Package 'ppLasso'

April 3, 2023

Type Package

Title Variable selection algorithm for generalized linear models with high dimensional center effects

Version 1.5

Date 2023-01-05

Author Yubo Shao

Maintainer Yubo Shao <ybshao@umich.edu>

Description
An efficient algorithm for solving GLM lasso problems with large number of health centers

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports Rcpp (>= 1.0.7)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.1

Depends R (>= 2.10)

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

coef.ppLasso *extract coefficients of a ppLasso or gr_ppLasso object*

Description

return the model coefficients of a ppLasso or gr_ppLasso object

Usage

```
## S3 method for class 'ppLasso'
coef(fit, lambda, which = 1:length(fit$lambda), drop = TRUE, ...)

## S3 method for class 'gr_ppLasso'
coef(fit, lambda, which = 1:length(fit$lambda), drop = TRUE, ...)
```

Arguments

fit	a ppLasso or gr_ppLasso object.
lambda	values of the regularization parameter lambda at which coefficients 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.
drop	whether to keep coefficient names
...	

Examples

```
#fit glm without grouped covariates
data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
fit <- pp.lasso(data, Y.char, Z.char, prov.char)
coef(fit, lambda = fit$lambda)$beta[, 1:10]
coef(fit, lambda = fit$lambda)$gamma[1:10, 1:5]
#fit glm with grouped covariates
data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
group <- GLM_Data$group
fit <- grp.lasso(data, Y.char, Z.char, prov.char, group = group)
coef(fit, lambda = fit$lambda)$beta[, 1:5]
coef(fit, lambda = fit$lambda)$gamma[1:10, 1:5]
```

`cv.grp.lasso` *cross-validation for grp.lasso*

Description

performs k-fold cross validation for group penalized regression models over a grid of values of regularization parameter lambda.

Usage

```
cv.grp.lasso(
  data,
  Y.char,
  Z.char,
  prov.char,
  group = 1:length(Z.char),
  ...,
  nfolds = 10,
  seed,
  fold,
  trace.cv = FALSE
)
```

Arguments

`data`	an 'dataframe' or 'list' object that contains the variables in the model.
`Y.char`	name of the response variable from 'data' as a character string, as in `grp.lasso` function.
`Z.char`	names of covariates from 'data' as vector of character strings, as in `grp.lasso` function.
`prov.char`	name of provider IDs variable from 'data' as a character string, as in `grp.lasso` function.
`group`	a vector describing the grouping of the coefficients. If there are coefficients to be included in the model without being penalized, assign them to group 0 (or "0").
`...`	extra arguments to be passed to function.
`nfolds`	the number of cross-validation folds. Default is 10.
`seed`	the seed of the random number generator in order to obtain reproducible results.
`fold`	a vector that specifies the fold that observations belongs to. By default the observations are randomly assigned.
`trace.cv`	`cv.grp.lasso` will provide user with the progress of cross validation if 'trace.cv = TRUE'. Default is FALSE.

Value

An object with S3 class `cv.gr_ppLasso`.

`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 gr_ppLasso object for the whole data.
fold	the fold assignments for cross-validation for each observation
min	the index of lambda corresponding to lambda.min.
lambda.min	the value of lambda with the minimum cross-validation error.

References

K. He, J. Kalbfleisch, Y. Li, and et al. (2013) Evaluating hospital readmission rates in dialysis facilities; adjusting for hospital effects. *Lifetime Data Analysis*, **19**: 490-512.

Examples

```
data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
group <- GLM_Data$group
fit <- cv.grp.lasso(data, Y.char, Z.char, prov.char, group = group, nfolds = 10)
# the best lambda using cross validation
fit$lambda.min
```

---

cv.pp.lasso *cross-validation for pp.lasso*

Description

performs k-fold cross validation for penalized regression models over a grid of values of regularization parameter lambda.

