Package 'grpreg'

April 8, 2018

Title Regularization Paths for Regression Models with Grouped Covariates

Version 3.1-3

Date 2018-04-07

Author Patrick Breheny [aut, cre], Yaohui Zeng [ctb]

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

Depends R (>= 3.1.0), Matrix

Suggests grpregOverlap, knitr, survival

VignetteBuilder knitr

Description Efficient algorithms for fitting the regularization path of linear or logistic 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

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-04-08 11:13:01 UTC

R topics documented:

`grpreg-package` *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

The following penalties are available:

- `grLasso`: Group lasso (Yuan and Lin, 2006)
- `grMCP`: Group MCP; like the group lasso, but with an MCP penalty on the norm of each group
- `grSCAD`: 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.

See the "Quick start guide" for a brief overview of how the package works, and "Penalties in grpreg" for a more detailed description of the individual penalties availabe in the package.

Author(s)

Patrick Breheny <patrick-breheny@uiowa.edu>

References

- Yuan, M. and Lin, Y. (2006) Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B*, **68**: 49-67.
- Huang, J., Ma, S., Xie, H., and Zhang, C. (2009) A group bridge approach for variable selection. *Biometrika*, **96**: 339-355.
- Breheny, P. and Huang, J. (2009) Penalized methods for bi-level variable selection. *Statistics and its interface*, **2**: 369-380. myweb.uiowa.edu/pbreheny/publications/Breheny2009.pdf
- Huang J., Breheny, P. and Ma, S. (2012). A selective review of group selection in high dimensional models. *Statistical Science*, **27**: 481-499. myweb.uiowa.edu/pbreheny/publications/Huang2012.pdf
- Breheny, P. and Huang, J. (2015) Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Statistics and Computing*, **25**: 173-187. www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s11222-013-9424-2
- Breheny, P. (2015) The group exponential lasso for bi-level variable selection. *Biometrics*, **71**: 731-740. http://dx.doi.org/10.1111/biom.12300

Examples

```
vignette("quick-start", "grpreg")
vignette("penalties", "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

van Houwelingen H, Putter H (2011). Dynamic Prediction in Clinical Survival Analysis. CRC Press.

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), xlim=rev(range(ll)), lwd=3, type='l',
     xlab=expression(log(lambda)), ylab='AUC')
```

Birthwt *Risk Factors Associated with Low Infant Birth Weight*

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,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.

Source

`MASS`. `R` package. Available at `http://cran.r-project.org`.

References

- Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.
- Hosmer, D.W. and Lemeshow, S. (1989) *Applied Logistic Regression.* New York: Wiley

See Also

`birthwt`, `grpreg`

Examples

```
data(Birthwt)
hist(Birthwt$bwt, xlab="Child's birth weight", main="")
table(Birthwt$low)
## See examples in ?birthwt (MASS package)
##   for more about the data set
## See examples in ?grpreg for use of this data set
##   with group penalized regression models
```

`birthwt.grpreg` *Risk Factors Associated with Low Infant Birth Weight*

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*

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, cv.ind,
returnY=FALSE, trace=FALSE)
```

Arguments

X	The design matrix, as in grpreg.
y	The response vector (or matrix), as in grpreg.
group	The grouping vector, as in grpreg.
...	Additional arguments to grpreg.
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.
cv.ind	Which fold each observation belongs to. By default the observations are randomly assigned by cv.grpreg.
returnY	Should cv.grpreg 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.
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.

Details

The function calls grpreg nfolds times, each time leaving out 1/nfolds of the data. The cross-validation error is based on the residual sum of squares when family="gaussian" and the deviance when family="binomial" or family="poisson".

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.

As in 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 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 "cv.grpreg" containing:

cve	The error for each value of lambda, averaged across the cross-validation folds.
cvse	The estimated standard error associated with each value of for cve.
lambda	The sequence of regularization parameter values along which the cross-validation error was calculated.
fit	The fitted grpreg object for the whole data.
min	The index of lambda corresponding to lambda.min.
lambda.min	The value of lambda with the minimum cross-validation error.
null.dev	The deviance for the intercept-only model.
pe	If family="binomial", the cross-validation prediction error for each value of lambda.

Author(s)

Patrick Breheny <patrick-breheny@uiowa.edu>

See Also

grpreg, plot.cv.grpreg, summary.cv.grpreg, predict.cv.grpreg

Examples

