PRACTICAL 6

Aim:

The aim of this practical is to illustrate how both simple linear regression and multiple linear regression can be implemented in R.

Before you start

Load the `tidyverse`.

Also install (if working on your own computer) and load the package `gridExtra`. This package allows you to arrange multiple plots next to each other.

Instructions:

This document contains discussions of some basic concepts, and instructions which you should execute in RStudio. **Text in bold** indicate actions that are to be taken within RStudio. `Text highlighted in grey` shows the code you should enter in RStudio.

Due date:

For Tutorial 6, it will be assumed that you have mastered the basic concepts discussed here.

Simple linear regression

We will first work with the `cars` dataset, which is already available in R so you don't have to import any data. The data was recorded in the 1920s and contains the speed of cars and the distances taken to stop. There are 50 observations, and 2 variables: `speed` (measuring the speed of the car in miles per hour) and `dist` (stopping distance of the car, measured in feet).

Since we will be using this dataset to illustrate simple linear regression, we should think about what the target variable (response) will be. In this case, we will aim to predict the stopping distance of a car, based on the car's speed.

1. **Examine the dataset by looking at the first few rows of the data. Also look at the variable types, and view summary statistics for the data.**

```
head(cars)
str(cars)
summary(cars)
```

2. **Create a scatterplot of the data. Put `dist` on the *y* axis and `speed` on the *x* axis.**

```
ggplot(data = cars, mapping = aes(x = speed, y = dist))+
geom_point()
```

3. **Visualise the distribution of each variable by creating boxplots. Arrange these boxplots next to each other, using the `grid.arrange()` function from the `gridExtra` package.**

```
box1 <- ggplot(data = cars) + geom_boxplot(mapping =
aes(x = speed)) + coord_flip()
box2 <- ggplot(data = cars) + geom_boxplot(mapping =
aes(x = dist)) + coord_flip()
grid.arrange(box1, box2, ncol = 2)
```

4. **Calculate the correlation between `speed` and `dist`.**

```
cor(cars$speed, cars$dist)
```

5. **Fit a simple linear regression model with `dist` as the target variable (response) and `speed` as explanatory variable. Save the model output in an object called `model1`, and view summary output for this model.**

```
model1 <- lm(dist~speed, data = cars)
summary(model1)
```

6. The summary output from Step 5 above shows that stopping distance of a car can be predicted using the following equation:

predicted stopping distance (in feet) = -17.5791 + (3.9324 x speed of car)

This means that the average predicted stopping distance for a car travelling at a speed of 20 miles per hour is

-17.5791 + (3.9324 x 20) = 61.0689 feet.

This predicted value can also be obtained in R.

6.1 **First create a data frame containing a variable called `speed` and one observation with `speed` equal to 20. Store this data frame in an object called `newdata`. Print this data frame to your screen.**

```
newdata <- data.frame(speed = 20)
newdata
```

6.2 **Now use the model created in Step 5 (i.e. the `model1` object) and the `predict()` function to make a prediction for the `newdata` data frame. Store your prediction in an object called `predict1` and view the prediction.**

```
predict1 <- predict(model1, newdata)
predict1
```

7. The least squares regression line can be visualised on a scatterplot by using the `geom_smooth()` function, and setting the argument `model = "lm"`. `"lm"` stands for *linear model*.

Add the regression line to the scatterplot you created in Step 2. You can use the same code from Step 2 and just add a `geom_smooth()` plot layer.

```
ggplot(data = cars, mapping = aes(x = speed, y = dist)) +
geom_point() + geom_smooth(method = "lm")
```

Quadratic regression

8. In the theoretical lectures, we only considered fitting straight lines. However, curvature can be introduced by for instance including a quadratic term and therefore fitting a parabola.

Use a quadratic polynomial to predict a car's stopping distance by also including the square of `speed` in the model. Store the model output in an object named `model2`. Also view the summary output for this model.

Note that even though the regression equation is no longer a straight line, it is still referred to as a linear model since it is linear in the parameters. You don't have to manually create a new variable containing the square of `speed`, you can specify this directly in the `lm()` formula. Using `I()` is required in the formula so that R correctly interprets that it should perform the specified arithmetic operation.

```
model2 <- lm(dist~speed + I(speed^2), data = cars)
summary(model2)
```

The summary output shows that stopping distance of a car can also be predicted using the following equation:

predicted stopping distance (in feet) = 2.47014 + (0.91329 x speed of car) + (0.09996 x (speed of car)^2)

This means that the average predicted stopping distance for a car travelling at a speed of 20 miles per hour is

$$2.4701 + (0.9133 \times 20) + (0.09996 \times 20^2) = 60.72 \text{ feet.}$$

9. **Verify this prediction in R.**

```
predict2 <- predict(model2, newdata)
predict2
```

10. **Visualise the quadratic regression line by including it on the scatterplot created in Step 2.**

    You can use the `geom_smooth()` function again, but this time you will have to specify the formula to use as well, so that a quadratic polynomial is plotted and not a straight line.

```
ggplot(data = cars, mapping = aes(x = speed, y = dist)) +
geom_point() + geom_smooth(method = "lm", formula = y ~ x
+ I(x^2))
```

11. **Create a plot containing both the linear regression line and the quadratic regression line. (Plot the straight line in red.) Compare the two lines and note how they differ.**

```
ggplot(data = cars, mapping = aes(x = speed, y = dist)) +
geom_point() + geom_smooth(method = "lm", colour = "red")
+ geom_smooth(method = "lm", formula = y ~ x + I(x^2))
```

    At this stage we are not yet discussing how to evaluate models, and therefore we will not have a discussion about which of the two provide a better fit to the data. We will only consider the evaluation of models at the end of the semester. However, you can study the plot so long and try to form an opinion regarding whether the straight line or the quadratic polynomial provides a better fit.

Multiple linear regression

To study multiple linear regression in R, we will use the `mtcars` dataset which we have used before. The aim is to use the 10 attributes in the dataset to predict fuel consumption (in miles per gallon). This means that `mpg` is the target variable (response) and the other variables in the dataset will be the explanatory variables. Recall that there are only 32 observations in this dataset.

12. **Examine the dataset by looking at the variable types and also summary statistics for the variables.**

```
str(mtcars)
summary(mtcars)
```

13. **Calculate the correlation between the `mpg` variable and the other variables in the dataset.**

```
cor(mtcars$mpg, mtcars[,-1])
```

14. **Fit a linear model to the dataset using `mpg` as target variable and all the other attributes as explanatory variables. Store the model output in an object named `model3`, and view summary output for this model.**

```
model3 <- lm(mpg~., data = mtcars)
summary(model3)
```

The predicted regression equation is

`mpg` = 12.30337 + (-0.11144 x `cyl`) + (0.01334 x `disp`) + … + (-0.19942 x `carb`)

Making predictions can be done in a similar way as for simple linear regression.

Also take careful note of the interpretation of the regression coefficients in multiple linear regression. For instance, for a 1 unit increase in `cyl` (i.e. for each additional

cylinder a car has), the average fuel consumption (in miles per gallon) will decrease by 0.11144, keeping all other variables constant.

You should now have a good grasp of the following:

*** Fitting simple linear regression models in R**

*** Utilising regression models to make predictions**

*** Visualing regression curves for simple linear regression**

*** Fitting multiple regression models in R**

*** Interpreting output from regression models**