Package ‘c060’

February 19, 2013

Version 0.15

Date 2012-06-09

Author Martin Sill, Thomas Hielscher, Manuela Zucknick, Natalia Becker.

Maintainer Martin Sill <m.sill@dkfz.de>

Title Additional variable selection, model validation and parameter tuning functions for glmnet models

Depends glmnet, survival, parallel, mlegp, tgp, peperr, penalizedSVM

Imports glmnet, survival, parallel, mlegp, tgp, peperr, penalizedSVM

Suggests survival

Description c060 provides additional functions to perform stability selection, model validation and parameter tuning for glmnet models

License GPL-2

LazyLoad yes

R topics documented:

aggregation.auc *Determine the area under the ROC curve for a fitted model*

Description

Evaluate the area under the ROC curve for a fitted model on new data. To be used as argument `aggregation.fun` in `peperr` call.

Usage

```
aggregation.auc(full.data=NULL, response, x, model, cplx=NULL,  type=c("apparent", "noinf"),
   fullsample.attr = NULL, ...)
```

Arguments

`full.data`	passed from `peperr`, but not used for calculation.
`response`	vector of binary response.
`x`	`n*p` matrix of covariates.
`model`	model fitted as returned by a `fit.fun`, as used in a call to `peperr`.
`cplx`	passed from `peperr`, but not necessary for calculation.
`type`	character.
`fullsample.attr`	passed from `peperr`, but not necessary for calculation.
`...`	additional arguments, passed to `predict` function.

Details

Area under the ROC curve is calculated based on internal `glmnet:::auc` function from package `glmnet`.

Value

Scalar, indicating the area under the ROC curve.

Author(s)

Thomas Hielscher \ `<t.hielscher@dkfz.de>`

See Also

`peperr`

`complexity.glmnet`	*Interface for determination of penalty lambda in penalized regression model via cross-validation*

Description

Determines the amount of shrinkage for a penalized regression model fitted by glmnet via cross-validation, conforming to the calling convention required by argument `complexity` in `peperr` call.

Usage

```
complexity.glmnet(response, x, full.data, ...)
```

Arguments

`response`	a survival object (with `Surv(time, status)`, or a binary vector with entries 0 and 1).
`x`	`n*p` matrix of covariates.
`full.data`	data frame containing response and covariates of the full data set.
`...`	additional arguments passed to `cv.glmnet` call such as `family`.

Details

Function is basically a wrapper for `cv.glmnet` of package `glmnet`. A n-fold cross-validation (default n=10) is performed to determine the optimal penalty lambda. For Cox PH regression models the deviance based on penalized partial log-likelihood is used as loss function. For binary endpoints other loss functions are available as well (see `type.measure`). Deviance is default. Calling `peperr`, the default arguments of `cv.glmnet` can be changed by passing a named list containing these as argument `args.complexity`. Note that only penalized Cox PH (`family="cox"`) and logistic regression models (`family="binomial"`) are sensible for prediction error evaluation with package `peperr`.

Value

Scalar value giving the optimal lambda.

Author(s)

Thomas Hielscher \ `<t.hielscher@dkfz.de>`

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*, `http://www.stanford.edu/~hastie/Papers/glmnet.pdf`
Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010
`http://www.jstatsoft.org/v33/i01/`

Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2011) *Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5) 1-13*
http://www.jstatsoft.org/v39/i05/
Porzelius, C., Binder, H., and Schumacher, M. (2009) *Parallelized prediction error estimation for evaluation of high-dimensional models, Bioinformatics, Vol. 25(6), 827-829.*

See Also

`peperr`, `cv.glmnet`

EPSGO — *Finds an optimal solution*

Description

Finds an optimal solution for the Q.func function.

Usage

```
EPSGO(Q.func, bounds,  round.n=5, parms.coding="none", fminlower=0, flag.find.one.min =FALSE,
show=c("none", "final", "all"), N= NULL, maxevals = 500,
    pdf.name=NULL,  pdf.width=12,  pdf.height=12,   my.mfrow=c(1,1),
    verbose=TRUE, seed=123,  ...  )
```