```
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)
```

---

cv.grpsurv *Cross-validation for grpsurv*

Description

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

Usage

```
cv.grpsurv(X, y, group, ..., nfolds=10, seed, cv.ind, returnY=FALSE,
trace=FALSE)
```

Arguments

X	The design matrix, as in grpsurv.
y	The response matrix, as in grpsurv.
group	The grouping vector, as in grpsurv.
...	Additional arguments to 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.
cv.ind	Which fold each observation belongs to. By default the observations are randomly assigned by cv.grpsurv.
returnY	Should cv.grpsurv return the linear predictors 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: The rows of Y are ordered by time on study, and therefore do not correspond to the original order of observations pased to cv.grpsurv.
trace	If set to TRUE, cv.grpsurv will inform the user of its progress by announcing the beginning of each CV fold. Default is FALSE.

Details

The function calls grpsurv nfolds times, each time leaving out 1/nfolds of the data. Because of the semiparametric nature of Cox regression, cross-validation is not clearly defined. cv.grpsurv uses the approach of calculating the full Cox partial likelihood using the cross-validated set of linear predictors. Unfortunately, using this approach there is no clear way (yet) of determining standard errors, so cv.grpsurv, unlike cv.grpreg, does not provide any.

Other approaches to cross-validation for the Cox regression model have been proposed; the strenghts and weaknesses of the various methods for penalized regression in the Cox model are not well understood. Because of this, the approach used by cv.grpsurv may change in the future as additional research is carried out.

Value

An object with S3 class "cv.grpsurv" inheriting from "cv.grpreg" and containing:

cve	The error for each value of lambda, averaged across the cross-validation folds.
lambda	The sequence of regularization parameter values along which the cross-validation error was calculated.
fit	The fitted grpsurv object for the whole data.
min	The index of lambda corresponding to lambda.min.

lambda.min	The value of lambda with the minimum cross-validation error.
null.dev	The cross-validated deviance for the first model along the grid of lambda (i.e., the cross-validated deviance for max(lambda), unless you have supplied your own lambda sequence, in which case this quantity is probably not meaningful). Although the actual null deviance can be calculated, it cannot be compared with the cross-validated deviance due to the manner in which deviance must be calculated for Cox regression models (see details).

Author(s)

Patrick Breheny <patrick-breheny@uiowa.edu>

References

- Verweij PJ and van Houwelingen HC. (1993) Cross-validation in survival analysis. *Statistics in Medicine*, **12**: 2305-2314.

See Also

grpsurv, plot.cv.grpreg, summary.cv.grpreg

Examples

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

cvfit <- cv.grpsurv(X, y, group)
plot(cvfit)
coef(cvfit)
plot(cvfit$fit)
plot(cvfit, type="rsq")
```

