Package ‘grpreg’

September 27, 2018

Title Regularization Paths for Regression Models with Grouped Covariates

Version 3.2-0

Date 2018-09-27

Depends R (>= 3.1.0)

Imports Matrix

Suggests grpregOverlap, knitr, survival

VignetteBuilder knitr

Description Efficient algorithms for fitting the regularization path of linear regression, GLM, and Cox regression models with grouped penalties. This includes group selection methods such as group lasso, group MCP, and group SCAD as well as bi-level selection methods such as the group exponential lasso, the composite MCP, and the group bridge.

BugReports http://github.com/pbreheny/grpreg/issues

License GPL-3

URL http://pbreheny.github.io/grpreg,
    https://github.com/pbreheny/grpreg

LazyData TRUE

RoxygenNote 6.1.0

NeedsCompilation yes

Author Patrick Breheny [aut, cre] (<https://orcid.org/000-0002-0650-1119>), Yaohui Zeng [ctb]

Maintainer Patrick Breheny <patrick-breheny@uiowa.edu>

Repository CRAN

Date/Publication 2018-09-27 14:40:11 UTC
Regularization paths for regression models with grouped covariates

Description

This package fits regularization paths for linear, logistic, and Cox regression models with grouped penalties, such as the group lasso, group MCP, group SCAD, group exponential lasso, and group bridge. The algorithms are based on the idea of either locally approximated coordinate descent or group descent, depending on the penalty. All of the algorithms (with the exception of group bridge) are stable and fast.

Details

Given a design matrix $X$ in which the features consist of non-overlapping groups and vector of responses $y$, `grpreg` solves the regularization path for a variety of penalties. The package also provides methods for plotting and cross-validation.

See the "Getting started" vignette for a brief overview of how the package works.

The following penalties are available:

- `gLasso`: Group lasso (Yuan and Lin, 2006)
- `gMCP`: Group MCP; like the group lasso, but with an MCP penalty on the norm of each group
- `gSCAD`: Group SCAD; like the group lasso, but with a SCAD penalty on the norm of each group
- `cMCP`: A hierarchical penalty which places an outer MCP penalty on a sum of inner MCP penalties for each group (Breheny & Huang, 2009)
• gel: Group exponential lasso (Breheny, 2015)
• gBridge: A penalty which places a bridge penalty on the L1-norm of each group (Huang et al., 2009)

The cmcp, gel, and gBridge penalties carry out bi-level selection, meaning that they carry out variable selection at the group level and at the level of individual covariates (i.e., they select important groups as well as important members of those groups). The grLasso, grMCP, and grSCAD penalties carry out group selection, meaning that within a group, coefficients will either all be zero or all nonzero. A variety of supporting methods for selecting lambda and plotting the paths are provided also.

For group penalization involving overlapping groups, see the companion package grpregOverlap.

Author(s)
Patrick Breheny

References

Examples
vignette("getting-started", "grpreg")

AUC.cv.grpsurv Calculates AUC for cv.grpsurv objects

Description
Calculates the cross-validated AUC (concordance) from a "cv.grpsurv" object.
usage

## S3 method for class 'cv.grpsurv'
AUC(obj, ...)

Arguments

obj A cv.grpsurv object. You must run cv.grpsurv with the option returnY=TRUE in order for AUC to work.

... For S3 method compatibility.

details

The area under the curve (AUC), or equivalently, the concordance statistic (C), is calculated according to the procedure outlined in the reference below. This calls the survConcordance function in the survival package, except the cross-validated linear predictors are used to guard against overfitting. Thus, the values returned by AUC.cv.grpsurv will be lower than those you would obtain with survConcordance if you fit the full (unpenalized) model.

Author(s)

Patrick Breheny

References


See Also

cv.grpsurv, survConcordance

Examples

data(Lung)
X <- Lung$X
y <- Lung$y
group <- Lung$group

cvfit <- cv.grpsurv(X, y, group, returnY=TRUE)
head(AUC(cvfit))
ll <- log(cvfit$fit$lambda)
plot(ll, AUC(cvfit), xlab=rev(range(ll)), lwd=3, type='l',
     xlab=expression(log(lambda)), ylab='AUC', las=1)
Description

The Birthwt data contains 189 observations, 16 predictors, and an outcome, birthweight, available both as a continuous measure and a binary indicator for low birth weight. The data were collected at Baystate Medical Center, Springfield, Mass during 1986. This data frame is a reparameterization of the birthwt data frame from the MASS package.

Usage

data(Birthwt)

Format

The Birthwt object is a list containing four elements:

- **bwt**: Birth weight in kilograms
- **low**: Indicator of birth weight less than 2.5kg
- **X**: Matrix of predictors
- **group**: Vector describing how the columns of X are grouped

The matrix X contains the following columns:

- **age1, age2, age3**: Orthogonal polynomials of first, second, and third degree representing mother’s age in years
- **lwt1, lwt2, lwt3**: Orthogonal polynomials of first, second, and third degree representing mother’s weight in pounds at last menstrual period
- **white, black**: Indicator functions for mother’s race; "other" is reference group
- **smoke**: Smoking status during pregnancy
- **ptl1, ptl2, ptl3**: Indicator functions for one or for two or more previous premature labors, respectively. No previous premature labors is the reference category.
- **ht**: History of hypertension
- **ui**: Presence of uterine irritability
- **ftv1, ftv2, ftv3**: Indicator functions for one, for two, or for three or more physician visits during the first trimester, respectively. No visits is the reference category.

Source

MASS. R package. [https://cran.r-project.org/package=MASS](https://cran.r-project.org/package=MASS)

References

See Also

birthwt, grpreg

Examples

data(Birthwt)
hist(Birthwt$bwt, xlab="Child's birth weight", main=""

See examples in ?birthwt (MASS package)

See examples in ?grpreg for use of this data set

See also

birthwt

Description

This version of the data set has been deprecated and will not be supported in future versions. Please use Birthwt instead.

