Lecture 2: Chapter 3 by Kieran Healy

Marc Kaufmann

September 23, 2019

Team Assignments: Choosing First Group

For the first group, I will pick 4 students who will complete a group assignment to share in 2 weeks time (lecture 4). Since it is the first group, I am happy to provide a bit more feedback if needed. So let's do this.

If you want to switch with someone else and someone else is happy to, then please go ahead and swap - just let me know before the weekend.

Note to the interested student

Try to follow along by typing it yourself, adding comments as you make mistakes or realize things. Write the code out in chunks:

How Ggplot Works

```
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------------------------- tidyverse 1.2
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts -------------------------------------------------------------------------- tidyverse_conflicts()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

The code specifies the connections between the variables in the data on one hand and the colors, points, and shapes you see on the screen. These logical connections are called *aesthetic mappings* or simply *aesthetics*.

How to use ggplot:

- `data = gapminder`: Tell it what your data is
- `mapping = aes(...)`: How to map the variables in the data to aesthetics
  - axes, size of points, intensities of colors, which colors, shape of points, lines/points
- Then say what type of plot you want:
  - boxplot, scatterplot, histogram, ...
  - these are called 'geoms' in ggplot's grammar, such as `geom_point()` giving scatter plots

```
library(ggplot2)
... + geom_point() # Produces scatterplots
... + geom_bar() # Bar plots
.... + geom_boxplot() # boxplots
... #
```

You link these steps by *literally* adding them together with `+` as we'll see.

Exercise: What other types of plots are there? Try to find several more `geom_` functions.

Ans: A few other kinds of plots and their corresponding Geo_ functions are: 1. Area Graph 2. Step Chart 3. Line Chart 4. Dot Plot 5. Density Chart

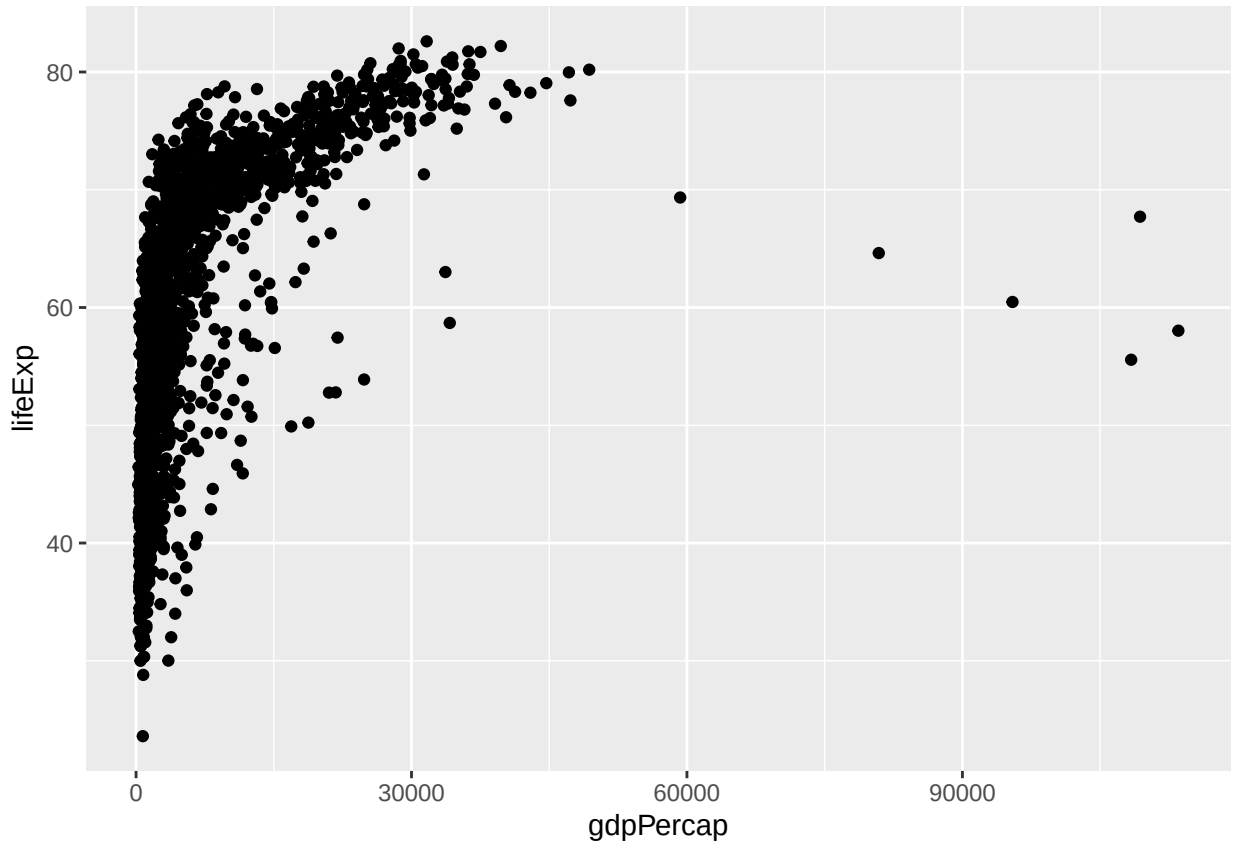
```
... + geom_area()
... + geom_step()
... + geom_line()
... + geom_dotplot()
... + geom_density()
#Error: '...' used in an incorrect context
```

Mappings Link Data to Things You See

```
library(gapminder)
library(ggplot2)
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp))
p + geom_point()
```



In detail:

- `data = gapminder` tells ggplot to use gapminder dataset, so if variable names are mentioned, they should be looked up in gapminder
- `mapping = aes(...)` shows that the mapping is a function call. Simply accept that this is how you write it
  - Kieran Healy: "The `mapping = aes(...)` argument *links variables* to *things you will see* on the plot"
- `aes(x = gdpPercap, y = lifeExp)` maps the GDP data onto `x`, which is a known aesthetic (the x-coordinate) and life expectancy data onto `x`
  - `x` and `y` are predefined names that are used by `ggplot` and friends

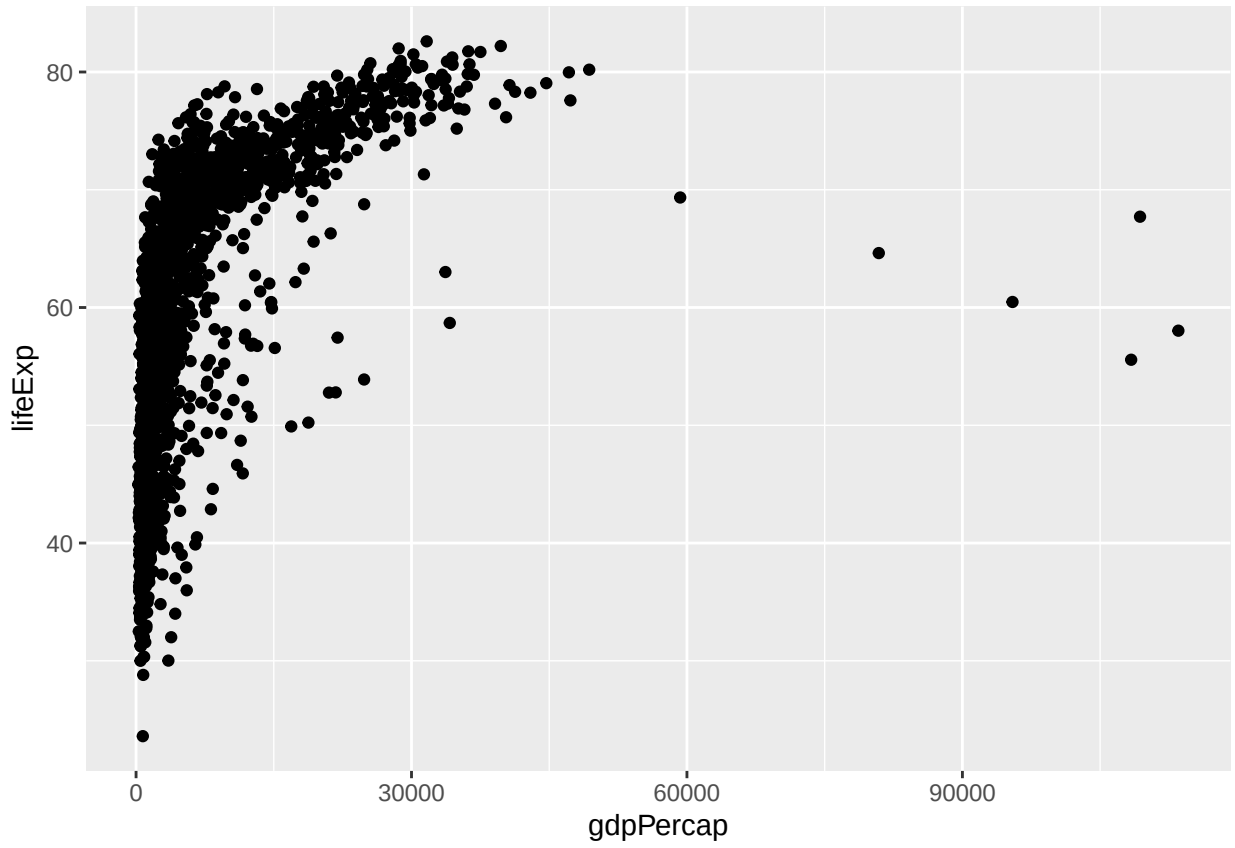
Importantly, mappings don't say *what* color or shape some variable will have – rather, it says that a given dataset will be mapped *to* the color or *to* the shape.

