Practical 2

Jumping Rivers

During this practical we will mainly use the **caret** package, we should load that 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.[1]

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

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

[1] Further information can be found in the help page, `help("cars2010", package = "AppliedPredictiveModeling")`.

Exploring the data

1. Prior to any analysis we should get an idea of the relationships between variables in the data. [2] Use the `pairs()` function to explore the data. The first few are shown in figure @**??**fig:fig1_1).

2. 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

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

[2] The `FE ~ .` notation is shorthand for `FE` against all variables in the data frame specified by the `data` argument.

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 is also a good idea to make smaller margins.

1. Create a simple linear model fit of `FE` against `EngDispl` using the `train()` function[3]. Call your model `m1`.

2. Examine the residuals of this fitted model, plotting residuals against fitted values

3. We can add the lines showing where we expect the standardised residuals to fall to aid graphical inspection

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

4. What do the residuals tell us about the model fit using this plot?

[3] Hint: use the `train()` function with the `lm` method.

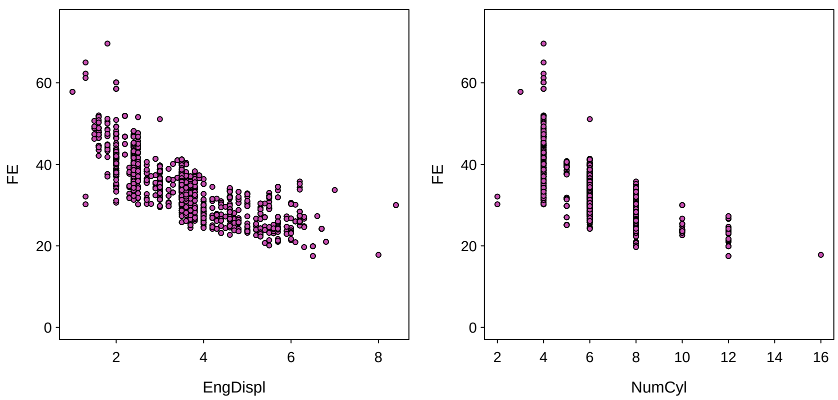

Figure 1: Plotting the response against some of the predictor variables in the ‘cars2010‘ data set.

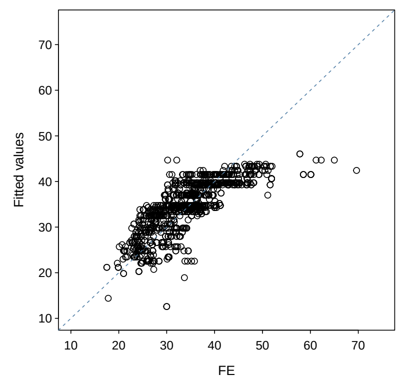

Figure 2: Plot of fitted against observed values. It’s always important to pay attention to the scales.

1. Plot the fitted values vs the observed values
2. What does this plot tell us about the predictive performance of this model across the range of the response?
3. Produce other diagnostic plots of this fitted model, e.g. a q-q plot
4. Are the modelling assumptions justified?

Extending the model

1. Do you think adding a quadratic term will improve the model fit?
2. Fit a model with the linear and quadratic terms for `EngDispl` and call it `m2`
3. Assess the modelling assumptions for this new model. How do the two models compare?
4. Add `NumCyl` as a predictor to the simple linear regression model `m1` and call it `m3`
5. Examine model fit and compare to the original.
6. Does the model improve with the addition of an extra variable?

Visualising the model

The **jrAnalytics** package contains a `plot3d()` function to help with viewing these surfaces in 3D.[4]

```
## points = TRUE to also show the points
plot3d(m3, cars2010$EngDispl, cars2010$NumCyl, cars2010$FE,
    points = FALSE)
```

We can also examine just the data interactively, via

[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.

```r
threejs::scatterplot3js(cars2010$EngDispl, cars2010$NumCyl,
    cars2010$FE, size = 0.5)
```

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

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