Usage

data(birthwt.grpreg)

Format

This data frame contains the following columns:

- `low` Indicator of birth weight less than 2.5kg
- `bwt` Birth weight in kilograms
- `age1`, `age2`, `age3` Orthogonal polynomials of first, second, and third degree representing mother’s age in years
- `lwt1`, `lwt2`, `lwt3` Orthogonal polynomials of first, second, and third degree representing mother’s weight in pounds at last menstrual period
- `white`, `black` Indicator functions for mother’s race; "other" is reference group
- `smoke` smoking status during pregnancy
- `ptl1`, `ptl2m` Indicator functions for one or for two or more previous premature labors, respectively. No previous premature labors is the reference category.
- `ht` History of hypertension
- `ui` Presence of uterine irritability
- `ftv1`, `ftv2`, `ftv3m` Indicator functions for one, for two, or for three or more physician visits during the first trimester, respectively. No visits is the reference category.

See Also

Birthwt
cv.grpreg

Cross-validation for grpreg/grpsurv

Description

Performs k-fold cross validation for penalized regression models with grouped covariates over a grid of values for the regularization parameter lambda.

Usage

    cv.grpreg(X, y, group=1:ncol(X), ..., nfolds=10, seed, fold, returnY=FALSE, trace=FALSE)
    cv.grpsurv(X, y, group, ..., nfolds=10, seed, fold, se=c('quick', 'bootstrap'), returnY=FALSE, trace=FALSE)

Arguments

- **X**: The design matrix, as in grpreg/grpsurv.
- **y**: The response vector (or matrix), as in grpreg/grpsurv.
- **group**: The grouping vector, as in grpreg/grpsurv.
- **...**: Additional arguments to grpreg/grpsurv.
- **nfolds**: The number of cross-validation folds. Default is 10.
- **seed**: You may set the seed of the random number generator in order to obtain reproducible results.
- **fold**: Which fold each observation belongs to. By default the observations are randomly assigned.
- **returnY**: Should cv.grpreg/grpsurv return the fitted values from the cross-validation folds? Default is FALSE; if TRUE, this will return a matrix in which the element for row i, column j is the fitted value for observation i from the fold in which observation i was excluded from the fit, at the jth value of lambda. NOTE: For cv.grpsurv, the rows of Y are ordered by time on study, and therefore will not correspond to the original order of observations passed to cv.grpsurv.
- **trace**: If set to TRUE, cv.grpreg will inform the user of its progress by announcing the beginning of each CV fold. Default is FALSE.
- **se**: For cv.grpsurv, the method by which the cross-validation standard error (CVSE) is calculated. The 'quick' approach is based on a rough approximation, but can be calculated more or less instantly. The 'bootstrap' approach is more accurate, but requires additional computing time.

Details

The function calls grpreg/cv.grpsurv nfolds times, each time leaving out 1/nfolds of the data. The cross-validation error is based on the deviance; see here for more details.
For Gaussian and Poisson responses, the folds are chosen according to simple random sampling. For binomial responses, the numbers for each outcome class are balanced across the folds; i.e., the number of outcomes in which \( y \) is equal to 1 is the same for each fold, or possibly off by 1 if the numbers do not divide evenly. This approach is used for Cox regression as well to balance the amount of censoring cross each fold.

For Cox models, \texttt{cv.grpsurv} uses the approach of calculating the full Cox partial likelihood using the cross-validated set of linear predictors. Other approaches to cross-validation for the Cox regression model have been proposed in the literature; the strengths and weaknesses of the various methods for penalized regression in the Cox model are the subject of current research. A simple approximation to the standard error is provided, although an option to bootstrap the standard error (\texttt{se='bootstrap'}) is also available.

As in \texttt{grpreg}, seemingly unrelated regressions/multitask learning can be carried out by setting \( y \) to be a matrix, in which case groups are set up automatically (see \texttt{grpreg} for details), and cross-validation is carried out with respect to rows of \( y \). As mentioned in the details there, it is recommended to standardize the responses prior to fitting.

**Value**

An object with S3 class "\texttt{cv.grpreg}" containing:

- \texttt{cve} The error for each value of \( \lambda \), averaged across the cross-validation folds.
- \texttt{cvse} The estimated standard error associated with each value of \( \lambda \).
- \texttt{lambda} The sequence of regularization parameter values along which the cross-validation error was calculated.
- \texttt{fit} The fitted \texttt{grpreg} object for the whole data.
- \texttt{fold} The fold assignments for cross-validation for each observation; note that for \texttt{cv.grpsurv}, these are in terms of the ordered observations, not the original observations.
- \texttt{min} The index of \( \lambda \) corresponding to \( \lambda_{\text{min}} \).
- \texttt{lambda.min} The value of \( \lambda \) with the minimum cross-validation error.
- \texttt{null.dev} The deviance for the intercept-only model.
- \texttt{pe} If \texttt{family="binomial"}, the cross-validation prediction error for each value of \( \lambda \).

**Author(s)**

Patrick Breheny

**See Also**

\texttt{grpreg, plot.cv.grpreg, summary.cv.grpreg, predict.cv.grpreg}
gBridge  

**Examples**

```r
data(Birthwt)
X <- Birthwt$X
y <- Birthwt$bwt
group <- Birthwt$group

cvfit <- cv.grpreg(X, y, group)
plot(cvfit)
summary(cvfit)
coef(cvfit) ## Beta at minimum CVE

cvfit <- cv.grpreg(X, y, group, penalty="gel")
plot(cvfit)
summary(cvfit)
```

---

**gBridge**  

*Fit a group bridge regression path*

---

**Description**

Fit regularization paths for linear and logistic group bridge-penalized regression models over a grid of values for the regularization parameter lambda.

