Package 'modelObj'

May 13, 2017

Type Package

Title A Model Object Framework for Regression Analysis

Version 3.0

Date 2017-05-12

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

License GPL-2

LazyData TRUE

NeedsCompilation no

Repository CRAN

Date/Publication 2017-05-12 23:29:18 UTC

R topics documented:

`modelObj-package` *A Model Object Framework for Regression Analysis*

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.

Details

Package:	modelObj
Type:	Package
Version:	3.0
Date:	2017-05-10
License:	GPL-2
Depends:	methods

Often, new statistical methods are developed on the framework of traditional regression or classification methods. To simplify the creation of new R implementations of these methods, researchers and software developers often make choices regarding the types of models that can be used by the user; hard-coding the regression method into the library and limiting or eliminating the ability of the user to modify regression control parameters. These choices artificially limit the general application of new methods. In addition, if a new method requires multiple models, a developer is often forced to artificially break the method into multiple function calls, each for a specific regression/classification step, or is forced to provide a cumbersome and/or confusing interface for the user. **modelObj** is an R package developed to facilitate the use of existing and future R regression and classification libraries that simplifies the development of general, non-model-specific implementations of new statistical methods.

Author(s)

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

`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,

$$predict.method = \text{“}predict.glm\text{”}$$

$$predict.args = list(\text{“}type\text{”} = \text{“}response\text{”}).$$

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 type modelObj, which contains a complete description of the model, the method to be used for parameter estimates, and the prediction method.

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'))
```

---

coef *Extract Model Coefficients*

Description

Extracts model coefficients from an object of class modelObjFit created by a call to modelObj::fit().

Usage

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

Arguments

object	an object of class modelObjFit.
...	passed through to coef() of the modelObj regression method.

Value

Coefficients extracted from the modelObjFit object 'object'. For standard model fitting classes, the value returned is a named vector. If no coef() method is defined for the regression method specified in the governing modelObj, NULL is returned.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

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)
```

---

fit *Execute Regression Step*

Description

Executes the model fitting procedure for an object of class modelObj.

Usage

```
## S4 method for signature 'modelObj,data.frame,vector'
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 a modelObjFit object to obtain standard fit results: `coef`, `plot`, `predict`, `print` `residuals`, `show`, and `summary`.

Value

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

Slots of the modelObjFit object can be retrieved using `fitObject(object)`, `predictor(object)`, and `predictorArgs(object)`.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

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

```
 ## 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.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

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 Object*

Description

Retrieve the model slot from an object of class modelObj.

Usage

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

Arguments

`object`	an object of class modelObj.
`...`	ignored.

Value

Returns an object of class "formula."

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

Examples

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


model(mo)
```

---

modelObj-class *Class* "modelObj"

Description

A class for model objects.

Objects from the Class

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.

S4Methods

fit `signature(object = "modelObj", data="data.frame", response="vector", ...)`: Executes regression step.

model `signature(object = "modelObj")`: Retrieve model.

solver `signature(object = "modelObj")`: Retrieve regression method name.

solverArgs `signature(object = "modelObj")`: Retrieve arguments to be sent to regression method.

solverArgs(object)<- `signature(object = "modelObj")`: Set arguments to be sent to regression method.

predictor `signature(object = "modelObj")`: Retrieve prediction method name.

predictorArgs `signature(object = "modelObj")`: Retrieve arguments to be sent to prediction method.

predictorArgs(object)<- `signature(object = "modelObj")`: Set arguments to be sent to prediction method.

Examples

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

`modelObjFit-class` *Class* `"modelObjFit"`

Description

A class containing the fit results for a model object defined as an object of class modelObj.

Objects from the Class

Objects should not be created directly. Objects are created by a call to fit().

Slots

`fitObj`: Value object of modeling function (i.e., class(fitObj)="lm" if solver.method=lm).

`func`: Object of class `methodObj` the method to be used to obtain predicted values.

`model`: Object of class `formula` model.

Methods

coef `signature(object = "modelObjFit")`: if, defined for the modeling function, extracts model coefficients. Else, returns NULL.

fitObject `signature(object = "modelObjFit")`: Retrieve fit object.

plot `signature(x = "modelObjFit", y=NULL)`: if defined for the modeling function, invokes plot(modelObj@fitObj). Else, returns a message that a plot(modelObj@fitObj) method is not defined.

predict `signature(object = "modelObjFit")`: returns predicted values obtained using fitObj and the newdata data.frame.

predictor `signature(object = "modelObjFit")`: Retrieve prediction method name.

predictorArgs `signature(object = "modelObjFit")`: Retrieve arguments to be sent to prediction method.

residuals `signature(object = "modelObjFit")`: if defined for the modeling function, returns the residuals(modelObjFit@fitObj).

show `signature(x = "modelObjFit")`: displays standard results returned by modeling function.

summary `signature(x = "modelObjFit")`: if, defined for the modeling function, returns summary(modelObjFit@fitObj). Else, returns a message that a summary(modelObjFit@fitObj) method is not defined.

Examples

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

plot *Plotting*

Description

Generates the plot from an object of class modelObjFit.

Usage

```
 ## S4 method for signature 'modelObjFit'
plot(x, y, ...)
```

Arguments

x	an object of class modelObjFit.
y	ignored
...	ignored

Value

If plot is defined for the regression method that was used to obtain the regression parameters, generates the plot as defined by the regression method. If method does not exist, a warning messages is printed.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

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)

plot(fit.obj)
```

---

predict *Model Prediction.*

Description

Executes the prediction method for an object of class modelObjFit.