```
str(p)
str(p + geom_point())
```

Exercise: Make sure that your knitted version doesn't include all the output from the `str(...)` commands, it's too tedious.

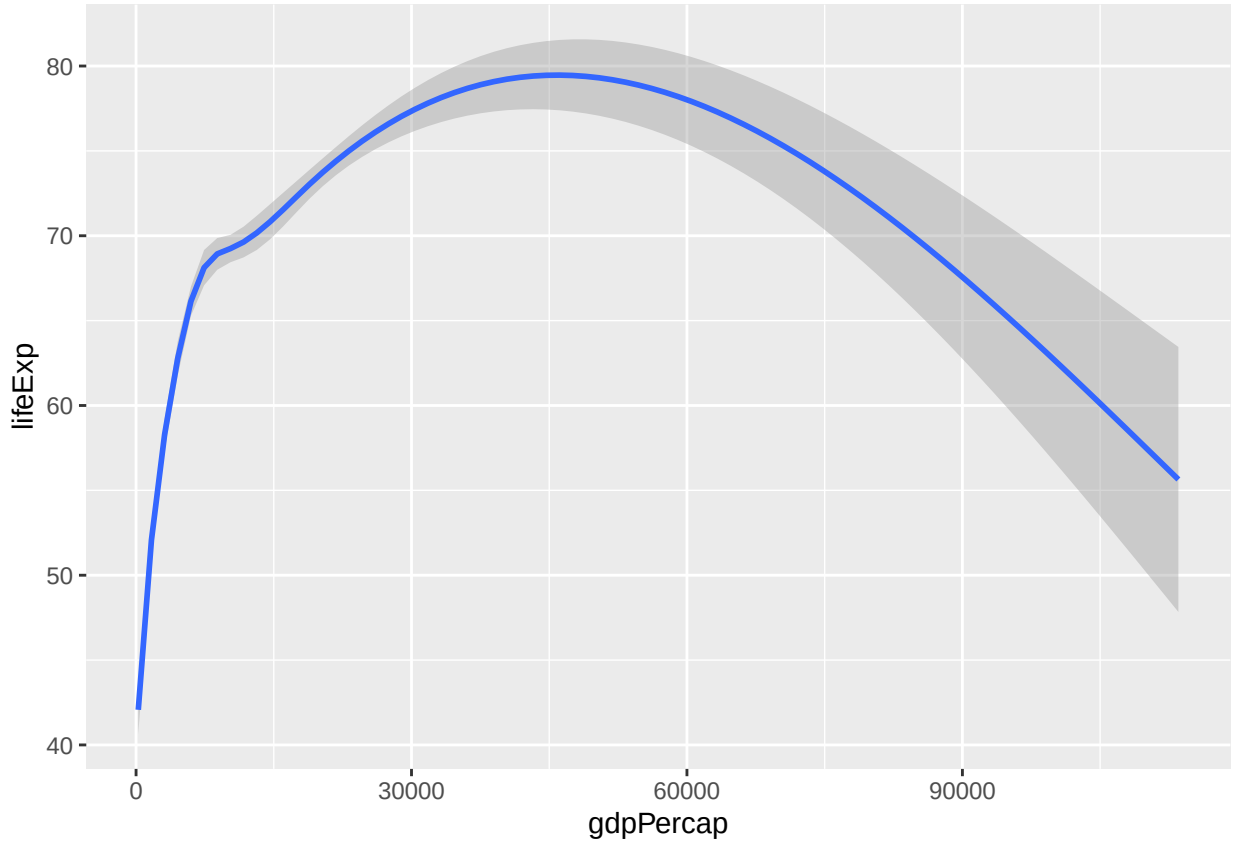
Finally, we add a *layer*. This says how some data gets turned into concrete visual aspects.

```
p + geom_point()
```




```
p + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Note: Both geom's use the same mapping, where the x-axis represents ... and the y-axis ... . But the first one maps the data to individual points, the other one maps it to a smooth line with error ranges.

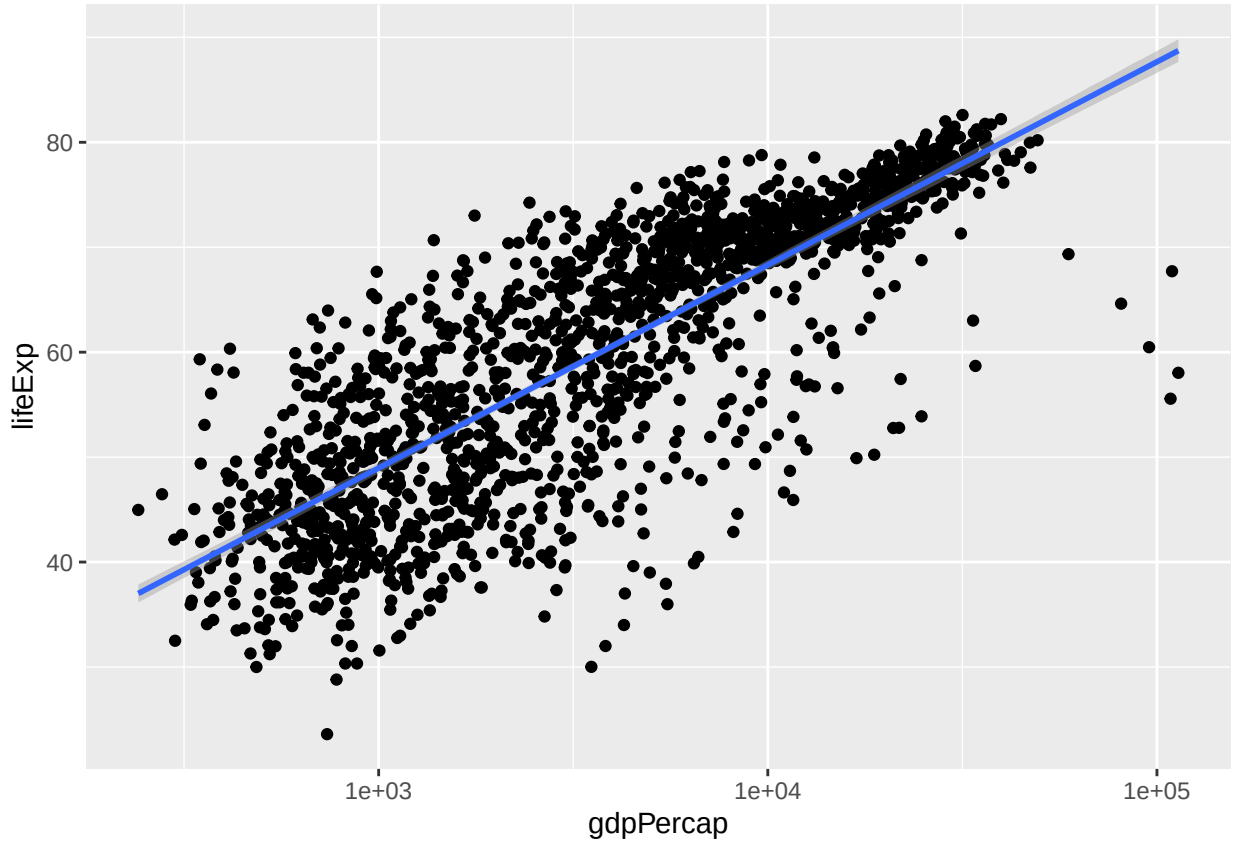
We get a message that tells us that `geom_smooth()` is using the method = 'gam', so presumably we can use other methods. Let's see if we can figure out which other methods there are.

```
?geom_smooth
p + geom_point() + geom_smooth() + geom_smooth(method = ...) + geom_smooth(method = ...)
p + geom_point() + geom_smooth() + geom_smooth(method = ...) + geom_smooth(method = ..., color = "red")
```

You may start to see why ggplots way of breaking up tasks is quite powerful: the geometric objects (long for geoms) can all reuse the *same* mapping of data to aesthetics, yet the results are quite different. And if we want later geoms to use different mappings, then we can override them – but it isn't necessary.

One thing about the data is that most of it is bunched to the left. If we instead used a logarithmic scale, we should be able to spread the data out better.

```
p + geom_point() + geom_smooth(method = "lm") + scale_x_log10()
```



Exercise: Describe what the `scale_x_log10()` does. Why is it a more evenly distributed cloud of points now? (2-3 sentences.)

Ans: The `scale_x_log10()` function scales the x-axis of the plot to a log 10 basis. This makes the graph better because most of the data points are concentrated on the left side of the x-axis and a logrithamic scale allowed the graph to be focused on this side with an even distribution.