**Usage**

```r
 gBridge(X, y, group=1:ncol(X), family=c("gaussian", "binomial", "poisson"), nlambda=100, lambda, lambda.min=(if (nrow(X) > ncol(X)) .001 else .05), lambda.max, alpha=1, eps=.001, delta=1e-7, max.iter=10000, gamma=0.5, group.multiplier, warn=TRUE)
```

**Arguments**

- **X**  
The design matrix, as in grpreg.
- **y**  
The response vector (or matrix), as in grpreg.
- **group**  
The grouping vector, as in grpreg.
- **family**  
Either "gaussian" or "binomial", depending on the response.
- **nlambda**  
The number of lambda values, as in grpreg.
- **lambda**  
A user supplied sequence of lambda values, as in grpreg.
- **lambda.min**  
The smallest value for lambda, as in grpreg.
- **lambda.max**  
The maximum value for lambda. Unlike the penalties in grpreg, it is not possible to solve for lambda.max directly with group bridge models. Thus, it must be specified by the user. If it is not specified, gBridge will attempt a guess at lambda.max, but this is not particularly accurate.
- **alpha**  
Tuning parameter for the balance between the group penalty and the L2 penalty, as in grpreg.
eps
Convergence threshold, as in grpreg.

delta
The group bridge penalty is not differentiable at zero, and requires a small number delta to bound it away from zero. There is typically no need to change this value.

max.iter
Maximum number of iterations, as in grpreg.

gamma
Tuning parameter of the group bridge penalty (the exponent to which the L1 norm of the coefficients in the group are raised). Default is 0.5, the square root.

group.multiplier
The multiplicative factor by which each group’s penalty is to be multiplied, as in grpreg.

warn
Should the function give a warning if it fails to converge? As in grpreg.

Details
This method fits the group bridge method of Huang et al. (2009). Unlike the penalties in grpreg, the group bridge is not differentiable at zero; because of this, a number of changes must be made to the algorithm, which is why it has its own function. Most notably, the method is unable to start at lambda.max; it must start at lambda.max and proceed in the opposite direction.

In other respects, the usage and behavior of the function is similar to the rest of the grpreg package.

Value
An object with S3 class "gBreg", as in grpreg.

Author(s)
Patrick Breheny

References


See Also
gpreg

Examples
data(Birthwt)
X <- Birthwt$X
group <- Birthwt$group

## Linear regression
y <- Birthwt$bw
fit <- gBridge(X, y, group)
grpreg

Plot the fitted values
select(fit)

## Logistic regression
y <- Birthwt$low
fit <- gBridge(X, y, group, family="binomial")
plot(fit)
select(fit)

---

grpreg

Fit a group penalized regression path

Description

Fit regularization paths for models with grouped penalties over a grid of values for the regularization parameter lambda. Fits linear and logistic regression models.

