Package ‘modelObj’

February 26, 2019

Type Package

Title A Model Object Framework for Regression Analysis

Version 4.0

Date 2019-02-26

Author Shannon T. Holloway

Maintainer Shannon T. Holloway <sthollow@ncsu.edu>

Description A utility library to facilitate the generalization of statistical methods built on a regression framework. Package developers can use 'modelObj' methods to initiate a regression analysis without concern for the details of the regression model and the method to be used to obtain parameter estimates. The specifics of the regression step are left to the user to define when calling the function. The user of a function developed within the 'modelObj' framework creates as input a 'modelObj' that contains the model and the R methods to be used to obtain parameter estimates and to obtain predictions. In this way, a user can easily go from linear to non-linear models within the same package.

Depends methods, stats, graphics

License GPL-2

LazyData TRUE

Encoding UTF-8

NeedsCompilation no

RoxygenNote 6.1.1

Repository CRAN

Date/Publication 2019-02-26 20:50:14 UTC

R topics documented:

`buildModelObj` *Create an Object of Class modelObj*

Description

A utility function to transfer user defined models and estimation methods to an object of class modelObj.

Usage

```
buildModelObj(model, solver.method = NULL, solver.args = NULL,
  predict.method = NULL, predict.args = NULL)
```

Arguments

`model`	An object of class formula; the model.
`solver.method`	An object of class character specifying the name of the R function to be used to obtain parameter estimates. Or, the function to be used to obtain parameter estimates. For example, 'lm', 'glm', or 'rpart'. The specified modeling function MUST have a corresponding predict method.
`solver.args`	An object of class list containing additional arguments to be sent to solver.method. Arguments must be provided as a list, where the name of each element matches a formal argument of solver.method. For example, if a logistic regression using glm is desired, $$solver.method = \text{“}glm\text{”}$$ $$solver.args = list(\text{“}family\text{”} = binomial)$$ A solver.method can takes formal arguments 'formula' and 'data' as inputs, such as lm and glm. Some R methods do not use formal names 'formula' and 'data'; a user can indicate if a different naming convention is used for these two input arguments. For example, if a method expects the formula object to be passed through input variable `x`, `solver.args` <- list("x"="formula") A solver.method can also take formal arguments 'x' and 'y' as inputs, such as glmnet. Some R methods do not use formal names 'x' and 'y' to indicate the covariate and response; a user can indicate if a different naming convention is used for these two input arguments. For example, if a method expects the covariate matrix to be passed through input variable `X`, `solver.args` <- list("X"="x")

`predict.method`	A character. The name of the R function or the function to be used to obtain predictions. For example, ‘predict.lm’, ‘predict’, or ‘predict.glm’. If no function is explicitly given, the generic `predict` is assumed. For many methods, the generic method is appropriate.
`predict.args`	A list. Additional arguments to be sent to predict.method. This must be provided as a list, where the name of each element matches a formal argument of predict.method. For example, if a logistic regression using glm was used to fit the model formula object and predictions on the scale of the response are desired, $$predict.method = “predict.glm”$$ $$predict.args = list(“type” = “response”).$$ It is assumed that the predict.method has formal arguments “object" and “newdata". If predict.method does not use these formal arguments, predict.args must explicitly indicate the variable names used for these inputs. For example, list(“newx"=“newdata") if the new data is passed to predict.method through input argument “newx".

Details

Unless changed by the user in solver.args and/or predict.args, default settings are assumed for the specified regression and prediction methods.

Value

An object of class `modelObjFormula` or `modelObjXY`, which inherit directly from modelObj.

Examples

```
#----------------------------------------------------#
# Create modeling object using a formula
#----------------------------------------------------#
mo <- buildModelObj(model=Y ~ X1 + X2 + X3 + X4,
                    solver.method='lm',
                    predict.method='predict.lm',
                    predict.args=list(type='response'))
```

`fit` *Obtain parameter estimates*

Description

Performs specified regression analysis.

Usage

```
fit(object, data, response, ...)

## S4 method for signature 'modelObj,data.frame'
fit(object, data, response, ...)
```

Arguments

`object`	An object of class `modelObj` as returned by the buildModelObj function.
`data`	An object of class data.frame containing the variables in the model.
`response`	An object of class vector containing the response variable.
`...`	ignored

Details

If defined by the modeling function, the following methods can be applied to the value object returned: `coef`, `plot`, `predict`, `print`, `residuals`, `show`, and `summary`.

Value

An object of class `modelObjFit`, which contains the object returned by the modeling function and the method to be used to obtain predictions.

Examples

```
# generate data
X <- matrix(rnorm(1000,0,1),
            ncol=4,
            dimnames=list(NULL,c("X1","X2","X3","X4")))

Y <- X \%*\% c(0.1, 0.2, 0.3, 0.4) + rnorm(250)

X <- data.frame(X)

# create modeling object using a formula
mo <- buildModelObj(model=Y ~ X1 + X2 + X3 + X4,
               solver.method='lm')

# fit model
fit.obj <- fit(object=mo, data=X, response=Y)

coef(fit.obj)
head(residuals(fit.obj))
plot(fit.obj)
head(predict(fit.obj,X))
summary(fit.obj)
```

`fitObject` *Retrieve Regression Object*

Description

Retrieves the value object returned by the regression method used to obtain parameter estimates.