Nice! The x-axis now has scientific notation, let's change that.

```
library(scales)
p + geom_point() +
  geom_smooth(method = "lm") +
  scale_x_log10(labels = scales::dollar)
```

Exercise: What does the `dollar()` call do?

Ans: This function will format a vector of values as currency. If accuracy is not specified, values are rounded to the nearest cent, and cents are displayed if any of the values has a non-zero cents and the largest value is less than largest_with_cents which by default is 100,000.

```
?dollar()
```

Exercise: How can you find other ways of relabeling the scales when using `scale_x_log10()`?

Ans: We can use ?scale_x_log10() to find out the various parameters of the function. One of these allows us to pass a vector with character labels. Alternatively, we can also look into the documentation of the scales library to find other built-in functions like dollar() and comma().

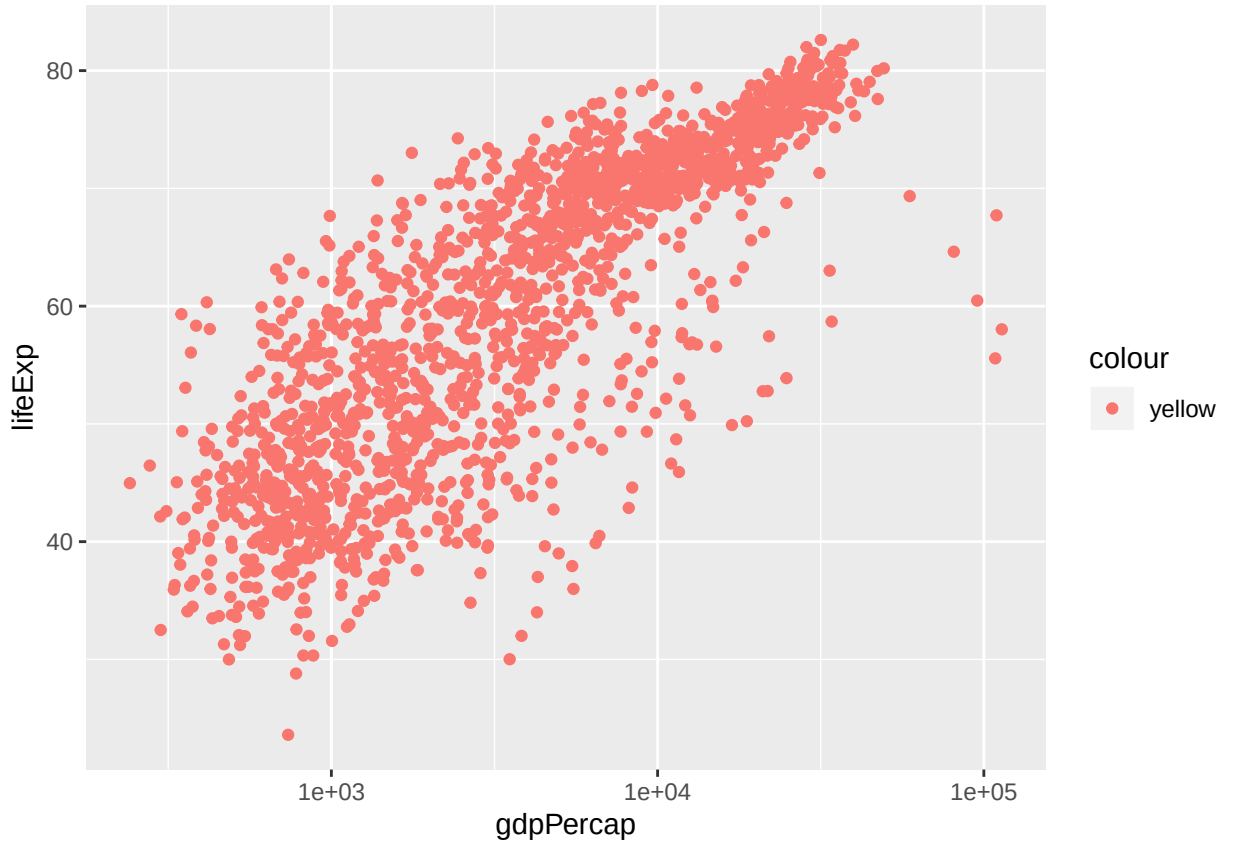
The Ggplot Recipe

1. Tell the `ggplot()` function what our data is.
2. Tell `ggplot()` *what* relationships we want to see. For convenience we will put the results of the first two steps in an object called `p`.
3. Tell `ggplot` *how* we want to see the relationships in our data.
4. Layer on geoms as needed, by adding them on the `p` object one at a time.
5. Use some additional functions to adjust scales, labels, tickmarks, titles.

- The `scale_`, `labs()`, and `guides()` functions

Mapping Aesthetics vs Setting them

```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp, color = 'yellow'))
p + geom_point() + scale_x_log10()
```



This is interesting (or annoying): the points are not yellow. How can we tell ggplot to draw yellow points?

```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp))
p + geom_point(color = 'yellow') + scale_x_log10()
```

Exercise: Based on the discussion in Chapter 3 of *Data Visualization* (read it), describe in your words what is going on.

Ans: When we define the color property inside the aes() function it treats it as a variable/data and plots it as a constant. It also makes a legend for it and uses the deault colour to plot it.

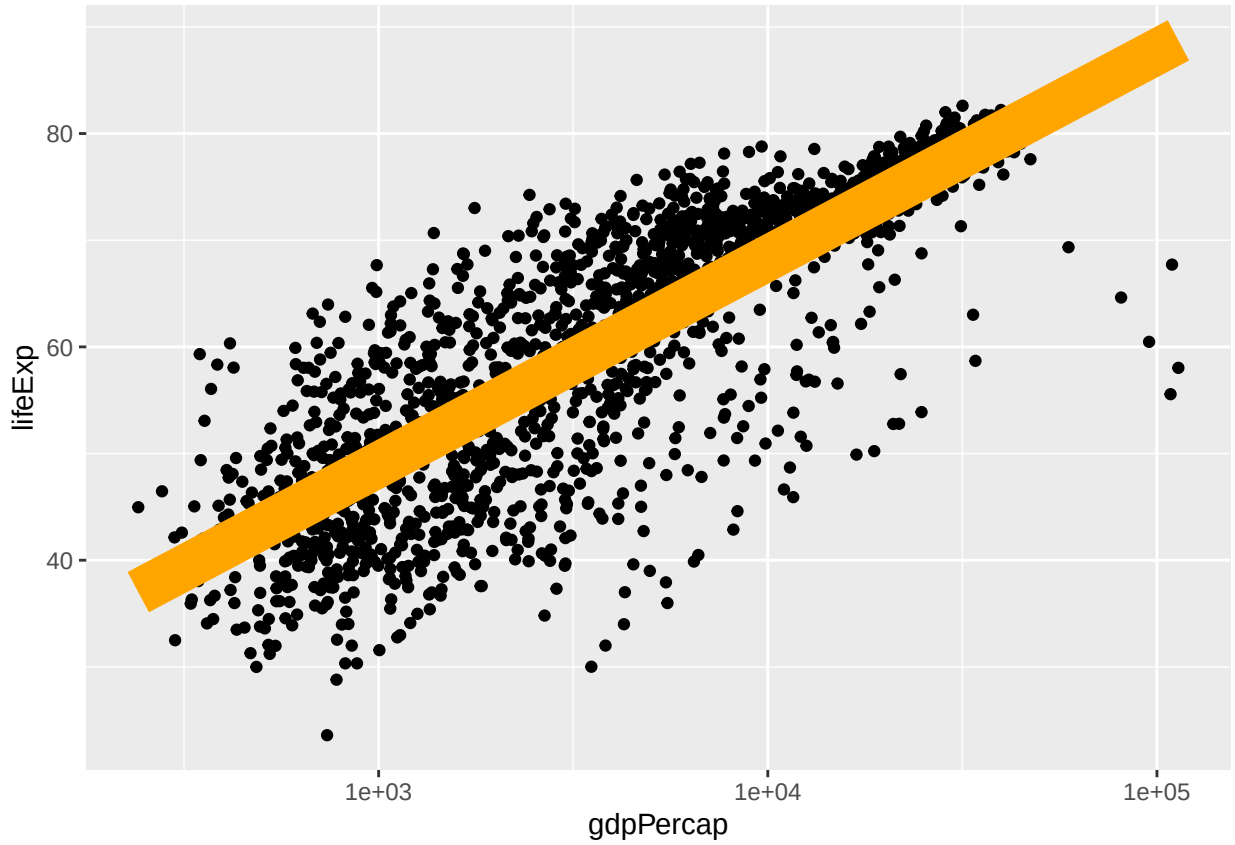
One way to avoid such mistakes is to read arguments inside `aes(<property> = <variable>)`as *the property in the graph is determined by the data in* .

Exercise: Write the above sentence for the original call `aes(x = gdpPercap, y = lifeExp, color = 'yellow')`.

Aesthetics convey information about a variable in the dataset, whereas setting the color of all points to yellow conveys no information about the dataset - it changes the appearance of the plot in a way that is independent of the underlying data.