Arguments

`Q.func`	name of the function to be minimized.
`bounds`	bounds for parameters
`round.n`	number of digits after comma, default: 5
`parms.coding`	parmeters coding: none or log2, default: none.
`fminlower`	minimal value for the function Q.func, default is 0.
`flag.find.one.min`	do you want to find one min value and stop? Default: FALSE
`show`	show plots of DIRECT algorithm: none, final iteration, all iterations. Default: none
`N`	define the number of start points, see details.
`maxevals`	the maximum number of DIRECT function evaluations, default: 500.
`pdf.name`	pdf name
`pdf.width`	default 12
`pdf.height`	default 12
`my.mfrow`	default c(1,1)
`verbose`	verbose? default TRUE.
`seed`	seed
`...`	additional argument(s)

Details

if the number of start points (N) is not defined by the user, it will be defined dependent on the dimensionality of the parameter space. N=10D+1, where D is the number of parameters, but for high dimensional parameter space with more than 6 dimensions, the initial set is restricted to 65. However for one-dimensional parameter space the N is set to 21 due to stability reasons.

The idea of EPSGO (Efficient Parameter Selection via Global Optimization): Beginning from an intial Latin hypercube sampling containing N starting points we train an Online GP, look for the point with the maximal expected improvement, sample there and update the Gaussian Process(GP). Thereby it is not so important that GP really correctly models the error surface of the SVM in parameter space, but that it can give a us information about potentially interesting points in parameter space where we should sample next. We continue with sampling points until some convergence criterion is met.

DIRECT is a sampling algorithm which requires no knowledge of the objective function gradient. Instead, the algorithm samples points in the domain, and uses the information it has obtained to decide where to search next. The DIRECT algorithm will globally converge to the maximal value of the objective function. The name DIRECT comes from the shortening of the phrase 'DIviding RECTangles', which describes the way the algorithm moves towards the optimum.

The code source was adopted from MATLAB originals, special thanks to Holger Froehlich.

Value

`fmin`	minimal value of Q.func on the interval defined by bounds.
`xmin`	coreesponding parameters for the minimum
`iter`	number of iterations
`neval`	number of visited points
`maxevals`	the maximum number of DIRECT function evaluations
`seed`	seed
`bounds`	bounds for parameters
`Q.func`	name of the function to be minimized.
`points.fmin`	the set of points with the same fmin
`Xtrain`	visited points
`Ytrain`	the output of Q.func at visited points Xtrain
`gp.seed`	seed for Gaussian Process
`model.list`	detailed information of the search process

Author(s)

Natalia Becker natalia.becker at dkfz.de

References

Froehlich, H. and Zell, A. (2005) "Effcient parameter selection for support vector machines in classification and regression via model-based global optimization" *In Proc. Int. Joint Conf. Neural Networks, 1431-1438* .

fit.glmnet *Interface function for fitting a penalized regression model with* glmnet

Description

Interface for fitting penalized regression models for binary of survival endpoint using glmnet, conforming to the requirements for argument fit.fun in peperr call.

Usage

```
fit.glmnet(response, x, cplx, ...)
```

Arguments

response	a survival object (with Surv(time, status), or a binary vector with entries 0 and 1).
x	n*p matrix of covariates.
cplx	lambda penalty value.
...	additional arguments passed to glmnet call such as family.

Details

Function is basically a wrapper for glmnet of package **glmnet**. Note that only penalized Cox PH (family="cox") and logistic regression models (family="binomial") are sensible for prediction error evaluation with package peperr.

Value

glmnet object

Author(s)

Thomas Hielscher \ <t.hielscher@dkfz.de>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*, http://www.stanford.edu/~hastie/Papers/glmnet.pdf
Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010
http://www.jstatsoft.org/v33/i01/
Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2011) *Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5) 1-13*
http://www.jstatsoft.org/v39/i05/
Porzelius, C., Binder, H., and Schumacher, M. (2009) *Parallelized prediction error estimation for evaluation of high-dimensional models, Bioinformatics, Vol. 25(6), 827-829.*

See Also

peperr, glmnet

PLL.coxnet *Predictive partial log-likelihood for glmnet Cox PH model fit*

Description

Extracts the predictive partial log-likelihood from a glmnet Cox PH model fit.

Usage

```
PLL.coxnet(object, newdata, newtime, newstatus, complexity, ...)
```

Arguments

`object`	fitted model of class `coxnet`.
`newdata`	`n_new*p` matrix of covariates.
`newtime`	`n_new`-vector of censored survival times.
`newstatus`	`n_new`-vector of survival status, coded with 0 and .1
`complexity`	lambda penalty value.
`...`	additional arguments, not used.

