Package ‘BranchGLM’

April 8, 2024

Type Package

Title Efficient Best Subset Selection for GLMs via Branch and Bound Algorithms

Version 2.1.5

Date 2024-4-7

Maintainer Jacob Seedorff <jacob-seedorff@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 2024-04-08 03:53:06 UTC

R topics documented:

`BranchGLM` *Fits GLMs*

Description

Fits generalized linear models (GLMs) via RcppArmadillo with the ability to perform some computation in parallel with 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 data.frame, list or environment (or object coercible by as.data.frame to a data.frame), containing the variables in formula. Neither a matrix nor an array will be accepted.
`family`	the distribution used to model the data, one of "gaussian", "gamma", "binomial", or "poisson".
`link`	the link used to link the mean structure to the linear predictors. One of "identity", "logit", "probit", "cloglog", "sqrt", "inverse", or "log". The accepted links depend on the specified family, see more in details.
`offset`	the 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`	a positive integer to denote the number of gradients used to approximate the inverse information with, only for `method = "LBFGS"`.
`parallel`	a logical value to indicate if parallelization should be used.
`nthreads`	a positive integer to denote the number of threads used with OpenMP, only used if `parallel = TRUE`.

`tol`	a positive number to denote the tolerance used to determine model convergence.
`maxit`	a positive integer to denote the maximum number of iterations performed. The default for Fisher's scoring is 50 and for the other methods the default is 200.
`init`	a numeric vector of initial values for the betas, if not specified then they are automatically selected via linear regression with the transformation specified by the link function. This is ignored for linear regression models.
`fit`	a logical value to indicate whether to fit the model or not.
`contrasts`	see `contrasts.arg` of `model.matrix.default`.
`keepData`	a logical value to indicate whether or not to store a copy of data and the design matrix, the default is TRUE. If this is FALSE, then the results from this cannot be used inside of `VariableSelection`.
`keepY`	a logical value to indicate 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

Fitting:

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.

`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.

Dispersion Parameter:

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

Families and Links:

The binomial family accepts "cloglog", "log", "logit", and "probit" as possible link functions. The gamma and gaussian families accept "identity", "inverse", "log", and "sqrt" as possible link functions. The Poisson family accepts "identity", "log", and "sqrt" as possible link functions.

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 `data` argument 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 predictor 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

References

McCullagh, P., & Nelder, J. A. (1989). Generalized Linear Models (2nd ed.). Chapman & Hall.

See Also

predict.BranchGLM, coef.BranchGLM, VariableSelection, confint.BranchGLM, logLik.BranchGLM

Examples

```
Data <- iris

# Linear regression
## 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")

# Gamma regression
## Using BranchGLM
BranchGLM(Sepal.Length ~ ., data = Data, family = "gamma", link = "log")

### init
BranchGLM(Sepal.Length ~ ., data = Data, family = "gamma", link = "log",
init = rep(0, 6), maxit = 50, tol = 1e-6, contrasts = NULL)

### method
BranchGLM(Sepal.Length ~ ., data = Data, family = "gamma", link = "log",
init = rep(0, 6), maxit = 50, tol = 1e-6, contrasts = NULL, method = "LBFGS")

### offset
BranchGLM(Sepal.Length ~ ., data = Data, family = "gamma", link = "log",
init = rep(0, 6), maxit = 50, tol = 1e-6, contrasts = NULL,
offset = Data$Sepal.Width)
```

```
## Using BranchGLM.fit
x <- model.matrix(Sepal.Length ~ ., data = Data)
y <- Data$Sepal.Length
BranchGLM.fit(x, y, family = "gamma", link = "log", init = rep(0, 6),
maxit = 50, tol = 1e-6, offset = Data$Sepal.Width)
```

`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 from BranchGLM Objects*

Description

Extracts beta coefficients from BranchGLM objects.

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 from BranchGLMVS or summary.BranchGLMVS Objects*

Description

Extracts beta coefficients from BranchGLMVS or summary.BranchGLMVS objects.

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`	a numeric vector of indices or "all" to indicate which models to get coefficients from, the default is 1 which is used for the best model. For the branch and bound algorithms the number k is used for the kth best model and for the stepwise algorithms the number k is used for the model that is k - 1 steps away from the final model.
`...`	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 for BranchGLM Objects*

Description

Finds profile likelihood ratio confidence intervals for beta coefficients with the ability to calculate the intervals in parallel.

