits of the inner loops. This can be used to replicate model fits. Whether specified or NULL, the seed is stored in the output object for future reference. The stored seed is a list with two vectors seedr for the seeds used in generating the random fold splits, and seedt for generating the random initial weights and biases in the torch neural network models. The first element in each of these vectors is for the all data fits and remaining elements for the folds of the inner cross validation. The integers assigned to seed should be positive (>= 1) and not more than 2147483647. Beginning in version 0.5-5 the values from $seedr and $seedt the first last element in each vector was used when fitting each model on the whole data set and the other values were used for the outer cross validation or the bootstrap sample generation. For version 0.4-2 through 0.5-4 the last element from each vector was used when fitting each model on the whole dataset. This change was made so that the set of full models numerical fits do not depend on whether or not resampling is performed (resample is set to 0 or 1 2) or the number of bootstrap resamples. The seeds are generated with the glmnetr_seed() function.

`foldid` a vector of integers to associate each record to a fold. Should be integers from 1 and folds_n. These will only be used in the outer folds.

`limit` limit the small values for lambda after the initial fit. This will 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 validation many times to more finely tune the hyper paramters. See the 'glmnet' package documentation

`ties` method for handling ties in Cox model for relaxed model component. Default is "efron", optionally "breslow". For penalized fits "breslow" is always used as derived form to 'glmnet' package.

`keepdata` 0 (default) to delete the input data (xs, start, y_, event) from the output objects from the random forest fit and the glm() fit for the stepwise AIC model, 1 to keep.

keepxbetas	1 (default) to retain in the output object a copy of the functional outcome variable, i.e. y_ for "gaussian" and "binomial" data, and the Surv(y_,event) or Surv(start,y_,event) for "cox" data. This allows calibration studies of the models, going beyond the linear calibration information calculated by the function. The xbetas are calculated both for the model derived using all data as well as for the hold out sets (1/k of the data each) for the models derived within the cross validation ((k-1)/k of the data for each fit).
bootstrap	0 (default) to use nested cross validation, a positive integer to perform as many iterations of the bootstrap for model evaluation.
unique	0 to use the bootstrap sample as is as training data, 1 to include the unique sample elements only once. A fractional value between 0.5 and 0.9 will sample without replacement a fraction of this value for training and use the remaining as test data.
id	optional vector identifying dependent observations. Can be used, for example, when some study subjects have more than one row in the data. No values should be NA. Default is NULL where all rows can be regarded as independent.
track	1 (default) to track progress by printing to console elapsed and split times, 0 to not track
do_ncv	Deprecated, and replaced by resample
int_file	A file name at which to save the intermediate results at the end of each outer loop of the resampling. This may be useful when fitting one of the machine learning models crashes or hangs in one of the iterations of the outer loop (resampling of the nested cross validation or the bootstrap). The value for int_file must be a valid file name for your operating system and installation.
...	additional arguments that can be passed to glmnet()

Value

- Model fit performance for LASSO, GBM, Random Forest, Oblique Random Forest, RPART, artificial neural network (ANN) or STEPWISE models are estimated using k-cross validation or bootstrap. Full data model fits for these models are also calculated independently (prior to) the performance evaluation, often using a second layer of resampling validation.

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

glmnetr.simdata , summary.nested.glmnetr , nested.compare , plot.nested.glmnetr , predict.nested.glmnetr , predict_ann_tab, cv.glmnetr , xgb.tuned , rf_tune , orf_tune , ann_tab_cv , cv.stepreg , glmnetr_seed

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, type="devrat", ylim=c(0.7,1))
plot(nested.glmnetr.fit, type="lincal", ylim=c(0.9,1.1))
plot(nested.glmnetr.fit, type="lasso")
plot(nested.glmnetr.fit, type="coef")
summary(nested.glmnetr.fit)
nested.compare(nested.glmnetr.fit)
summary(nested.glmnetr.fit, cvfit=TRUE)
```

`orf_tune` *Fit a Random Forest model on data provided in matrix and vector formats.*

Description

Fit an Random Forest model using the orsf() function of the aorsf package.

Usage