Usage

```
grpreg(X, y, group=1:ncol(X), penalty=c("grLasso", "grMCP", "grSCAD", "gel", "cMCP"), family=c("gaussian", "binomial", "poisson"), nlambda=100, lambda, lambda.min={if (nrow(X) > ncol(X)) 1e-4 else .05}, log.lambda = TRUE, alpha=1, eps=1e-4, max.iter=10000, dfmax=p, gmax=length(unique(group)), gamma=ifelse(penalty == "grSCAD", 4, 3), tau = 1/3, group.multiplier, warn=TRUE, returnX = FALSE, ...)```

Arguments

- **x**  
The design matrix, without an intercept. `grpreg` standardizes the data and includes an intercept by default.

- **y**  
The response vector, or a matrix in the case of multitask learning (see details).

- **group**  
A vector describing the grouping of the coefficients. For greatest efficiency and least ambiguity (see details), it is best if `group` is a factor or vector of consecutive integers, although unordered groups and character vectors are also allowed. If there are coefficients to be included in the model without being penalized, assign them to group 0 (or `0`).

- **penalty**  
The penalty to be applied to the model. For group selection, one of `grLasso`, `grMCP`, or `grSCAD`. For bi-level selection, one of `gel` or `cMCP`. See below for details.

- **family**  
Either "gaussian" or "binomial", depending on the response.

- **nlambda**  
The number of lambda values. Default is 100.

- **lambda**  
A user supplied sequence of lambda values. Typically, this is left unspecified, and the function automatically computes a grid of lambda values that ranges uniformly on the log scale over the relevant range of lambda values.
lambda.min
  The smallest value for lambda, as a fraction of lambda.max. Default is .0001
  if the number of observations is larger than the number of covariates and .05
  otherwise.

log.lambda
  Whether compute the grid values of lambda on log scale (default) or linear scale.

alpha
  grpreg allows for both a group penalty and an L2 (ridge) penalty; alpha con-
  trols the proportional weight of the regularization parameters of these two penal-
  ties. The group penalties’ regularization parameter is lambda*alpha, while the
  regularization parameter of the ridge penalty is lambda*(1-alpha). Default is
  1: no ridge penalty.

eps
  Convergence threshhold. The algorithm iterates until the RMSD for the change
  in linear predictors for each coefficient is less than eps. Default is 1e-4. See
  details.

max.iter
  Maximum number of iterations (total across entire path). Default is 10000. See
  details.

dfmax
  Limit on the number of parameters allowed to be nonzero. If this limit is ex-
  ceeded, the algorithm will exit early from the regularization path.

Gmax
  Limit on the number of groups allowed to have nonzero elements. If this limit
  is exceeded, the algorithm will exit early from the regularization path.

gamma
  Tuning parameter of the group or composite MCP/SCAD penalty (see details).
  Default is 3 for MCP and 4 for SCAD.

tau
  Tuning parameter for the group exponential lasso; defaults to 1/3.

group.multiplier
  A vector of values representing multiplicative factors by which each group’s
  penalty is to be multiplied. Often, this is a function (such as the square root)
  of the number of predictors in each group. The default is to use the square
  root of group size for the group selection methods, and a vector of 1’s (i.e., no
  adjustment for group size) for bi-level selection.

warn
  Should the function give a warning if it fails to converge? Default is TRUE. See
  details.

returnX
  Return the standardized design matrix (and associated group structure informa-
  tion)? Default is FALSE.

...  Arguments passed to other functions (such as gBridge).

Details

There are two general classes of methods involving grouped penalties: those that carry out bi-level
selection and those that carry out group selection. Bi-level means carrying out variable selection
at the group level as well as the level of individual covariates (i.e., selecting important groups as
well as important members of those groups). Group selection selects important groups, and not
members within the group – i.e., within a group, coefficients will either all be zero or all nonzero.
The grLasso, grMCP, and grSCAD penalties carry out group selection, while the gel and cmCP
penalties carry out bi-level selection. For bi-level selection, see also the gBridge function. For
historical reasons and backwards compatibility, some of these penalties have aliases; e.g., gLasso
will do the same thing as grLasso, but users are encouraged to use grLasso.
Please note the distinction between grMCP and cMCP. The former involves an MCP penalty being applied to an L2-norm of each group. The latter involves a hierarchical penalty which places an outer MCP penalty on a sum of inner MCP penalties for each group, as proposed in Breheny & Huang, 2009. Either penalty may be referred to as the "group MCP", depending on the publication. To resolve this confusion, Huang et al. (2012) proposed the name "composite MCP" for the cMCP penalty.

For more information about the penalties and their properties, please consult the references below, many of which contain discussion, case studies, and simulation studies comparing the methods. If you use grpreg for an analysis, please cite the appropriate reference.

In keeping with the notation from the original MCP paper, the tuning parameter of the MCP penalty is denoted 'gamma'. Note, however, that in Breheny and Huang (2009), gamma is denoted 'a'.

The objective function for grpreg optimization is defined to be

\[ Q(\beta|X,y) = \frac{1}{n} L(\beta|X,y) + P_\lambda(\beta) \]

where the loss function \( L \) is the deviance (-2 times the log likelihood) for the specified outcome distribution (gaussian/binomial/poisson). See here for more details.

For the bi-level selection methods, a locally approximated coordinate descent algorithm is employed. For the group selection methods, group descent algorithms are employed.

The algorithms employed by grpreg are stable and generally converge quite rapidly to values close to the solution. However, especially when \( p \) is large compared with \( n \), grpreg may fail to converge at low values of \( \lambda \), where models are nonidentifiable or nearly singular. Often, this is not the region of the coefficient path that is most interesting. The default behavior warning the user when convergence criteria are not met may be distracting in these cases, and can be modified with \texttt{warn} (convergence can always be checked later by inspecting the value of \texttt{iter}).

If models are not converging, increasing \texttt{max.iter} may not be the most efficient way to correct this problem. Consider increasing \texttt{n.lambda} or \texttt{lambda.min} in addition to increasing \texttt{max.iter}.

Although grpreg allows groups to be unordered and given arbitrary names, it is recommended that you specify groups as consecutive integers. The first reason is efficiency: if groups are out of order, \( X \) must be reordered prior to fitting, then this process reversed to return coefficients according to the original order of \( X \). This is inefficient if \( X \) is very large. The second reason is ambiguity with respect to other arguments such as \texttt{group.multiplier}. With consecutive integers, \texttt{group=3} unambiguously denotes the third element of \texttt{group.multiplier}.

Seemingly unrelated regressions/multitask learning can be carried out using grpreg by passing a matrix to \( y \). In this case, \( X \) will be used in separate regressions for each column of \( y \), with the coefficients grouped across the responses. In other words, each column of \( X \) will form a group with \( m \) members, where \( m \) is the number of columns of \( y \). For multiple Gaussian responses, it is recommended to standardize the columns of \( y \) prior to fitting, in order to apply the penalization equally across columns.

grpreg requires groups to be non-overlapping; for model fitting involving overlapping groups, see the companion package grpregOverlap.

\textbf{Value}

An object with S3 class "grpreg" containing:
beta  The fitted matrix of coefficients. The number of rows is equal to the number of coefficients, and the number of columns is equal to nlambda.

family  Same as above.

group  Same as above.

lambda  The sequence of lambda values in the path.

alpha  Same as above.

loss  A vector containing either the residual sum of squares ("gaussian") or negative log-likelihood ("binomial") of the fitted model at each value of lambda.

n  Number of observations.

penalty  Same as above.

df  A vector of length nlambda containing estimates of effective number of model parameters all the points along the regularization path. For details on how this is calculated, see Breheny and Huang (2009).

iter  A vector of length nlambda containing the number of iterations until convergence at each value of lambda.

group.multiplier  A named vector containing the multiplicative constant applied to each group's penalty.

Author(s)

Patrick Breheny

References


See Also

cv.grpreg, as well as plot and select methods.

Examples

# Birthweight data
data(Birthwt)
X <- Birthwt$X
group <- Birthwt$group
# Linear regression
y <- Birthwt$bwt
fit <- grpreg(X, y, group, penalty="grLasso")
plot(fit)
fit <- grpreg(X, y, group, penalty="grMCP")
plot(fit)
fit <- grpreg(X, y, group, penalty="grSCAD")
plot(fit)
fit <- grpreg(X, y, group, penalty="gel")
plot(fit)
fit <- grpreg(X, y, group, penalty="cMCP")
plot(fit)
select(fit, "AIC")

# Logistic regression
y <- Birthwt$low
fit <- grpreg(X, y, group, penalty="grLasso", family="binomial")
plot(fit)
fit <- grpreg(X, y, group, penalty="grMCP", family="binomial")
plot(fit)
fit <- grpreg(X, y, group, penalty="grSCAD", family="binomial")
plot(fit)
fit <- grpreg(X, y, group, penalty="gel", family="binomial")
plot(fit)
fit <- grpreg(X, y, group, penalty="cMCP", family="binomial")
plot(fit)
select(fit, "BIC")

# Multitask learning (simulated example)
set.seed(1)
n <- 50
p <- 10
k <- 5
X <- matrix(runif(n*p), n, p)
y <- matrix(rnorm(n*k, X[,1] + X[,2]), n, k)
fit <- grpreg(X, y)
# Note that group is set up automatically
fit$group
plot(fit)

# Overlapping groups
## Not run:
library(grpregOverlap)
data(pathway.dat)
X <- pathway.dat$expression
group <- pathway.dat$pathways
y <- pathway.dat$mutation
fit <- grpregOverlap(X, y, group, family='binomial')
plot(fit)
## End(Not run)
Fit an group penalized survival model

Description

Fit regularization paths for Cox models with grouped penalties over a grid of values for the regularization parameter lambda.

Usage