Remember: `color = 'yellow'` and `aes(color = 'yellow')` are very different, and the second makes usually no sense, as `'yellow'` is treated as *data*.

```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp))
p + geom_point() + geom_smooth(color = "orange", se = FALSE, size = 8, method = "lm") + scale_x_log10()
```

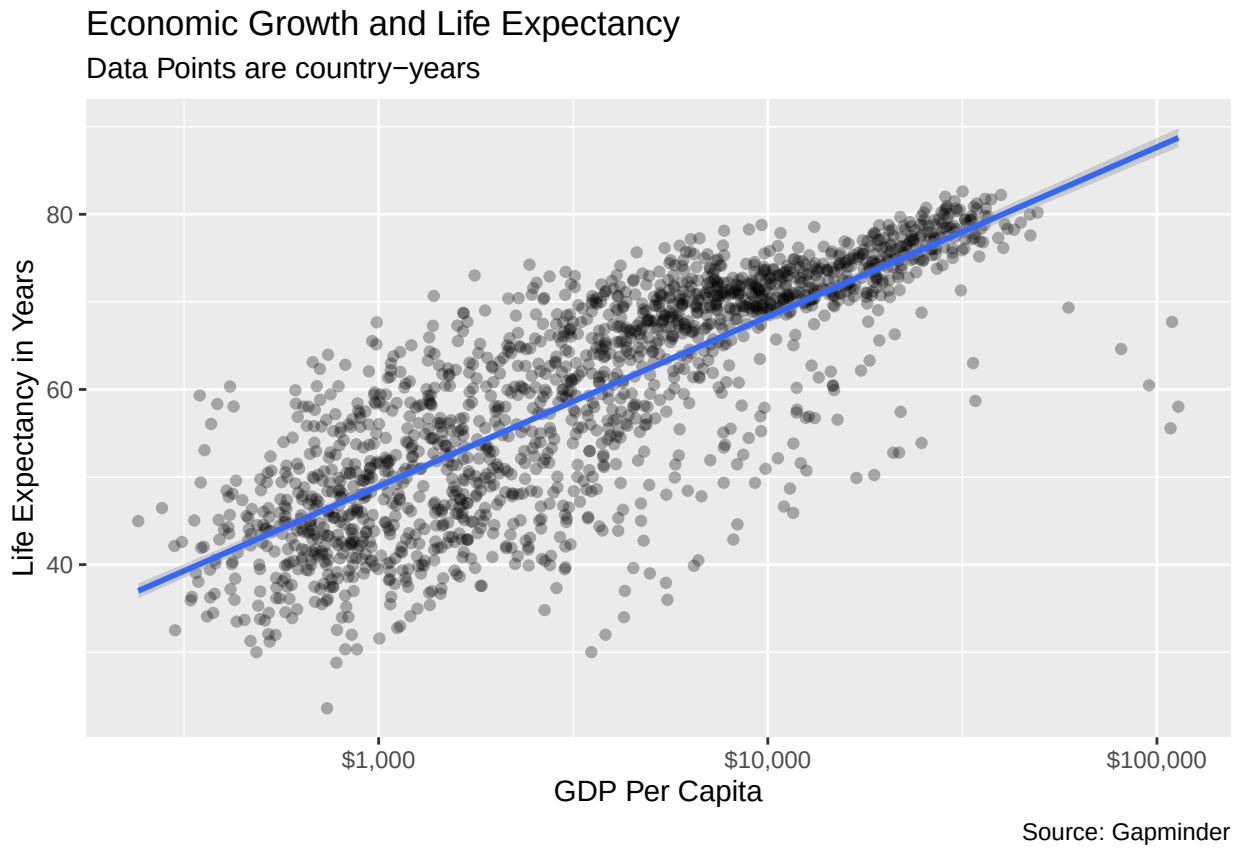


Exercise: Write down what all those arguments in `geom_smooth(...)` do.

Ans: 1. color = "Orange" sets the color of the line to orange. 2. se = FALSE turns off the standard error display/function on the line. Hence, standard error is not displayed anymore. 3. method = "lm" tells it to use a linear model to plot the line instead of the default gam model.

```
p + geom_point(alpha = 0.3) +
  geom_smooth(method = "gam") +
  scale_x_log10(labels = scales::dollar) +
  labs(x = "GDP Per Capita", y = "Life Expectancy in Years",
       title = "Economic Growth and Life Expectancy",
       subtitle = "Data Points are country-years",
```

```
       caption = "Source: Gapminder")
```



Coloring by continent:

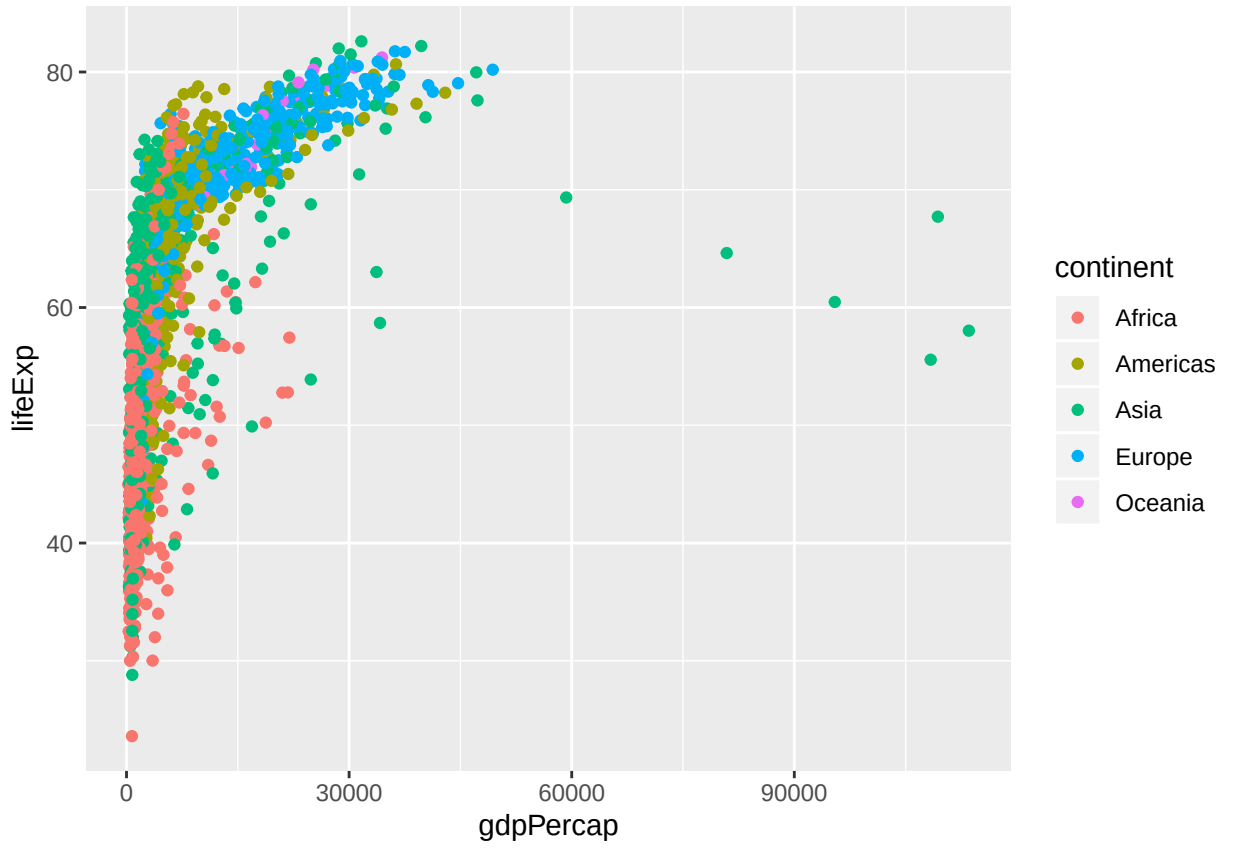
```
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp, color = continent, fill = continent))
p + geom_point()
```




```
p + geom_point() + scale_x_log10(labels = dollar)
```


```
p + geom_point() + scale_x_log10(labels = dollar) + geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

Exercise: What does `fill = continent` do? What do you think about the match of colors between lines and error bands?

Ans: The fill = continent property sets the color of the Standard Error error bar/region/ribbon according to the continent thus allowing it to match the colour set by the color = continent property. This results in matching coloured points, lines and standard error ribbons for theses line.

```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp))
p + geom_point(mapping = aes(color = continent)) + geom_smooth() + scale_x_log10()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

Exercise: Notice how the above code leads to a single smooth line, not one per continent. Why?

