Lecture 2: Chapter 3 by Kieran Healy

Veronika Palotai

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

```r
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------------------

## 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 ---------------------------------------------------------------------------------- tid
## 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

```r
library(ggplot2)
library(gapminder)
p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))
p + geom_point() # Produces scatterplots
p + geom_bar() # Bar plots
p + 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.

Answer:

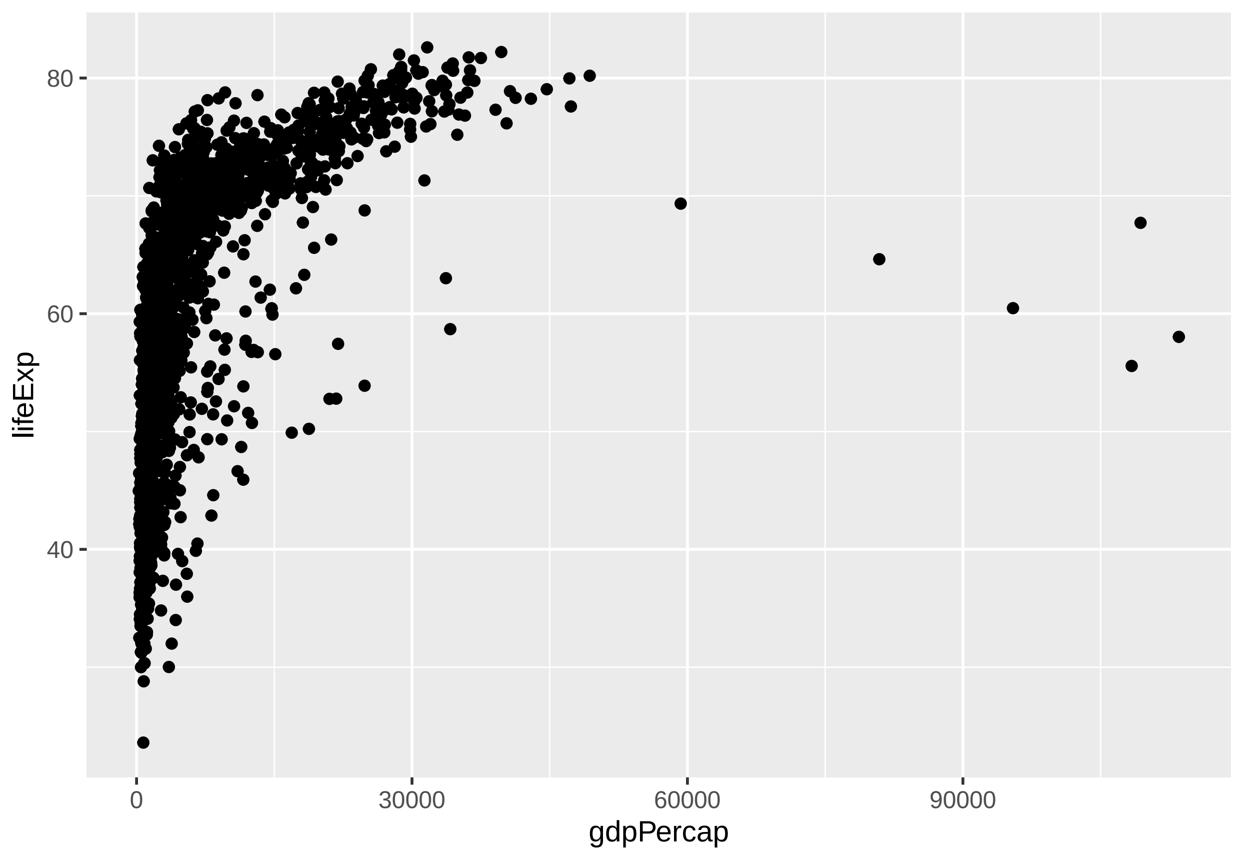
- *geom_histogram()*: visualizes the distribution of a single continuous variable by dividing the x axis into bins and counting the nr. of observations in each bin
- *geom_qq()*: so-called quantile-quantile plot, this is used to determine if your data is close to being normally distributed
- *geom_line()*: a plot type used for time series and trend lines

Mappings Link Data to Things You See

```r
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
```

```r
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.

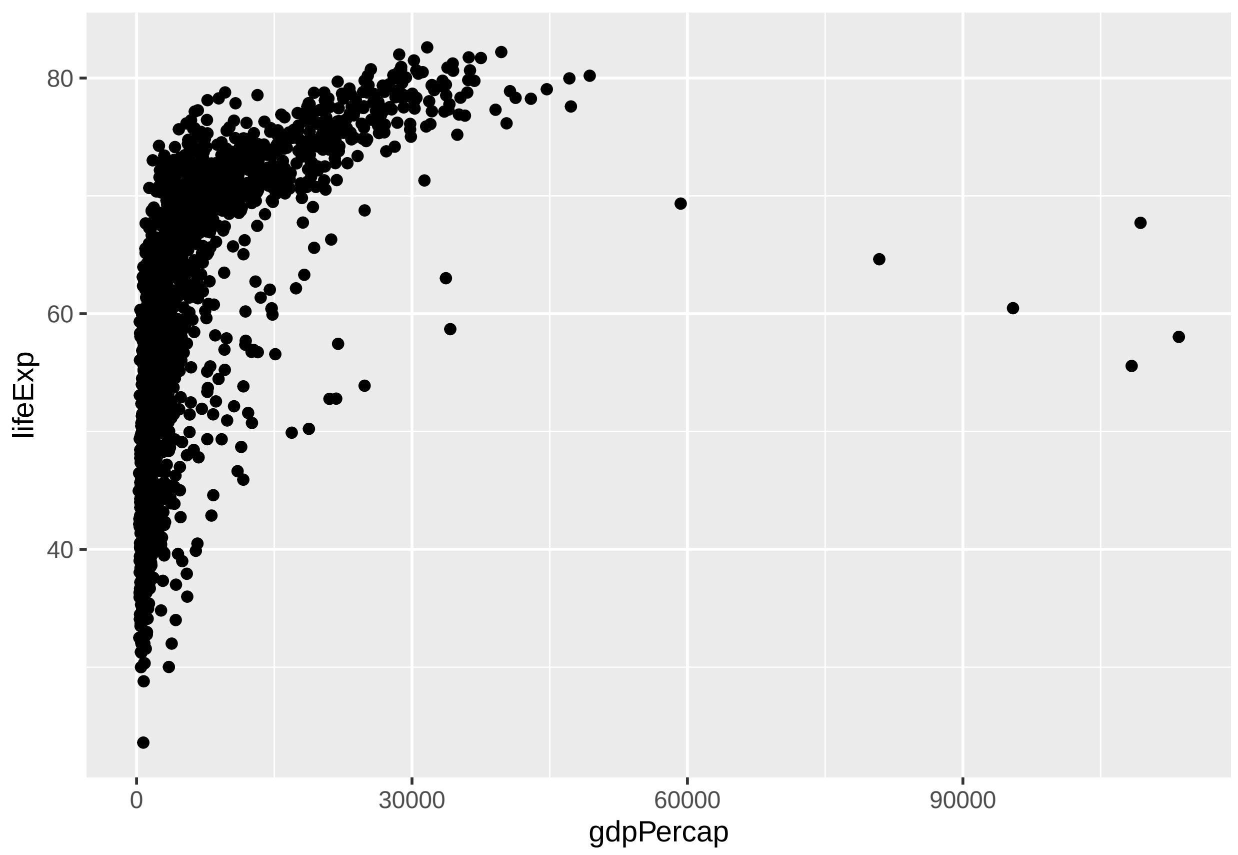
```
str(p)
str(p + geom_point())
```

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

Answer: In order to make this sure, it is enough to set the evaluation to false at the beginning of the code chunk if we don't want to run the code at all. If we want to run the code but don't want to see the results then results='hide' should be added at the beginning of the chunk.

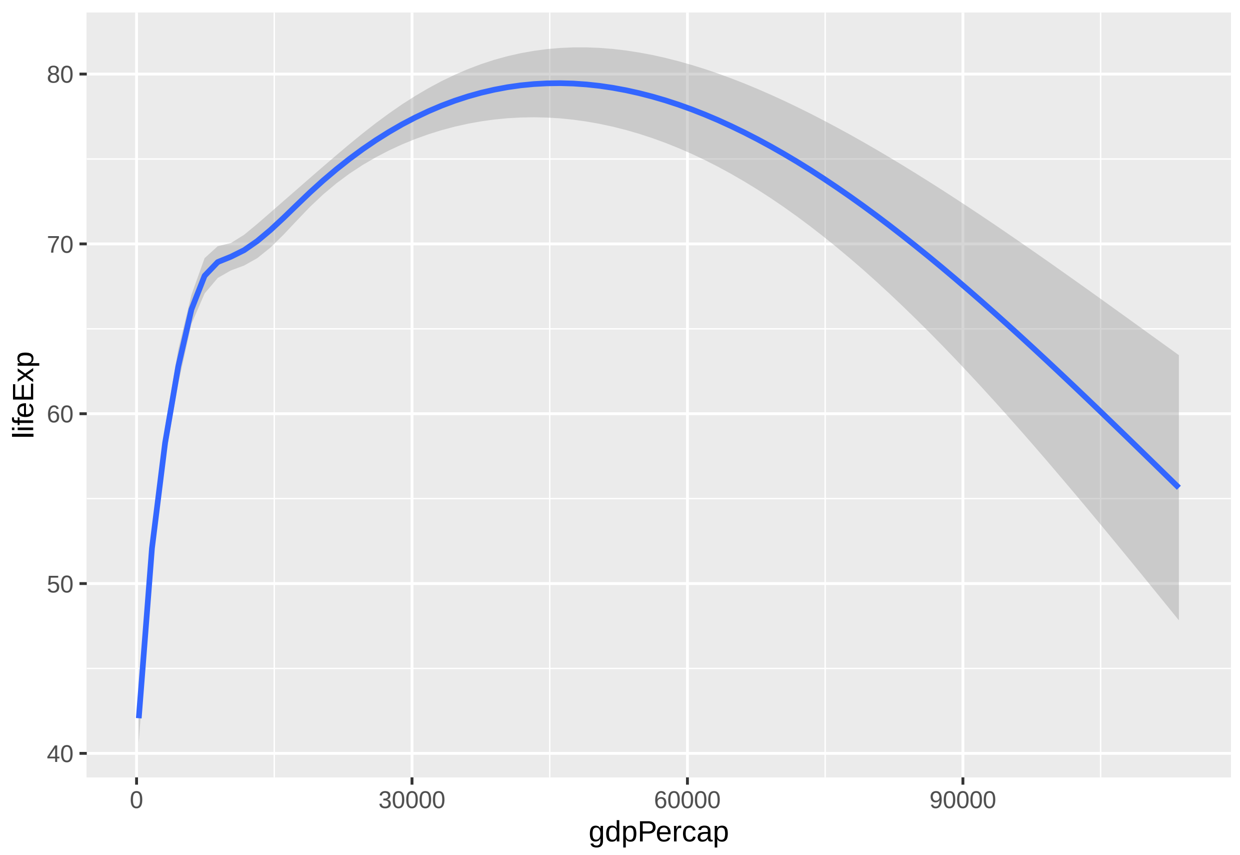
Finally, we add a *layer*. This says how some data gets turned into concrete visual aspects.