```r
grpsurv(X, y, group=1:ncol(X), penalty=c("grLasso", "grMCP", "grSCAD", "gel", "cMCP"), gamma=ifelse(penalty=="grSCAD", 4, 3), alpha=1, nlambda=100, lambda, lambda.min=(if (nrow(X) > ncol(X)) 0.001 else .05), eps=.001, max.iter=10000, dfmax=p, gmax=length(unique(group)), tau=1/3, group.multiplier, warn=TRUE, returnX=FALSE, ...)
```

Arguments

- `X` The design matrix.
- `y` The time-to-event outcome, as a two-column matrix or `Surv` object. The first column should be time on study (follow up time); the second column should be a binary variable with 1 indicating that the event has occurred and 0 indicating (right) censoring.
- `group` A vector describing the grouping of the coefficients. For greatest efficiency and least ambiguity (see details), it is best if `group` is a factor or vector of consecutive integers, although unordered groups and character vectors are also allowed. If there are coefficients to be included in the model without being penalized, assign them to group 0 (or `BPB`).
- `penalty` The penalty to be applied to the model. For group selection, one of `grLasso`, `grMCP`, or `grSCAD`. For bi-level selection, one of `gel` or `cmcp`. See below for details.
- `gamma` Tuning parameter of the group or composite MCP/SCAD penalty (see details). Default is 3 for MCP and 4 for SCAD.
- `alpha` `grpsurv` allows for both a group penalty and an L2 (ridge) penalty; `alpha` controls the proportional weight of the regularization parameters of these two penalties. The group penalties' regularization parameter is `lambda*alpha`, while the regularization parameter of the ridge penalty is `lambda*(1-alpha)`. Default is 1: no ridge penalty.
- `nlambda` The number of lambda values. Default is 100.
- `lambda.min` The smallest value for lambda, as a fraction of lambda.max. Default is .001 if the number of observations is larger than the number of covariates and .05 otherwise.
- `lambda` A user-specified sequence of lambda values. By default, a sequence of values of length `nlambda` is computed automatically, equally spaced on the log scale.
eps
Convergence threshold. The algorithm iterates until the RMSD for the change in linear predictors for each coefficient is less than eps. Default is 0.001.

max.iter
Maximum number of iterations (total across entire path). Default is 10000.

dfmax
Limit on the number of parameters allowed to be nonzero. If this limit is exceeded, the algorithm will exit early from the regularization path.

gmax
Limit on the number of groups allowed to have nonzero elements. If this limit is exceeded, the algorithm will exit early from the regularization path.

tau
Tuning parameter for the group exponential lasso; defaults to 1/3.

group.multiplier
A vector of values representing multiplicative factors by which each group’s penalty is to be multiplied. Often, this is a function (such as the square root) of the number of predictors in each group. The default is to use the square root of group size for the group selection methods, and a vector of 1’s (i.e., no adjustment for group size) for bi-level selection.

warn
Return warning messages for failures to converge and model saturation? Default is TRUE.

returnX
Return the standardized design matrix? Default is FALSE.

Details
The sequence of models indexed by the regularization parameter lambda is fit using a coordinate descent algorithm. In order to accomplish this, the second derivative (Hessian) of the Cox partial log-likelihood is diagonalized (see references for details). The objective function is defined to be

\[ Q(\beta |X, y) = \frac{1}{n}L(\beta |X, y) + P(\lambda(\beta)) \]

where the loss function L is the deviance (-2 times the partial log-likelihood) from the Cox regression mode. See here for more details.

Presently, ties are not handled by grpsurv in a particularly sophisticated manner. This will be improved upon in a future release of grpreg.

Value
An object with S3 class "grpsurv" containing:

beta
The fitted matrix of coefficients. The number of rows is equal to the number of coefficients, and the number of columns is equal to nlambda.

group
Same as above.

lambda
The sequence of lambda values in the path.

penalty
Same as above.

gamma
Same as above.

alpha
Same as above.

loss
The negative partial log-likelihood of the fitted model at each value of lambda.
n
The number of observations.

df
A vector of length nlambda containing estimates of effective number of model parameters all the points along the regularization path. For details on how this is calculated, see Breheny and Huang (2009).

iter
A vector of length nlambda containing the number of iterations until convergence at each value of lambda.

group.multiplier
A named vector containing the multiplicative constant applied to each group’s penalty.

For Cox models, the following objects are also returned (and are necessary to estimate baseline survival conditional on the estimated regression coefficients), all of which are ordered by time on study. I.e., the ith row of W does not correspond to the ith row of X):

W
Matrix of exp(betahat) values for each subject over all lambda values.

time
Times on study.

fail
Failure event indicator.

Author(s)

Patrick Breheny

References


See Also

plot.grpreg, predict.grpsurv, cv.grpsurv,

Examples

data(Lung)
X <- Lung$X
y <- Lung$y
group <- Lung$group
logLik.grpreg

Description
Calculates the log likelihood and degrees of freedom for a fitted grpreg object.

Usage

## S3 method for class 'grpreg'
logLik(object, df.method=c("default","active"),
       REML=FALSE, ...)

Arguments

object A fitted grpreg object.
df.method How should effective model parameters be calculated? One of: "active", which counts the number of nonzero coefficients; or "default", which uses the calculated df returned by grpreg. Default is "default".
REML Use restricted MLE for estimation of the scale parameter in a gaussian model? Default is FALSE.
... For S3 method compatibility.

Details

Exists mainly for use with 'AIC' and 'BIC'.

Value

Returns an object of class 'logLik', in this case consisting of a number (or vector of numbers) with two attributes: 'df' (the estimated degrees of freedom in the model) and 'nobs' (number of observations).

The 'print' method for 'logLik' objects is not intended to handle vectors; consequently, the value of the function does not necessarily display correctly. However, it works with 'AIC' and 'BIC' without any glitches and returns the expected vectorized output.

Author(s)

Patrick Breheny
See Also
gpreg

Examples
data(Birthwt)  
X <- Birthwt$X  
y <- Birthwt$bwt  
group <- Birthwt$group  
fit <- grpreg(X, y, group, penalty = "cMCP")  
logLik(fit)  
## Display is glitchy for vectors  
AIC(fit)  
BIC(fit)

Lung

VA lung cancer data set

Description

Data from a randomised trial of two treatment regimens for lung cancer. This is a standard survival analysis data set from the classic textbook by Kalbfleisch and Prentice.

Usage
data(Lung)

Format

The Lung object is a list containing three elements:

- **y**: A two column matrix (Surv object) containing the follow-up time (in days) and an indicator variable for whether the patient died while on the study or not.
- **X**: Matrix of predictors
- **group**: Vector describing how the columns of X are grouped

The matrix X contains the following columns:

- **trt**: Treatment indicator (1=control group, 2=treatment group)
- **karno1, karno2, karno3**: Orthogonal polynomials of first, second, and third degree representing Karnofsky performance score (0=bad, 100=good)
- **diagtime1, diagtime2**: Orthogonal polynomials of first and second degree representing time from diagnosis to randomization (months)
- **age1, age2, age3**: Orthogonal polynomials of first, second, and third degree representing the patient’s age in years
- **prior**: Prior therapy (0=no, 1=yes)
- **squamous, small, adeno, large**: Indicators for the lung cancer cell type. For each subject, exactly one of these columns will be 1 and the rest 0.
Source

https://cran.r-project.org/package=survival

References


See Also

veteran, grpsurv

Examples

