Package 'BranchGLM'

December 7, 2023

Type Package

Title Efficient Branch and Bound Variable Selection for GLMs using 'RcppArmadillo'

Version 2.1.3

Date 2023-12-6

Maintainer Jacob Seedorff <jwseedorff@uiowa.edu>

URL https://github.com/JacobSeedorff21/BranchGLM

BugReports https://github.com/JacobSeedorff21/BranchGLM/issues

Description Performs efficient and scalable glm best subset selection using a novel implementation of a branch and bound algorithm. To speed up the model fitting process, a range of optimization methods are implemented in 'RcppArmadillo'. Parallel computation is available using 'OpenMP'.

License Apache License (>= 2)

Depends R (>= 3.3.0)

Imports Rcpp (>= 1.0.7), methods, stats, graphics

LinkingTo Rcpp, RcppArmadillo, BH

RoxygenNote 7.2.3

Encoding UTF-8

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Jacob Seedorff [aut, cre]

Repository CRAN

Date/Publication 2023-12-07 01:30:02 UTC

R topics documented:

`BranchGLM` *Fits GLMs*

Description

Fits generalized linear models via RcppArmadillo. Also has the ability to fit the models with parallelization via OpenMP.

Usage

```
BranchGLM(
  formula,
  data,
  family,
  link,
  offset = NULL,
  method = "Fisher",
  grads = 10,
```

```
  parallel = FALSE,
  nthreads = 8,
  tol = 1e-06,
  maxit = NULL,
  init = NULL,
  fit = TRUE,
  contrasts = NULL,
  keepData = TRUE,
  keepY = TRUE
)

BranchGLM.fit(
  x,
  y,
  family,
  link,
  offset = NULL,
  method = "Fisher",
  grads = 10,
  parallel = FALSE,
  nthreads = 8,
  init = NULL,
  maxit = NULL,
  tol = 1e-06
)
```

Arguments

formula	a formula for the model.
data	a dataframe that contains the response and predictor variables.
family	distribution used to model the data, one of "gaussian", "gamma", "binomial", or "poisson".
link	link used to link mean structure to linear predictors. One of "identity", "logit", "probit", "cloglog", "sqrt", "inverse", or "log".
offset	offset vector, by default the zero vector is used.
method	one of "Fisher", "BFGS", or "LBFGS". BFGS and L-BFGS are quasi-newton methods which are typically faster than Fisher's scoring when there are many covariates (at least 50).
grads	number of gradients used to approximate inverse information with, only for `method = "LBFGS"`.
parallel	whether or not to make use of parallelization via OpenMP.
nthreads	number of threads used with OpenMP, only used if `parallel = TRUE`.
tol	tolerance used to determine model convergence.
maxit	maximum number of iterations performed. The default for Fisher's scoring is 50 and for the other methods the default is 200.

init	initial values for the betas, if not specified then they are automatically selected via linear regression with the transformation specified by link. This is ignored for linear regression models.
fit	a logical value to indicate whether to fit the model or not. Setting this to false will make it so no coefficients matrix or variance-covariance matrix are returned.
contrasts	see contrasts.arg of model.matrix.default.
keepData	Whether or not to store a copy of data and design matrix, the default is TRUE. If this is FALSE, then the results from this cannot be used inside of VariableSelection.
keepY	Whether or not to store a copy of y, the default is TRUE. If this is FALSE, then the binomial GLM helper functions may not work and this cannot be used inside of VariableSelection.
x	design matrix used for the fit, must be numeric.
y	outcome vector, must be numeric.

Details

Can use BFGS, L-BFGS, or Fisher's scoring to fit the GLM. BFGS and L-BFGS are typically faster than Fisher's scoring when there are at least 50 covariates and Fisher's scoring is typically best when there are fewer than 50 covariates. This function does not currently support the use of weights. In the special case of gaussian regression with identity link the method argument is ignored and the normal equations are solved directly.

The models are fit in C++ by using Rcpp and RcppArmadillo. In order to help convergence, each of the methods makes use of a backtracking line-search using the strong Wolfe conditions to find an adequate step size. There are three conditions used to determine convergence, the first is whether there is a sufficient decrease in the negative log-likelihood, the second is whether the l2-norm of the score is sufficiently small, and the last condition is whether the change in each of the beta coefficients is sufficiently small. The tol argument controls all of these criteria. If the algorithm fails to converge, then iterations will be -1.