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`	a logical value to indicate if parallelization should be used.
`nthreads`	a positive integer to denote the 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 numeric matrix with the confidence intervals
`level`	the supplied level
`MLE`	a numeric vector of the MLEs of the coefficients

See Also

plot.BranchGLMCIs, plotCI

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)
```

`formula.BranchGLM` *Extract Model Formula from BranchGLM Objects*

Description

Extracts model formula from BranchGLM objects.

Usage

```
## S3 method for class 'BranchGLM'
formula(x, ...)
```

Arguments

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

Value

a formula representing the model used to obtain `object`.

`logLik.BranchGLM` *Extract Log-Likelihood from BranchGLM Objects*

Description

Extracts log-likelihood from BranchGLM objects.

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

This only produces a plot, nothing is returned.

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"))
```

`nobs.BranchGLM` *Extract Number of Observations from BranchGLM Objects*

Description

Extracts number of observations from BranchGLM objects.

Usage

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

Arguments

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

Value

A single number indicating the number of observations used to fit the model.

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

Description

Creates a plot to visualize confidence intervals from 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 a numeric vector of indices, a character vector of names of desired variables, or "all" to plot all intervals.
`mary`	a numeric 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.

See Also

plotCI

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` *Plot Method for BranchGLMROC Objects*

Description

This plots a ROC curve.

Usage

```
## S3 method for class 'BranchGLMROC'
plot(x, xlab = "1 - Specificity", ylab = "Sensitivity", type = "l", ...)
```

Arguments

`x`	a `BranchGLMROC` object.
`xlab`	label for the x-axis.
`ylab`	label for the y-axis.
`type`	what type of plot to draw, see more details at plot.default.
`...`	further arguments passed to plot.default.

Value

This only produces a plot, nothing is returned.

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

Creates plots to help visualize variable selection results from BranchGLMVS or summary.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`	a numeric 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`	a 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`	a logical value to indicate whether models should be displayed horizontally for 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 plot.default for the "metrics" plot and image.default for the "variables" plot.

Details

The different values for ptype are as follows

- "metrics" for a plot that displays the metric values ordered by rank for the branch and bound algorithms or a plot which displays the metric values in the path taken by the stepwise algorithms
- "variables" for a plot that displays which variables are in each of the top models for the branch and bound algorithms or a plot which displays the path taken by the stepwise algorithms
- "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

Creates a plot to display confidence intervals.

Usage

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

Arguments

`CIs`	a numeric matrix of confidence intervals, must have exactly 2 columns. The variable names displayed in the plot are taken from the column names.
`points`	points to be plotted in the middle of the CIs, typically means or medians. The default is to plot the midpoints of the intervals.
`ylab`	a label for the y-axis.
`ylas`	the style of the y-axis label, see more about this at `las` in 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 plot.default.

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

Obtains predictions from `BranchGLM` objects.

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 data.frame, 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" which is on the scale of the linear predictors or "response" which is on the scale of the response variable. 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.

Value

A numeric vector of predictions.

Examples

```
Data <- airquality

# Example without offset
Fit <- BranchGLM(Temp ~ ., data = Data, family = "gaussian", link = "identity")

## Using default na.action
predict(Fit)

## Using na.omit
predict(Fit, na.action = na.omit)

## Using new data
predict(Fit, newdata = Data[1:20, ], na.action = na.pass)

# Using offset
FitOffset <- BranchGLM(Temp ~ . - Day, data = Data, family = "gaussian",
link = "identity", offset = Data$Day * -0.1)

## Getting predictions for data used to fit model
### Don't need to supply offset vector
predict(FitOffset)

## Getting predictions for new dataset
### Need to include new offset vector since we are
### getting predictions for new dataset
predict(FitOffset, newdata = Data[1:20, ], offset = Data$Day[1:20] * -0.1)
```

`predict.BranchGLMVS` *Predict Method for BranchGLMVS or summary.BranchGLMVS Objects*

Description

Obtains predictions from BranchGLMVS or summary.BranchGLMVS objects.

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`	a positive integer to indicate which model to get predictions from, the default is 1 which is used for the best model. For the branch and bound algorithms the number k is used for the kth best model and for the stepwise algorithms the number k is used for the model that is k - 1 steps away from the final model.
`...`	further arguments passed to predict.BranchGLM.

Value

A numeric vector of predictions.

See Also

predict.BranchGLM

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 Objects*

Description

Print method for `BranchGLM` objects.

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 Objects*

Description

Print method for BranchGLMROC objects.

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 Objects*

Description

Print method for BranchGLMTable objects.

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 Objects*

Description

Print method for BranchGLMVS objects.

Usage

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

Arguments

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

Value

The supplied `BranchGLMVS` object.