```r
data(Lung)
hist(Lung$y[, 1], xlab="Follow-up time", main="")
table(Lung$y[, 2])
```

---

plot.cv.grpreg  
*Plots the cross-validation curve from a cv.grpreg object*

Description

Plots the cross-validation curve from a cv.grpreg object, along with standard error bars.

Usage

```r
# S3 method for class 'cv.grpreg'
plot(x, log.l=TRUE, type=c("cve", "rsq", "scale", "snr", "pred", "all"), selected=TRUE, vertical.line=TRUE, col="red", ...)
```

Arguments

- `x`: A cv.grpreg object.
- `log.l`: Should horizontal axis be on the log scale? Default is TRUE.
- `type`: What to plot on the vertical axis. cve plots the cross-validation error (deviance); rsq plots an estimate of the fraction of the deviance explained by the model (R-squared); snr plots an estimate of the signal-to-noise ratio; scale plots, for family="gaussian", an estimate of the scale parameter (standard deviation); pred plots, for family="binomial", the estimated prediction error; all produces all of the above.
- `selected`: If TRUE (the default), places an axis on top of the plot denoting the number of groups in the model (i.e., that contain a nonzero regression coefficient) at that value of lambda.
- `vertical.line`: If TRUE (the default), draws a vertical line at the value where cross-validation error is minimized.
plot.grpreg

### Description

Produces a plot of the coefficient paths for a fitted grpreg object.

### Usage

```r
## S3 method for class 'grpreg'
plot(x, alpha=1, legend.loc, label=FALSE, log.l=FALSE, norm=FALSE, ...)
```
plot.grpreg

**Arguments**

- **x**: Fitted "grpreg" model.
- **alpha**: Controls alpha-blending. Default is alpha=1.
- **legend.loc**: Where should the legend go? If left unspecified, no legend is drawn. See `legend` for details.
- **label**: If TRUE, annotates the plot with text labels in the right margin describing which variable/group the corresponding line belongs to.
- **log.l**: Should horizontal axis be on the log scale? Default is FALSE.
- **norm**: If TRUE, plot the norm of each group, rather than the individual coefficients.
- **...**: Other graphical parameters to `plot`, `matlines`, or `legend`.

**Author(s)**

Patrick Breheny

**See Also**

grpreg

**Examples**

```r
# Fit model to birthweight data
data(Birthwt)
X <- Birthwt$X
y <- Birthwt$bwt
group <- Birthwt$group
fit <- grpreg(X, y, group, penalty="grLasso")

# Plot (basic)
plot(fit)

# Plot group norms, with labels in right margin
plot(fit, norm=TRUE, label=TRUE)

# Plot (miscellaneous options)
myColors <- c("black", "red", "green", "blue", "yellow", "purple", "orange", "brown")
plot(fit, legend.loc="topleft", col=myColors)
labs <- c("Mother's Age", "# Phys. visits", "Hypertension", "Mother's weight", "# Premature", "Race", "Smoking", "Uterine irritability")
plot(fit, legend.loc="topleft", lwd=6, alpha=0.5, legend=labs)
plot(fit, norm=TRUE, legend.loc="topleft", lwd=6, alpha=0.5, legend= labs)
```
plot.grpsurv.func  

Plot survival curve for grpsurv model

Description

Plot survival curve for a model that has been fit using grpsurv followed by a prediction of the survival function using predict.grpsurv

Usage