```
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.

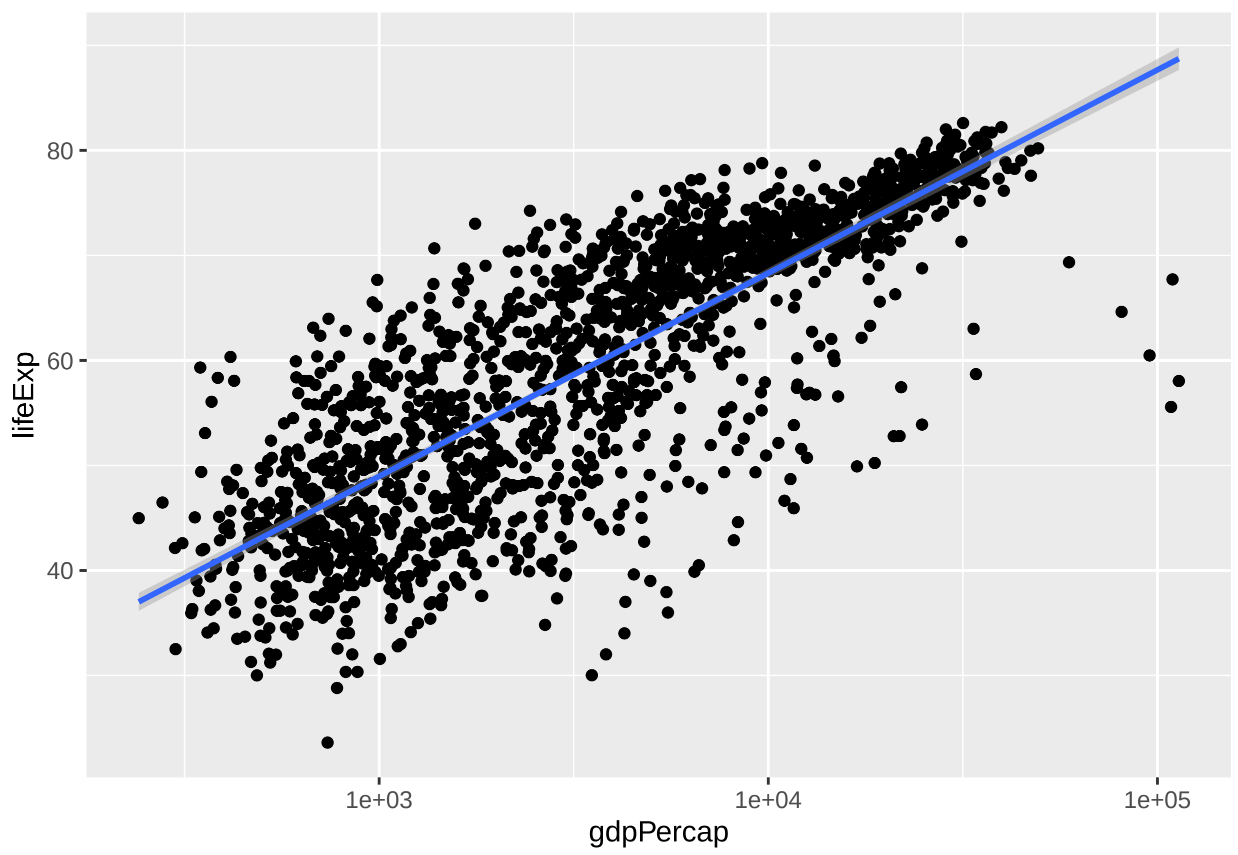
We get a message that tells us that `geom_smooth()` is using the method = 'gam', so presumably we can use other methods. Let's see if we can figure out which other methods there are.

```
?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.)

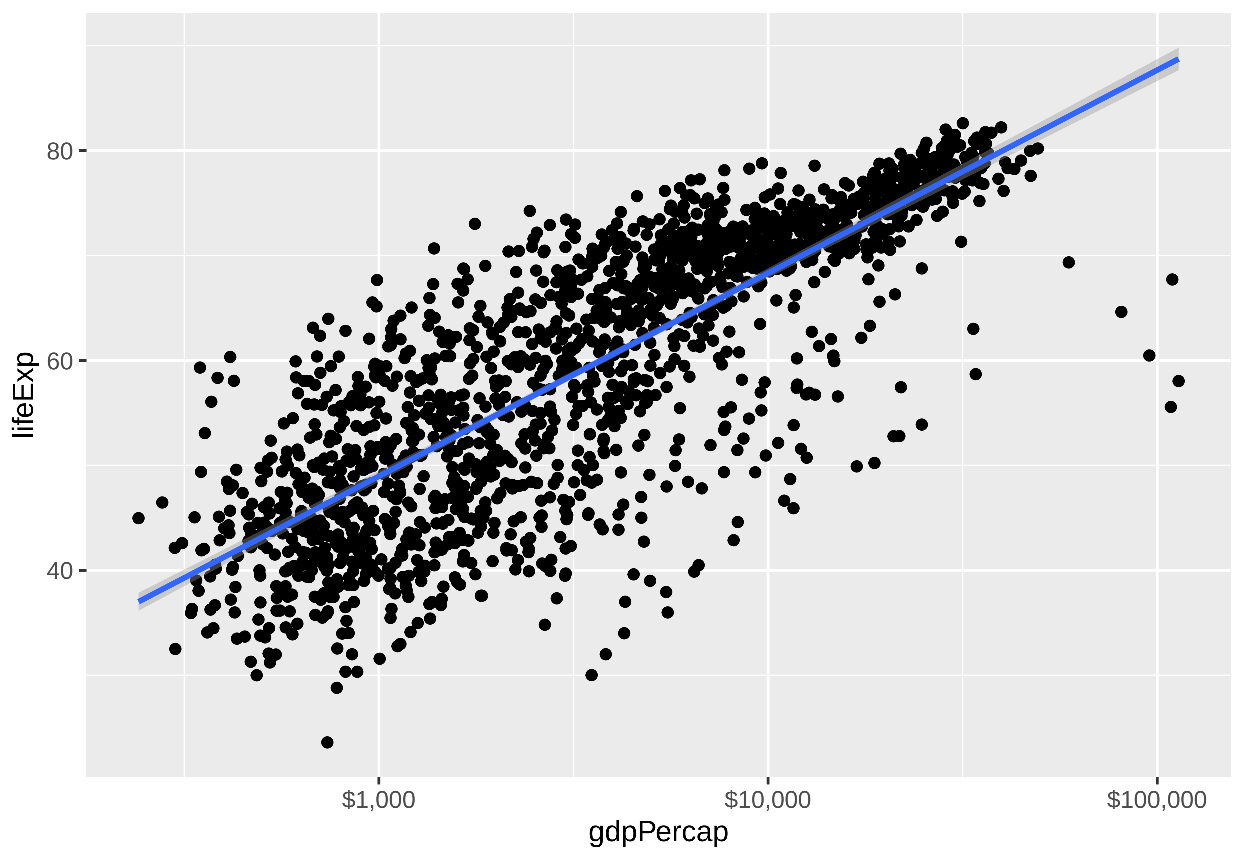
Answer: Transforms the x axis into a log scale with base 10. It is important to note that the this scale function transforms the data. In our case there are values which are much larger than the bulk of the data. These values are much further from the other data points on a standard linear scale. Unlike a linear scale, a logarithmic one is based on orders of magnitude therefore in this case using a logarithmic scale results in a more evenly distributed cloud of points.

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

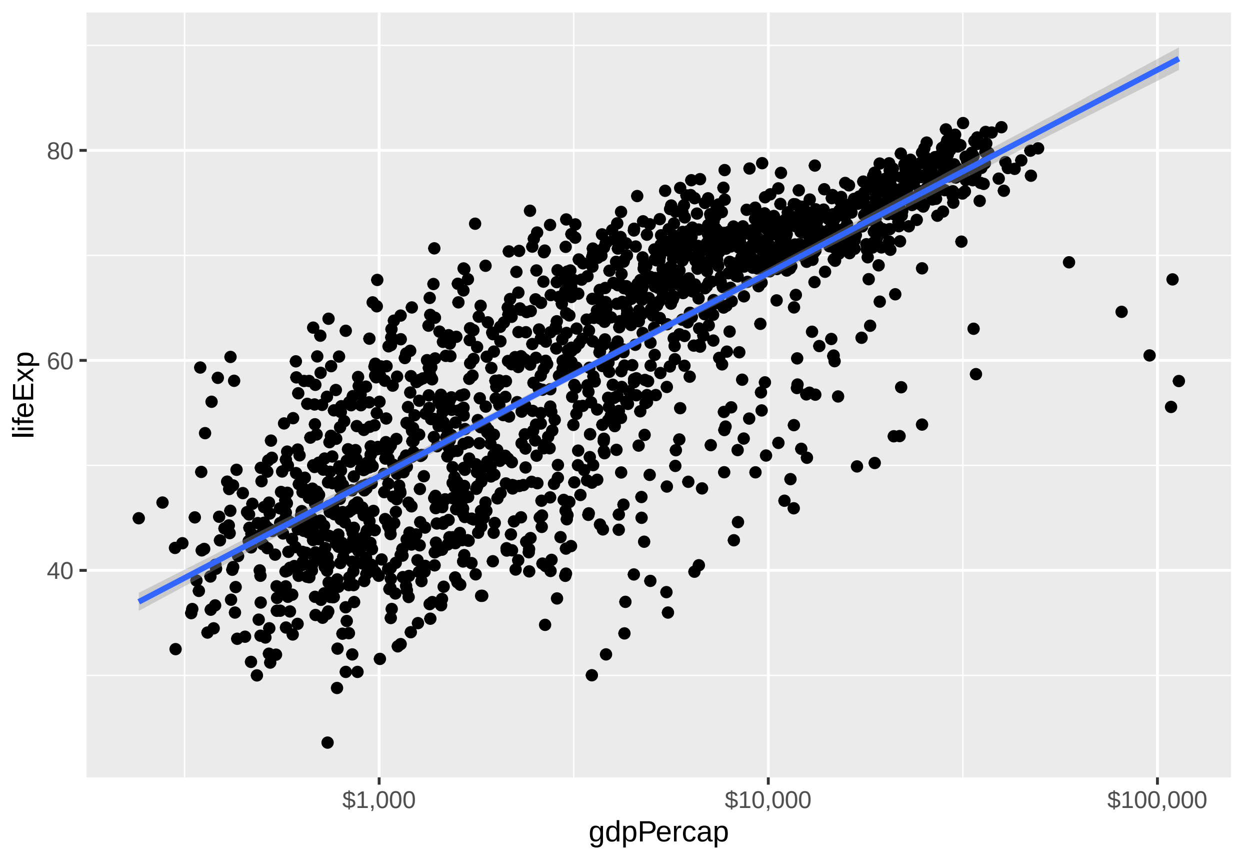
```
library(scales)
```

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

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

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

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



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



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

Answer: According to the description: *The returned 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()
```

```
## starting httpd help server ... done
```

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

Answer: By using the ?scale_x_log10() command and reading the labels section of the help page.