Usage

```
cv.pp.lasso(
  data,
  Y.char,
  Z.char,
  prov.char,
  penalize.x = rep(1, length(Z.char)),
  ...,
  nfolds = 10,
  seed,
  fold,
  trace.cv = FALSE
)
```

Arguments

data	an 'dataframe' or 'list' object that contains the variables in the model.
Y.char	name of the response variable from 'data' as a character string, as in pp.lasso function.
Z.char	names of covariates from 'data' as vector of character strings, as in pp.lasso function.
prov.char	name of provider IDs variable from 'data' as a character string, as in pp.lasso function.
penalize.x	a vector indicates whether the corresponding covariate will be penalized, as in pp.lasso function.
...	extra arguments to be passed to function.
nfolds	the number of cross-validation folds. Default is 10.
seed	the seed of the random number generator in order to obtain reproducible results.
fold	a vector that specifies the fold that observations belongs to. By default the observations are randomly assigned.
trace.cv	cv.pp.lasso will provide user with the progress of cross validation if 'trace.cv = TRUE'. Default is FALSE.

Value

An object with S3 class cv.ppLasso.

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 pp.lasso object for the whole data.
fold	the fold assignments for cross-validation for each observation
min	the index of lambda corresponding to lambda.min.
lambda.min	the value of lambda with the minimum cross-validation error.

References

K. He, J. Kalbfleisch, Y. Li, and et al. (2013) Evaluating hospital readmission rates in dialysis facilities; adjusting for hospital effects. *Lifetime Data Analysis*, **19**: 490-512.

Examples

```
data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
fit <- cv.pp.lasso(data, Y.char, Z.char, prov.char, nfolds = 10)
# the best lambda using cross validation
fit$lambda.min
```

GLM_Data *example data for pp.lasso and grp.lasso*

Description

A simulated data set containing response variable, provider information and 5 covariates.

Usage

```
data(GLM_Data)
```

Format

A list containing the following elements:

data example data. `Y` is the response variable; `Prov.ID` is the provider indicator; `Z1`, ..., `Z5` are 5 continuous covariates.

Y.char variable name of the response variable.

prov.char variable name of the provider indicator.

Z.char variable names of covariates.

group vector describing how the covariates are grouped.

grp.lasso *fit a group penalized generalized regression model*

Description

main function for fitting a group penalized generalized regression model

Usage

```
grp.lasso(
  data,
  Y.char,
  Z.char,
  prov.char,
  group = 1:length(Z.char),
  group.multiplier,
  standardize = T,
  lambda,
  nlambda = 100,
  lambda.min.ratio = 1e-04,
  lambda.early.stop = FALSE,
  nvar.max = p,
  group.max = length(unique(group)),
  stop.dev.ratio = 0.001,
  bound = 10,
  backtrack = FALSE,
```

```
  tol = 1e-04,
  max.each.iter = 10000,
  max.total.iter = (max.each.iter * nlambda),
  actSet = TRUE,
  actIter = max.each.iter,
  actGroupNum = sum(unique(group) != 0),
  actSetRemove = F,
  returnX = FALSE,
  trace.lambda = FALSE,
  threads = 1,
  ...
)
```

Arguments

data	an ‘dataframe‘ or ‘list‘ object that contains the variables in the model.
Y.char	name of the response variable in ‘data‘ as a character string.
Z.char	names of covariates in ‘data‘ as vector of character strings.
prov.char	name of provider IDs variable in ‘data‘ as a character string..
group	a vector describing the grouping of the coefficients. If there are coefficients to be included in the model without being penalized, assign them to group 0 (or "0").
group.multiplier	A vector of values representing multiplicative factors by which each covariate's penalty is to be multiplied. Default is a vector of 1's.
standardize	logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is ‘standardize=TRUE‘.
lambda	a user supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence based on ‘nlambda‘ and ‘lambda.min.ratio‘.
nlambda	the number of lambda values. Default is 100.
lambda.min.ratio	the fraction of the smallest value for lambda with ‘lambda.max‘ (smallest lambda for which all coefficients are zero) on log scale. Default is 1e-04.
lambda.early.stop	whether the program stop before running the entire sequence of lambda. Early stop based on the ratio of deviance for models under two successive lambda. Default is ‘FALSE‘.
nvar.max	number of maximum selected variables. Default is the number of all covariates.
group.max	number of maximum selected groups. Default is the number of all groups.
stop.dev.ratio	if ‘lambda.early.stop = TRUE‘, the ratio of deviance for early stopping. Default is 1e-3.
bound	a positive number to avoid inflation of provider effect. Default is 10.
backtrack	for updating the provider effect, whether to use the "backtracking line search" with Newton method.
tol	convergence threshold. For each lambda, the program will stop if the maximum change of covariate coefficient is smaller than ‘tol‘. Default is 1e-4.
max.each.iter	maximum number of iterations for each lambda. Default is 1e4.

