SMI105, Week 1: Introduction to ggplot2

What is this document?

Welcome to SMI105: Data Visualisation! This document accompanies the first workshop.

There'll be a document like this every week on Thursday. Each Thursday, we'll work through how to do different bits of data viz, and each week will build on the previous week.

While the documents will give you a sense of what we're doing, and (crucially) will provide all the code that you need to carry out the tasks you'll need on this module, the activity during the workshops will put it into context and help you to understand. For this reason, *please make notes* throughout the workshops, and afterwards.

If you manage to get a copy of the exercises that is not printed out, I would encourage you to **not** copy and paste the code into your R script/console, but to write it yourself. This will help you remember the key functions and get a sense for the syntax and arguments used in `R`

What happens in these workshops?

Today, as with most Thursdays, we're making a bunch of different graphs.

This involves working with three main things:

- R
- RStudio
- ggplot2

Let's explain what each of these is.

`R` is a statistical programming language, which has been developed over several years by a huge range of people.

RStudio is what's called an "Integrated Development Environment" (or IDE) for R. In practice, this means that RStudio is a more manageable way to work in R.

ggplot2's a *package* for R. R packages allow R to do more different things than what you get when you download R on its own, and there's thousands of them; different people install different packages based on their own needs.

I know these terms are a bit dry, and might make the module seem more technical and less design-y than you might have expected. So let's start drawing some graphs.

How do we draw graphs?

In order to draw graphs in the way we're doing it, you need to be on a machine with R, RStudio, and ggplot2 installed. In this workshop you're probably sitting in a computer lab at a machine with those things installed on it, though you might also be on your personal machine. (This is fine, though there's a chance that we can't give you as much support.)

Hopefully your machine now looks like the one you can see on the projector. If it doesn't, try Ctrl-Shift-N.

Let's draw some graphs

The way this handout works is that when you see plain-text – the following's an example –

```
library(ggplot2)
```

– you should type it into the top left pane (your 'R script') **exactly as written**, and hit Ctrl-Enter (or the "Run" button) (or Cmd-Enter if you're on an Apple machine). (It's very common to think you've copied something exactly as written, but made a small error in transcription – this is totally normal, don't panic if it happens.)

It's likely that we'll need to **install** some packages before we get any further. To start, let's type and run the following:

```
install.packages("tidyverse")
```

What this does is install the **tidyverse**, which is a wide range of different packages, many of which we'll be using this semester.

Once it's installed - and this process might take a couple of minutes - please load it by typing the following:

```
library(tidyverse)
```

By running the above command, we've loaded the tidyverse, which includes **ggplot2**, the main package we'll be using this semester. By loading the ggplot2 package, we've also loaded a bunch of different **datasets** (I'll explain this more in the coming weeks). So, let's draw some graphs.

The **economics** data contains some information about, amongst other things, the number of people in the US who were unemployed over time. Let's make a graph of how it's changed.

```
ggplot(data = economics) +
  aes(x = date, y = unemploy) +
  geom_line()
```

The **midwest** data contains some information about different counties in the states of the Midwest of the US. Let's make a graph of how many counties there are in each state in the Midwest.

```
ggplot(data = midwest) +
  aes(x = state) +
  geom_bar()
```

The **diamonds** data contains information about diamonds sold in the US. Let's make a graph of how much they each sold for.

```
ggplot(data = diamonds) +
  aes(x = price) +
  geom_histogram()
```

…and another one…

```
ggplot(data = diamonds) +
  aes(price) +
  geom_density()
```

…and another one.

```
ggplot(data = diamonds)+
  aes(x = price) +
  geom_freqpoly()
```

Let's also use the **diamonds** data to make a graph of how diamond prices vary by what colour they are…

```
ggplot(data = diamonds) +
  aes(x = color, y = price) +
  geom_boxplot()
```

…and another one.

```
ggplot(data = diamonds) +
  aes(x = price, color = color) +
  geom_density()
```

Finallly, the **mpg** data contains information on cars, including the miles per gallon they get on highways, and their engine displacement. (I don't know anything about cars). Let's look at the relationship between those two variables.

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

What was that?

In each of the above cases, we wrote **three** lines of code, stitched them together with the + sign, and ran the lines of code we ran.

The lines began as follows:

- ggplot(
- aes(
- geom_

What does that mean?

Starting with **ggplot** means we're using the **ggplot** command: we're telling R that we're using the ggplot2 package, and - in practice - that we're drawing a graph. We then open a bracket where we specify what **data** we're using. We used a few different things: diamonds, mpg, economics, and so on. Finally on this line we closed the bracket and added a +, to indicate that something more was coming.

We then use an **aes** parenthesis, to indicate that we're adding **aesthetics**. This doesn't mean aesthetics in the sense of "making things look nice", but specifying what variables are going where. (We'll go through this in more detail in the coming weeks.) Within this bracket, we indicate exactly what's going where, using the equals sign =. Before the equals sign we indicate the location of the variable we're interested in – x, y, colour, there's more to come – and after the equals sign we indicate the variable we want to use – price, state, date, and so on.

Finally, we add a geometric object, starting with geom_. These geometric objects are the specific objects that our graphs are made up of. Bar charts are made up of bars, hence geom_bar() (and its cousin, geom_col(), for column, when totals have already been calculated), but scatterplots are made up of points, hence geom_point(). (For the moment these brackets are empty; that won't be the case all the way through the semester).

So, while we're going to be extending this a lot through the course of the semester, fundamentally what we're doing is: - declaring a graph using ggplot(), and declaring the data; - adding aesthetic mappings with aes(), pairing aesthetics with variables; - adding geometric objects.

Task

Get R and RStudio installed on your own machine if you've got one, and run this entire handout to check it works. If you don't have your own machine, please instead run this entire handout on a university computer in a different room from where this workshop has taken place. You can install R here (https://cloud.r-project.org/); if you're on a Windows machine click "Download R for Windows", if you're on a Mac click "Download R for (Mac) OS X". You can install RStudio here (https://www.rstudio.com/products/rstudio/download/); click the first "Download" button, under "RStudio Desktop".

Please install R **first** and RStudio **second**. (It can cause some problems if you don't.)

You'll also need to install the **tidyverse**, as explained towards the start of the handout.