The Ggplot Recipe

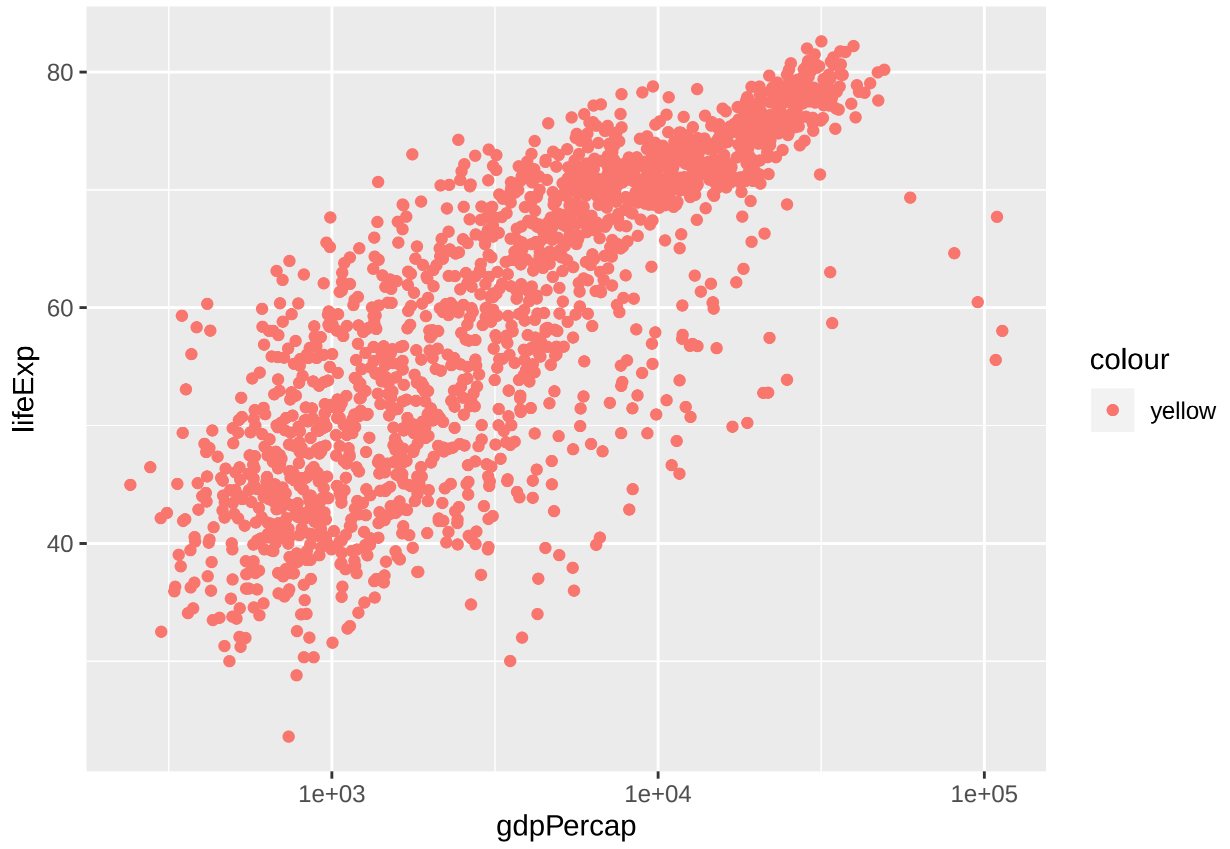
1. Tell the `ggplot()` function what our data is.
2. Tell `ggplot()` *what* relationships we want to see. For convenience we will put the results of the first two steps in an object called `p`.
3. Tell `ggplot` *how* we want to see the relationships in our data.
4. Layer on geoms as needed, by adding 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. 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* .

Answer: An aesthetic is basically the mapping of variables in our data to properties we can see on graphs. Technically, it defines the basic structure of the graphic. Some of the properties we might want to set, like color or size, have the same name as mappable elements. These, however, do not involve mapping variables to aesthetic elements therefore such arguments will never go inside the aes() function.

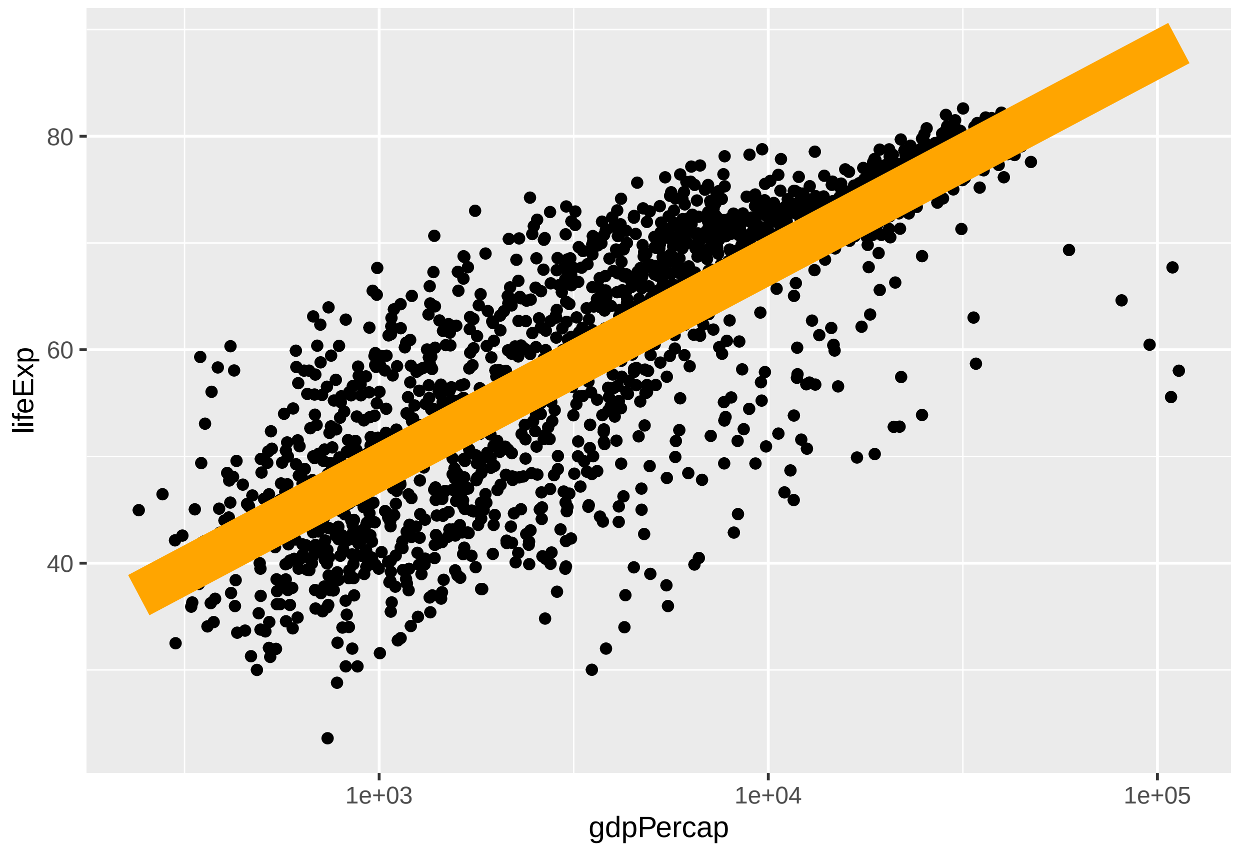
Exercise: Write the above sentence for the original call `aes(x = gdpPercap, y = lifeExp, color = 'yellow')`.