All observations with any missing values are removed before model fitting.

The dispersion parameter for gamma regression is estimated via maximum likelihood, very similar to the gamma.dispersion function from the MASS package.

BranchGLM.fit can be faster than calling BranchGLM if the x matrix and y vector are already available, but doesn't return as much information. The object returned by BranchGLM.fit is not of class BranchGLM, so all of the methods for BranchGLM objects such as predict or VariableSelection cannot be used.

Value

BranchGLM returns a BranchGLM object which is a list with the following components

coefficients	a matrix with the coefficient estimates, SEs, Wald test statistics, and p-values
iterations	number of iterations it took the algorithm to converge, if the algorithm failed to converge then this is -1
dispersion	the value of the dispersion parameter
logLik	the log-likelihood of the fitted model

vcov	the variance-covariance matrix of the fitted model
resDev	the residual deviance of the fitted model
AIC	the AIC of the fitted model
preds	predictions from the fitted model
linpreds	linear predictors from the fitted model
tol	tolerance used to fit the model
maxit	maximum number of iterations used to fit the model
formula	formula used to fit the model
method	iterative method used to fit the model
grads	number of gradients used to approximate inverse information for L-BFGS
y	y vector used in the model, not included if keepY = FALSE
x	design matrix used to fit the model, not included if keepData = FALSE
offset	offset vector in the model, not included if keepData = FALSE
fulloffset	supplied offset vector, not included if keepData = FALSE
data	original dataframe supplied to the function, not included if keepData = FALSE
mf	the model frame, not included if keepData = FALSE
numobs	number of observations in the design matrix
names	names of the variables
yname	name of y variable
parallel	whether parallelization was employed to speed up model fitting process
missing	number of missing values removed from the original dataset
link	link function used to model the data
family	family used to model the data
ylevel	the levels of y, only included for binomial glms
xlev	the levels of the factors in the dataset
terms	the terms object used

BranchGLM.fit returns a list with the following components

coefficients	a matrix with the coefficients estimates, SEs, Wald test statistics, and p-values
iterations	number of iterations it took the algorithm to converge, if the algorithm failed to converge then this is -1
dispersion	the value of the dispersion parameter
logLik	the log-likelihood of the fitted model
vcov	the variance-covariance matrix of the fitted model
resDev	the residual deviance of the fitted model
AIC	the AIC of the fitted model
preds	predictions from the fitted model
linpreds	linear predictors from the fitted model
tol	tolerance used to fit the model
maxit	maximum number of iterations used to fit the model

Examples

```
Data <- iris
### Using BranchGLM
BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")

### Using BranchGLM.fit
x <- model.matrix(Sepal.Length ~ ., data = Data)
y <- Data$Sepal.Length
BranchGLM.fit(x, y, family = "gaussian", link = "identity")
```

`Cindex` *Cindex/AUC*

Description

Calculates the c-index/AUC.

Usage

```
Cindex(object, ...)

AUC(object, ...)

## S3 method for class 'numeric'
Cindex(object, y, ...)

## S3 method for class 'BranchGLM'
Cindex(object, ...)

## S3 method for class 'BranchGLMROC'
Cindex(object, ...)
```

Arguments

`object`	a `BranchGLM` object, a `BranchGLMROC` object, or a numeric vector.
`...`	further arguments passed to other methods.
`y`	Observed values, can be a numeric vector of 0s and 1s, a two-level factor vector, or a logical vector.

Details

Uses trapezoidal rule to calculate AUC when given a BranchGLMROC object and uses Mann-Whitney U to calculate it otherwise. The trapezoidal rule method is less accurate, so the two methods may give different results.

Value

A number corresponding to the c-index/AUC.

Examples

```
Data <- ToothGrowth
Fit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "logit")
Cindex(Fit)
AUC(Fit)
```

`coef.BranchGLM` *Extract Coefficients*

Description

Extract Coefficients

Usage

```
## S3 method for class 'BranchGLM'
coef(object, ...)
```

Arguments

`object`	a `BranchGLM` object.
`...`	further arguments passed to or from other methods.