```
orf_tune(
  xs,
  start = NULL,
  y_,
  event = NULL,
  family = NULL,
  mtryc = NULL,
  ntreec = NULL,
  nsplitc = 8,
  seed = NULL,
  tol = 1e-05,
  track = 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`	an optional vector of start times in case of a Cox model. Class numeric of length same as number of patients (n)
`y_`	dependent variable as a vector: time, or stop time for Cox model, Y_ 0 or 1 for binomial (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" (default)
mtryc	a vector (numeric) of values to search over for optimization of the Random Forest fit. This if for the mtry input variable of the orsf() program specifying the number of terms to consider in each step of teh Random Forest fit.
ntreec	a vector (numeric) of 2 values, the first for the number of forests (ntree from orsf()) to use when searhcing for a better bit and the second to use when fitting the final model. More trees should give a better fit but require more computations and storage for the final. model.
nsplitc	This nsplit of orsf(), a non-negative integer for the number of random splits for a predictor.
seed	a seed for set.seed() so one can reproduce the model fit. If NULL the program will generate a random seed. Whether specified or NULL, the seed is stored in the output object for future reference. Note, for the default this randomly generated seed depends on the seed in memory at that time so will depend on any calls of set.seed prior to the call of this function.
tol	a small number, a lower bound to avoid division by 0
track	1 to output a brief summary of the final selected model, 3 to output a brief summary on each model fit in search of a better model or 0 (default) to not output this information.

Value

a Random Forest model fit

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

summary.orf_tune , rederive_orf , nested.glmnetr

---

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 deviance based upon a cv.glmnetr() output object. Optionally, with coefs=TRUE, plots the relaxed 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() 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 be 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 results from a nested.glmnetr() output*

Description

Plot the nested cross validation performance numbers, cross validated relaxed lasso deviances or coefficients from a nested.glmnetr() call.

Usage

```
## S3 method for class 'nested.glmnetr'
plot(
  x,
  type = "devrat",
  gam = NULL,
  lambda.lo = NULL,
  title = NULL,
  plup = 0,
  coefs = FALSE,
  comment = TRUE,
  pow = 2,
  ylim = 1,
  plot = 1,
  fold = 1,
  xgbsimple = 0,
  ...
)
```

Arguments

`x`	A nested.glmnetr output object
`type`	type of plot to be produced form the (nested) cross validation performance measures, and the lasso model tuning or lasso model coefficients. For the lasso model the options include "lasso" to plot deviances informing hyperparmeter choice or "coef" to plot lasso parameter estimates. Else nested cross validation performance measures are plotted. To show cross validation performance

	measures the options include "devrat" to plot deviance ratios, i.e. the fractional reduction in deviance relative to the null model deviance, "agree" to plot agreement, "lincal" to plot the linear calibration slope coefficients, "intcal" to plot the linear calibration intercept coefficients or "devian" to plot the deviances from the nested cross validation. For each performance measure estimates from the individual (outer) cross validation fold are depicted by thin lines of different colors and styles, while the composite value from all fol=ds is depicted by a thicker black line, and the performance measures naively calculated on the all data using the model derived from all data is depicted in a thicker red line.
gam	A specific level of gamma for plotting. By default gamma.min will be used. Applies only for type = "lasso".
lambda.lo	A lower limit of lambda when plotting. Applies only for type = "lasso".
title	A title
plup	Plot upper 95 percent two-sided confidence intervals for the deviance plots. Applies only for type = "lasso".
coefs	Depricated. See option 'type'. To plot coefficients specify type = "coef".
comment	Default of TRUE to write to console information on lam and gam selected for output. FALSE will suppress this write to console. Applies only for type = "lasso".
pow	Power to which agreement is to be raised when the "gaussian" model is fit, i.e. 2 for R-square, 1 for correlation. Does not apply to type = "lasso".
ylim	y axis limits for model performance plots, i.e. does not apply to type = "lasso". The ridge model may calibrate very poorly obscuring plots for type of "lincal" or "intcal", so one may specify the ylim value. If ylim is set to 1, then the program will derive a reasonable range for ylim. If ylim is set to 0, then the entire range for all models will be displayed. Does not apply to type = "lasso".
plot	By default 1 to produce a plot, 0 to return the data used in the plot in the form of a list.
fold	By default 1 to display model performance estimates form individual folds (or replicaitons for boostrap evaluations) when type of "agree", "intcal", "lincal", "devrat" for "devian". If 0 then the individual fold calculations are not displayed. When there are many replications as sometimes the case when using bootstrap, one may specify the number of randomly selected lines for plotting.
xgbsimple	1 (default) to include results for the untuned XGB model, 0 to not include.
...	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_perf_glmnetr , calplot , 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, type="coef")
```

---

plot_perf_glmnetr *Plot nested cross validation performance summaries*

Description

This function plots summary information from a nested.glmnetr() output object, that is from a nested cross validation performance. Alternamvely one can output the numbers otherwise displayed to a list for extraction or customized plotting. Performance measures for plotting include "devrat" the deviance ratio, i.e. the fractional reduction in deviance relative to the null model deviance, "agree" a measure of agreement, "lincal" the slope from a linear calibration and "intcal" the intercept from a linear calibration. Performance measure estimates from the individual (outer) cross validation fold are depicted by thin lines of different colors and styles, while the composite value from all folds is depicted by a thicker black line, and the performance measures naively calculated on the all data using the model derived from all data is depicted by a thicker red line.

Usage