max.total.iter	maximum number of iterations for entire path. Default is 'max.each.iter' * 'nlambda'.
actSet	whether to use the active method for variable selection. Default is TRUE.
actIter	if 'actSet = TRUE', the maximum number of iterations for a new updated active set. Default is 'max.each.iter' (i.e. we will update the current active set until convergence).
actGroupNum	if 'actSet = TRUE', the maximum number of variables that can be selected into the new active set for each time when the active set is updated. Default is number of groups.
actSetRemove	if 'actSet = TRUE', whether we remove the zero coefficients from the current active set. Default is FALSE.
returnX	whether return the standardized design matrix. Default is FALSE.
trace.lambda	whether display the progress for fitting the entire path. Default is FALSE.
threads	number of cores that are used for parallel computing.
...	extra arguments to be passed to function.

Details

The model is fit by Newton method and coordinate descent method.

Value

An object with S3 class gr_ppLasso.

beta	the fitted matrix of covariate coefficients. The number of rows is equal to the number of coefficients, and the number of columns is equal to nlambda.
gamma	the fitted value of provider effects.
group	a vector describing the grouping of the coefficients.
lambda	the sequence of 'lambda' values in the path.
loss	the loss of the fitted model at each value of 'lambda'.
linear.predictors	the linear predictors of the fitted model at each value of 'lambda'.
df	the estimates of effective number of selected variables all the points along the regularization path.
iter	the number of iterations until convergence at each value of 'lambda'.

References

K. He, J. Kalbfleisch, Y. Li, and et al. (2013) Evaluating hospital readmission rates in dialysis facilities; adjusting for hospital effects. *Lifetime Data Analysis*, **19**: 490-512.

See Also

coef function.

Examples

```
data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
group <- GLM_Data$group
fit <- grp.lasso(data, Y.char, Z.char, prov.char, group = group)
# fitted values of covariate coefficients (under the lambda sequence that was automatically generated by the pac
round(fit$beta[1:5, 1:5], 5)
# estimated center effects
round(fit$gamma[1:5, 1:5], 5)
```

---

plot.cv.ppLasso *plot the cross entropy Loss from a cv.ppLasso or cv.gr_ppLasso object*

Description

return the plot of the cross entropy Loss from a `cv.ppLasso` or `cv.gr_ppLasso` object

Usage

```
## S3 method for class 'cv.ppLasso'
plot(
  fit,
  log.x = T,
  vertical.line = T,
  col.vertical.line = "blue",
  col.dot = "red"
)

## S3 method for class 'cv.gr_ppLasso'
plot(
  fit,
  log.x = T,
  vertical.line = T,
  col.vertical.line = "blue",
  col.dot = "red"
)
```

Arguments

`fit`	a `cv.gr_ppLasso` object.
`log.x`	whether the horizontal axis be on the log scale.
`vertical.line`	whether draws a vertical line at the value where cross-validaton error is minimized.

Examples

```
data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
fit <- cv.pp.lasso(data, Y.char, Z.char, prov.char, nfolds = 10)
plot(fit)

data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
group <- GLM_Data$group
fit <- cv.grp.lasso(data, Y.char, Z.char, prov.char, group = group, nfolds = 10)
plot(fit)
```

`plot.ppLasso` *plot regularization path of coefficients from a ppLasso or gr_ppLasso object*

Description

return the plot the regularization path from a `ppLasso` or `gr_ppLasso` object

Usage

```
## S3 method for class 'ppLasso'
plot(fit, log.x = T, label = F)

## S3 method for class 'gr_ppLasso'
plot(fit, log.x = T, label = F)
```

Arguments

`fit`	a `gr_ppLasso` object.
`log.x`	whether the horizontal axis be on the log scale.
`label`	whether annotates the plot with labels.

