Data Science Lab 2: Common R datasets

michaelferrie@edinburghcollege.ac.uk

1.0 Introduction

“We want to use realistics datasets, stop teaching people about cars from the 70's, stop speaking about automatic vs manual transmission, and give them a dataset they can relate to (University of Edinburgh, 2021).”

There are some criticisms of the datasets introduced in this lab, however we still think it is important to know some of the most common datasets used when learning R such as mtcars. As climate change becomes an ever more real threat to the Earth in the minds of people across the world, many governments are taking action to mitigate greenhouse gas emissions. The car industry in particular is targeted for regulations, and in many instances fuel economy standards are set in place for car manufacturers to meet. If manufacturers continue to produce cars with poor fuel economy in spite of standards, they are often made to pay a tax. In order to meet these standards and avoid excess taxation, engineers and businesses must come up with creative solutions to reduce fuel consumption without compromising on performance.

Since this issue is relevant to today’s engineering challenges and protecting the environment, our team has decided to analyse data in order to explore which avenues may be considered by engineering teams when attempting to meet fuel economy standards. To do this, we have used the MTcars data set, which has data on the design, performance and fuel economy for 32 automobiles from 1973 - 1974. All of the data was extracted from the 1974 Motor Trend US magazine.

1.1 Access the dataset

R-Studio comes with a lot of built in datasets - in this lab we will look at the mtcars dataset, click into the console and enter, after you type this press the CTRL + return key. This loads the data set into R

```
data("mtcars")
```

1.2 Access help for this dataset, each built in dataset comes with a help file, access this with, notice the help file opens and describes the data - we can see it is from a 1974 magazine:

```
help(mtcars)
```

From the help output we can see that the dataset has a data frame with 32 observations on 11 (numeric) variables.

Column Number	Abbreviation	Description
[, 1]	mpg	Miles/(US) gallon
[, 2]	cyl	Number of cylinders

```
[, 3]            disp            Displacement (cu.in.)
[, 4]            hp              Gross horsepower
[, 5]            drat            Rear axle ratio
[, 6]            wt              Weight (1000 lbs)
[, 7]            qsec            1/4 mile time
[, 8]            vs              Engine (0 = V-shaped, 1 = straight)
[, 9]            am              Transmission (0 =automatic, 1 =manual)
[,10]            gear            Number of forward gears
[,11]            carb            Number of carburetors
```

A dataframe is an organised two dimensional data structure, similar to a spreadsheet, in the output above the numbers at the start of each variable refer to the columns in the dataframe, each is numbered and has a header. The header is usually the first row.

1.3 Let's look at the top (the head) of the dataframe, think of this as the first few rows - enter:

```
head(mtcars, 6)
```

1.4 Can you see the first line, how many mpg was the Mazda RX4?

1.5 Think of the dataset as a large spreadsheet, to get the number of rows the command is nrow and to get the number of columns it is ncol - try those now, remember to tell R the dataset name. These are functions, and you can send a value (in this case a whole datset) into a function using ().

1.6 Let's see the real magic of R now, pass the number of gears for each car into a table, using the table function, the $ special character lets us select a particular column from the dataset.

```
mtcarsgear <- table(mtcars$gear)
```

Display the table as a bar chart:

```
barplot(mtcarsgear)
```

1.7 You should see a bar chart now for the number of cars and how many gears each car had (yes cars had 3 gears in the 70's!!!)

1.8 In the console in R-Studio press the up arrow on your keyboard to get the last command back, let's add some detail to the bar chart. Modify your barplot command to the following then run it to see the output - Main is the title, axis labels are set with xlabel and ylabel:

```
barplot(mtcarsgear, main = "Car Gears", ylab = 'Cars', xlab =
"Gears Type")
```

1.9 Add this to your command to make the bars change colour

```
barplot(mtcarsgear, main = "Car Gears", ylab = 'Cars' ,xlab =
"Gears Type", col=c('green','red','yellow'))
```

1.10 Now try it yourself, create the following. A table with the weight of each car, then plot the weight table as a barplot. Once you have the barplot add a main title, and an x and y label.

Review

In this first part we learned the following:

1. How to load, show the head and show the help file for a built in dataset
2. How to make a table and turn this into a bar chart
3. How to add colours and labels to the bar chart
4. The <- is used to assign a variable in R - this was done in step 1.6

Part 2: Histograms with Iris

2.0 A histogram is a plot that lets you discover, and show, the underlying frequency distribution (shape) of a set of continuous data. This allows the inspection of the data for its underlying distribution. A built-in dataset that is often used when learning R is the iris dataset. The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher 1936.

2.1 Start by loading the dataset into R and then load the top of the data. This will show the columns and the first three rows. This is useful to get an overview of the data available.

```
head(iris,3)
```

```
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
```

2.2 If the head shows the top rows of a dataset, R also has a summary command that should provide a broader overview of the whole dataset, from the summary we can see that R understands the variables in the dataset.

```
summary(iris)
```

```
 Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa     :50
```

```
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50

 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
```