```
plot_perf_glmnetr(
  x,
  type = "devrat",
  pow = 2,
  ylim = 1,
  fold = 1,
  xgbsimple = 0,
  plot = 1
)
```

Arguments

x	A nested.glmnetr output object

type	determines what type of nested cross validation performance measures are plotted. Possible values are "devrat" to plot the deviance ratio, i.e. the fractional reduction in deviance relative to the null model deviance, "agree" to plot agreement in terms of concordance, correlation or R-square, "lincal" to plot the linear calibration slope coefficients, "intcal" to plot the linear calibration intercept coefficients, from the (nested) cross validation.
pow	Power to which agreement is to be raised when the "gaussian" model is fit, i.e. 2 for R-square, 1 for correlation. Does not apply to type = "lasso".
ylim	y axis limits for model perforamnce plots, i.e. does not apply to type = "lasso". The ridge model may calibrate very poorly obscuring plots for type of "lincal" or "intcal", so one may specify the ylim value. If ylim is set to 1, then the program will derive a reasonable range for ylim. If ylim is set to 0, then the entire range for all models will be displayed. Does not apply to type = "lasso".
fold	By default 1 to display using a spaghetti the performance as calculated from the individual folds, 0 to display using dots only the composite values calculated using all folds.
xgbsimple	1 (default) to include results for the untuned XGB model, 0 to not include.
plot	By default 1 to produce a plot, 0 to return the data used in the plot in the form of a list.

Value

This program returns a plot to the graphics window by default, and returns a list with data used in teh plots if the plot=1 is specified.

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

plot.nested.glmnetr , nested.glmnetr

---

predict.cv.glmnetr *Give predicteds based upon a cv.glmnetr() output object.*

Description

Give predicteds based upon a cv.glmnetr() output object. By default lambda and gamma are chosen as the minimizing values for the relaxed lasso model. If gam=1 and lam=NULL then the best unrelaxed lasso model is chosen and if gam=0 and lam=NULL then the best fully relaxed lasso model is selected.

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_new*beta estimates based upon the predictor matrix xs_new) 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

summary.cv.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.cv.stepreg *Beta's or predicteds based upon a cv.stepreg() output object.*

Description

Give predicteds or Beta's based upon a cv.stepreg() output object. If an input data matrix is specified the X*Beta's are output. If an input data matrix is not specified then the Beta's are output. In the first column values are given based upon df as a tuning parameter and in the second column values based upon p as a tuning parameter.

Usage

```
## S3 method for class 'cv.stepreg'
predict(object, xs = NULL, ...)
```

Arguments

object	cv.stepreg() output object
xs	dataset for predictions. Must have the same columns as the input predictor matrix in the call to cv.stepreg().
...	pass through parameters

Value

a matrix of beta's or predicteds

See Also

summary.cv.stepreg, cv.stepreg , nested.glmnetr

---

predict.glmnetr *Get predicteds or coefficients using a glmnetr output object*

Description

Give predicteds based upon a glmnetr() output object. Because the glmnetr() function has no cross validation information, lambda and gamma must be specified. To choose lambda and gamma based upon cross validation one may use the cv.glmnetr() or nested.glmnetr() and the corresponding predict() functions.

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. Required.
gam	The value for gamma for determining the lasso fit. Required.
...	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 validation informed 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 validation informed 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_new*Beta estimates based upon the predictor matrix, or model coefficients.

See Also

predict.cv.glmnetr , predict_ann_tab , nested.glmnetr

Examples

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

---

predict_ann_tab *Get predicteds for an Artificial Neural Network model fit in nested.glmnetr()*

Description

All but one of the Artificial Neural Network (ANNs) fit by nested.glmnetr() are based upon a neural network model and input from a lasso model. Thus a simple model(xs) statement will not give the proper predicted values. This function process information form the lasso and ANN model fits to give the correct predicteds. Whereas the ann_tab_cv() function ca be used to fit a model based upon an input data set it does not fit a lasso model to allow an informed starting point for the ANN fit. The pieces fo this are in nested.glmnetr(). To fit a cross validation (CV) informed ANN model fit one can run nested.glmnetr() with folds_n = 0 to derive the full data models without doing a cross validation.

Usage