Examples

```
data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
fit <- pp.lasso(data, Y.char, Z.char, prov.char)
plot(fit)
data(GLM_Data)
data <- GLM_Data$data
```

```
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
group <- GLM_Data$group
fit <- grp.lasso(data, Y.char, Z.char, prov.char, group = group)
plot(fit)
```

`pp.DiscSurv` *fit a penalized discrete survival model*

Description

main function for fitting a penalized discrete survival model

Usage

```
pp.DiscSurv(
  data,
  Event.char,
  prov.char,
  Z.char,
  Time.char,
  lambda,
  nlambda = 100,
  lambda.min.ratio = 1e-04,
  penalize.x = rep(1, length(Z.char)),
  penalized.multiplier,
  lambda.early.stop = FALSE,
  nvar.max = p,
  stop.dev.ratio = 0.001,
  bound = 10,
  backtrack = FALSE,
  tol = 1e-04,
  max.each.iter = 10000,
  max.total.iter = (max.each.iter * nlambda),
  actSet = TRUE,
  actIter = max.each.iter,
  actVarNum = sum(penalize.x == 1),
  actSetRemove = F,
  returnX = FALSE,
  trace.lambda = FALSE,
  threads = 1,
  MM = FALSE,
  ...
)
```

Arguments

`data`	an ‘dataframe‘ or ‘list‘ object that contains the variables in the model.
`Event.char`	name of the event indicator in ‘data‘ as a character string.
`prov.char`	name of provider IDs variable in ‘data‘ as a character string.

Z.char	names of covariates in ‘data‘ as vector of character strings.
Time.char	name of the observation time in ‘data‘ as a character string.
lambda	a user supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence based on ‘nlambda‘ and ‘lambda.min.ratio‘.
nlambda	the number of lambda values. Default is 100.
lambda.min.ratio	the fraction of the smallest value for lambda with ‘lambda.max‘ (smallest lambda for which all coefficients are zero) on log scale. Default is 1e-04.
penalize.x	a vector indicates whether the corresponding covariate will be penalized. If equals 0, variable is unpenalized, else is penalized. Default is a vector of 1’s (all covariates are penalized).
penalized.multiplier	A vector of values representing multiplicative factors by which each covariate’s penalty is to be multiplied. Default is a vector of 1’s.
lambda.early.stop	whether the program stop before running the entire sequence of lambda. Early stop based on the ratio of deviance for models under two successive lambda. Default is ‘FALSE‘.
nvar.max	number of maximum selected variables. Default is the number of all covariates.
stop.dev.ratio	if ‘lambda.early.stop = TRUE‘, the ratio of deviance for early stopping. Default is 1e-3.
bound	a positive number to avoid inflation of provider effect. Default is 10.
backtrack	for updating the provider effect, whether to use the "backtracking line search" with Newton method.
tol	convergence threshold. For each lambda, the program will stop if the maximum change of covariate coefficient is smaller than ‘tol‘. Default is 1e-4.
max.each.iter	maximum number of iterations for each lambda. Default is 1e4.
max.total.iter	maximum number of iterations for entire path. Default is ‘max.each.iter‘ * ‘nlambda‘.
actSet	whether to use the active method for variable selection. Default is TRUE.
actIter	if ‘actSet = TRUE‘, the maximum number of iterations for a new updated active set. Default is ‘max.each.iter‘ (i.e. we will update the current active set until convergence).
actSetRemove	if ‘actSet = TRUE‘, whether we remove the zero coefficients from the current active set. Default is FALSE.
returnX	whether return the standardized design matrix. Default is FALSE.
trace.lambda	whether display the progress for fitting the entire path. Default is FALSE.
threads	number of cores that are used for parallel computing.
MM	whether we use the "Majorize-Minimization" algorithm to optimize the objective function.
...	extra arguments to be passed to function.

Details

The model is fit by Newton method and coordinate descent method.

Value

An object with S3 class ppDiscSurv.

beta	the fitted matrix of covariate coefficients. The number of rows is equal to the number of coefficients, and the number of columns is equal to nlambda.
alpha	the fitted value of logit-transformed baseline hazard.
gamma	the fitted value of provider effects. The effect of the first provider is set to be reference group.
lambda	the sequence of 'lambda' values in the path.
df	the estimates of effective number of selected variables all the points along the regularization path.
iter	the number of iterations until convergence at each value of 'lambda'.