---

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

```
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 threshhold, 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 "grpreg", as in grpreg.

Author(s)

Patrick Breheny <patrick-breheny@uiowa.edu>

References

- Huang, J., Ma, S., Xie, H., and Zhang, C. (2009) A group bridge approach for variable selection. *Biometrika*, **96**: 339-355.
- Breheny, P. and Huang, J. (2009) Penalized methods for bi-level variable selection. *Statistics and its interface*, **2**: 369-380.

See Also

grpreg

Examples

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

## Linear regression
y <- Birthwt$bwt
fit <- gBridge(X, y, group)
plot(fit)
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 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.
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 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.
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 information)? 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 is defined to be

$$\frac{1}{2n}RSS + penalty$$

for `"gaussian"` and

$$-\frac{1}{n}loglik + penalty$$

for `"binomial"`, where the likelihood is from a traditional generalized linear model for the log-odds of an event. For logistic regression models, some care is taken to avoid model saturation; the algorithm may exit early in this setting.

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 `warn` (convergence can always be checked later by inspecting the value of `iter`).

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

Although grpreg allows groups to be unordered and given arbitary 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 group.multiplier. With consecutive integers, group=3 unambiguously denotes the third element of 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.

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 <patrick-breheny@uiowa.edu>

References

- Breheny, P. and Huang, J. (2009) Penalized methods for bi-level variable selection. *Statistics and its interface*, **2**: 369-380. myweb.uiowa.edu/pbreheny/publications/Breheny2009.pdf
- Huang J., Breheny, P. and Ma, S. (2012). A selective review of group selection in high dimensional models. *Statistical Science*, **27**: 481-499. myweb.uiowa.edu/pbreheny/publications/Huang2012.pdf
- Breheny, P. and Huang, J. (2015) Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Statistics and Computing*, **25**: 173-187. www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s11222-013-9424-2
- Breheny, P. (2015) The group exponential lasso for bi-level variable selection. *Biometrics*, **71**: 731-740. http://dx.doi.org/10.1111/biom.12300

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)
```

---

grpsurv *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

```
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 "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.

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 threshhold. 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.
...	Not used.

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

$$-\frac{1}{n}L(\beta|X,y) + \text{penalty},$$

where L is the partial log-likelihood from the Cox regression model.

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 conditonal 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(beta) values for each subject over all lambda values.
time	Times on study.
fail	Failure event indicator.

Author(s)

Patrick Breheny <patrick-breheny@uiowa.edu>

References

- Breheny, P. and Huang, J. (2009) Penalized methods for bi-level variable selection. *Statistics and its interface*, **2**: 369-380. myweb.uiowa.edu/pbreheny/publications/Breheny2009.pdf
- Huang J., Breheny, P. and Ma, S. (2012). A selective review of group selection in high dimensional models. *Statistical Science*, **27**: 481-499. myweb.uiowa.edu/pbreheny/publications/Huang2012.pdf
- Breheny, P. and Huang, J. (2015) Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Statistics and Computing*, **25**: 173-187. www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s11222-013-9424-2

- Breheny, P. (2015) The group exponential lasso for bi-level variable selection. *Biometrics*, **71**: 731-740. http://dx.doi.org/10.1111/biom.12300
- Simon N, Friedman JH, Hastie T, and Tibshirani R. (2011) Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. *Journal of Statistical Software*, **39**: 1-13. http://www.jstatsoft.org/v39/i05

See Also

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

Examples

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

fit <- grpsurv(X, y, group)
plot(fit)

S <- predict(fit, X, type='survival', lambda=0.05)
plot(S, xlim=c(0,200))
```

---

logLik.grpreg *logLik method for 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 <patrick-breheny@uiowa.edu>

See Also

grpreg

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

- Kalbfleisch D and Prentice RL (1980), *The Statistical Analysis of Failure Time Data*. Wiley, New York.

See Also

veteran, grpsurv

Examples

```
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

```
## 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-validaton error is minimized.
col	Controls the color of the dots (CV estimates).
...	Other graphical parameters to plot

Details

Error bars representing approximate +/- 1 SE (68% confidence intervals) are plotted along with the estimates at value of lambda. For rsq and snr, these confidence intervals are quite crude, especially near zero, and will hopefully be improved upon in later versions of grpreg.