Value

A named vector with the corresponding coefficient estimates.

`coef.BranchGLMVS` *Extract Coefficients*

Description

Extract Coefficients

Usage

```
## S3 method for class 'BranchGLMVS'
coef(object, which = 1, ...)

## S3 method for class 'summary.BranchGLMVS'
coef(object, which = 1, ...)
```

Arguments

object	a BranchGLMVS or summary.BranchGLMVS object.
which	which models to get coefficients from, the default is the best model. Can specify "all" to get coefficients from all of the best models.
...	ignored.

Value

A numeric matrix with the corresponding coefficient estimates.

Examples

```
Data <- iris
Fit <- BranchGLM(Sepal.Length ~ ., data = Data,
family = "gaussian", link = "identity")

# Doing branch and bound selection
VS <- VariableSelection(Fit, type = "branch and bound", metric = "BIC",
bestmodels = 10, showprogress = FALSE)

## Getting coefficients from best model
coef(VS, which = 1)

## Getting coefficients from all best models
coef(VS, which = "all")
```

---

confint.BranchGLM *Likelihood Ratio Confidence Intervals for Beta Coefficients*

Description

Likelihood Ratio Confidence Intervals for Beta Coefficients

Usage

```
## S3 method for class 'BranchGLM'
confint(object, parm, level = 0.95, parallel = FALSE, nthreads = 8, ...)
```

Arguments

object	a BranchGLM object.
parm	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
level	the confidence level required.
parallel	whether or not to make use of parallelization via OpenMP.

nthreads	number of threads used with OpenMP, only used if parallel = TRUE.
...	further arguments passed from other methods.

Details

Endpoints of the confidence intervals that couldn't be found by the algorithm are filled in with NA. When there is a lot of multicollinearity in the data the algorithm may have problems finding many of the intervals.

Value

An object of class BranchGLMCIs which is a list with the following components.

CIs	a matrix with the confidence intervals
level	the supplied level
MLE	a numeric vector of the MLEs of the coefficients

Examples

```
Data <- iris
### Fitting linear regression model
mymodel <- BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")

### Getting confidence intervals
CIs <- confint(mymodel, level = 0.95)
CIs

### Plotting CIs
plot(CIs, mary = 7, cex.y = 0.9)
```

---

fit.BranchGLMVS *Fits GLMs for summary.BranchGLMVS and BranchGLMVS objects*

Description

Fits GLMs for summary.BranchGLMVS and BranchGLMVS objects

Usage

```
## S3 method for class 'BranchGLMVS'
fit(object, which = 1, keepData = TRUE, keepY = TRUE, ...)

fit(object, ...)

## S3 method for class 'summary.BranchGLMVS'
fit(object, which = 1, keepData = TRUE, keepY = TRUE, ...)
```

Arguments

object	a summary.BranchGLMVS or BranchGLMVS object.
which	a positive integer indicating which model to fit, the default is to fit the first model .
keepData	Whether or not to store a copy of data and design matrix, the default is TRUE. If this is FALSE, then the results from this cannot be used inside of VariableSelection.
keepY	Whether or not to store a copy of y, the default is TRUE. If this is FALSE, then the binomial GLM helper functions may not work and this cannot be used inside of VariableSelection.
...	further arguments passed to other methods.

Details

The information needed to fit the GLM is taken from the original information supplied to the VariableSelection function.

The fitted models do not have standard errors or p-values since these are biased due to the selection process.

Value

An object of class BranchGLM.

---

logLik.BranchGLM *Extract Log-Likelihood*

Description

Extract Log-Likelihood

Usage

```
## S3 method for class 'BranchGLM'
logLik(object, ...)
```

Arguments

object	a BranchGLM object.
...	further arguments passed to or from other methods.

Value

An object of class logLik which is a number corresponding to the log-likelihood with the following attributes: "df" (degrees of freedom) and "nobs" (number of observations).

