PRACTICAL 3

Aim:

The aim of this practical is to introduce you to data visualisation in R, specifically using `ggplot2`, which is part of the `tidyverse` group of packages.

Mode of study:

Self-study, using this set of notes / instructions, together with the video provided on SUNLearn.

Before you start:

Load the `tidyverse`.

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 3, it will be assumed that you have mastered the basic concepts discussed here.

About `ggplot2`

`ggplot2` is one of the core packages in the `tidyverse`. `ggplot2` implements the *grammar of graphics* and is a tool that makes it easy to concisely describe the components of a graphic. `ggplot2` uses a *layered* grammar, meaning that it builds plots layer by layer.

You can read more about the theoretical underpinnings of ggplot2 at https://vita.had.co.nz/papers/layered-grammar.pdf.

There are different components to a plot created with `ggplot2`. These are:

- Dataset – You need to tell `ggplot2` which dataset to use when constructing the plot.
- *Mappings* (from variables to aesthetics) – *Aesthetics* are visual properties of the objects in your plots, and include things like the size, shape, or colour. You add a variable to a plot by mapping it to an aesthetic.
- Layer(s) (each with *geometric object*, statistical transformation, and position adjustment) – In `ggplot2`, we use a geom (geometrical object) to specify the type of plot that should be created. We will explore several different geoms in this practical. You can also use multiple geoms in the same plot, for instance to overlay a line on a scatterplot. Geoms can use statistical transformations to calculate required values for a plot, but we will not need to specify these explicitly for the plots we will be creating.
- *Scale* for each aesthetic mapping (optional).
- *Coordinate* system – This allows you to use a coordinate system other than the default Cartesian coordinate system (but we will mostly ignore this).
- *Facet* specification – While variables can be added through aesthetics, another useful way of adding additional variables to a plot is to split the plot into facets, in other words subplots that each display one subset of the data.

We will mostly focus on the first three aspects in the plots we create.

A basic scatterplot

1. In Practical 2 we considered the `mpg` dataset. We will use this dataset in this practical as well.

1.1 **Refresh your memory about the variables contained in the `mpg` dataset by printing the dataset to the console.** (Refer to Practical 2 if you cannot recall how to do this.)

1.2 **Create a scatterplot with the `displ` variable on the *x*-axis and the `hwy` variable on the *y*-axis.**

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy))
```

Take note of the following:

- While the package we use is called `ggplot2`, the function we call is `ggplot()`.
- Within the `ggplot` function call, we include the name of the dataset. I have typed the argument out in full for the sake of clarity, but there is no need to include the `data =`; `ggplot` will automatically assume that the first argument you provide is the name of the dataset.
- After the `ggplot()` call, we add the geom layer; in this case we want a scatterplot so we use `geom_point`.
- We place the aesthetic mapping inside the `geom_point()` function call. Here you specify which variable you want on the *x*-axis and which variable on the *y*-axis. [Note that you could also include the aesthetic mapping in the `ggplot()` call instead of inside the geom; the difference will be explained a little later in this practical.]

1.3 **Change the colour of the points on the scatterplot to red.**

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy),
color = "red")
```

Note that R has a built-in list of colour names which you can use. You can view this list at the following link (scroll down to see the colour names):
https://rstudio-pubs-static.s3.amazonaws.com/3486_79191ad32cf74955b4502b8530aad627.html

1.4 Look at the `mpg` dataset again by typing `mpg` in the console. Note that many values of `displ` and `hwy` are rounded. When plotting these values, there are therefore many overlapping points (i.e. *overplotting*). This can be avoided by **adding a small amount of random noise to each point** using a *position adjustment*; specifically, using `position = "jitter"`:

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy),
color = "red", position = "jitter")
```

A more complex scatterplot

2. With a scatterplot we plot the values of 2 numeric variables. However, we can include additional categorical variables by adding additional aesthetics to the plot. For instance, we can change the colour of the points in the scatterplot according to the categories of a third categorical variable. Alternatively (or additionally), we can change the size and/or shape of the points as well.

2.1 **Redraw the scatterplot from (1.2), but this time use a different colour for each different type of car.** (In other words, add a *colour aesthetic* for the variable `class`).

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class))
```

Note that this time we are adding a colour *aesthetic*, so the colour specification is inside the aesthetic mapping, and we don't specify a specific colour but instead the name of a variable. The values of this variable will be used to vary the colour of the points in the scatterplot. In (1.3), when we wanted to change the colour of all of the points to a different colour, we did not use a colour aesthetic; instead, the colour specification was

outside the aesthetic mapping. [Compare the position of the brackets in (1.3) and (2.1) and make sure you understand the difference.]

Also note that the legend is automatically created for you, showing which values of `class` are represented by which colour.