```
predict_ann_tab(object, xs, modl = NULL)
```

Arguments

object	a output object from the nested.glmnetr() function
xs	new data of the same form used as input to nested.glmnetr()
modl	ANN model entry an integer from 1 to 5 indicating which "lasso informed" ANN is to be used for calculations. The number corresponds to the position of the ensemble input from the nested.glmnetr() call. The model must already be fit to calculate predicteds: 1 for ensemble[1] = 1, for model based upon raw data

; 2 for ensemble[2] = 1, raw data plus lasso predicteds as a predictor variable (features) ; 4 for ensemble[3] = 1, raw data plus lasso predicteds and initial weights corresponding to offset and allowed to update ; 5 for ensemble[4] = 1, raw data plus lasso predicteds and initial weights corresponding to offset and not allowed to updated ; 6 for ensemble[5] = 1, nonzero relaxed lasso terms ; 7 for ensemble[6] = 1, nonzero relaxed lasso terms plus lasso predicteds as a predictor variable (features) ; 8 for ensemble[7] = 1, nonzero relaxed lasso terms plus lasso predicteds with initial weights corresponding to offset and allowed to update ; 9 for ensemble[8] = 1, nonzero relaxed lasso terms plus lasso predicteds with initial weights corresponding to offset and not allowed to update.

Value

a vector of predicteds

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

ann_tab_cv , nested.glmnetr

`print.nested.glmnetr` *A redirect to the summary() function for nested.glmnetr() output objects*

Description

A redirect to the summary() function for nested.glmnetr() output objects

Usage

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

Arguments

x	a nested.glmnetr() output object.
...	additional pass through inputs for the print function.

Value

- a nested cross validation fit summary, or a cross validation model summary.

See Also

summary.nested.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)
print(fit3)
```

`print.orf_tune` *Print output from orf_tune() function*

Description

Print output from orf_tune() function

Usage

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

Arguments

x	output from an orf_tune() function
...	optional pass through parameters to pass to print.orf()

Value

summary to console

See Also

summary.orf_tune , orf_tune , nested.glmnetr

print.rf_tune *Print output from rf_tune() function*

Description

Print output from rf_tune() function

Usage

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

Arguments

x	output from an rf_tune() function
...	optional pass through parameters to pass to print.rfsrc()

Value

summary to console

See Also

summary.rf_tune , rf_tune , nested.glmnetr

rederive_orf *Rederive Oblique Random Forest models not kept in nested.glmnetr() output*

Description

Because the oblique random forest models sometimes take large amounts of storage one may decide to set keep=0 within the doorf list passed to nested.glmnetr(). This function allows the user to rederive the oblique random forest models without doing the search. Note, the oblique random forest fitting for survival data routine does not allow for (start,stop) times.

Usage

```
rederive_orf(object, xs, y_, event = NULL, type = NULL)
```

Arguments

object	A nested.glmnetr() output object
xs	Same xs used as input to ntested.glmnetr() for input object.
y_	Same y_ used as input to ntested.glmnetr() for input object.
event	Same event used as input to ntested.glmnetr() for input object.
type	Same type used as input to ntested.glmnetr() for input object.

Value

an output like nested.glmnetr()$rf_tuned_fitX for X in c("", "F", "O")

See Also

orf_tune , nested.glmnetr

---

rederive_rf *Rederive Random Forest models not kept in nested.glmnetr() output*

Description

Because the random forest models sometimes take large amounts of storage one may decide to set keep=0 within the dorf list passed to nested.glmnetr(). This function allows the user to rederive the random forest models without doing the search. Note, the random forest fitting routine does not allow for (start,stop) times.

Usage

```
rederive_rf(object, xs, y_, event = NULL, type = NULL)
```

Arguments

object	A nested.glmnetr() output object
xs	Same xs used as input to ntested.glmnetr() for input object.
y_	Same y_ used as input to ntested.glmnetr() for input object.
event	Same event used as input to ntested.glmnetr() for input object.
type	Same type used as input to ntested.glmnetr() for input object.