Details

Used by function `peperr`, if function `fit.glmnet` and `family="cox"` is used for model fit, which gives a class `coxnet` object. This is basically a wrapper based on the `coxnet.deviance` function from package `glmnet`.

Value

Vector of length `n_new`

Author(s)

Thomas Hielscher \ `<t.hielscher@dkfz.de>`

plot.peperr.curves *Plot method for prediction error curves of a peperr object*

Description

Plots individual and aggregated prediction error estimates based on bootstrap samples.

Usage

```
plot.peperr.curves(x, at.risk=TRUE, allErrors=FALSE,...)
```

Arguments

`x`	`peperr` object.
`at.risk`	number at risk to be display. default is TRUE.
`allErrors`	Display .632, no information and average out-of-bag error in addition. default is FALSE.
`...`	additional arguments, not used.

Details

This function is literally taken from `plot.peperr` in the `peperr` package. The display of prediction error curves is adapted to allow for numbers at risk.

Author(s)

Thomas Hielscher <`t.hielscher@dkfz.de`>

See Also

`peperr`

Examples

```
## Not run:

# example from glmnet package
set.seed(10101)
library(glmnet)
library(survival)
library(peperr)

N=1000;p=30
nzc=p/3
x=matrix(rnorm(N*p),N,p)
beta=rnorm(nzc)
fx=x[,seq(nzc)]
hx=exp(fx)
```

```
ty=rexp(N,hx)
tcens=rbinom(n=N,prob=.3,size=1)# censoring indicator
y=Surv(ty,1-tcens)

peperr.object <- peperr(response=y, x=x,
                        fit.fun=fit.glmnet, args.fit=list(family="cox"),
                        complexity=complexity.glmnet,
                        args.complexity=list(family="cox",nfolds=10),
                        indices=resample.indices(n=N, method="sub632", sample.n=10))

plot.peperr.curves(peperr.object)
# peperr
plot(peperr.object)

## End(Not run)
```

plot.summary.int.search

Plot Summary object for interval search models

Description

Produces a plot for summary object of a fitted interval search model. Plot 'visited' points against iteration steps. start.N points are initial points selected before interval search starts.

Usage

```
plot.summary.int.search(summary.int)
plot.points.int.search(summary.int, start.N=21)
```

Arguments

`fit`	an object as returned by the function `summary.int.search`.
`srat.N`	number of initial points, default 21.

Author(s)

Natalia Becker \ <natalia.becker@dkfz.de>

See Also

EPSGO, summary.int.search

`plotstabpath` *function to plot a stability path*

Description

Given a desired family-wise error rate (FWER) and a stability path calculated with `stability.path` the function selects an stable set of features and plots the stability path and the corresponding regularization path.

Usage

```
plotstabpath(x, fwer=0.5,pi_thr=0.6, xvar=c("lambda", "norm", "dev"), col.all="black", col.sel="red
```

Arguments

`x`	an object of class "stabpath" as returned by the function `stability.path`.
`fwer`	the desired family-wise error rate (fwer), e.g. the probability that at least one feature in the estimated set of stable features has been falsely selected.
`pi_thr`	the threshold used for the stability selection, should be in the range of $0.5 > pi_thr < 1$.
`xvar`	the variable used for the xaxis, e.g. for "lambda" the selection probabilities are plotted along the log of the penalization parameters, for "norm" along the L1-norm and for "dev" along the fraction of explained deviance.
`col.all`	the color used for the variables that are not in the estimated stable set
`col.sel`	the color used for the variables in the estimated stable set
`...`	further arguments that are passed to matplot

Value

a list of four objects

`stable`	a vector giving the positions of the estimated stable variables
`lambda`	the penalization parameter used for the stability selection
`lpos`	the position of the penalization parameter in the regularization path
`fwer`	the controlled family-wise error rate (fwer), e.g. the probability that at least one feature in the estimated set of stable features has been falsely selected.

Author(s)

Martin Sill \ `<m.sill@dkfz.de>`

References

Meinshausen N. and B\"uhlmann P. (2010), Stability Selection, Journal of the Royal Statistical Society: Series B (Statistical Methodology) Volume 72, Issue 4, pages 417???473.