`MultipleROCCurves` *Plotting Multiple ROC Curves*

Description

Plotting Multiple ROC Curves

Usage

```
MultipleROCCurves(
  ...,
  legendpos = "bottomright",
  title = "ROC Curves",
  colors = NULL,
  names = NULL,
  lty = 1,
  lwd = 1
)
```

Arguments

`...`	any number of `BranchGLMROC` objects.
`legendpos`	a keyword to describe where to place the legend, such as "bottomright". The default is "bottomright"
`title`	title for the plot.
`colors`	vector of colors to be used on the ROC curves.
`names`	vector of names used to create a legend for the ROC curves.
`lty`	vector of linetypes used to create the ROC curves or a single linetype to be used for all ROC curves.
`lwd`	vector of linewidths used to create the ROC curves or a single linewidth to be used for all ROC curves.

Value

No return value, called to create the plot.

Examples

```
Data <- ToothGrowth

### Logistic ROC
LogisticFit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "logit")
LogisticROC <- ROC(LogisticFit)

### Probit ROC
ProbitFit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "probit")
```

```
ProbitROC <- ROC(ProbitFit)

### Cloglog ROC
CloglogFit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "cloglog")
CloglogROC <- ROC(CloglogFit)

### Plotting ROC curves

MultipleROCCurves(LogisticROC, ProbitROC, CloglogROC,
                  names = c("Logistic ROC", "Probit ROC", "Cloglog ROC"))
```

`plot.BranchGLMCIs` *Plot Method for BranchGLMCIs Objects*

Description

Plot Method for BranchGLMCIs Objects

Usage

```
## S3 method for class 'BranchGLMCIs'
plot(x, which = "all", mary = 5, ...)
```

Arguments

`x`	a `BranchGLMCIs` object.
`which`	which intervals to plot, can use indices or names of desired variables.
`mary`	value used to determine how large to make margin of y-axis. If variable names are cut-off, consider increasing this from the default value of 5.
`...`	further arguments passed to plotCI.

Value

This only produces a plot, nothing is returned.

Examples

```
Data <- iris
### Fitting linear regression model
mymodel <- BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")

### Getting confidence intervals
CIs <- confint(mymodel, level = 0.95)
CIs

### Plotting CIs
plot(CIs, mary = 7, cex.y = 0.9)
```

plot.BranchGLMROC *Plotting ROC Curve*

Description

This plots a ROC curve.

Usage

```
## S3 method for class 'BranchGLMROC'
plot(x, ...)
```

Arguments

x	a BranchGLMROC object.
...	arguments passed to generic plot function.

Value

No return value, called to create the plot.

Examples

```
Data <- ToothGrowth
Fit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "logit")
MyROC <- ROC(Fit)
plot(MyROC)
```

plot.BranchGLMVS *Plot Method for summary.BranchGLMVS and BranchGLMVS objects*

Description

Plot Method for summary.BranchGLMVS and BranchGLMVS objects

Usage

```
## S3 method for class 'BranchGLMVS'
plot(
  x,
  ptype = "both",
  marnames = 7,
  addLines = TRUE,
  type = "b",
  horiz = FALSE,
  cex.names = 1,
```

```
  cex.lab = 1,
  cex.axis = 1,
  cex.legend = 1,
  cols = c("deepskyblue", "indianred", "forestgreen"),
  ...
)

## S3 method for class 'summary.BranchGLMVS'
plot(
  x,
  ptype = "both",
  marnames = 7,
  addLines = TRUE,
  type = "b",
  horiz = FALSE,
  cex.names = 1,
  cex.lab = 1,
  cex.axis = 1,
  cex.legend = 1,
  cols = c("deepskyblue", "indianred", "forestgreen"),
  ...
)
```

Arguments

x	a summary.BranchGLMVS or BranchGLMVS object.
ptype	the type of plot to produce, look at details for more explanation.
marnames	value used to determine how large to make margin of axis with variable names, this is only for the "variables" plot. If variable names are cut-off, consider increasing this from the default value of 7.
addLines	logical value to indicate whether or not to add black lines to separate the models for the "variables" plot. This is typically useful for smaller amounts of models, but can be annoying if there are many models.
type	what type of plot to draw for the "metrics" plot, see more details at plot.default.
horiz	whether models should be displayed horizontally or vertically in the "variables" plot.
cex.names	how big to make variable names in the "variables" plot.
cex.lab	how big to make axis labels.
cex.axis	how big to make axis annotation.
cex.legend	how big to make legend labels.
cols	the colors used to create the "variables" plot. Should be a character vector of length 3, the first color will be used for included variables, the second color will be used for excluded variables, and the third color will be used for kept variables.
...	further arguments passed to the generic plot and image methods.

