# Ggplot2 handout

*Adam Bartonicek*

*17/08/2019*

## Setup

To start coding in R, you first need to run RStudio. You should be able to find it either on the desktop or when you type "RStudio" into the Windows start menu. After your get RStudio running, go to the top left corner & click File >> New File >> R Script. Alternatively, you can just press CTRL + SHIFT + N. You should now have in front of you a fresh window titled "Untitled1". You are now ready to start coding!

First things first, we're going to load some packages. There are many packages that extend R's capabilities. You can think of a "package" like a toolbox - it's a single place to store many tools designed for a specific purpose. Ggplot2 is a package, and it's a part of an even bigger package called `tidyverse`. Tidyverse contains a lot of (great) packages, so it's like a toolbox that contains other toolboxes. Or a shed.

To load tidyverse, simply write:

```r
library(tidyverse)
```

To run a line of code, highlight it with your mouse and press CTRL + ENTER. Or, you can also go to the top right corner of your script and press "Run" (but that gets tedious after a while - I highly recommend the CTRL + ENTER method).

After you load tidyverse, you will see a message about packages being attached and about conflicts. That's good! You can ignore it.

## Your first ggplot2 plot

Because you have loaded tidyverse, the `ggplot2` package was loaded automatically with it. We are going to be using the `diamonds` dataset that comes preloaded with `ggplot2`. Try to make your first plot by writing and running the following two lines of code:

```r
ggplot(data = diamonds, aes(x = carat, y = price)) +
  geom_point()
```

You should now see a scatterplot showing what looks like a positive relationship between carat (a diamond's weight) and price. Congrats! You have just made your first ggplot2 plot.

## Exploring `diamonds`

Write `diamonds` on a new line of your script and try running it. R will print out the first few lines of the `diamonds` dataset into the console in the lower-left panel, and this should give you an idea of how the dataset looks like. You can also try running `str(diamonds)` and `glimpse(diamods)`. The functions `str()` and `glimpse()` give you a high level overview of the dataset, and they're especially handy for figuring out what types of variables you have in your dataset. For example, `str(diamods)` shows that `cut`, `color`, & `clarity` are 'ordered factors', which means they are categorical variables. The other variables are either 'numerical' (`num`) or integer (`int`) which means they are continuous. Knowing the type of the variables in your dataset can help you know what sort of plots you can make.

**Further info & ideas**

- To make a boxplot, use `+ geom_boxplot()`. Make sure that you specify a categorical x variable and a continuous y variable in your `aes()` call.
- To make a violin plot, use `+ geom_violin()`. Violin plots take in a categorical x variable and continuous y variable, just like boxplots.
- To make a histogram, use `+ geom_histogram`. Histogram only takes one continuous variable - x (the y variable represents discrete counts in each bin of x). So make sure to specify only the x variable in your `aes()` call.
- `+ geom_density()` works almost exactly the same as histogram, except that it fits a smoothed density curve instead of discrete bins. Try it out!
- Try specifying a categorical variable as either colour (`col =`) or fill (`fill =`) in your `aes()` call.
- Try playing around with the breaks and limits of your y axis. To do this, first add `+ scale_y_continuous()`. Then, write `limits =` into the brackets, e.g. `+ scale_y_continous(limits = c(0, 10000))` will set the y axis limits from 0 to 10,000. You can use `seq()` to specify your breaks as a sequence, e.g. `scale_y_continous(breaks = seq(0, 15000, by = 2000))` will give you a break every 2000 points along the y axis.
- What does adding `+ theme_bw()` or `+ theme_minimal()` do?
- Play around with the `+ labs()` function, e.g. to change the name of the x axis to 'Boring x axis', use `+ labs(x = 'Boring x axis')`. Try changing the label of the x axis, title, or colour/fill if you have it in your plot!
- What does `+ scale_y_log10()` do? How might it be useful in the diamonds dataset?
- Can you guess what does `+ geom_smooth()` do? What does `+ geom_smooth(method = 'lm')` do?