See Also

`stability.selection`, `stability.path`

Examples

```
## Not run:
#gaussian
set.seed(1234)
x=matrix(rnorm(100*1000,0,1),100,1000)
y <- x[1:100,1:1000]%*%c(rep(2,5),rep(-2,5),rep(.1,990))
res <- stability.path(y,x,weakness=1,mc.cores=2)
plotstabpath(res,fwer=.5)

## End(Not run)
```

`predictProb.coxnet` *Extract predicted survival probabilities from a glmnet fit*

Description

Extracts predicted survival probabilities from survival model fitted by glmnet, providing an interface as required by `pmpec`.

Usage

```
predictProb.coxnet(object, x, times, complexity, ...)
```

Arguments

`object`	a fitted model of class `glmnet`.
`x`	`n*p` matrix of covariates.
`times`	vector of evaluation time points.
`complexity`	lambda penalty value.
`...`	additional arguments, currently not used.

Value

Matrix with probabilities for each evaluation time point in `times` (columns) and each new observation (rows).

Author(s)

Thomas Hielscher \ `<t.hielscher@dkfz.de>`

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*, http://www.stanford.edu/~hastie/Papers/glmnet.pdf
Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010
http://www.jstatsoft.org/v33/i01/
Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2011) *Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5) 1-13*
http://www.jstatsoft.org/v39/i05/
Porzelius, C., Binder, H., and Schumacher, M. (2009) *Parallelized prediction error estimation for evaluation of high-dimensional models, Bioinformatics, Vol. 25(6), 827-829.*

See Also

peperr, glmnet

---

predictProb.glmnet *Extract predicted survival probabilities from a glmnet fit*

Description

Extracts predicted survival probabilities from survival model fitted by glmnet, providing an interface as required by pmpec.

Usage

```
predictProb.glmnet(object, x, times, complexity, ...)
```

Arguments

object	a fitted model of class glmnet.
x	n*p matrix of covariates.
times	vector of evaluation time points.
complexity	lambda penalty value.
...	additional arguments, currently not used.

Value

Matrix with probabilities for each evaluation time point in times (columns) and each new observation (rows).

Author(s)

Thomas Hielscher \ <t.hielscher@dkfz.de>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*, http://www.stanford.edu/~hastie/Papers/glmnet.pdf
Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010
http://www.jstatsoft.org/v33/i01/
Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2011) *Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5) 1-13*
http://www.jstatsoft.org/v39/i05/
Porzelius, C., Binder, H., and Schumacher, M. (2009) *Parallelized prediction error estimation for evaluation of high-dimensional models, Bioinformatics, Vol. 25(6), 827-829.*

See Also

`peperr`, `glmnet`

`stability.path` *Stability path for glmnet models*

Description

The function calculates the stability path for glmnet models, e.g. the selection probabilities of the features along the range of regularization parameters.

Usage

```
stability.path(y,x,size=0.632,steps=100,weakness=1,mc.cores=getOption("mc.cores", 2L),...)
```

Arguments

`y`	response variable. Like for the glment function: Quantitative for `family="gaussian"` or `family="poisson"` (non-negative counts). For `family="binomial"` should be either a factor with two levels, or a two-column matrix of counts or proportions. For `family="multinomial"`, can be a `nc>=2` level factor, or a matrix with `nc` columns of counts or proportions. For `family="cox"`, `y` should be a two-column matrix with columns named 'time' and 'status'. The latter is a binary variable, with '1' indicating death, and '0' indicating right censored. The function `Surv()` in package `survival` produces such a matrix
`x`	input matrix. Like for the glmnet function: of dimension nobs x nvars; each row is an observation vector. Can be in sparse matrix format (inherit from class `"sparseMatrix"` as in package `Matrix`; not yet available for `family="cox"`)
`size`	proportion of samples drawn in every subsample used for the stability selection.
`steps`	number of subsamples used for the stability selection.
`weakness`	weakness parameter used for the randomised lasso as described in Meinshausen and B\"uhlmann (2010). For each subsample the features are reweighted by a random weight uniformly sampled in [weakness,1]. This additional randomisation leads to a more consistent estimation of the stable set of features.

`mc.cores`	number of cores used for the parallelization. For unix like system the parallelization is done by forking using the function `mclapply`. For windows systems socket cluster are used.
`...`	further arguments that are passed to the `glmnet` function.

Value

an object of class "stabpath", which is a list of three objects

`fit`	the fit object of class "glmnet" as returned from the glmnet function when applied to the complete data set.
`stabpath`	a matrix which represents the stability path.
`qs`	a vector holding the values of the average number of non-zero coefficients w.r.t to the lambdas in the regularization path.