Answer: The property *x* in the graph is determined by data in *gdpPercap*, the property *y* in the graph is determined by data in *lifeExp*. For the last one it would be: the property *color* in the graph is determined by the data in *yellow*, however, this is usually meaningless unless there is actual data in 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()
```



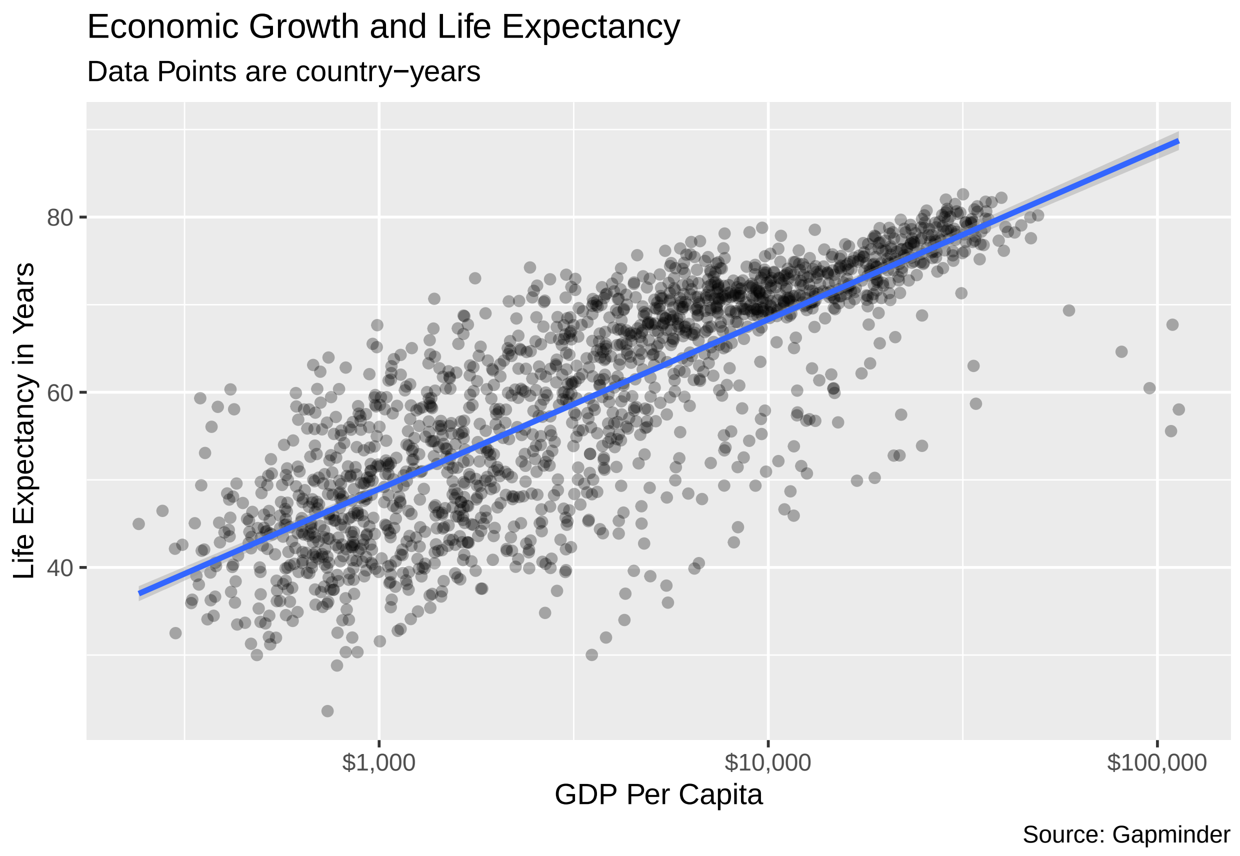
Exercise: Write down what all those arguments in `geom_smooth(...)` do.

Answer:

- mapping: Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
- data: The data to be displayed in this layer. There are three options: NULL, data.frame or other object, function.
- position: Position adjustment, either as a string, or the result of a call to a position adjustment function.
- . . . : Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
- method: Smoothing method (function) to use, accepts either a character vector, e.g. "auto", "lm", "glm", "gam", "loess" or a function, e.g. MASS::rlm or mgcv::gam, stats::lm, or stats::loess.
- formula: Formula to use in smoothing function, eg. y ~ x, y ~ poly(x, 2), y ~ log(x)
- se: Display confidence interval around smooth? (TRUE by default, see level to control.)
- na.rm: If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

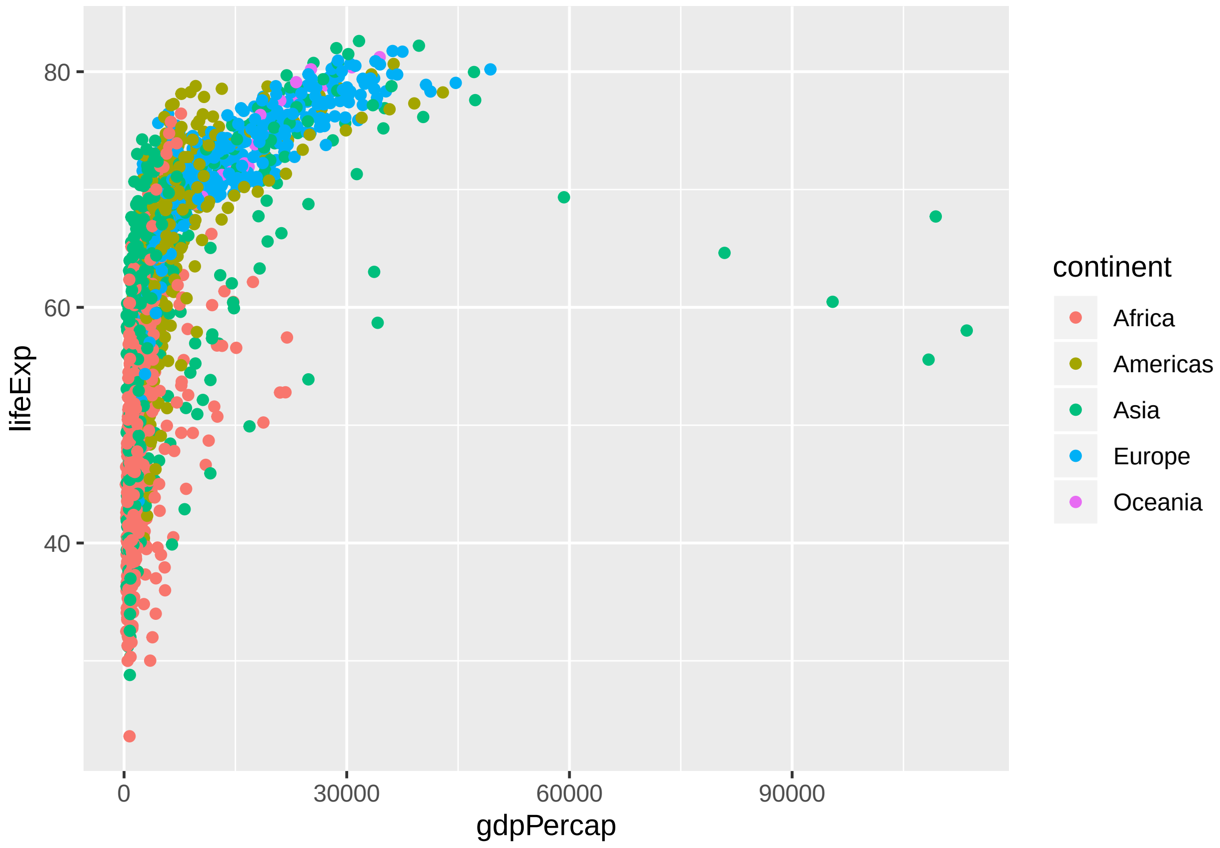
- show.legend: logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
- inherit.aes: If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
- geom, stat: Use to override the default connection between geom_smooth() and stat_smooth().
- n: Number of points at which to evaluate smoother.
- span: Controls the amount of smoothing for the default loess smoother. Smaller numbers produce wigglier lines, larger numbers produce smoother lines.
- fullrange: Should the fit span the full range of the plot, or just the data?
- level: Level of confidence interval to use (0.95 by default).
- method.args: List of additional arguments passed on to the modelling function defined by method.

```r
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")
```

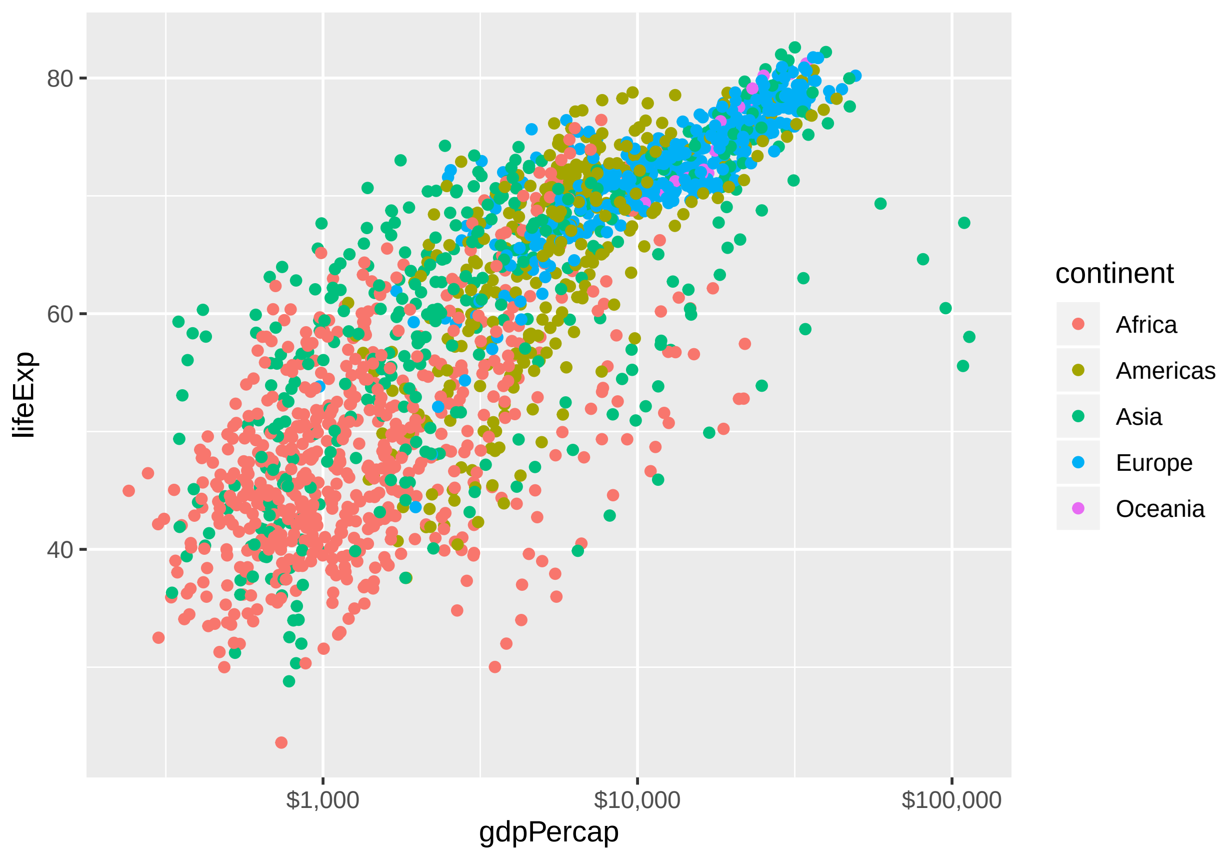


Coloring by continent:

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

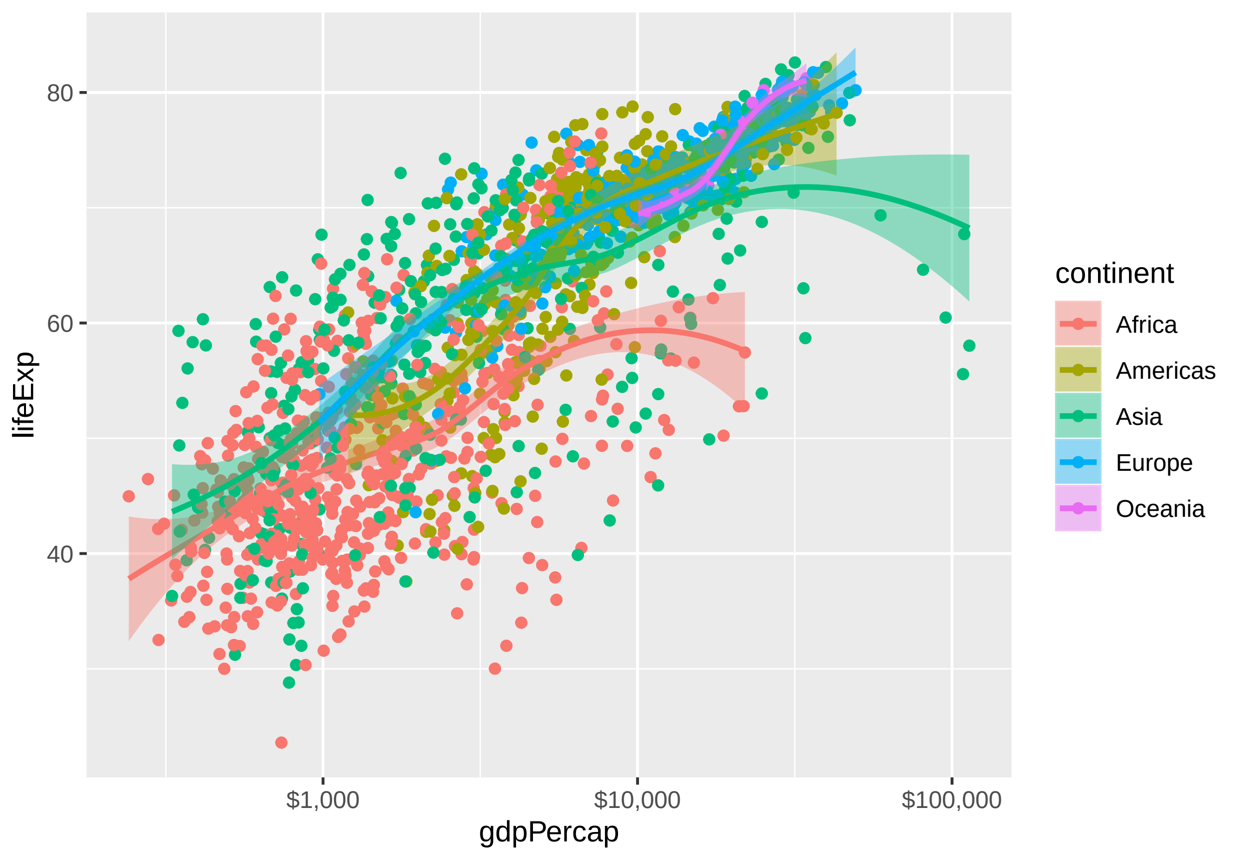