References

K. He, J. Kalbfleisch, Y. Li, and et al. (2013) Evaluating hospital readmission rates in dialysis facilities; adjusting for hospital effects. *Lifetime Data Analysis*, **19**: 490-512.

See Also

coef function.

Examples

```
data(Surv_Data)
data <- Surv_Data$data
Event.char <- Surv_Data$Event.char
prov.char <- Surv_Data$prov.char
Z.char <- Surv_Data$Z.char
Time.char <- Surv_Data$Time.char
fit <- pp.DiscSurv(data, Event.char, prov.char, Z.char, Time.char, lambda = 0.01)
fit$beta
fit$alpha
fit$gamma #effect of the first provider is set to be zero
```

---

pp.lasso *fit a penalized generalized regression model*

Description

main function for fitting a penalized generalized regression

Usage

```
pp.lasso(
  data,
  Y.char,
  Z.char,
  prov.char,
  standardize = T,
  lambda,
  nlambda = 100,
  lambda.min.ratio = 1e-04,
  penalize.x = rep(1, length(Z.char)),
  penalized.multiplier,
  lambda.early.stop = FALSE,
  nvar.max = p,
  stop.dev.ratio = 0.001,
  bound = 10,
  backtrack = FALSE,
  tol = 1e-04,
  max.each.iter = 10000,
  max.total.iter = (max.each.iter * nlambda),
  actSet = TRUE,
  actIter = max.each.iter,
  actVarNum = sum(penalize.x == 1),
  actSetRemove = F,
  returnX = FALSE,
  trace.lambda = FALSE,
  threads = 1,
  MM = FALSE,
  ...
)
```

Arguments

data	an ‘dataframe‘ or ‘list‘ object that contains the variables in the model.
Y.char	name of the response variable in ‘data‘ as a character string.
Z.char	names of covariates in ‘data‘ as vector of character strings.
prov.char	name of provider IDs variable in ‘data‘ as a character string.
standardize	logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is ‘standardize=TRUE‘.
lambda	a user supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence based on ‘nlambda‘ and ‘lambda.min.ratio‘.
nlambda	the number of lambda values. Default is 100.
lambda.min.ratio	the fraction of the smallest value for lambda with ‘lambda.max‘ (smallest lambda for which all coefficients are zero) on log scale. Default is 1e-04.
penalize.x	a vector indicates whether the corresponding covariate will be penalized. If equals 0, variable is unpenalized, else is penalized. Default is a vector of 1’s (all covariates are penalized).

`penalized.multiplier`
: A vector of values representing multiplicative factors by which each covariate's penalty is to be multiplied. Default is a vector of 1's.

`lambda.early.stop`
: whether the program stop before running the entire sequence of lambda. Early stop based on the ratio of deviance for models under two successive lambda. Default is 'FALSE'.

`nvar.max`
: number of maximum selected variables. Default is the number of all covariates.

`stop.dev.ratio`
: if 'lambda.early.stop = TRUE', the ratio of deviance for early stopping. Default is 1e-3.

`bound`
: a positive number to avoid inflation of provider effect. Default is 10.

`backtrack`
: for updating the provider effect, whether to use the "backtracking line search" with Newton method.

`tol`
: convergence threshold. For each lambda, the program will stop if the maximum change of covariate coefficient is smaller than 'tol'. Default is 1e-4.

`max.each.iter`
: maximum number of iterations for each lambda. Default is 1e4.

`max.total.iter`
: maximum number of iterations for entire path. Default is 'max.each.iter' * 'nlambda'.

`actSet`
: whether to use the active method for variable selection. Default is TRUE.

`actIter`
: if 'actSet = TRUE', the maximum number of iterations for a new updated active set. Default is 'max.each.iter' (i.e. we will update the current active set until convergence).

`actVarNum`
: if 'actSet = TRUE', the maximum number of variables that can be selected into the new active set for each time when the active set is updated. Default is 'nvar.max'.

`actSetRemove`
: if 'actSet = TRUE', whether we remove the zero coefficients from the current active set. Default is FALSE.

`returnX`
: whether return the standardized design matrix. Default is FALSE.

`trace.lambda`
: whether display the progress for fitting the entire path. Default is FALSE.

`threads`
: number of cores that are used for parallel computing.