…

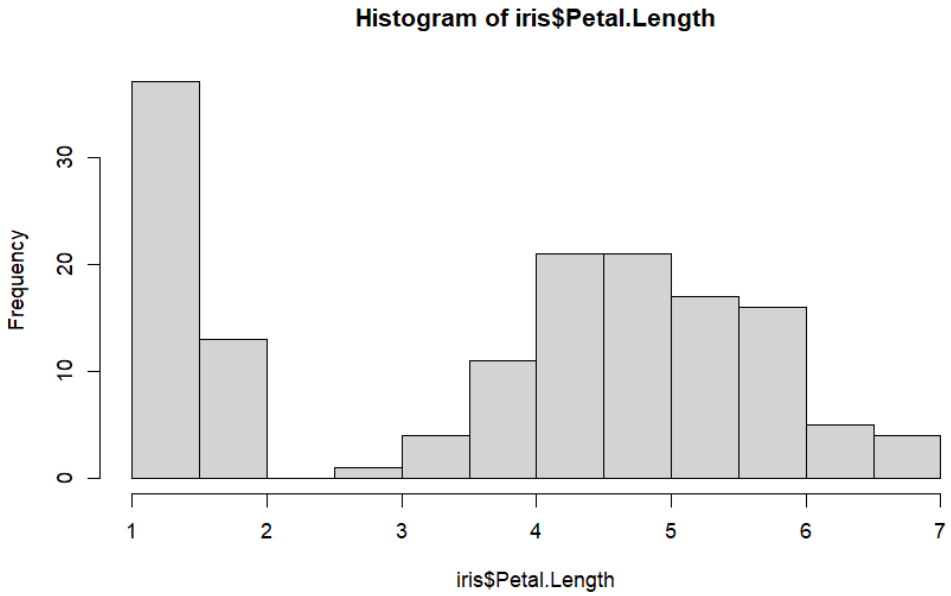
2.3 We can have a random sample of the data and ask for 10 rows to give a neater gird of information:

```
iris[sample(nrow(iris),10),]
```

2.4 Petal length is one of the variables in the dataset, let's create a histogram of the petal length, in R we can specify the type of chart, then the dataset and the variable or column in the dataset like this:

```
hist(iris$Petal.Length)
```

You should see something that looks like this, it is quite a basic chart.



The histogram created in step 2.4 is functional but it is not the best design, let's add a library and make it look nicer. The great thing about R is the number of packages available. In R studio choose file and new and select a new R script and install the following packages by running the following with CTRL+RETURN:

```
install.packages("ggplot2")
```

2.6 R will have some output in the console while it downloads the packages we requested from the CRAN (comprehensive R archive network). Now create a new R script and at the top load the libraries we just downloaded. You should only have to download the libraries to your local machine once, then you can just import them when needed.

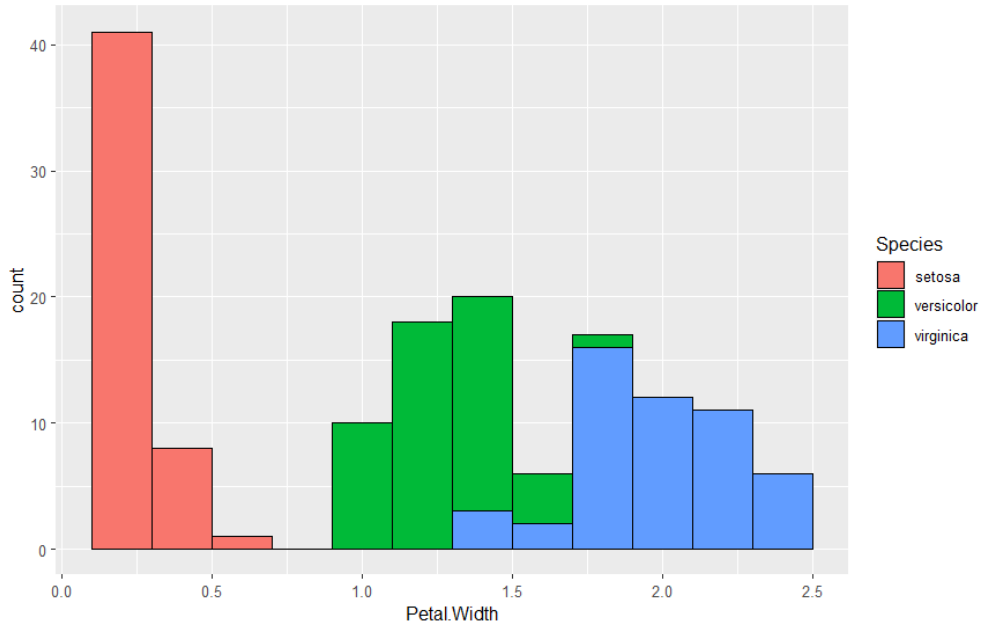
```
library(ggplot2)
```

2.7 Now lets create a new variable called histogram:

```
histogram <- ggplot(data=iris, aes(x=Petal.Width))
```

2.8 The bars in a histogram are called bins - set the bin width to 0.2 and plot with ggplot2

```
histogram + geom_histogram(binwidth=0.2, aes(fill=Species))
```



2.9 This should plot a nice looking histogram, now to add some details, adding black should put a border round the bins:

```
histogram + geom_histogram(binwidth=0.2, color="black",
aes(fill=Species))
```

2.10 Now add some labels to the plot:

```
histogram + geom_histogram(binwidth=0.2, color="black",
aes(fill=Species)) + xlab("Petal Width") + ylab("Frequency")
+ ggtitle("Histogram of Petal Width")
```

2.11 Re-run the command and this time, set the bin width to 0.5 and the colour to red.

2.12 Challenge

See if you can create a histogram using the Sepal Width column of the iris dataset, using ggplot2 with a binwidth of 0.5. Modify your code from the previous examples to achieve this.