```r
p + geom_point() + scale_x_log10(labels = dollar)
```



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

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

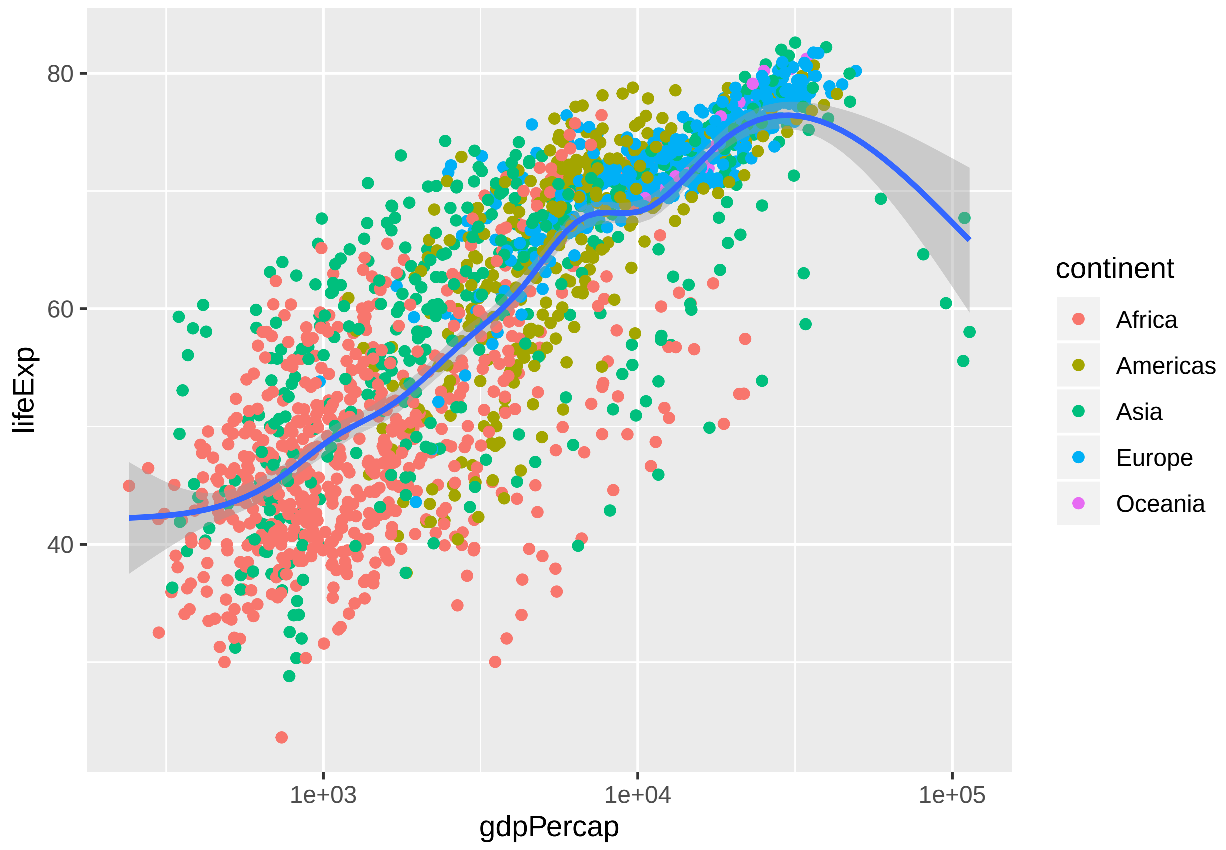


Exercise: What does `fill = continent` do? What do you think about the match of colors between lines and error bands?

Answer: The property color=continent displays the data points corresponding to countries from the same continent with the same colour. Multiple lines are created to fit to the data by continent and the property *fill* also sets the color of the confidence interval for each line based on continents so these also match the colours used for each continent.

```
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?

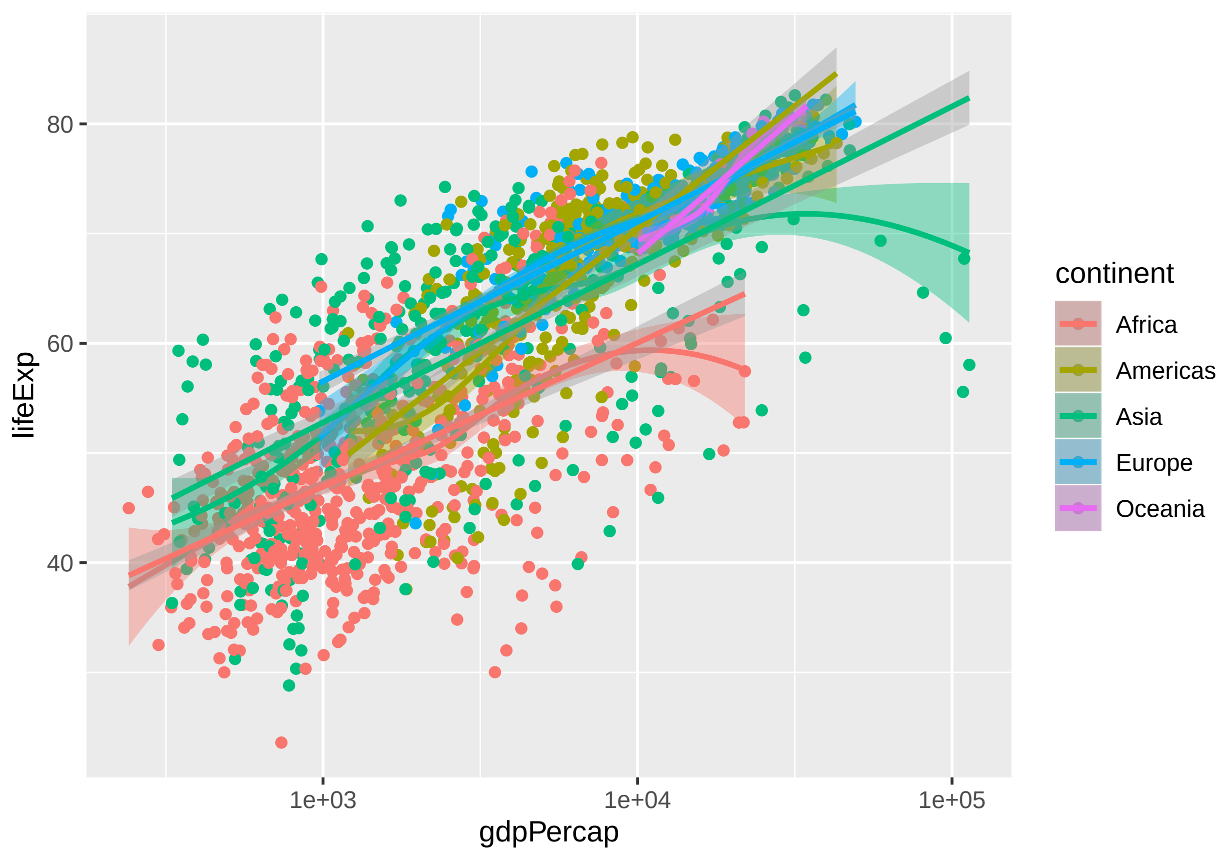
Answer: Because we specified *color=continent* in geom_point() not in ggplot().

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.

Answer: mapping = aes(color = continent) is used in both geom_smooth() functions therefore smoothing is done twice based on continents. In the first geom_smooth() where the fill property is also specified, the confidence intervals are also colourful. In case of the second geom_smooth() fill is not specified so the error bands belonging to these lines are grey. To correct the plot, both color and fill should be specified in the aes() in ggplot().