Details

The different values for ptype are as follows

- "metrics" for a plot that displays the metric values ordered by rank
- "variables" for a plot that displays which variables are in each of the top models
- "both" for both plots

If there are so many models that the "variables" plot appears to be entirely black, then set addLines to FALSE.

Value

This only produces plots, nothing is returned.

Examples

```
Data <- iris
Fit <- BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")

# Doing branch and bound selection
VS <- VariableSelection(Fit, type = "branch and bound", metric = "BIC", bestmodels = 10,
showprogress = FALSE)
VS

## Getting summary of the process
Summ <- summary(VS)
Summ

## Plotting the BIC of best models
plot(Summ, type = "b", ptype = "metrics")

## Plotting the BIC of the best models
plot(Summ, ptype = "variables")

### Alternative colors
plot(Summ, ptype = "variables",
cols = c("yellowgreen", "purple1", "grey50"))

### Smaller text size for names
plot(Summ, ptype = "variables", cex.names = 0.75)
```

---

plotCI *Plot Confidence Intervals*

Description

Plot Confidence Intervals

Usage

```
plotCI(
  CIs,
  points = NULL,
  ylab = "",
  las = 2,
  cex.y = 1,
  decreasing = FALSE,
  ...
)
```

Arguments

CIs	a matrix of confidence intervals, must have either 2 rows or 2 columns. The variable names displayed in the plot are taken from either the column names or row names of this.
points	points to be plotted in the middle of the CIs, typically means or medians. The default is to plot the midpoint of the intervals.
ylab	axis label for y-axis.
las	the style of the y-axis label, the default is horizontal, see more about this at par.
cex.y	font size used for variable names on y-axis.
decreasing	a logical value indicating if confidence intervals should be displayed in decreasing or increasing order according to points. Can use NA if no ordering is desired.
...	further arguments passed to default plot method.

Value

This only produces a plot, nothing is returned.

Examples

```
Data <- iris
### Fitting linear regression model
mymodel <- BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")

### Getting confidence intervals
CIs <- confint.default(mymodel, level = 0.95)
xlim <- c(min(CIs), max(CIs))

### Plotting CIs
par(mar = c(5, 7, 3, 1) + 0.1)
plotCI(CIs, main = "95% Confidence Intervals", xlim = xlim, cex.y = 0.9,
xlab = "Beta Coefficients")
abline(v = 0)
```

predict.BranchGLM *Predict Method for BranchGLM Objects*

Description

Gets predictions from a BranchGLM object.

Usage

```
## S3 method for class 'BranchGLM'
predict(
  object,
  newdata = NULL,
  offset = NULL,
  type = "response",
  na.action = na.pass,
  ...
)
```

Arguments

object	a BranchGLM object.
newdata	a dataframe, if not specified then the data the model was fit on is used.
offset	a numeric vector containing the offset variable, this is ignored if newdata is not supplied.
type	one of "linpreds" or "response", if not specified then "response" is used.
na.action	a function which indicates what should happen when the data contains NAs. The default is na.pass. This is ignored if newdata is not supplied and data isn't included in the supplied BranchGLM object.
...	further arguments passed to or from other methods.

Details

linpreds corresponds to the linear predictors and response is on the scale of the response variable. Offset variables are ignored for predictions on new data.

Value

A numeric vector of predictions.

Examples

```
Data <- iris
Fit <- BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")
predict(Fit)
### Example with new data
predict(Fit, newdata = iris[1:20,])
```

predict.BranchGLMVS *Predict Method for BranchGLMVS Objects*

Description

Gets predictions from a BranchGLMVS or summary.BranchGLMVS object.

Usage

```
## S3 method for class 'BranchGLMVS'
predict(object, which = 1, ...)

## S3 method for class 'summary.BranchGLMVS'
predict(object, which = 1, ...)
```

Arguments

object	a BranchGLMVS or summary.BranchGLMVS object.
which	which model to get predictions from, the default is the best model.
...	further arguments passed to predict.BranchGLM.

Value

A numeric vector of predictions.