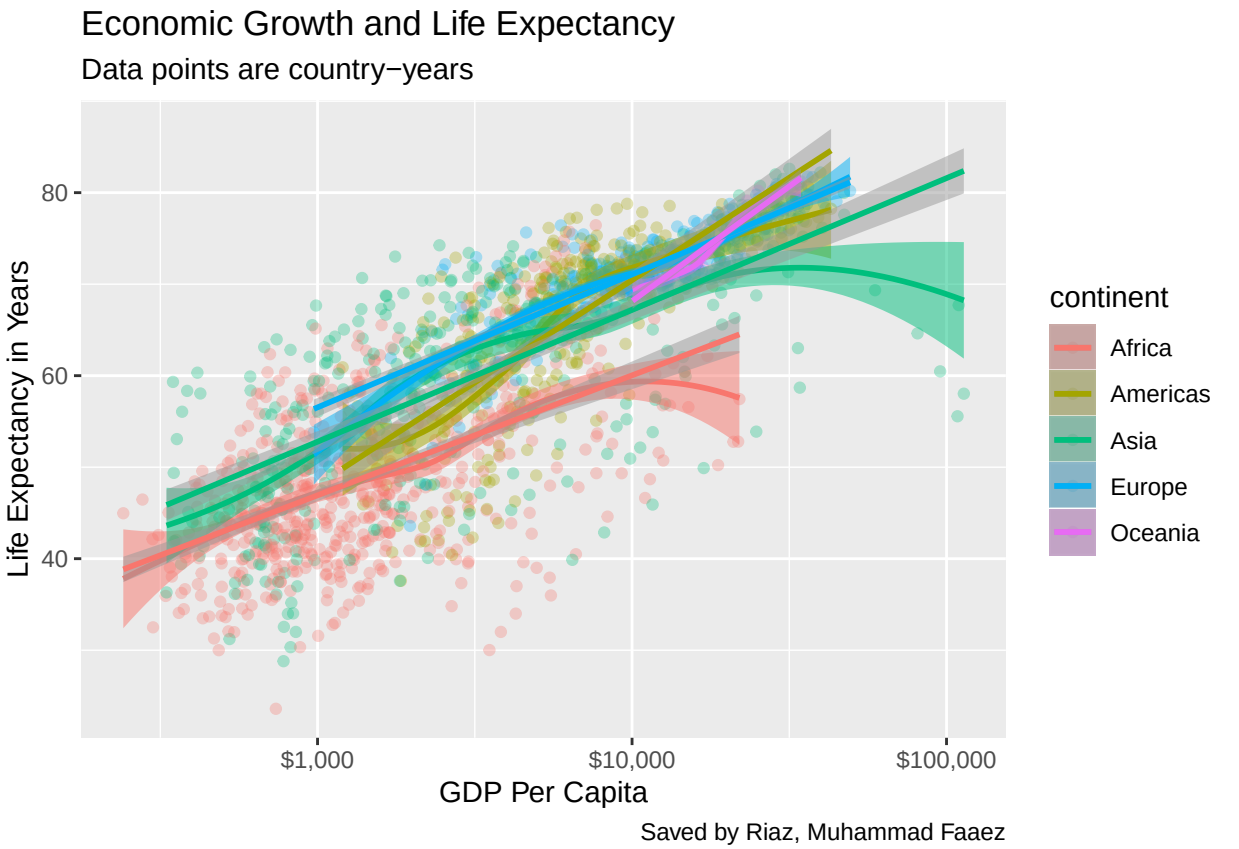
Ans: All geoms that are added to a ggplot() function inherit the mappings and properties of the function. However, in the above code the mapping is for color = continent is defined inside the geom_point function but not inside the geom_smooth function. Therefore, the smooth function does not inherit the color mapping and that results in just one smooth line.

Exercise: What is bad about the following example, assuming the graph is the one we want? This is why you should set aesthetics at the top level rather than at the individual geometry level if that's your intent.

Ans: There is a lot of repeated redundant code that could easily be avoided by setting the common aesthetics mapping within the ggplot() function. color = continent should be set in the ggplot function..

```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp, color = continent))
p + geom_point(alpha = 0.3) +
  geom_smooth(mapping = aes(fill = continent), alpha = 0.5) +
  scale_x_log10(labels = scales::dollar) +
  geom_smooth(method = "gam", alpha = 0.5) + labs(x = "GDP Per Capita", y = "Life Expectancy in Years",
         title = "Economic Growth and Life Expectancy",
         subtitle = "Data points are country-years",
         caption = "Saved by Riaz, Muhammad Faaez")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Additional (not Optional) Exercises

Exercise (Discourse): Find ways to save the figures that you made so that you can use them elsewhere too. Create a new folder to save only images. Use the command for saving to save the picture for the last image in your new folder, after you have updated the axes, title, subtitle, and caption of the image. Post your solution on Discourse and use it to include the final image above with a caption saying "Saved by " inside your Discourse post.

```
library(here)
```

```
## here() starts at C:/Users/faaez/OneDrive - Central European University/Current Courses/Data Coding 1
```

```
ggsave(here("lecture2/images", "lifExpVsGDP.png"))
```

```
## Saving 6.5 x 4.5 in image
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

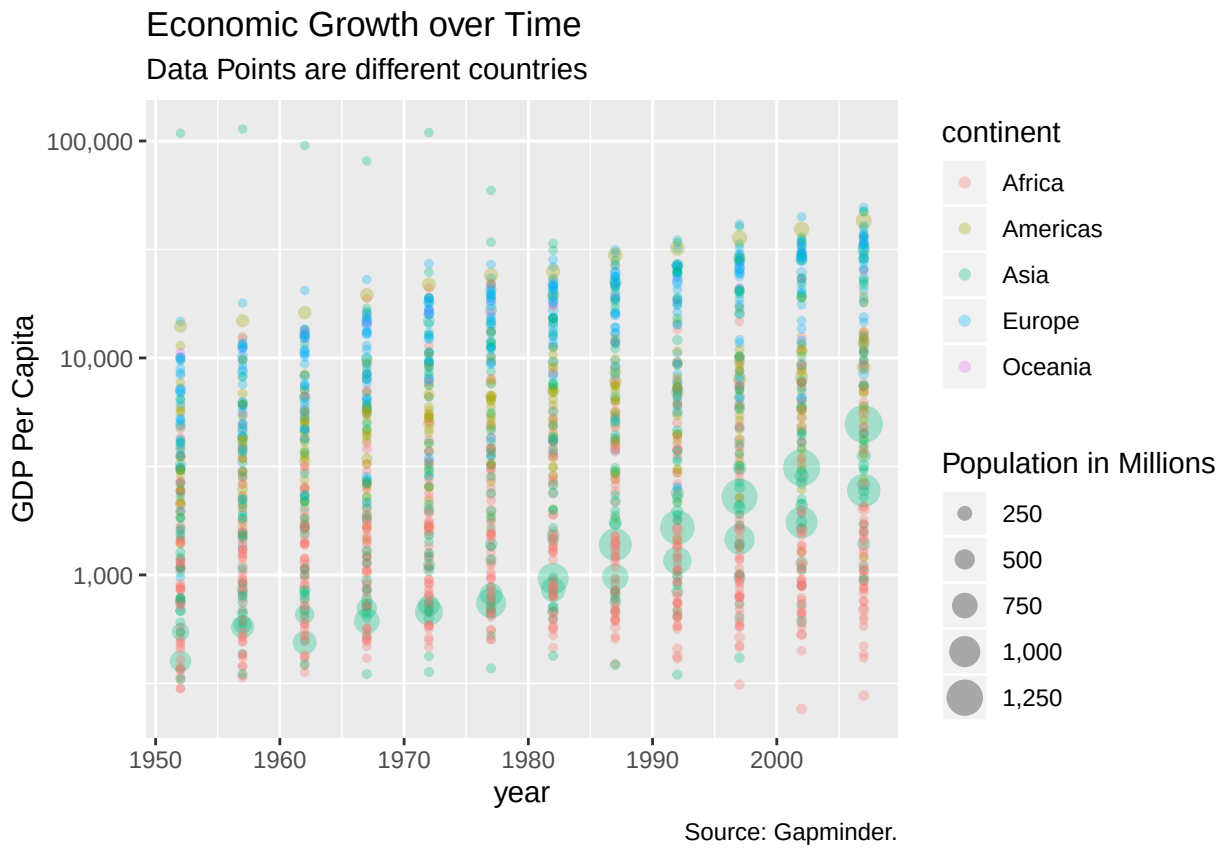
Exercise: Read section 3.8 "Where to go next" from DV. Based on those ideas, experiment and create two different graphs with the gapminder data. Describe each briefly in one sentence.