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

Description

Print method for summary.BranchGLMVS objects.

Usage

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

Arguments

`x`	a `summary.BranchGLMVS` object.
`digits`	number of digits to display.
`...`	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 Objects*

Description

Summary method for BranchGLMVS objects.

Usage

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

Arguments

`object`	a `BranchGLMVS` object.
`...`	further arguments passed to or from 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 sets of variables included in each model
`VS`	the supplied `BranchGLMVS` object
`formulas`	a list containing the formulas of the best models
`metric`	the metric used to perform variable selection

See Also

plot.summary.BranchGLMVS, coef.summary.BranchGLMVS, predict.summary.BranchGLMVS

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 a 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 subset variable selection with information criterion for generalized linear models (GLMs). Best subset 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.
`data`	a data.frame, list or environment (or object coercible by as.data.frame to a data.frame), containing the variables in formula. Neither a matrix nor an array will be accepted.
`family`	the distribution used to model the data, one of "gaussian", "gamma", "binomial", or "poisson".
`link`	the link used to link the mean structure to the linear predictors. One of "identity", "logit", "probit", "cloglog", "sqrt", "inverse", or "log".
`offset`	the 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 the branch and bound algorithms 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". See more about these algorithms in details
`metric`	the 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 integer 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 algorithms.
`cutoff`	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 algorithms.
`keep`	a character vector of names to denote variables that must be in the models.
`keepintercept`	a logical value to indicate whether to keep the intercept in all models, only used if an intercept is included in the formula.
`maxsize`	a positive integer to denote the 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"`. This argument is now deprecated.

`grads`	a positive integer to denote the number of gradients used to approximate the inverse information with, only for `method = "LBFGS"`..
`parallel`	a logical value to indicate if parallelization should be used.
`nthreads`	a positive integer to denote the number of threads used with OpenMP, only used if `parallel = TRUE`.
`tol`	a positive number to denote the tolerance used to determine model convergence.
`maxit`	a positive integer to denote the maximum number of iterations performed. 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`	a logical value to indicate whether to show progress updates for branch and bound algorithms.

Details

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

Branch and Bound Algorithms:

The branch and bound algorithm is an efficient algorithm used to find the optimal models. The backward branch and bound algorithm is very similar to the branch and bound algorithm, except it tends to be faster when the best models contain most of the variables. The switch branch and bound algorithm is a combination of the two algorithms and is typically the fastest of the 3 branch and bound algorithms. All of the branch and bound algorithms are guaranteed to find the optimal models (up to numerical precision).

GLM Fitting:

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.

Value

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

`initmodel`	the `BranchGLM` object corresponding to the upper model
`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 only included for the stepwise algorithms
`type`	type of variable selection employed
`optType`	whether the type specified used a heuristic or exact algorithm
`metric`	metric used to select best models

`bestmodels`	numeric matrix used to describe the best models for the branch and bound algorithms or a numeric matrix describing the models along the path taken for stepwise algorithms
`bestmetrics`	numeric vector with the best metrics found in the search for the branch and bound algorithms or a numeric vector with the metric values along the path taken for stepwise algorithms
`beta`	numeric matrix of beta coefficients for the models in bestmodels
`cutoff`	the cutoff that was used, this is set to -1 if bestmodels was used instead or if a stepwise algorithm was used
`keep`	vector of which variables were kept through the selection process

See Also

plot.BranchGLMVS, coef.BranchGLMVS, predict.BranchGLMVS, summary.BranchGLMVS

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 = c("Species", "Petal.Width"), metric = "BIC", bestmodels = 4,
showprogress = FALSE)
keepVS

## Getting the coefficients from the fourth best model according to BIC when
## keeping Petal.Width and Species in every model
FinalModel <- coef(keepVS, which = 4)
FinalModel

# Treating categorical variable beta parameters separately
## This function automatically groups together parameters from a categorical variable
## to avoid this, you need to create the indicator variables yourself
x <- model.matrix(Sepal.Length ~ ., data = iris)
Sepal.Length <- iris$Sepal.Length
Data <- cbind.data.frame(Sepal.Length, x[, -1])
VSCat <- VariableSelection(Sepal.Length ~ ., data = Data, family = "gaussian",
link = "identity", metric = "BIC", bestmodels = 10, showprogress = FALSE)
VSCat

## Plotting results
plot(VSCat, cex.names = 0.75)
```

`vcov.BranchGLM` *Extract covariance matrix from BranchGLM Objects*

Description

Extracts covariance matrix of beta coefficients from BranchGLM objects.

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