2.2 **Redraw the scatterplot from (1.2), but this time use a different shape for each different type of car.** (In other words, add a *shape aesthetic* for the variable `class`).

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, shape = class))
```

---

3. Draw a scatterplot showing city miles per gallon on the *x*-axis and highway miles per gallon on the *y*-axis. Use different sized points for engines with different engine displacement. (In other words, add a *size aesthetic* for the variable `displ`).

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = cty, y = hwy, size = displ))
```

Study the scatterplot you obtained. Note that the size of the point changes according to the value of the engine displacement variable; lower `displ` values have a smaller circle.

Geoms

There are almost 50 different geoms to use in `ggplot2`. In this module, we will consider the following geoms:

<table>
<tr><td><code>geom_point()</code></td><td colspan="2">Scatterplot</td></tr>
<tr><td><code>geom_bar()</code></td><td colspan="2">Bar chart</td></tr>
<tr><td><code>geom_histogram()</code></td><td colspan="2">Histogram</td></tr>
<tr><td><code>geom_boxplot()</code></td><td colspan="2">Boxplot</td></tr>
<tr><td><code>geom_line()</code></td><td colspan="2">Line chart</td></tr>
<tr><td><code>geom_hline()</code></td><td>Add horizontal line</td><td rowspan="4">These are usually combined with another geom</td></tr>
<tr><td><code>geom_vline()</code></td><td>Add vertical line</td></tr>
<tr><td><code>geom_abline()</code></td><td>Add diagonal line</td></tr>
<tr><td><code>geom_smooth()</code></td><td>Add smoothed line</td></tr>
</table>

Global vs. local mappings

In the previous plots, we used *local mappings*; in other words, we added the aesthetic mapping to the `geom_point()` call. A powerful aspect of `ggplot2` is that you can display multiple geoms in the same plot by adding multiple geom functions. If these geoms all use the same aesthetic mapping, you can avoid unnecessary typing by using a *global mapping*, i.e. by adding the aesthetic mapping to the initial `ggplot()` call.

Layering plots by using multiple geoms

4. We will explore the difference between using a local and a global mapping.

4.1 **Create a scatterplot with `displ` on the *x*-axis and `hwy` on the *y*-axis. Also add a smoothed line to aid in identifying the pattern in the scatterplot. Use a local mapping.** [Hint: use `geom_smooth()`]

```
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy)) + geom_smooth(mapping = aes(x = displ, y = hwy))
```

4.2 **Repeat (4.1) above, but this time use a global mapping.**

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) + geom_point() + geom_smooth()
```

Bar charts

5.1 **Draw a bar chart showing the type of car**.

```
ggplot(data = mpg) + geom_bar(mapping = aes(x = class))
```

5.2 Try **changing the colours of the bars using the color argument**.

```
ggplot(data = mpg) + geom_bar(mapping = aes(x = class, color = class))
```

Note that this only changes the outline colour of each bar, so this is not really what we want.

5.3 In a bar chart we can **change the colour of each individual bar by using the fill argument**.

```
ggplot(data = mpg) + geom_bar(mapping = aes(x = class, fill = class))
```

Adding labels to plots

Good labels are crucial to make a chart easy to understand. Labels can be added in `ggplot2` by using the `labs()` function. Labels can be added to many chart components, such as titles, subtitles and axes labels.

6.1 **Add a suitable title to the plot created in (2.1).**

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class)) + labs(title = "Engine displacement and fuel efficiency")
```

Note that the title is left aligned, which might not be what you require for your graph. You can center align the title by adding the following to the instruction above:

```
+ theme(plot.title = element_text(hjust = 0.5))
```

Tip:

Don't just copy the code from this document into your own R console. Firstly, you learn better by actually doing – in this case, typing. Secondly, you will run into issues in this question, as the quotation marks will probably copy incorrectly into RStudio!

6.2 **Add a suitable title to the *x*- and *y*-axes as well.**

To do this, you change the `labs()` function call in (6.1) as follows:

```
labs(title = "Engine displacement and fuel efficiency", x = "Engine displacement (in litres)", y = "Highway miles per gallon")
```

6.3 **Add a suitable title for the legend.**

This is done using the color argument in `labs()`, so include the following in your `labs()` function call in (6.2):

```
color = "Type of car"
```

6.4 **Experiment with adding a suitable subtitle as well, using the argument `subtitle =` in the `labs()` function call.**

Note that when including a subtitle you might not want to center align your title.

You should now understand the general syntax of `ggplot` and be able to create and customise scatterplots and bar charts.
Once you have mastered the basics of creating plots in `ggplot`, it is not difficult to create other types of plots, since the basic principles remain the same.

In Tutorial 3, you will practice these skills, and also encounter some different plot types and geoms.