Graph 1: Plots a scatter plot of Population growth over time, size of data points depends on population of countries and their colour depends on the continent variable.

```
p <- ggplot(gapminder, aes(year, gdpPercap, size = pop / 1000000, color = continent))

p + geom_point(alpha = 0.3) +
  scale_y_log10(labels = scales::comma) +
  scale_radius(labels = scales::comma) +
```

```
  labs(y = "GDP Per Capita", Time = "Life Expectancy in Years",
       title = "Economic Growth over Time",
       subtitle = "Data Points are different countries",
       caption = "Source: Gapminder.", size = "Population in Millions")
```

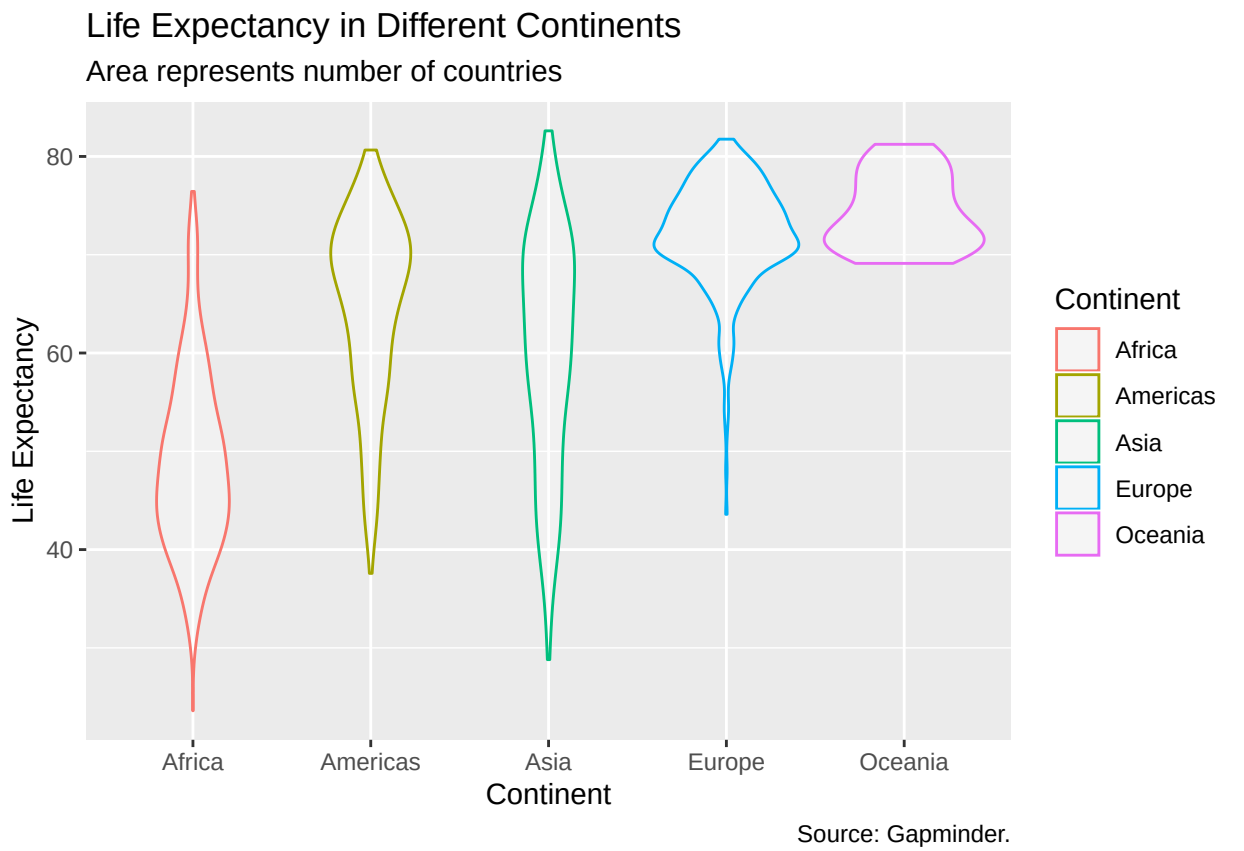


Graph 2: Plots a voilin graph of life expectancy in different continents.

```
q <- ggplot(gapminder, aes(continent, lifeExp, color = continent))

q + geom_violin(scale = "area", alpha = 0.3) +

   labs(x = "Continent", y = "Life Expectancy",
        title = "Life Expectancy in Different Continents",
        subtitle = "Area represents number of countries",
        caption = "Source: Gapminder.", color = "Continent")
```



Exercise: Read section 1.6 of R for Data Science on *Getting help and learning more*. Go back to an error from your previous assignment – or pick a new one – and post a reproducible error as described in that section on the discourse forum.

Exercise: Do exercise 3.2.4 from R for Data Science. Include your code in chunks, describe the output and code (where necessary) in the surrounding text.

Run ggplot(data = mpg). What do you see?

```
ggplot(data = mpg)
```

How many rows are in mpg? How many columns?

Ans: mpg has 234 rows and 11 columns

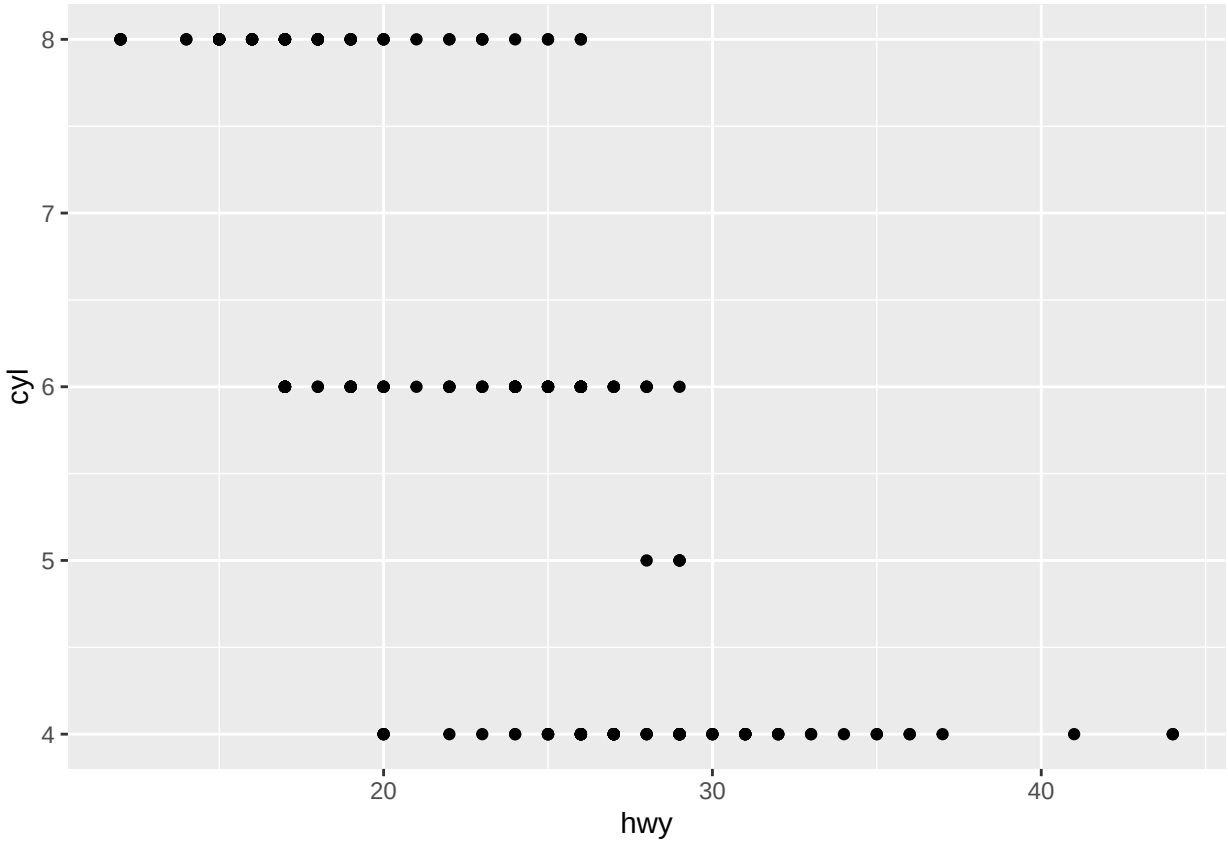
What does the drv variable describe? Read the help for ?mpg to find out.

Ans: It describes whether the car is front-wheel drive, rear-wheel drive or 4-wheel drive f = front-wheel drive, r = rear wheel drive, 4 = 4wd

Make a scatterplot of hwy vs cyl.

```
hwy_cyl <- ggplot(mpg, aes(hwy, cyl))

hwy_cyl + geom_point()
```