Examples

```
Data <- iris
Fit <- BranchGLM(Sepal.Length ~ ., data = Data,
family = "gamma", link = "log")

# Doing branch and bound selection
VS <- VariableSelection(Fit, type = "branch and bound", metric = "BIC",
bestmodels = 10, showprogress = FALSE)

## Getting predictions from best model
predict(VS, which = 1)

## Getting linear predictors from 5th best model
predict(VS, which = 5, type = "linpreds")
```

print.BranchGLM *Print Method for BranchGLM*

Description

Print Method for BranchGLM

Usage

```
## S3 method for class 'BranchGLM'
print(x, coefdigits = 4, digits = 2, ...)
```

Arguments

x	a BranchGLM object.
coefdigits	number of digits to display for coefficients table.
digits	number of digits to display for information after table.
...	further arguments passed to or from other methods.

Value

The supplied BranchGLM object.

print.BranchGLMCIs *Print Method for BranchGLMCIs Objects*

Description

Print Method for BranchGLMCIs Objects

Usage

```
## S3 method for class 'BranchGLMCIs'
print(x, digits = 4, ...)
```

Arguments

x	a BranchGLMCIs object.
digits	number of significant digits to display.
...	further arguments passed from other methods.

Value

The supplied BranchGLMCIs object.

print.BranchGLMROC *Print Method for BranchGLMROC*

Description

Print Method for BranchGLMROC

Usage

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

Arguments

x	a BranchGLMROC object.
...	further arguments passed to other methods.

Value

The supplied BranchGLMROC object.

print.BranchGLMTable *Print Method for BranchGLMTable*

Description

Print Method for BranchGLMTable

Usage

```
## S3 method for class 'BranchGLMTable'
print(x, digits = 4, ...)
```

Arguments

x	a BranchGLMTable object.
digits	number of digits to display.
...	further arguments passed to other methods.

Value

The supplied BranchGLMTable object.

print.BranchGLMVS *Print Method for BranchGLMVS*

Description

Print Method for BranchGLMVS

Usage

```
## S3 method for class 'BranchGLMVS'
print(x, coefdigits = 4, digits = 2, ...)
```

Arguments

x	a BranchGLMVS object.
coefdigits	number of digits to display for coefficients table.
digits	number of digits to display for information not in the table.
...	further arguments passed to other methods.

Value

The supplied BranchGLMVS object.

print.summary.BranchGLMVS *Print Method for summary.BranchGLMVS*

Description

Print Method for summary.BranchGLMVS

Usage

```
## S3 method for class 'summary.BranchGLMVS'
print(x, digits = 4, ...)
```

Arguments

x	a summary.BranchGLMVS object.
digits	number of digits to display for information in the table.
...	further arguments passed to other methods.

Value

The supplied summary.BranchGLMVS object.

ROC *ROC Curve*

Description

Creates an ROC curve.

Usage

```
ROC(object, ...)

## S3 method for class 'numeric'
ROC(object, y, ...)

## S3 method for class 'BranchGLM'
ROC(object, ...)
```

Arguments

object	a BranchGLM object or a numeric vector.
...	further arguments passed to other methods.
y	observed values, can be a numeric vector of 0s and 1s, a two-level factor vector, or a logical vector.

Value

A BranchGLMROC object which can be plotted with plot(). The AUC can also be calculated using AUC().

Examples

```
Data <- ToothGrowth
Fit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "logit")
MyROC <- ROC(Fit)
plot(MyROC)
```

summary.BranchGLMVS *Summary Method for BranchGLMVS*

Description

Summary Method for BranchGLMVS

Usage

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

Arguments

`object`	a `BranchGLMVS` object.
`...`	further arguments passed to other methods.

Value

An object of class `summary.BranchGLMVS` which is a list with the following components

`results`	a data frame which has the metric values for the best models along with the variables included in each model
`initmodel`	the initial `BranchGLM` object that was supplied to the `VariableSelection` function
`VS`	the supplied `BranchGLMVS` object
`formulas`	a list containing the formulas of the best models
`metric`	the metric used to perform variable selection

Examples

```
Data <- iris
Fit <- BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")

# Doing branch and bound selection
VS <- VariableSelection(Fit, type = "branch and bound", metric = "BIC",
bestmodels = 10, showprogress = FALSE)
VS

## Getting summary of the process
Summ <- summary(VS)
Summ

## Plotting the BIC of the best models
plot(Summ, type = "b")

## Plotting the variables in the best models
plot(Summ, ptype = "variables")

## Getting coefficients
coef(Summ)
```

`Table` *Confusion Matrix*

Description

Creates confusion matrix and calculates related measures.