Value

an output like nested.glmnetr()$rf_tuned_fitX for X in c("", "F", "O")

See Also

rf_tune , nested.glmnetr

rederive_xgb *Rederive XGB models not kept in nested.glmnetr() output*

Description

Because the XGBoost models sometimes take large amounts of storage one may decide to set keep=0 with in the doxgb list passed to nested.glmnetr(). This function allows the user to rederive the XGBoost models without doing the search. Note, the random forest fitting routine does not allow for (start,stop) times.

Usage

```
rederive_xgb(object, xs, y_, event = NULL, type = "base", tuned = 1)
```

Arguments

object	A nested.glmnetr() output object
xs	Same xs used as input to ntested.glmnetr() for input object.
y_	Same y_ used as input to ntested.glmnetr() for input object.
event	Same event used as input to ntested.glmnetr() for input object.
type	Same type used as input to ntested.glmnetr() for input object.
tuned	1 (default) to derive the tuned model like with xgb.tuned(), 0 to derive the basic models like with xgb.simple().

Value

an output like nested.glmnetr()$xgb.simple.fitX or nested.glmnetr()$xgb.tuned.fitX for X in c("", "F", "O")

See Also

xgb.tuned , xgb.simple , nested.glmnetr

rf_tune *Fit a Random Forest model on data provided in matrix and vector formats.*

Description

Fit an Random Forest model using the rfsrc() function of the randomForestSRC package.

Usage

```
rf_tune(
  xs,
  start = NULL,
  y_,
  event = NULL,
  family = NULL,
  mtryc = NULL,
  ntreec = NULL,
  nsplitc = 8,
  seed = NULL,
  track = 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	an optional vector of start times in case of a Cox model. Class numeric of length same as number of patients (n)
y_	dependent variable as a vector: time, or stop time for Cox model, Y_ 0 or 1 for binomial (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" (default)
mtryc	a vector (numeric) of values to search over for optimization of the Random Forest fit. This if for the mtry input variable of the rfsrc() program specifying the number of terms to consider in each step of teh Random Forest fit.
ntreec	a vector (numeric) of 2 values, the first for the number of forests (ntree from rfsrc()) to use when searhcing for a better bit and the second to use when fitting the final model. More trees should give a better fit but require more computations and storage for the final. model.
nsplitc	This nsplit of rfsrc(), a non-negative integer for the number of random splits for a predictor.
seed	a seed for set.seed() so one can reproduce the model fit. If NULL the program will generate a random seed. Whether specified or NULL, the seed is stored in the output object for future reference. Note, for the default this randomly generated seed depends on the seed in memory at that time so will depend on any calls of set.seed prior to the call of this function.
track	1 to output a brief summary of the final selected model, 2 to output a brief summary on each model fit in search of a better model or 0 (default) to not output this information.

Value

a Random Forest model fit

Author(s)

Walter Kremers (kremers.walter@mayo.edu)

See Also

summary.rf_tune , rederive_rf , nested.glmnetr

---

roundperf *round elements of a summary.glmnetr() output*

Description

round elements of a summary.glmnetr() output

Usage

```
roundperf(summdf, digits = 3, resample = 1)
```

Arguments

summdf	a summary data frame from summary.nested.glmnetr() obtained using the option table=0
digits	the minimum number of decimals to display the elements of the data frame
resample	1 (default) if the summdf object is a summary for an analysis including nested cross validation, 0 if only the full data models were fit.

Value

a data frame with same form as the input but with rounding for easier display

See Also

summary.nested.glmnetr , nested.glmnetr

stepreg *Fit the steps of a stepwise regression.*

Description

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,
  track = 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"
track	1 to output stepwise fit program, 0 (default) to suppress

Value

does a stepwise regression of depth maximum depth steps_n

See Also

summary.stepreg , aicreg , cv.stepreg , nested.glmnetr

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-validation informed model fit. 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 FALSE 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

`predict.cv.glmnetr` , `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 results from a cv.stepreg() output object.*

Description

Summarize results from a cv.stepreg() output object.

Usage

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

Arguments

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

Value

Summary of a stepreg() (stepwise regression) output object.

See Also

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

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
# for this example we use small numbers for steps_n and folds_n to shorten run time
```

```
cv.stepreg.fit = cv.stepreg(xs, NULL, y_, event, steps_n=10, folds_n=3, track=0)
summary(cv.stepreg.fit)
```

---

summary.nested.glmnetr *Summarize a nested.glmnetr() output object*

Description

Summarize the model fit from a nested.glmnetr() output object, i.e. the fit of a cross-validation informed relaxed lasso model fit, inferred by nested cross validation. Else summarize the cross-validated model fit.