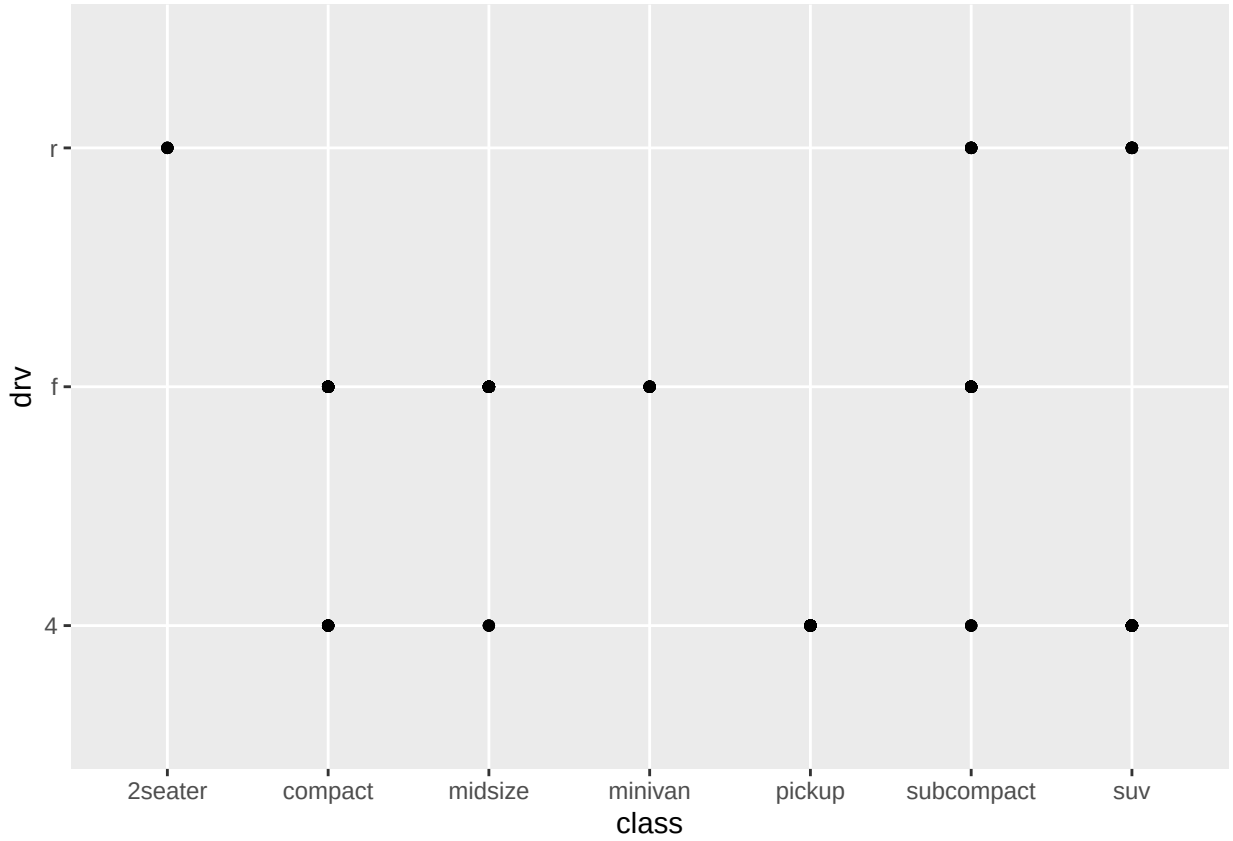


What happens if you make a scatterplot of class vs drv? Why is the plot not useful?

Ans: The graph has been drawn below. It is not useful because both of the variables are discrete and scatter plots are used for visualizing data that has continuous variables on both axis.

```
class_drv <- ggplot(mpg, aes(class, drv))

class_drv + geom_point()
```

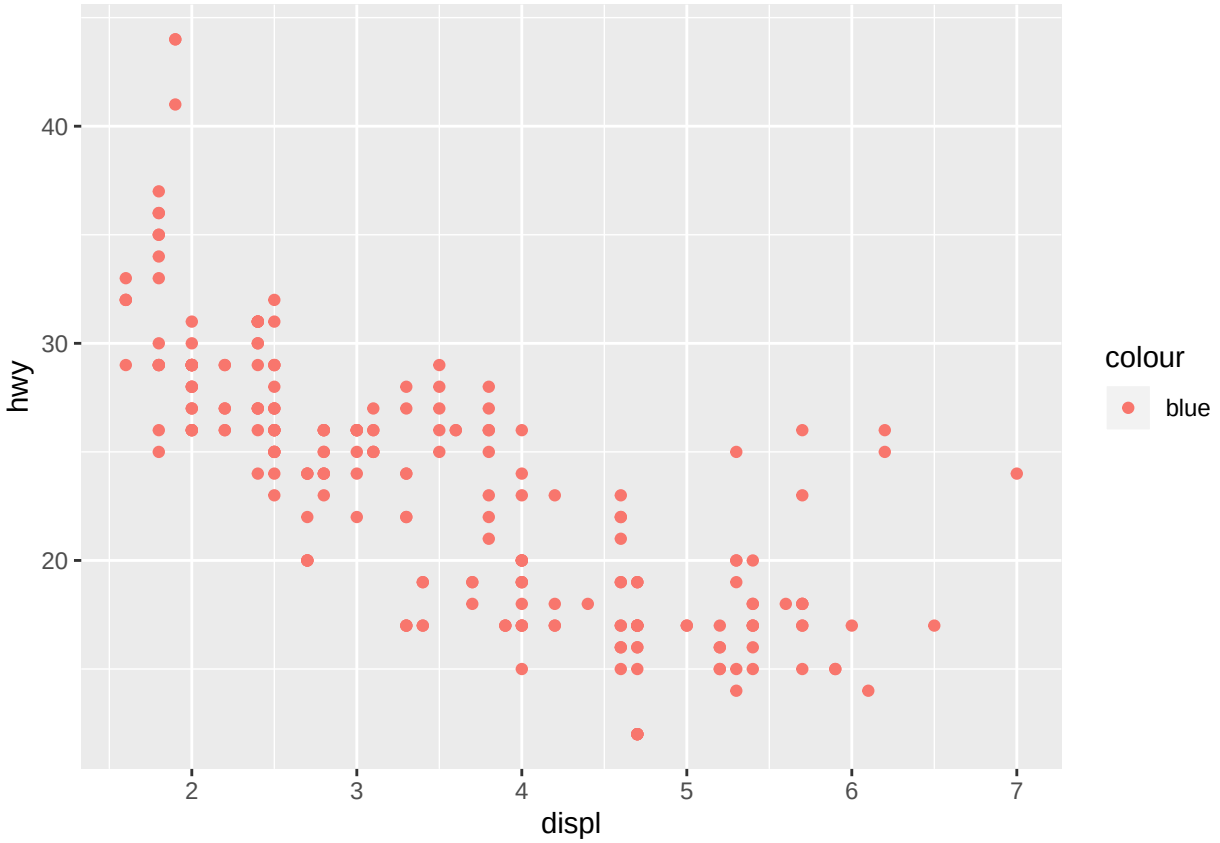


Exercise: Go through Exercises in 3.3.1. If an exercise does not make sense immediately (such as you don't know what a categorical variable is), replace the question by a question that addresses that point (in the case of the caregorical variable "What are categorical and continuous variables and how are they different in R?"). Write it down, try to answer that question, and ignore the original question. That way you don't end up spending too much time on this one exercise.

1. What's gone wrong with this code? Why are the points not blue?

Ans: When the color property is defined inside the aes() function it treats it as a variable/data and plots it as a constant. It also makes a legend for it and uses the deault colour to plot it. The color needs to be passed seperately (outside the aes func) to the geom function to make the points blue.

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



2. What are categorical and continuous variables?

Ans: Categorical are the same as discrete variables that is any variables that can take only a limited number of values. Continuous variables are numeric variables that can take on an unlimited number of values.

In this data there are 8 categorical (manufacturer, model, displ, cyl, trans, drv, fl, class) and 3 continuous variables (year, cty, hwy).

3. Map a continuous variable to color, size, and shape. How do these aesthetics behave differently for categorical vs. continuous variables?

Ans: For color, continuos variables are diffrentiated by a changing color gradient while categorical variables are mapped onto discrete/seperate colors.

For size, both categorical and continuos behave in the same way as even continuos variables are divided into discrete categories by size before being plotted.

For shape, continuous variables are not accepted in the aesthetic.

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

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

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

4. What happens if you map the same variable to multiple aesthetics?

Ans: It is acceptable to map the same variable to multiple aesthetics, for example cty can be mapped to both size and color.