Usage

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

Arguments

object	an object of class modelObjFit containing the regression object which which predictions are to be obtained.
newdata	An optional data frame of variables with which to make predictions. If not provided, fitted values from the regression analysis are returned.
...	ignored

Value

A matrix of the predicted response for the fitted model.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

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)

pred <- predict(object=fit.obj, newdata=X)

head(pred)
```

`predictor` *Retrieve Prediction Method*

Description

Retrieve the prediction method from an object of class modelObj. This capability is intended for package developers making use of the “model object" framework.

Usage

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

Arguments

`object`	an object of class modelObj.
`...`	ignored.

Value

Returns an object of class "character" or "function"; the prediction method specified in the modelObj object.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

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'))

predictor(mo)
```

---

predictorArgs *Retrieve Arguments Specified for the Prediction Method*

Description

Retrieve from objects of class modelObj created by function buildModelObj the arguments that will be sent to the prediction method. This capability is intended for package developers making use of the "model object" framework.

Usage

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

Arguments

object	an object of class modelObj.
...	ignored.

Value

Returns an object of class "list" containing the arguments specified to be passed to the prediction method.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

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'))

predictorArgs(mo)
```

predictorArgs<- *Reset Arguments for Prediction Method*

Description

Reset the arguments to be sent to the prediction method. This capability is intended for package developers making use of the “model object" framework.

Usage

```
## S4 replacement method for signature 'modelObj'
predictorArgs(object) <- value
```

Arguments

object	object of class modelObj, for which the arguments sent to the prediction method are to be reset.
value	A named list containing the new arguments to be sent to the prediction method.

Details

The first two elements of the list contain the name for the object returned by the regression method and the name for the data.frame for which predictions are desired. buildModelObj creates modelObj using an internal convention, which is critical to the correct implementation of this package. These two elements should not be changed and will be carried over from the original argument list.

Value

modelObj with an updated argument list to be passed to the prediction method.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

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'))

predictorArgs(mo)
argList <- list("type"='terms')
predictorArgs(mo) <- argList
predictorArgs(mo)
```

`residuals` *Extract Model Residuals.*

Description

Returns the residuals from an object of class modelObjFit.

Usage

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

Arguments

`object`	an object of class modelObjFit containing the regression object from which residuals are to be retrieved.
`...`	ignored

Value

If `residuals()` is defined for the regression method used to obtain parameter estimate, returns the residuals as defined by the regression method. If method does not exist, a warning messages is printed.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

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)

res <- residuals(fit.obj)
head(res)
```

`solver` *Retrieve Regression Method.*

Description

Retrieve the regression method from an object of class modelObj.

Usage

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

Arguments

`object`	an object of class modelObj.
`...`	ignored.

Value

Returns an object of class "character" giving the name of the regression method to be used to obtain parameter estimates.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

Examples

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

solver(mo)
```

solverArgs *Retrieve Arguments for Regression Method.*

Description

Retrieve the arguments to be sent to the regression method.

Usage

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

Arguments

object	object of class modelObj, from which the arguments sent to the regression method are to be retrieved.
...	ignored.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

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'))

solverArgs(mo)
solverArgs(mo) <- list(model=TRUE)
solverArgs(mo)
```

solverArgs<- *Reset Arguments for Regression Method*

Description

Reset the arguments to be sent to the regression method. This capability is intended for package developers making use of the “model object" framework.

Usage

```
 ## S4 replacement method for signature 'modelObj'
solverArgs(object) <- value
```

Arguments

object	object of class modelObj, for which the arguments sent to the regression method are to be reset.
value	A named list containing the new arguments to be sent to the regression method.

Details

The first two elements of the list contain the name for the formula object or covariate matrix and the name for the data.frame or response. buildModelObj creates modelObj using an internal convention, which is critical to the correct implementation of this package. These two elements should not be changed and will be carried over from the original argument list.

Value

modelObj is updated with new argument list.

Author(s)

Shannon T. Holloway <sthollow@ncsu.edu>

Examples

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

solverArgs(mo)
argList <- list("x"=TRUE)
solverArgs(mo) <- argList
solverArgs(mo)
```

Index