```r
## S3 method for class 'grpsurv.func'
plot(x, alpha=1, ...)
```

Arguments

- `x`  
  A 'grpsurv.func' object, which is returned by predict.grpsurv if type='survival' is specified. See examples.

- `alpha`  
  Controls alpha-blending (i.e., transparency). Useful if many overlapping lines are present.

- `...`  
  Other graphical parameters to pass to `plot`

Author(s)

Patrick Breheny

See Also

ggrpsurv, predict.grpsurv

Examples

data(Lung)
X <- Lung$X
y <- Lung$y
group <- Lung$group
fit <- grpsurv(X, y, group)

# A single survival curve
S <- predict(fit, X[1,], type='survival', lambda=0.05)
plot(S, xlim=c(0,200))

# Lots of survival curves
S <- predict(fit, X, type='survival', lambda=0.05)
plot(S, xlim=c(0,200), alpha=0.3)
predict.grpreg

Model predictions based on a fitted grpreg object

Description

Similar to other predict methods, this function returns predictions from a fitted "grpreg" object.

Usage

```r
## S3 method for class 'grpreg'
predict(object, X, type=c("link", "response", "class", "coefficients", "vars", "groups", "nvars", "ngroups", "norm"), lambda,
which=1:length(object$lambda), ...)  
## S3 method for class 'grpreg'
coef(object, lambda, which=1:length(object$lambda), drop=TRUE, ...)  
## S3 method for class 'cv.grpreg'
predict(object, X, lambda=object$lambda.min,
which=object$min, type=c("link", "response", "class", "coefficients", "vars", "groups", "nvars", "ngroups", "norm"), ...)  
## S3 method for class 'cv.grpreg'
coef(object, lambda=object$lambda.min,
which=object$min, ...)  
```

Arguments

- **object**: Fitted "grpreg" or "cv.grpreg" model object.
- **X**: Matrix of values at which predictions are to be made. Not used for `type="coefficients"`
- **lambda**: Values of the regularization parameter lambda at which predictions are requested. For values of lambda not in the sequence of fitted models, linear interpolation is used.
- **which**: Indices of the penalty parameter lambda at which predictions are required. By default, all indices are returned. If lambda is specified, this will override which.
- **type**: Type of prediction: "link" returns the linear predictors; "response" gives the fitted values; "class" returns the binomial outcome with the highest probability; "coefficients" returns the coefficients; "vars" returns the indices for the nonzero coefficients; "groups" returns the indices for the groups with at least one nonzero coefficient; "nvars" returns the number of nonzero coefficients; "ngroups" returns the number of groups with at least one nonzero coefficient; "norm" returns the L2 norm of the coefficients in each group.
- **drop**: By default, if a single value of lambda is supplied, a vector of coefficients is returned. Set drop=FALSE if you wish to have coef always return a matrix (see drop).
- **...**: Not used.
predict.grpsurv

Details

coe and predict methods are provided for "cv.grpreg" options as a convenience. They simply call coef.grpreg and predict.grpreg with lambda set to the value that minimizes the cross-validation error.

Value

The object returned depends on type.

Author(s)

Patrick Breheny

See Also

grpreg

Examples

# Fit penalized logistic regression model to birthweight data
data(Birthwt)
X <- Birthwt$X
y <- Birthwt$low
group <- Birthwt$group
fit <- grpreg(X, y, group, penalty="grlasso", family="binomial")

# Coef and predict methods
coeff(fit, lambda=.001)
predict(fit, X, type="link", lambda=.07)[1:10]
predict(fit, X, type="response", lambda=.07)[1:10]
predict(fit, X, type="class", lambda=.01)[1:15]
predict(fit, type="vars", lambda=.07)
predict(fit, type="groups", lambda=.07)
predict(fit, type="norm", lambda=.07)

# Coef and predict methods for cross-validation
cvfit <- cv.grpreg(X, y, group, family="binomial", penalty="grMCP")
coeff(cvfit)
predict(cvfit, X)[1:10]
predict(cvfit, X, type="response")[1:10]
predict(cvfit, type="groups")

Model predictions based on a fitted "grpsurv" object.

Description

Similar to other predict methods, this function returns predictions from a fitted "grpsurv" object.
Usage

```r
predict(object, x, type=c("link", "response", "survival",
               "median", "norm", "coefficients", "vars", "nvars", "groups", "ngroups"),
lambda, which=1:length(object$lambda), ...)
```

Arguments

- **object**: Fitted `grpsurv` model object.
- **x**: Matrix of values at which predictions are to be made. Not used for `type="coefficients"` or for some of the `type` settings in `predict`.
- **lambda**: Values of the regularization parameter `lambda` at which predictions are requested. For values of `lambda` not in the sequence of fitted models, linear interpolation is used.
- **which**: Indices of the penalty parameter `lambda` at which predictions are required. By default, all indices are returned. If `lambda` is specified, this will override `which`.
- **type**: Type of prediction: "link" returns the linear predictors; "response" gives the risk (i.e., \(\exp(\text{link})\)); "survival" returns the estimated survival function; "median" estimates median survival times. The other options are all identical to their `grpreg` counterparts: "coefficients" returns the coefficients; "vars" returns the indices for the nonzero coefficients; "groups" returns the indices for the groups with at least one nonzero coefficient; "nvars" returns the number of nonzero coefficients; "ngroups" returns the number of groups with at least one nonzero coefficient; "norm" returns the L2 norm of the coefficients in each group.
- **...**: Not used.

Details

Estimation of baseline survival function conditional on the estimated values of \(\beta\) is carried out according to the method described in Chapter 4.3 of Kalbfleish and Prentice. In particular, it agrees exactly with the results returned by `survfit.coxph(..., type='kalbfleisch-prentice')` in the survival package.

Value

The object returned depends on `type`.

Author(s)

Patrick Breheny

References

select.grpreg

Select an value of lambda along a grpreg path

Description

Selects a point along the regularization path of a fitted grpreg object according to the AIC, BIC, or GCV criteria.

Usage