Usage

```
## S3 method for class 'nested.glmnetr'
summary(
  object,
  cvfit = FALSE,
  pow = 2,
  printg1 = FALSE,
  digits = 4,
  call = NULL,
  onese = 0,
  table = 1,
  tuning = 0,
  width = 84,
  cal = 0,
  ...
)
```

Arguments

object	a nested.glmnetr() output object.
cvfit	default of FALSE to summarize fit of a cross validation informed relaxed lasso model fit, inferred by nested cross validation. Option of TRUE will describe the cross validation informed relaxed lasso model itself.
pow	the power to which the average of correlations is to be raised. Only applies to the "gaussian" model. Default is 2 to yield R-square but can be on to show correlations. Pow is ignored for the family of "cox" and "binomial".
printg1	TRUE to also print out the fully penalized lasso beta, else to suppress. Only applies to cvfit=TRUE.
digits	digits for printing of deviances, linear calibration coefficients and agreement (concordances and R-squares).

call	1 to print call used in generation of the object, 0 or NULL to not print
onese	0 (default) to not include summary for 1se lasso fits in tables, 1 to include
table	1 to print table to console, 0 to output the tabled information to a data frame
tuning	1 to print tuning parameters, 0 (default) to not print
width	character width of the text body preceding the performance measures which can be adjusted between 60 and 120.
cal	1 print performance statistics for lasso models calibrated on training data, 2 to print performance statistics for lasso and random forest models calibrated on training data, 0 (default) to not print. Note, despite any intuitive appeal these training data calibrated models may sometimes do rather poorly.
...	Additional arguments passed to the summary function.

Value

- a nested cross validation fit summary, or a cross validation model summary.

See Also

nested.compare , nested.cis , summary.cv.glmnetr , roundperf , plot.nested.glmnetr , calplot , 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.orf_tune *Summarize output from rf_tune() function*

Description

Summarize output from rf_tune() function

Usage

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

Arguments

object	output from an rf_tune() function
...	optional pass through parameters to pass to summary.orsf()

Value

summary to console

See Also

rf_tune , nested.glmnetr

---

summary.rf_tune *Summarize output from rf_tune() function*

Description

Summarize output from rf_tune() function

Usage

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

Arguments

object	output from an rf_tune() function
...	optional pass through parameters to pass to summary.rfsrc()

Value

summary to console

See Also

rf_tune , nested.glmnetr

summary.stepreg — *Briefly summarize steps in a stepreg() output object, i.e. a stepwise regression fit*

Description

Briefly summarize steps in a stepreg() output object, i.e. 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

See Also

stepreg , cv.stepreg , nested.glmnetr

xgb.simple — *Get a simple XGBoost model fit (no tuning)*

Description

This fits a gradient boosting machine model using the XGBoost platform. If uses a single set of hyperparameters that have sometimes been reasonable so runs very fast. For a better fit one can use xgb.tuned() which searches for a set of hyperparameters using the mlrMBO package which will generally provide a better fit but take much longer. See xgb.tuned() for a description of the data format required for input.

Usage

```
xgb.simple(
  train.xgb.dat,
  booster = "gbtree",
  objective = "survival:cox",
  eval_metric = NULL,
  minimize = NULL,
```

```
  seed = NULL,
  folds = NULL,
  doxgb = NULL,
  track = 2
)
```

Arguments

train.xgb.dat	The data to be used for training the XGBoost model
booster	for now just "gbtree" (default)
objective	one of "survival:cox" (default), "binary:logistic" or "reg:squarederror"
eval_metric	one of "cox-nloglik" (default), "auc", "rmse" or NULL. Default of NULL will select an appropriate value based upon the objective value.
minimize	whether the eval_metric is to be minimized or maximized
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.
folds	an optional list where each element is a vector of indexes for a test fold. Default is NULL. If specified then doxgb$nfold is ignored as in xgb.cv().
doxgb	a list with parameters for passed to xgb.cv() including $nfold, $nrounds, and $early_stopping_rounds. If not provided defaults will be used. Defaults can be seen form the output object$doxgb element, again a list. In case not NULL, the seed and folds option values override the $seed and $folds values in doxgb.
track	0 (default) to not track progress, 2 to track progress.