Usage

```
Table(object, ...)

## S3 method for class 'numeric'
Table(object, y, cutoff = 0.5, ...)

## S3 method for class 'BranchGLM'
Table(object, cutoff = 0.5, ...)
```

Arguments

object	a BranchGLM object or a numeric vector.
...	further arguments passed to other methods.
y	observed values, can be a numeric vector of 0s and 1s, a two-level factor vector, or a logical vector.
cutoff	cutoff for predicted values, the default is 0.5.

Value

A BranchGLMTable object which is a list with the following components

table	a matrix corresponding to the confusion matrix
accuracy	a number corresponding to the accuracy
sensitivity	a number corresponding to the sensitivity
specificity	a number corresponding to the specificity
PPV	a number corresponding to the positive predictive value
levels	a vector corresponding to the levels of the response variable

Examples

```
Data <- ToothGrowth
Fit <- BranchGLM(supp ~ ., data = Data, family = "binomial", link = "logit")
Table(Fit)
```

VariableSelection *Variable Selection for GLMs*

Description

Performs forward selection, backward elimination, and efficient best subsets variable selection with information criterion for generalized linear models. Best subsets selection is performed with branch and bound algorithms to greatly speed up the process.

Usage

```
VariableSelection(object, ...)

## S3 method for class 'formula'
VariableSelection(
  object,
  data,
  family,
  link,
  offset = NULL,
  method = "Fisher",
  type = "switch branch and bound",
  metric = "AIC",
  bestmodels = NULL,
  cutoff = NULL,
  keep = NULL,
  keepintercept = TRUE,
  maxsize = NULL,
  grads = 10,
  parallel = FALSE,
  nthreads = 8,
  tol = 1e-06,
  maxit = NULL,
  contrasts = NULL,
  showprogress = TRUE,
  ...
)

## S3 method for class 'BranchGLM'
VariableSelection(
  object,
  type = "switch branch and bound",
  metric = "AIC",
  bestmodels = NULL,
  cutoff = NULL,
  keep = NULL,
  keepintercept = TRUE,
  maxsize = NULL,
  parallel = FALSE,
  nthreads = 8,
  showprogress = TRUE,
  ...
)
```

Arguments

object	a formula or a BranchGLM object.
...	further arguments passed from other methods.

data	a dataframe with the response and predictor variables.
family	distribution used to model the data, one of "gaussian", "gamma", "binomial", or "poisson".
link	link used to link mean structure to linear predictors. One of "identity", "logit", "probit", "cloglog", "sqrt", "inverse", or "log".
offset	offset vector, by default the zero vector is used.
method	one of "Fisher", "BFGS", or "LBFGS". Fisher's scoring is recommended for forward selection and branch and bound methods since they will typically fit many models with a small number of covariates.
type	one of "forward", "backward", "branch and bound", "backward branch and bound", or "switch branch and bound" to indicate the type of variable selection to perform. The default value is "switch branch and bound". The branch and bound algorithms are guaranteed to find the best models according to the metric while "forward" and "backward" are heuristic approaches that may not find the optimal model.
metric	metric used to choose the best models, the default is "AIC", but "BIC" and "HQIC" are also available. AIC is the Akaike information criterion, BIC is the Bayesian information criterion, and HQIC is the Hannan-Quinn information criterion.
bestmodels	a positive number to indicate the number of the best models to find according to the chosen metric or NULL. If this is NULL, then cutoff is used instead. This is only used for the branch and bound methods.
cutoff	this is a non-negative number which indicates that the function should return all models that have a metric value within cutoff of the best metric value or NULL. Only one of this or bestmodels should be specified and when both are NULL a cutoff of 0 is used. This is only used for the branch and bound methods.
keep	vector of names to denote variables that must be in the models.
keepintercept	whether to keep the intercept in all models, only used if an intercept is included in the formula.
maxsize	maximum number of variables to consider in a single model, the default is the total number of variables. This number adds onto any variables specified in keep. This argument only works for `type = "forward"` and `type = "branch and bound"`.
grads	number of gradients used to approximate inverse information with, only for `method = "LBFGS"`.
parallel	one of TRUE or FALSE to indicate if parallelization should be used
nthreads	number of threads used with OpenMP, only used if `parallel = TRUE`.
tol	tolerance used to determine model convergence when fitting GLMs.
maxit	maximum number of iterations performed when fitting GLMs. The default for Fisher's scoring is 50 and for the other methods the default is 200.
contrasts	see `contrasts.arg` of `model.matrix.default`.
showprogress	whether to show progress updates for branch and bound methods.