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

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



Additional (not Optional) Exercises

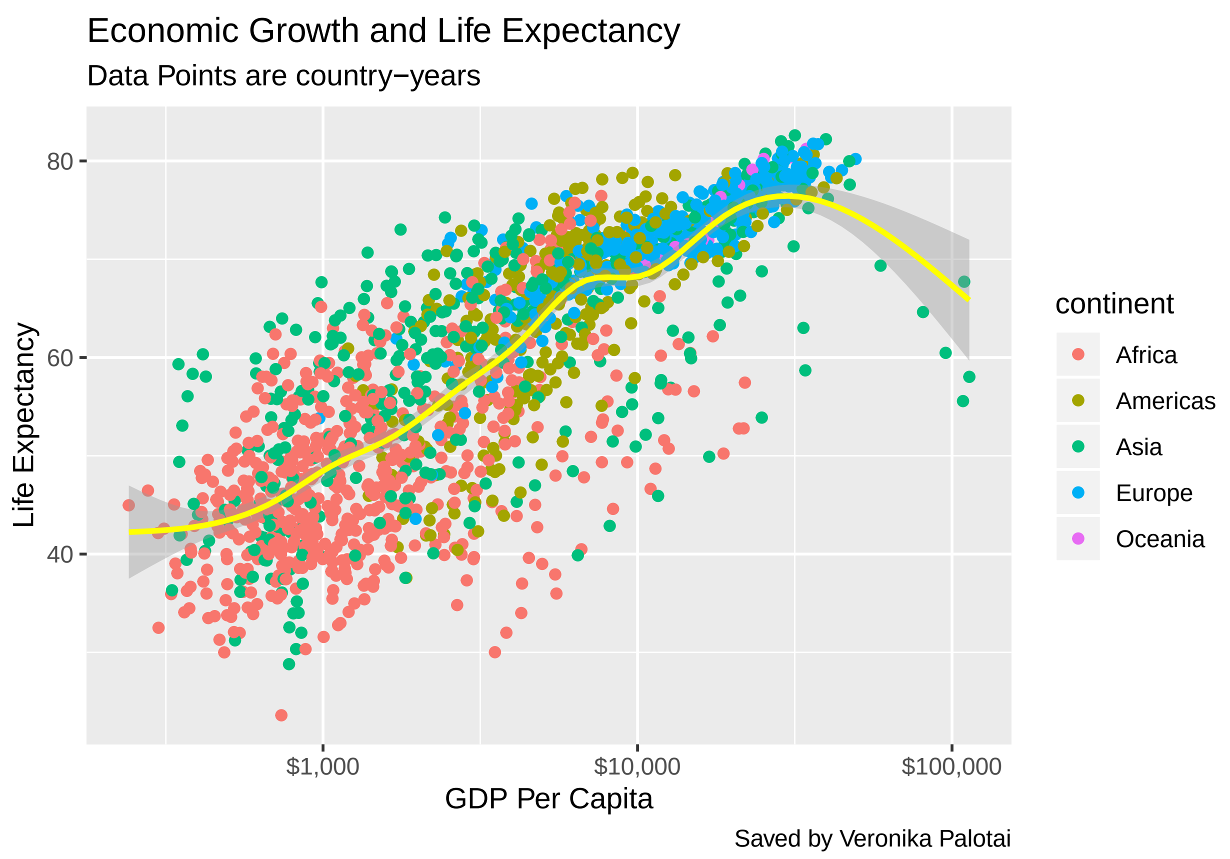
Exercise (Discourse): Find ways to save the figures that you made so that you can use them elsewhere too. Create a new folder to save only images. Use the command for saving to save the picture for the last image in your new folder, after you have updated the axes, title, subtitle, and caption of the 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.

```
getwd()
```

```
## [1] "D:/Egyetem/CEU/Coding_1/R-Coding/lecture2"
```

```
dir.create("./images")

p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp))
p + geom_point(mapping = aes(color = continent)) +
  geom_smooth(color = 'yellow') + scale_x_log10(labels = scales::dollar) +
  labs(x = "GDP Per Capita",
       y = "Life Expectancy",
       title = "Economic Growth and Life Expectancy",
       subtitle = "Data Points are country-years",
       caption = "Saved by Veronika Palotai")
```

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



```r
ggsave("./images/econ_growth-life_exp.png",plot = last_plot())
```

```
## Saving 6.5 x 4.5 in image
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

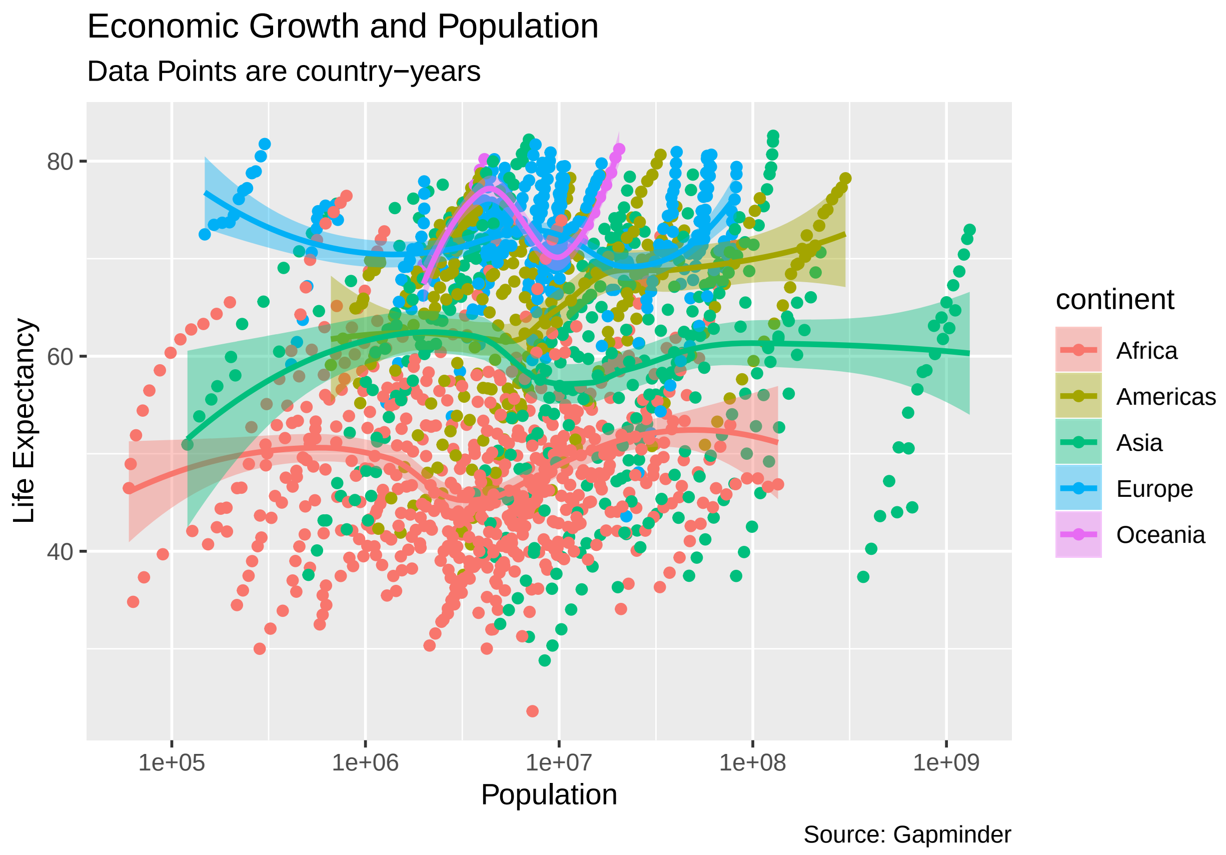
Exercise: Read section 3.8 “Where to go next” from DV. Based on those ideas, experiment and create two different graphs with the gapminder data. Describe each briefly in one sentence.

Answer:

Plotting Life Expectancy against Population rather than GDP per capita:

```r
p <- ggplot(data = gapminder,
            mapping = aes(x = pop, y = lifeExp, color = continent, fill = continent))
p + geom_point() +
  geom_smooth() + scale_x_log10() +
  labs(x = "Population",
       y = "Life Expectancy",
       title = "Economic Growth and Population",
       subtitle = "Data Points are country-years",
       caption = "Source: Gapminder")
```

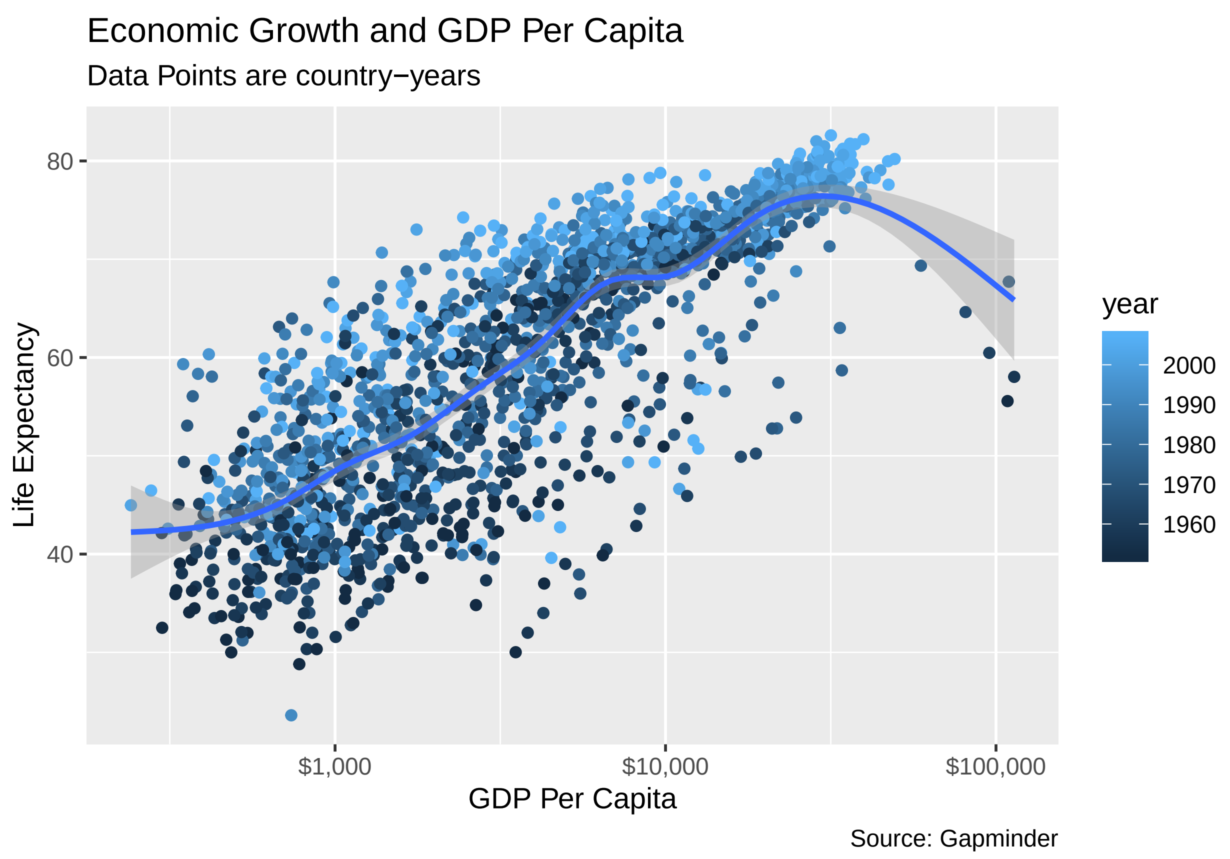
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Mapping color to year instead of continent:

```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp, color = year))
p + geom_point() + geom_smooth() + scale_x_log10(labels = scales::dollar) +
  labs(x = "GDP Per Capita",
       y = "Life Expectancy",
       title = "Economic Growth and GDP Per Capita",
       subtitle = "Data Points are country-years",
       caption = "Source: Gapminder")
```

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



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.

Answer:

```
#packages
None

#data
None

#code
help(class())

#error
Error in help(class()) : ‘topic’ should be a name, length-one character vector or reserved word

Reason: I should have put the name of class() function, ‘class’ as an argument of help(), not the actua
```

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.

Answer:

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

An empty graph.

How many rows are in mpg? How many columns?

```r
str(mpg)
```

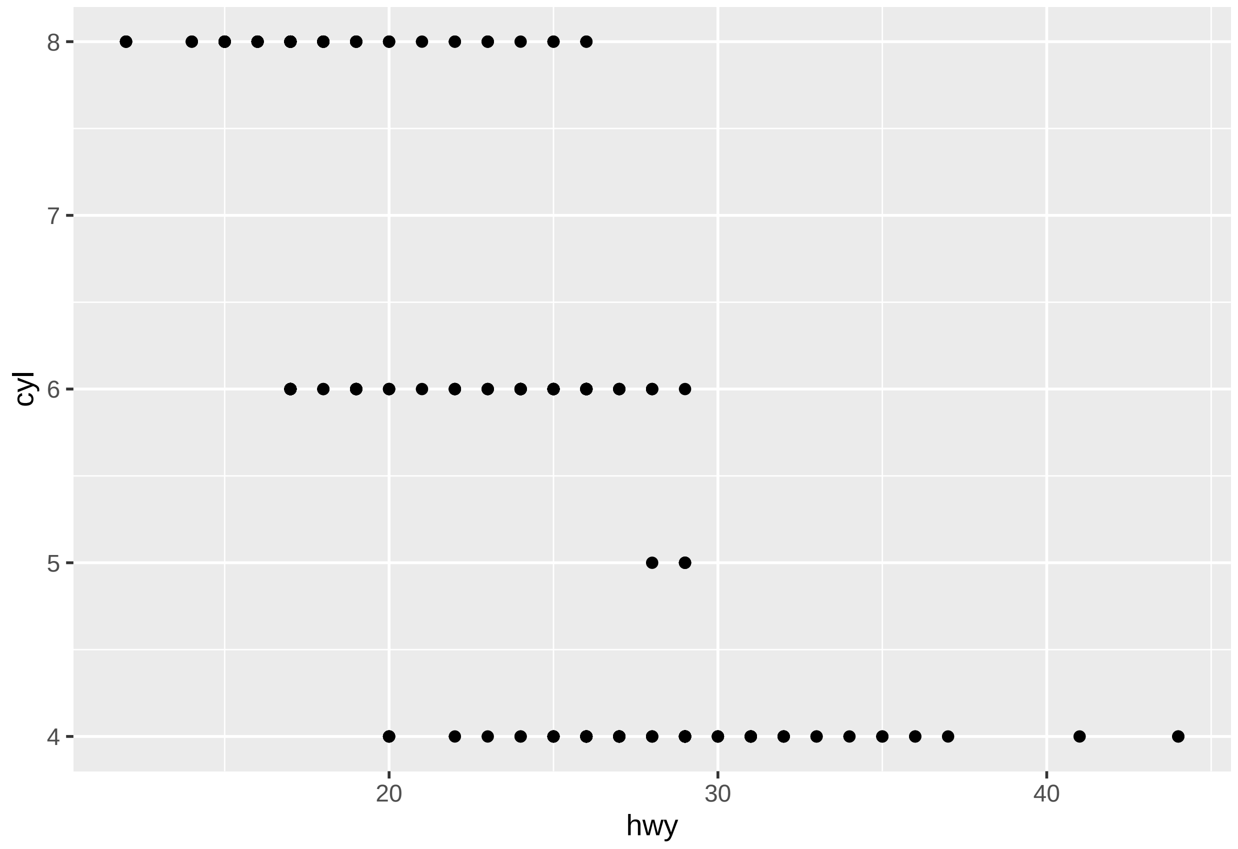
It has 234 rows and 11 columns.

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

Help says: f = front-wheel drive, r = rear wheel drive, 4 = 4wd. So it determines the type of wheel drive.

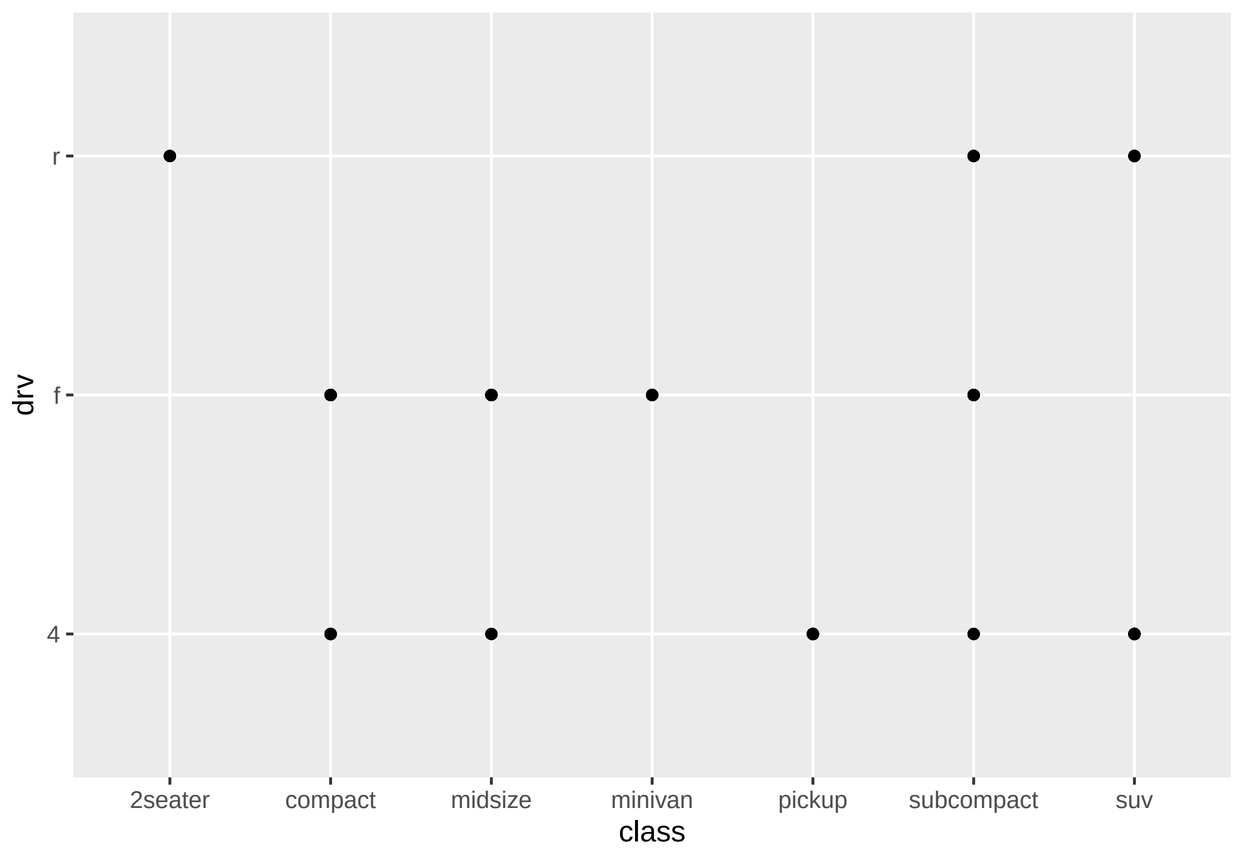
Make a scatterplot of hwy vs cyl.

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



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