`MM`
: whether we use the "Majorize-Minimization" algorithm to optimize the objective function.

`...`
: extra arguments to be passed to function.

Details

The model is fit by Newton method and coordinate descent method.

Value

An object with S3 class `ppLasso`.

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

`gamma`
: the fitted value of provider effects.

`lambda`
: the sequence of 'lambda' values in the path.

`loss`
: the loss of the fitted model at each value of 'lambda'.

`linear.predictors` the linear predictors of the fitted model at each value of 'lambda'.

`df` the estimates of effective number of selected variables all the points along the regularization path.

`iter` the number of iterations until convergence at each value of 'lambda'.

References

K. He, J. Kalbfleisch, Y. Li, and et al. (2013) Evaluating hospital readmission rates in dialysis facilities; adjusting for hospital effects. *Lifetime Data Analysis*, **19**: 490-512.

See Also

`coef` function.

Examples

```
data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
fit <- pp.lasso(data, Y.char, Z.char, prov.char)
# fitted values of covariate coefficients (under the lambda sequence that was automatically generated by the pac
round(fit$beta[1:5, 1:5], 5)
# estimated center effects
round(fit$gamma[1:5, 1:5], 5)
```

---

predict.ppLasso *predictions of a ppLasso or gr_ppLasso object*

Description

return the model predictions of a `ppLasso` or `gr_ppLasso` object

Usage

```
## S3 method for class 'ppLasso'
predict(
  fit,
  data,
  Z.char,
  prov.char,
  lambda,
  which = 1:length(fit$lambda),
  type = c("response", "class", "vars", "nvars"),
  ...
)

## S3 method for class 'gr_ppLasso'
```

```
predict(
  fit,
  data,
  Z.char,
  prov.char,
  lambda,
  which = 1:length(fit$lambda),
 type = c("response", "class", "vars", "groups", "nvars", "ngroups", "beta.norm"),
  ...
)
```

Arguments

fit	a ppLasso or gr_ppLasso or ppDiscSurv object.
data	an 'dataframe' or 'list' object that contains the variables for prediction.
Z.char	names of covariates in 'data' as vector of character strings.
prov.char	name of provider IDs variable in 'data' as a character string.
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: response provides the fitted value; class returns the binomial outcome with the largest probability; vars returns the indices for the non-zero coefficients; nvars returns the number of non-zero coefficients; groups returns the indices for the non-zero groups; ngroups returns the number of non-zero coefficients; beta.norm returns L2 norm of the coefficients in each group
...	
Y.char	name of the response variable in 'data' as a character string.

Examples

```
data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
fit <- pp.lasso(data, Y.char, Z.char, prov.char)
predict(fit, data, Z.char, prov.char, lambda = fit$lambda, type = "response")[1:10, 1:5]
predict(fit, data, Z.char, prov.char, lambda = 0.001, type = "class")[1:10]
predict(fit, data, Z.char, prov.char, lambda = 0.04, type = "vars")

data(GLM_Data)
data <- GLM_Data$data
Y.char <- GLM_Data$Y.char
prov.char <- GLM_Data$prov.char
Z.char <- GLM_Data$Z.char
group <- GLM_Data$group
fit <- grp.lasso(data, Y.char, Z.char, prov.char, group = group)
predict(fit, GLM_Data, Z.char, prov.char, lambda = fit$lambda, type = "response")[1:10, 1:5]
predict(fit, GLM_Data, Z.char, prov.char, lambda = 0.001, type = "class")[1:10]
predict(fit, GLM_Data, Z.char, prov.char, lambda = 0.04, type = "vars")
predict(fit, GLM_Data, Z.char, prov.char, lambda = 0.04, type = "groups")
```

`Surv_Data` *example data for pp.DiscSurv*

Description

A simulated data set containing observation time, event indicator, provider information and 5 covariates.

Usage

```
data(Surv_Data)
```

Format

A list containing the following elements:

data example data.`time` represents the observation time; `status` is the event indicator; `Prov.ID` is the provider indicator; `Z1`, ..., `Z5` are 5 continuous covariates.

Event.char variable name of the event indicator.

prov.char variable name of the provider indicator.

Z.char variable names of covariates.

Time.char variable name of the observation time.

Index