Value

a XGBoost model fit

Author(s)

Walter K Kremers with contributions from Nicholas B Larson

See Also

xgb.tuned , nested.glmnetr

Examples

```
# Simulate some data for a Cox model
sim.data=glmnetr.simdata(nrows=1000, ncols=100, beta=NULL)
Surv.xgb = ifelse( sim.data$event==1, sim.data$yt, -sim.data$yt )
data.full <- xgboost::xgb.DMatrix(data = sim.data$xs, label = Surv.xgb)
# for this example we use a small number for folds_n and nrounds to shorten run time
xgbfit = xgb.simple( data.full, objective = "survival:cox")
preds = predict(xgbfit, sim.data$xs)
summary( preds )
preds[1:8]
```

xgb.tuned *Get a tuned XGBoost model fit*

Description

This fits a gradient boosting machine model using the XGBoost platform. It uses the mlrMBO mlrMBO package to search for a well fitting set of hyperparameters and will generally provide a better fit than xgb.simple(). Both this program and xgb.simple() require data to be provided in a xgb.DMatrix() object. This object can be constructed with a command like data.full <- xgb.DMatrix(data=myxs, label=mylabel), where myxs object contains the predictors (features) in a numerical matrix format with no missing values, and mylabel is the outcome or dependent variable. For logistic regression this would typically be a vector of 0's and 1's. For linear regression this would be vector of numerical values. For a Cox proportional hazards model this would be in a format required for XGBoost, which is different than for the survival package or glmnet package. For the Cox model a vector is used where observations associated with an event are assigned the time of event, and observations associated with censoring are assigned the NEGATIVE of the time of censoring. In this way information about time and status are communicated in a single vector instead of two vectors. The xgb.tuned() function does not handle (start,stop) time, i.e. interval, data. To tune the xgboost model we use the mlrMBO package which "suggests" the DiceKriging and rgenoud packages, but doe not install these. Still, for xgb.tuned() to run it seems that one should install the DiceKriging and rgenoud packages.

Usage

```
xgb.tuned(
  train.xgb.dat,
  booster = "gbtree",
  objective = "survival:cox",
  eval_metric = NULL,
  minimize = NULL,
  seed = NULL,
  folds = NULL,
  doxgb = NULL,
  track = 0
)
```

Arguments

train.xgb.dat	The data to be used for training the XGBoost model
booster	for now just "gbtree" (default)
objective	one of "survival:cox" (default), "binary:logistic" or "reg:squarederror"
eval_metric	one of "cox-nloglik" (default), "auc" or "rmse",
minimize	whether the eval_metric is to be minimized or maximized

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.
folds	an optional list where each element is a vector of indeces for a test fold. Default is NULL. If specified then nfold is ignored a la xgb.cv().
doxgb	A list specifying how the program is to do the xgb tune and fit. The list can have elements $nfold, $nrounds, and $early_stopping_rounds, each numerical values of length 1, $folds, a list as used by xgb.cv() do identify folds for cross validation, and $eta, $gamma, $max_depth, $min_child_weight, $colsample_bytree, $lambda, $alpha and $subsample, each a numeric of length 2 giving the lower and upper values for the respective tuning parameter. The meaning of these terms is as in 'xgboost' xgb.train(). If not provided defaults will be used. Defaults can be seen from the output object$doxgb element, again a list. In case not NULL, the seed and folds option values override the $seed and $folds values.
track	0 (default) to not track progress, 2 to track progress.

Value

a tuned XGBoost model fit

Author(s)

Walter K Kremers with contributions from Nicholas B Larson

See Also

xgb.simple , rederive_xgb , nested.glmnetr

Examples

```
# Simulate some data for a Cox model
sim.data=glmnetr.simdata(nrows=1000, ncols=100, beta=NULL)
Surv.xgb = ifelse( sim.data$event==1, sim.data$yt, -sim.data$yt )
data.full <- xgboost::xgb.DMatrix(data = sim.data$xs, label = Surv.xgb)
# for this example we use a small number for folds_n and nrounds to shorten
# run time.  This may still take a minute or so.
# xgbfit=xgb.tuned(data.full,objective="survival:cox",nfold=5,nrounds=20)
# preds = predict(xgbfit, sim.data$xs)
# summary( preds )
```

Index