```r
ggplot(data = mpg) +
  geom_point(mapping = aes(x = class, y = drv))
```



A scatterplot is not the appropriate way of visualizing this type of data. This way it is difficult to read.

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.

Answer:

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

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

Which variables in mpg are categorical? Which variables are continuous? (Hint: type ?mpg to read the documentation for the dataset). How can you see this information when you run mpg?

- Categorical: model, trans, drv, fl, class
- Continuous: displ, cty, hwy, year, cyl

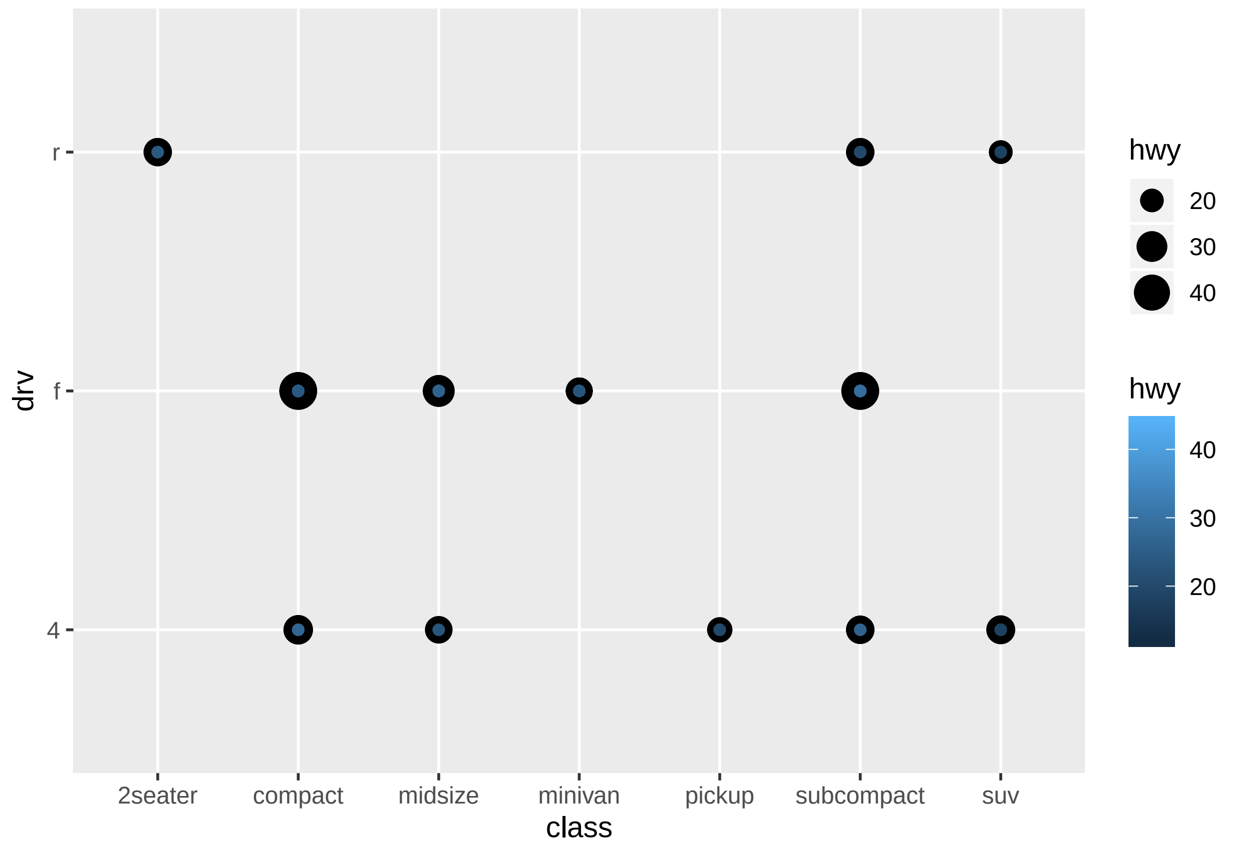
Map a continuous variable to color, size, and shape. How do these aesthetics behave differently for categorical vs. continuous variables?

A continuous variable can't be mapped to *shape*, *size* maps the variable to the area of a circle-shaped mark and *color* maps a continuous variable to the saturation of the colour blue.

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

It is possible to map the same variable to multiple aesthetics. For instance, using both *size* and *color* for the same variable means that values are distinguished based on both size and colour.

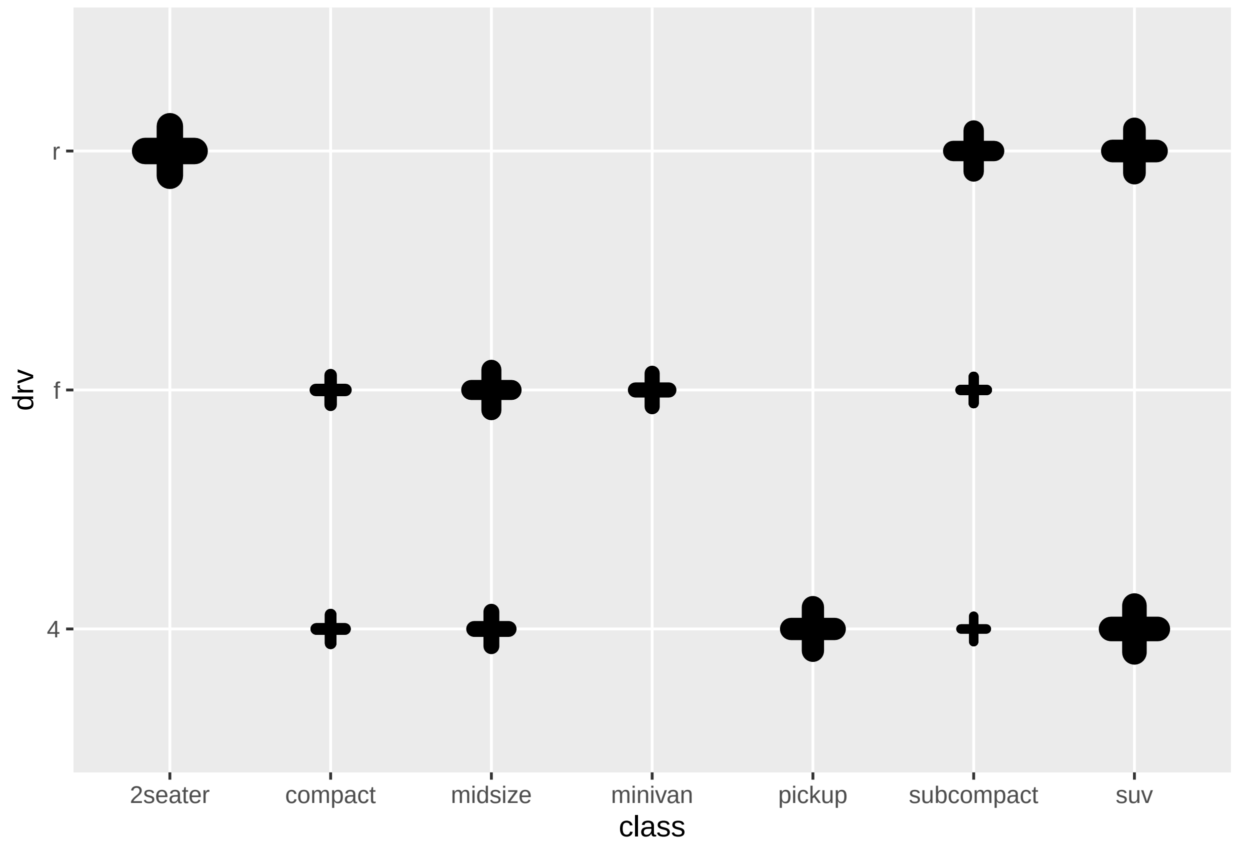
```r
ggplot(data = mpg) + geom_point(mapping = aes(x = class, y = drv, size = hwy)) +
  geom_point(mapping = aes(x = class, y = drv, color = hwy))
```



What does the `stroke` aesthetic do? What shapes does it work with?

Stroke controls the width of the border in case of those shapes that only have one.

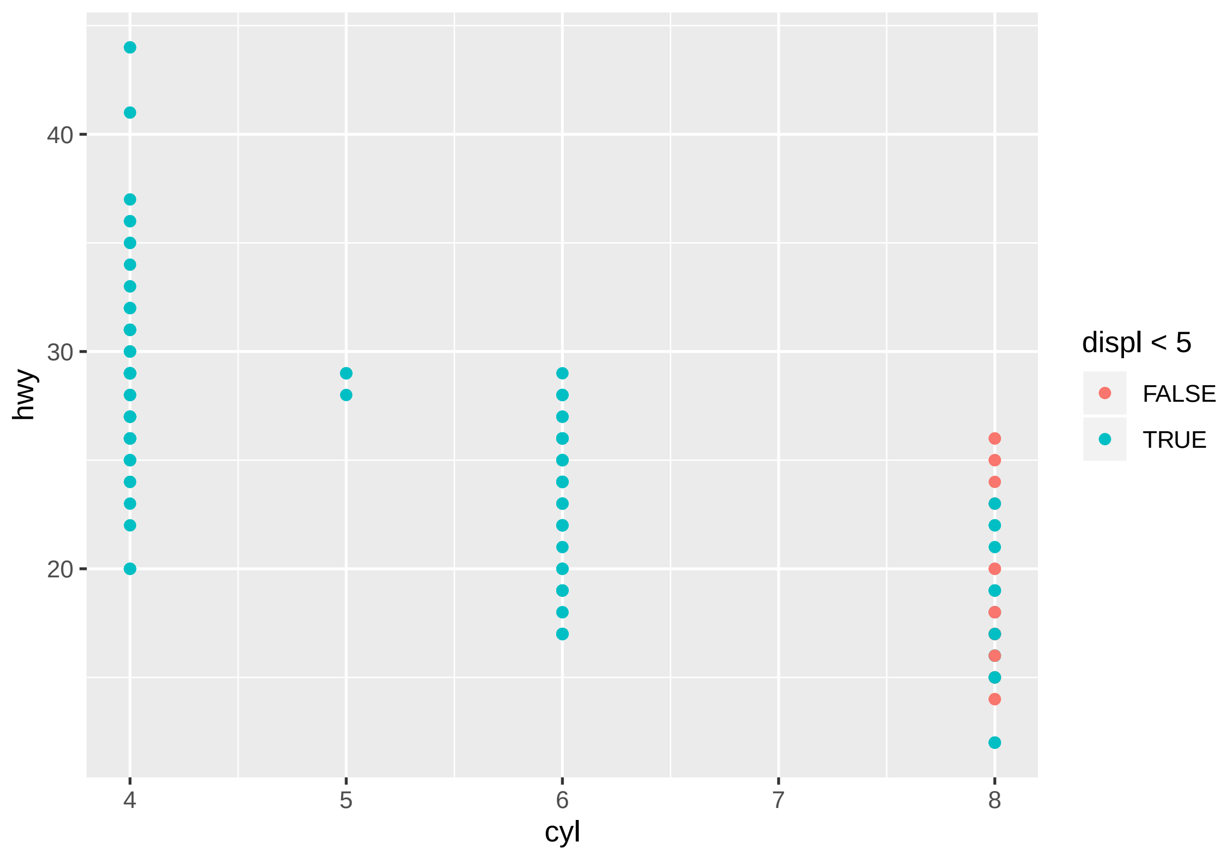
```r
ggplot(data = mpg) + geom_point(mapping = aes(x = class, y = drv, stroke = displ), shape = 3)
```



What happens if you set an aesthetic to something other than a variable name, like `displ < 5`?

The expression will be evaluated and the result will be plotted. In this case where displ < 5 is TRUE, the corresponding data points are coloured blue where it's FALSE they are coloured red.

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



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

Answer:

Why does this code not work?

```
my_variable <- 10
my_varıable
```

It's not working because the letter *i* in the second declaration of the variable is not the latin *i* used in the first one but some special character without the dot. As the two declarations are not the same, the first one has 10 as its value but the second one returns `not found` since it has never been used before.

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

Answer: I could not figure this one out. Most of my googling results explain how to customize `scale` but no results on knitting.

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.