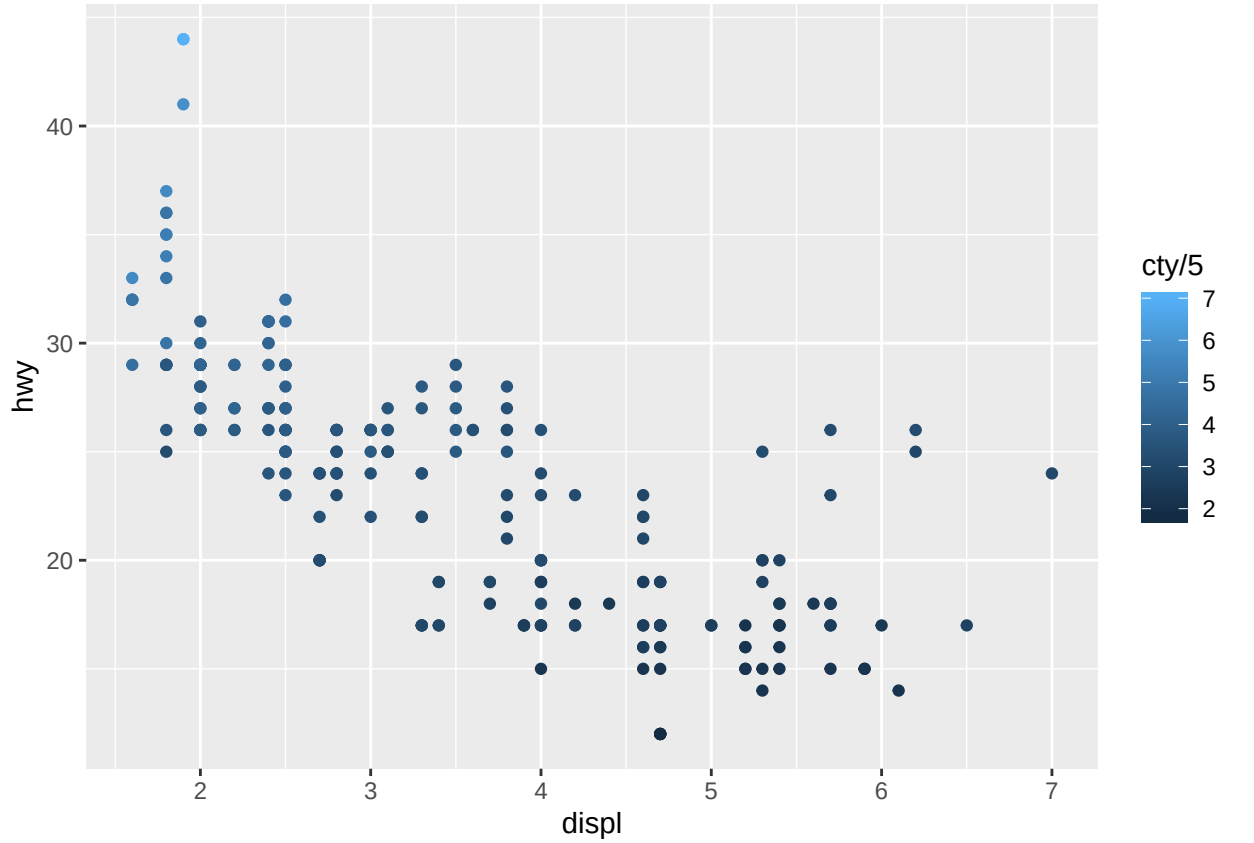
5. What does the stroke aesthetic do? What shapes does it work with? (Hint: use ?geom_point)

Ans: Stroke aesthetic sets the size of the stroke (non filled area) when plotting a scatter plot. All point based graphs can use the stroke aesthetic, this includes geom_point and geo_dotplot.

6. What happens if you map an aesthetic to something other than a variable name, like aes(colour = displ < 5)? Note, you'll also need to specify x and y.

Ans: It will compute the resultant value for each row and then plot the result. In this particular example it will compute a boolean value for each row marking True where displ is less than 5 and False if not and then plotting it.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = cty / 5)) +
  geom_point()
```



Exercise: Read the (very short) Chapter 4 of R for Data Science and try exercise 1 in section 4.4.

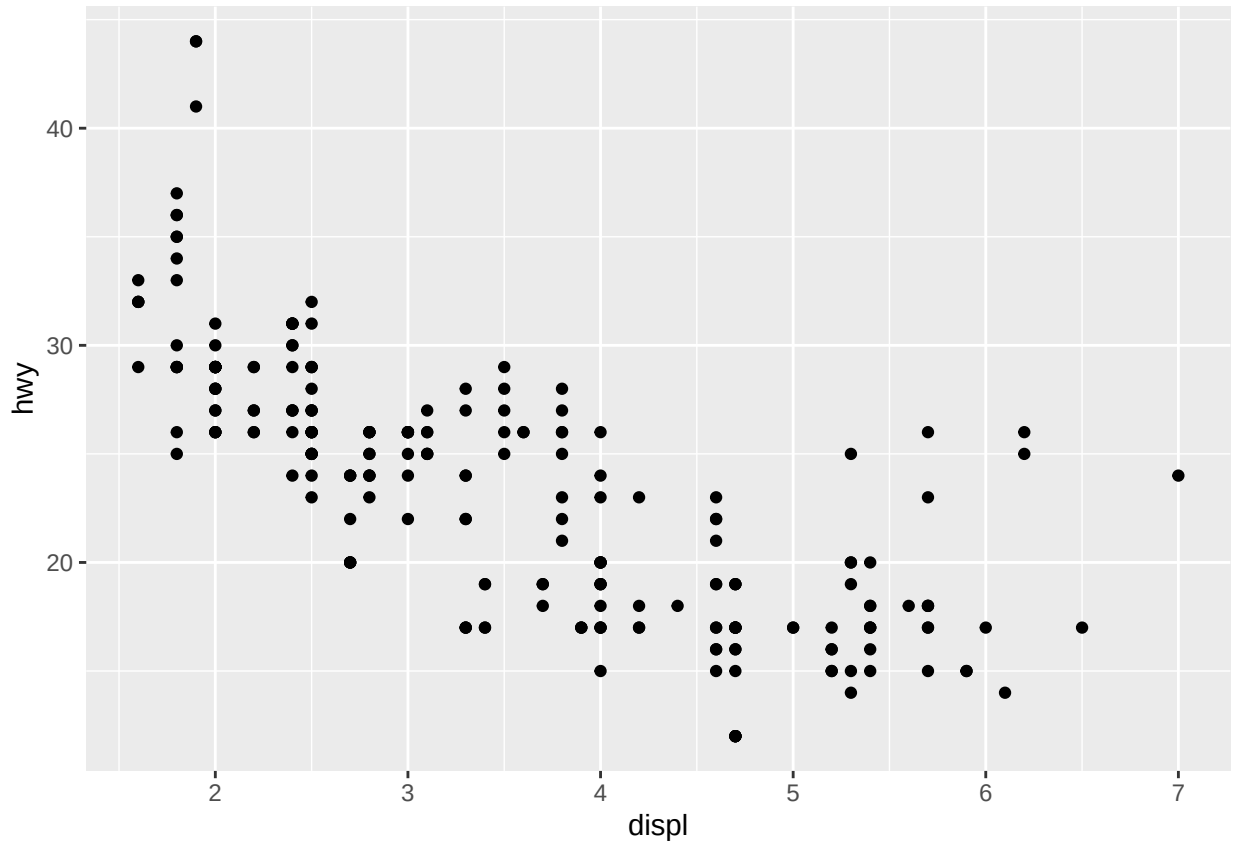
Ans:

1. Why does this code not work? Because there is a different character instead of the i in the name of the variable.

2: Tweak each of the following R commands so that they run correctly:

```
library(tidyverse)

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




```
filter(mpg, cyl == 8)
```

```
## # A tibble: 70 x 11
##    manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
##    <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a6 q~   4.2  2008     8 auto~ 4        16    23 p     mids~
##  2 chevrolet    c150~   5.3  2008     8 auto~ r        14    20 r     suv
##  3 chevrolet    c150~   5.3  2008     8 auto~ r        11    15 e     suv
##  4 chevrolet    c150~   5.3  2008     8 auto~ r        14    20 r     suv
##  5 chevrolet    c150~   5.7  1999     8 auto~ r        13    17 r     suv
##  6 chevrolet    c150~   6    2008     8 auto~ r        12    17 r     suv
##  7 chevrolet    corv~   5.7  1999     8 manu~ r        16    26 p     2sea~
##  8 chevrolet    corv~   5.7  1999     8 auto~ r        15    23 p     2sea~
##  9 chevrolet    corv~   6.2  2008     8 manu~ r        16    26 p     2sea~
## 10 chevrolet    corv~   6.2  2008     8 auto~ r        15    25 p     2sea~
## # ... with 60 more rows
```

```
filter(diamonds, carat > 3)
```

```
## # A tibble: 32 x 10
##    carat cut     color clarity depth table price     x     y     z
##    <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  3.01 Premium I     I1       62.7    58  8040  9.1   8.97  5.67
##  2  3.11 Fair    J     I1       65.9    57  9823  9.15  9.02  5.98
##  3  3.01 Premium F     I1       62.2    56  9925  9.24  9.13  5.73
##  4  3.05 Premium E     I1       60.9    58 10453  9.26  9.25  5.66
##  5  3.02 Fair    I     I1       65.2    56 10577  9.11  9.02  5.91
```

```
##  6  3.01 Fair    H     I1       56.1    62 10761  9.54  9.38  5.31
##  7  3.65 Fair    H     I1       67.1    53 11668  9.53  9.48  6.38
##  8  3.24 Premium H     I1       62.1    58 12300  9.44  9.4   5.85
##  9  3.22 Ideal   I     I1       62.6    55 12545  9.49  9.42  5.92
## 10  3.5  Ideal   H     I1       62.8    57 12587  9.65  9.59  6.03
## # ... with 22 more rows
```

3: Press Alt + Shift + K. What happens? How can you get to the same place using the menus?

Ans: Shows Keyboard shortcuts. We can get to the same using help->keyboard shortcuts help

Bonus Exercise: Why did I load the `scales` library twice via `library(scales)` to knit?

Ans: Because the first time you did it eval was set to FALSE, therefore the library did not load hence you had to do it a second time.

Assignment 3

1. Do the exercises in these lecture notes.
2. Knit lectures 2, making sure to get rid of those `eval=FALSE` that are just there because I didn't complete the code
3. Upload your pdf on Moodle
4. Grade assignment 2 on Moodle – let me know if you can't access Moodle!
5. If you are part of the team that does the first group assignment, start thinking about how you are going to do the assignment. You have until lecture 4.