

Predictive Analytics: practical 1

Course R package

Installing the course R package¹ is straightforward. First install drat

```
install.packages("drat")
```

Then

```
drat::addRepo("rcourses")  
install.packages("nclRpredictive", type="source")
```

This R package contains copies of the practicals, solutions and data sets that we require. It will also automatically install any packages that we use during the course. To load the package, use

```
library("nclRpredictive")
```

During this practical we will use the caret package

```
library("caret")
```

The cars2010 data set

The cars2010 data set contains information about car models in 2010. The aim is to model the FE variable which is a fuel economy measure based on 13 predictors.²

The data is part of the AppliedPredictiveModeling package and can be loaded by,

```
data(FuelEconomy, package = "AppliedPredictiveModeling")
```

Exploring the data

There are a lot of questions to be considered below marked out by bullet points in the document. Don't worry if you can't finish them all, I intended for there to be enough such that if anyone wants to continue trying things at home they can.

- Prior to any analysis we should get an idea of the relationships between variables in the data. Use the pairs function to explore the data. The first few are shown in figure 1.

An alternative to using pairs is to specify a plot device that has enough space for the number of plots required to plot the response against each predictor. We don't get all the pairwise information amongst predictors but it saves a lot of space on the plot and makes it easier to see what's going on. It's also a good idea to make smaller margins.

¹ A package is an *add-on* or a *module*. It provides additional functions and data.

² Further information can be found in the help page, `help(cars2010)`.

The $FE \sim .$ notation is shorthand for FE against all variables in the data frame specified by the data argument.

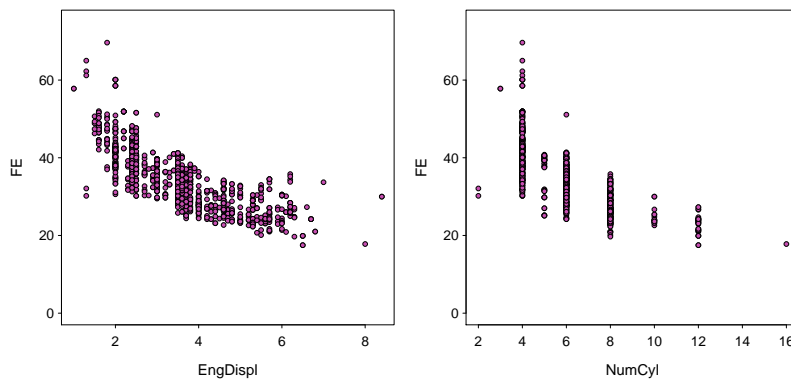


Figure 1: Plotting the response against some of the predictor variables in the cars2010 data set.

```
op = par(mfrow = c(3, 5), mar = c(4, 2, 1, 1.5))
plot(FE ~ ., data = cars2010)
par(op)
```

- Create a simple linear model fit of FE against EngDispl using the train function.³
- Examine the residuals of this fitted model, plotting residuals against fitted values
We can add the lines showing where we expect the residuals to fall to aid graphical inspection

³ Remember, to specify a particular model type we use the method argument.

```
abline(h = c(-2, 0, 2), col = c(2, 3, 2), lty = c(2, 1, 2))
```

- What do the residuals tell us about the model fit using this plot?
- Plot the fitted values vs the observed values
 - What does this plot tell us about the predictive performance of this model across the range of the response?
 - Produce other diagnostic plots of this fitted model
 - Are the modelling assumptions justified?

Extending the model

- Do you think adding a quadratic term will improve the model fit?
- Fit a model with the linear and quadratic terms for EngDispl and call it m2
 - Assess the modelling assumptions for this new model
 - How do the two models compare?
- How does transforming the response variable affect the fit?
- Add NumCyl as a predictor to the simple linear regression model m1 and call it m5
- Examine model fit and compare to the original.
- Does the model improve with the addition of an extra variable?

Common transformations may be a log or square root function.

Visualising the model

The `nclRpredictive` package contains a `plot3d` function to help with viewing these surfaces in 3D as in figure 2.⁴

```
plot3d(m5, cars2010$EngDispl, cars2010$NumCyl, cars2010$FE)
```

- Try fitting other variations of this model using these two predictors, how is prediction affected in each case? Don't forget to examine residuals, R squared values and the predictive surface.
- If you want to add an interaction term you can do so with the `:` operator, how does the interaction affect the surface?

One way to gauge how well your model is performing is to hold out a set of observations from the training data. Then examine how well your model extends to the data that wasn't used for training. We will see more of this in coming chapters of the notes.

```
# set up a set of indicies that will be included
# in the training data
trainIndex = sample(nrow(cars2010), 900)
# create two data frames, a training and a test set
# by taking subsets using this set of indicies
# here we use 900 observations to train the model
# and the rest for testing
carstrain = cars2010[trainIndex,]
carstest = cars2010[-trainIndex,]
# train the model and predict
mtrain = train(FE~EngDispl + NumCyl, data = carstrain,
               method = "lm")
prediction = predict(mtrain, carstest)

# residuals of the test set
res = prediction - carstest$FE
# calculate RMSE
sqrt(mean(res*res))

## [1] 4.626
```

Having a small value here indicates that my model does a good job of predicting for observations that weren't used to train the model.

In the spirit of competition ...

Try to fit the best model that you can using the `cars2010` data set and the above tools. I have a set of data that you haven't yet seen. Once you are happy with your model you can validate it using the `validate` function in the `nclRpredictive` package.

⁴ We can also add the observed points to the plot using the `points` argument to this function, see the help page for further information.

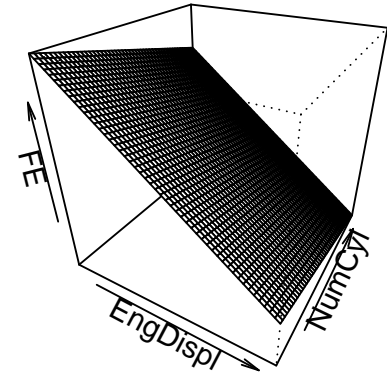


Figure 2: A surface plot from a linear model of fuel economy against the number of cylinders and engine displacement including the interaction term.

```
mlvalidated = validate(model = m1)
```

Other data sets

A couple of other data sets that can be used to try fitting linear regression models.

data set	Package	Response
diamonds	ggplot2	price
Wage	ISLR	wage
BostonHousing	mlbench	medv