Author(s)

Martin Sill <`m.sill@dkfz.de`>

References

Meinshausen N. and B\"uhlmann P. (2010), Stability Selection, Journal of the Royal Statistical Society: Series B (Statistical Methodology) Volume 72, Issue 4, pages 417???473.

See Also

`glmnet`,`stability.selection`,`plotstabpath`

Examples

```
## Not run:
#gaussian
set.seed(1234)
x <- matrix(rnorm(100*1000,0,1),100,1000)
y <- x[1:100,1:1000]%*% c(rep(2,5),rep(-2,5),rep(.1,990))
res <- stability.path(y,x,weakness=1,mc.cores=2)
plotstabpath(res)

#binomial
y=sample(1:2,100,replace=TRUE)
res <- stability.path(y,x,weakness=1,mc.cores=2,family="binomial")
plotstabpath(res)

#multinomial
y=sample(1:4,100,replace=TRUE)
res <- stability.path(y,x,weakness=1,mc.cores=2,family="multinomial")
plotstabpath(res)

#poisson
N=100; p=1000
nzc=5
```

```
x=matrix(rnorm(N*p),N,p)
beta=rnorm(nzc)
f = x[,seq(nzc)]%*%beta
mu=exp(f)
y=rpois(N,mu)
res <- stability.path(y,x,weakness=1,mc.cores=2,family="poisson")
plotstabpath(res)

#Cox
library(survival)
set.seed(10101)
N=100;p=1000
nzc=p/3
x=matrix(rnorm(N*p),N,p)
beta=rnorm(nzc)
fx=x[,seq(nzc)]%*%beta/3
hx=exp(fx)
ty=rexp(N,hx)
tcens=rbinom(n=N,prob=.3,size=1)
y=cbind(time=ty,status=1-tcens)
res <- stability.path(y,x,weakness=1,mc.cores=2,family="cox")
plotstabpath(res)

## End(Not run)
```

`stability.selection` *function to estimate a stable set of variables*

Description

Given a desired family-wise error rate (FWER) and a stability path calculated with `stability.path` the function selects a stable set of features.

Usage

```
stability.selection(x,fwer,pi_thr=0.6)
```

Arguments

`x`	an object of class "stabpath" as returned by the function `stability.path`.
`fwer`	the desired family-wise error rate (fwer), e.g. the probability that at least one feature in the estimated set of stable features has been falsely selected.
`pi_thr`	the threshold used for the stability selection, should be in the range of $0.5 > pi_thr < 1$.

Value

a list of four objects

`stable`	a vector giving the positions of the estimated stable variables
`lambda`	the penalization parameter used for the stability selection
`lpos`	the position of the penalization parameter in the regularization path
`fwer`	the controlled family-wise error rate (fwer), e.g. the probability that at least one feature in the estimated set of stable features has been falsely selected.

Author(s)

Martin Sill \ `<m.sill@dkfz.de>`

References

Meinshausen N. and B\"uhlmann P. (2010), Stability Selection, Journal of the Royal Statistical Society: Series B (Statistical Methodology) Volume 72, Issue 4, pages 417???473.

See Also

`plotstabpath`,`stability.path`

Examples

```
## Not run:
#gaussian
set.seed(1234)
x=matrix(rnorm(100*1000,0,1),100,1000)
y <- x[1:100,1:1000]%*%c(rep(2,5),rep(-2,5),rep(.1,990))
res <- stability.path(y,x,weakness=1,mc.cores=2)
stability.selection(res,fwer=.5)

## End(Not run)
```

`summary.int.search` *Summary method for interval search models*

Description

Produces a summary of a fitted interval search model

Usage

```
summary.int.search(fit,digits = max(3, getOption("digits") - 3), verbose=TRUE,first.n=5)
```

Arguments

`fit`	an object as returned by the function `EPSGO`.
`digits`	digits after the comma
`verbose`	default set to TRUE.
`first.n`	show first.n entries , default 5.