Author(s)

Patrick Breheny <patrick-breheny@uiowa.edu>

See Also

grpreg, cv.grpreg

Examples

```
# Birthweight data
data(Birthwt)
X <- Birthwt$X
group <- Birthwt$group

# Linear regression
y <- Birthwt$bwt
cvfit <- cv.grpreg(X, y, group)
plot(cvfit)
par(mfrow=c(2,2))
plot(cvfit, type="all")

## Logistic regression
y <- Birthwt$low
cvfit <- cv.grpreg(X, y, group, family="binomial")
plot(cvfit)
par(mfrow=c(2,2))
plot(cvfit, type="all")
```

`plot.grpreg` *Plot coefficients from a "grpreg" object*

Description

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

Usage

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

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 <patrick-breheny@uiowa.edu>

See Also

grpreg

Examples

```
# 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)
```

---

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

```
## 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.

Details

coef 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 <patrick-breheny@uiowa.edu>

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
coef(fit, lambda=.001)
predict(fit, X, type="link", lambda=.001)
predict(fit, X, type="response", lambda=.001)
predict(fit, X, type="class", lambda=.001)
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")
coef(cvfit)
predict(cvfit, X)
predict(cvfit, X, type="response")
predict(cvfit, type="groups")
```

`predict.grpsurv` *Model predictions based on a fitted "grpsurv" object.*

Description

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

Usage

```
## S3 method for class 'grpsurv'
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(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 <patrick-breheny@uiowa.edu>

References

- Kalbfleish JD and Prentice RL (2002). The Statistical Analysis of Failure Time Data, 2nd edition. Wiley.

See Also

grpsurv

Examples

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

fit <- grpsurv(X, y, group)
coef(fit, lambda=0.05)
head(predict(fit, X, type="link", lambda=0.05))
head(predict(fit, X, type="response", lambda=0.05))

# Survival function
S <- predict(fit, X[1,], type="survival", lambda=0.05)
S(100)
S <- predict(fit, X, type="survival", lambda=0.05)
```

```
plot(S, xlim=c(0,200))

# Medians
predict(fit, X[1,], type="median", lambda=0.05)
M <- predict(fit, X, type="median")
M[1:10, 1:10]

# Nonzero coefficients
predict(fit, type="vars", lambda=c(0.1, 0.01))
predict(fit, type="nvars", lambda=c(0.1, 0.01))
```

`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

```
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.
`...`	For S3 method compatibility.

Details

The criteria are defined as follows, where ℓ is the log likelihood, ν is the degrees of freedom, and n is the sample size:

$$AIC = 2L + 2df$$

$$BIC = 2L + log(n)df$$

$$GCV = 2\frac{L}{(1 - df/n)^2}$$

$$AICc = AIC + 2\frac{df(df+1)}{n-df-1}$$

$$EBIC = BIC + 2\log\binom{p}{df}$$

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 <patrick-breheny@uiowa.edu>

See Also

`grpreg`

Examples

```
data(Birthwt)
X <- Birthwt$X
y <- Birthwt$bwt
group <- Birthwt$group
fit <- grpreg(X, y, group, penalty="grLasso")
select(fit)
select(fit,crit="AIC",df="active")
plot(fit)
abline(v=select(fit)$lambda)
par(mfrow=c(1,3))
l <- fit$lambda
xlim <- rev(range(l))
plot(l, select(fit)$IC, xlim=xlim, pch=19, type="o", ylab="BIC")
plot(l, select(fit,"AIC")$IC, xlim=xlim, pch=19, type="o",ylab="AIC")
plot(l, select(fit,"GCV")$IC, xlim=xlim, pch=19, type="o",ylab="GCV")
```

summary.cv.grpreg *Summarizing inferences based on cross-validation*

Description

Summary method for cv.grpreg objects

Usage

```
## 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.cv.grpreg 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.
model	Either "linear" or "logistic", depending on the family option in grpreg.
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.
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 <patrick-breheny@uiowa.edu>

See Also

grpreg, cv.grpreg, plot.cv.grpreg

Examples

```
# Birthweight data
data(Birthwt)
X <- Birthwt$X
group <- Birthwt$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.grpsurv(X, y, group))
summary(cvfit)
```

Index