Usage

```
fitObject(object, ...)

## S4 method for signature 'modelObjFit'
fitObject(object, ...)
```

Arguments

`object`	An object of class modelObjFit.
`...`	ignored.

Details

This function is useful for accessing methods that are defined by the regression method but are not directly accessible from the modelObjFit object. For example, for many regression methods, users can retrieve the fitted values by calling fitted.values(object). This method is not directly accessible from a modelObjFit. However, fitted.values() can be applied to the object returned by fitObject().

Value

The Value returned by the regression method specified in the governing modelObj. The exact structure of the value will depend on the regression method. For example, if nls() is the regression method, a list is returned.

Examples

```
# Generate data
X <- matrix(rnorm(1000,0,1),
            ncol=4,
            dimnames=list(NULL,c("X1","X2","X3","X4")))

Y <- X \%*\% c(0.1, 0.2, 0.3, 0.4) + rnorm(250)

X <- data.frame(X)

# Create modeling object using a formula
mo <- buildModelObj(model=Y ~ X1 + X2 + X3 + X4,
                    solver.method='lm')

# Fit model
fit.obj <- fit(object=mo, data=X, response=Y)

obj <- fitObject(fit.obj)
fobj <- fitted.values(obj)
head(fobj)
```

`model` *Retrieve model*

Description

Retrieves model from modelObj

Usage

```
model(object, ...)

## S4 method for signature 'modelObjFit'
model(object, ...)
```

Arguments

`object`	A modelObj object
`...`	ignored

Value

The formula for the regression

`modelObj` *Class* `modelObj`

Description

A class for model objects.

Details

Objects should not be created directly. The utility function buildModelObj() should be used.

Slots

`model` Object of class `formula`

`solver` Object of class `methodObjSolver` method to obtain parameter estimates.

`predictor` Object of class `methodObjPredict` method to obtain predicted values.

Methods

fit : Executes regression step.

model : Retrieve model.

solver : Retrieve regression method name.

solverArgs : Retrieve arguments to be sent to regression method.

solverArgs(object)<- : Set arguments to be sent to regression method.

predictor : Retrieve prediction method name.

predictorArgs : Retrieve arguments to be sent to prediction method.

predictorArgs(object)<- : Set arguments to be sent to prediction method.

Examples

```
showClass("modelObj")
```

`modelObjFit` *Class* `modelObjFit`

Description

A class for storing regression analysis results.

Slots

`fitObj` Object returned by the regression analysis

`model` Object of class `formula`

`func` Object of class `methodObjPredict` method to obtain predicted values.

Methods

fitObject : Extracts regression step.

model : Retrieve model.

solver : Retrieve regression method name.

solverArgs : Retrieve arguments to be sent to regression method.

solverArgs(object)<- : Set arguments to be sent to regression method.

predictor : Retrieve prediction method name.

predictorArgs : Retrieve arguments to be sent to prediction method.

predictorArgs(object)<- : Set arguments to be sent to prediction method.

Examples

```
showClass("modelObjFit")
```

predict *Model Predictions*

Description

Predictions from the results of a fit object.

Usage

```
predict(object, ...)

## S4 method for signature 'modelObjFit'
predict(object, newdata, ...)
```

Arguments

`object`	An object of class `modelObjFit` as returned by the fit() function.
`newdata`	An object of class data.frame containing the variables in the model.
`...`	ignored

Value

Model predictions, the form of which depend on the regression analysis.

Examples

```
# generate data
X <- matrix(rnorm(1000,0,1),
            ncol=4,
            dimnames=list(NULL,c("X1","X2","X3","X4")))

Y <- X \%*\% c(0.1, 0.2, 0.3, 0.4) + rnorm(250)

X <- data.frame(X)

# create modeling object using a formula
mo <- buildModelObj(model=Y ~ X1 + X2 + X3 + X4,
               solver.method='lm')

# fit model
fit.obj <- fit(object=mo, data=X, response=Y)

predict(fit.obj)
predict(fit.obj, newdata = X[1:10,])
```

`predictor` *Retrieve Prediction Method*

Description

Retrieves method for prediction analysis

Usage

```
predictor(object, ...)

## S4 method for signature 'modelObjFit'
predictor(object, ...)
```

Arguments

`object`	A modelObj object
`...`	ignored

Value

An object of class character or function

`predictorArgs` *Retrieve Predictor Arguments*

Description

Retrieves the arguments that are to be passed to the prediction method when called.

Usage

```
predictorArgs(object, ...)

predictorArgs(object) <- value

## S4 method for signature 'modelObjFit'
predictorArgs(object, ...)
```

Arguments

`object`	A modelObj object
`...`	ignored
`value`	List to be stored in args

Value

A list

`solver` *Retrieve Solver Method*

Description

Retrieves method for regression analysis

Usage

```
solver(object, ...)

## S4 method for signature 'modelObjFit'
solver(object, ...)
```

Arguments

`object`	A modelObj object
`...`	ignored

Value

An object of class character or function

`solverArgs` *Retrieve Solver Arguments*

Description

Retrieves the arguments that are to be passed to the regression method when called.

Usage

```
solverArgs(object, ...)

solverArgs(object) <- value

## S4 method for signature 'modelObjFit'
solverArgs(object, ...)
```

Arguments

`object`	A modelObj object
`...`	ignored
`value`	List to be stored in args

Value

A list

Index