```r
select(obj,...)
## S3 method for class 'grpreg'
select(obj, criterion=c("BIC", "AIC", "GCV", "AICC", "EBIC"), df.method=c("default", "active"), smooth=FALSE, ...)
```
Arguments

- **obj**: A fitted grpreg object.
- **criterion**: The criterion by which to select the regularization parameter. One of "AIC", "BIC", "GCV", "AICC", or "EBIC"; default is "BIC".
- **df.method**: How should effective model parameters be calculated? One of: "active", which counts the number of nonzero coefficients; or "default", which uses the calculated df returned by grpreg. Default is "default".
- **smooth**: Applies a smoother to the information criteria before selecting the optimal value.

Details

The criteria are defined as follows, where $L$ is the deviance (i.e., -2 times the log-likelihood), $\nu$ is the degrees of freedom, and $n$ is the sample size:

\[
AIC = L + 2\nu \\
BIC = L + \log(n)\nu \\
GCV = \frac{L}{(1-\nu/n)^2} \\
AICC = AIC + \frac{2\nu(\nu+1)}{n-\nu-1} \\
EBIC = BIC + 2\log\left(\frac{p}{\nu}\right)
\]

Value

A list containing:

- **lambda**: The selected value of the regularization parameter, lambda.
- **beta**: The vector of coefficients at the chosen value of lambda.
- **df**: The effective number of model parameters at the chosen value of lambda.
- **ic**: A vector of the calculated model selection criteria for each point on the regularization path.

Author(s)

Patrick Breheny

See Also

grpreg
summary.cv.grpreg

### Examples

```r
data(Birthwt)
X <- Birthwt$X
y <- Birthwt$bwt
group <- Birthwt$group
fit <- grpreg(X, y, group, penalty="grLasso")
select(fit)
plot(fit)
```

### Description

Summary method for cv.grpreg or cv.grpsurv objects

### Usage

```r
## S3 method for class 'cv.grpreg'
summary(object, ...)
## S3 method for class 'summary.cv.grpreg'
print(x, digits, ...)
```

### Arguments

- `object`  
  A "cv.grpreg" object.
- `x`  
  A "summary.cv.grpreg" object.
- `digits`  
  Number of digits past the decimal point to print out. Can be a vector specifying different display digits for each of the five non-integer printed values.
- `...`  
  Further arguments passed to or from other methods.

### Value

`summary(cvfit)` produces an object with S3 class "summary.cv.grpreg". The class has its own print method and contains the following list elements:

- `penalty`  
  The penalty used by grpreg/grpsurv.
- `model`  
  The type of model: "linear", "logistic", "Poisson", "Cox", etc.
- `n`  
  Number of observations.
**p**  
Number of regression coefficients (not including the intercept).

**min**  
The index of \( \lambda \) with the smallest cross-validation error.

**lambda**  
The sequence of \( \lambda \) values used by `cv.grpreg/cv.gprpsurv`.

**cve**  
Cross-validation error (deviance).

**r.squared**  
Proportion of variance explained by the model, as estimated by cross-validation.

**snr**  
Signal to noise ratio, as estimated by cross-validation.

**sigma**  
For linear regression models, the scale parameter estimate.

**pe**  
For logistic regression models, the prediction error (misclassification error).

---

**Author(s)**

Patrick Breheny

---

**See Also**

`grpreg, cv.grpreg, cv.gprpsurv, plot.cv.grpreg`

---

**Examples**

```r
# Birthweight data
data(Birthwt)
X <- Birthwt[,]
group <- Birthwt$group

# Linear regression
y <- Birthwt$bwt
cvfit <- cv.grpreg(X, y, group)
summary(cvfit)

# Logistic regression
y <- Birthwt$low
cvfit <- cv.grpreg(X, y, group, family="binomial")
summary(cvfit)

# Cox regression
data(Lung)
cvfit <- with(Lung, cv.gprpsurv(X, y, group))
summary(cvfit)
```
Index

AUC (AUC.cv.grpsurv), 3
AUC.cv.grpsurv, 3
Surv, 16, 20
survConcordance, 4

Birthwt, 5, 6
birthwt, 6
birthwt.grpreg, 6

coef.cv.grpreg (predict.grpreg), 25
coeff.grpreg (predict.grpreg), 25
coeff.grpsurv (predict.grpsurv), 26
cv.grpreg, 7, 14, 22, 31
cv.grpsurv, 4, 18, 31
cv.grpsurv (cv.grpreg), 7
drop, 25
gBridge, 9, 12
gpreg, 6, 8, 10, 11, 22, 31
gpreg-package, 2
grpsurv, 16, 21, 24, 28

legend, 23
logLik (logLik.grpreg), 19
logLik.grpreg, 19
Lung, 20

plot, 14
plot.cv.grpreg, 8, 21, 31
plot.grpreg, 18, 22
plot.grpsurv.func, 24
predict.cv.grpreg, 8
predict.cv.grpreg (predict.grpreg), 25
predict.grpreg, 25
predict.grpsurv, 18, 24, 26
print.summary.cv.grpreg
 (summary.cv.grpreg), 30

select, 14
select (select.grpreg), 28
select.grpreg, 28
summary.cv.grpreg, 8, 30