Value

A List of following ellements

`info`	a data frame of four objects for optimal models \itemalpha a vector of alphas \itemlambda a vector of penalization parameter lambda \itemdeviances a vector of deviances \itemn.features a vector of number of features selected in each optimal model
`opt.alpha`	an optimal value for alpha
`opt.lambda`	an optimal value for lambda
`opt.error`	an optimal value for alpha
`opt.alpha`	an optimal value for error, hier minimal diviance
`opt.models`	a list of optimal models with the same optimal error

Author(s)

Natalia Becker \ `<natalia.becker@dkfz.de>`

See Also

`EPSGO`

Examples

```
## Not run:
#gaussian
#set.seed(1234)
#x=matrix(rnorm(100*1000,0,1),100,1000)
#y <- x[1:100,1:1000] %*% matrix(c(rep(2,5),rep(-2,5),rep(.1,990)))
#y<-ifelse(y>0, "1","-1")

#bounds <- t(data.frame(alpha=c(0, 1)))
#colnames(bounds) <- c("lower", "upper")

#Q.func<- "tune.glmnet.interval"
#fit <- EPSGO(Q.func,bounds,parms.coding="none",x=x,y=y)
#summary.int.search(fit)

## End(Not run)
```

tune.glmnet.interval *Wrapper function for* glmnet *objects.*

Description

Wrapper function for glmnet objects used by EPSGO function. This function is mainly used within the function EPSGO

Usage

```
tune.glmnet.interval(parms, x, y,
                     weights,
                     offset = NULL,
                     lambda = NULL,
                     type.measure = c("mse", "deviance", "class", "auc", "mae"),
                     seed=12345,
                     nfolds = 10,
                     foldid=NULL,
                     grouped = TRUE,
                     type.min=c("lambda.min", "lambda.1se"),
                     verbose=FALSE,
                     ...)
```

Arguments

parms	tuning parameter alpha for glmnet object
x,y	x is a matrix where each row refers to a sample a each column refers to a gene; y is a factor which includes the class for each sample
weights	observation weights. Can be total counts if responses are proportion matrices. Default is 1 for each observation
offset	A vector of length nobs that is included in the linear predictor (a nobs x nc matrix for the "multinomial" family). Useful for the "poisson" family (e.g. log of exposure time), or for refining a model by starting at a current fit. Default is NULL. If supplied, then values must also be supplied to the predict function.
lambda	A user supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Supplying a value of lambda overrides this. WARNING: use with care. Do not supply a single value for lambda (for predictions after CV use predict() instead). Supply instead a decreasing sequence of lambda values. glmnet relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.
type.measure	loss to use for cross-validation. Currently five options, not all available for all models. The default is type.measure="deviance", which uses squared-error for gaussian models (a.k.a type.measure="mse" there), deviance for logistic and poisson regression, and partial-likelihood for the Cox model. type.measure="class" applies to binomial and multinomial logistic regression only, and gives misclassification error. type.measure="auc" is for two-class logistic regression only, and

	gives area under the ROC curve. type.measure="mse" or type.measure="mae" (mean absolute error) can be used by all models except the "cox"; they measure the deviation from the fitted mean to the response.
`seed`	seed
`nfolds`	number of cross-validation's folds, default 10.
`foldid`	an optional vector of values between 1 and nfold identifying what fold each observation is in. If supplied, nfold can be missing.
`grouped`	This is an experimental argument, with default TRUE, and can be ignored by most users. For all models except the "cox", this refers to computing nfolds separate statistics, and then using their mean and estimated standard error to describe the CV curve. If grouped=FALSE, an error matrix is built up at the observation level from the predictions from the nfold fits, and then summarized (does not apply to type.measure="auc"). For the "cox" family, grouped=TRUE obtains the CV partial likelihood for the Kth fold by subtraction; by subtracting the log partial likelihood evaluated on the full dataset from that evaluated on the on the (K-1)/K dataset. This makes more efficient use of risk sets. With grouped=FALSE the log partial likelihood is computed only on the Kth fold
`type.min`	parameter for chosing optimal model: 'lambda.min'- value of lambda that gives minimum mean cross-validated error (cvm). 'lambda.1se' - largest value of lambda such that error is within one standard error of the minimum.
`verbose`	verbose
`...`	Further parameters

Value

`q.val` minimal value of Q.func on the interval defined by bounds. Here, q.val is minimum mean cross-validate d error (cvm)

`model` model list

- alpha - optimal alpha
- lambda - optimal lambda
- cvreg - `cv.glmnet` object for optimal alpha
- fit - `glmnet` object for optimal alpha and optimal lambda

Author(s)

Natalia Becker natalia.becker at dkfz.de

See Also

`EPSGO`

Index