Details

The supplied formula or the formula from the fitted model is treated as the upper model. The variables specified in keep along with an intercept (if included in formula and keepintercept = TRUE) is the lower model. Factor variables are either kept in their entirety or entirely removed and interaction terms are properly handled.

The branch and bound method makes use of an efficient branch and bound algorithm to find the optimal models. This is will find the best models according to the metric and can be much faster than an exhaustive search and can be made even faster with parallel computation. The backward branch and bound method is very similar to the branch and bound method, except it tends to be faster when the best models contain most of the variables. The switch branch and bound method is a combination of the two methods and is typically the fastest of the 3 branch and bound methods.

Fisher's scoring is recommended for branch and bound selection and forward selection. L-BFGS may be faster for backward elimination, especially when there are many variables.

All observations that have any missing values in the upper model are removed.

Value

A `BranchGLMVS` object which is a list with the following components

`initmodel`	the supplied `BranchGLM` object or a fake `BranchGLM` object if a formula is supplied
`numchecked`	number of models fit
`names`	character vector of the names of the predictor variables
`order`	the order the variables were added to the model or removed from the model, this is not included for branch and bound selection
`type`	type of variable selection employed
`metric`	metric used to select best models
`bestmodels`	numeric matrix used to describe the best models
`bestmetrics`	numeric vector with the best metrics found in the search
`beta`	numeric matrix of beta coefficients for the best models
`cutoff`	the cutoff that was used, this is set to -1 if bestmodels was used instead
`keep`	vector of which variables are kept through the selection process

Examples

```
Data <- iris
Fit <- BranchGLM(Sepal.Length ~ ., data = Data, family = "gaussian", link = "identity")

# Doing branch and bound selection
VS <- VariableSelection(Fit, type = "branch and bound", metric = "BIC",
bestmodels = 10, showprogress = FALSE)
VS

## Plotting the BIC of the best models
plot(VS, type = "b")
```

```
## Getting the coefficients of the best model according to BIC
FinalModel <- coef(VS, which = 1)
FinalModel

# Now doing it in parallel (although it isn't necessary for this dataset)
parVS <- VariableSelection(Fit, type = "branch and bound", parallel = TRUE, metric = "BIC",
bestmodels = 10, showprogress = FALSE)

## Getting the coefficients of the best model according to BIC
FinalModel <- coef(parVS, which = 1)
FinalModel

# Using a formula
formVS <- VariableSelection(Sepal.Length ~ ., data = Data, family = "gaussian",
link = "identity", metric = "BIC", type = "branch and bound", bestmodels = 10, showprogress = FALSE)

## Getting the coefficients of the best model according to BIC
FinalModel <- coef(formVS, which = 1)
FinalModel

# Using the keep argument
keepVS <- VariableSelection(Fit, type = "branch and bound", keep = "Petal.Width",
metric = "BIC", bestmodels = 5, showprogress = FALSE)
keepVS

## Getting the coefficients from the fifth best model according to BIC when
## keeping Petal.Width in every model
FinalModel <- coef(keepVS, which = 5)
FinalModel
```

`vcov.BranchGLM` *Extract covariance matrix*

Description

Extract covariance matrix

Usage

```
## S3 method for class 'BranchGLM'
vcov(object, ...)
```

Arguments

`object`	a `BranchGLM` object.
`...`	further arguments passed to or from other methods.

Value

A numeric matrix which is the covariance matrix of the beta coefficients.

Index