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 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     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. geom_ **Exercise:** What other types of plots are there? Try to find several more `geom_` functions.

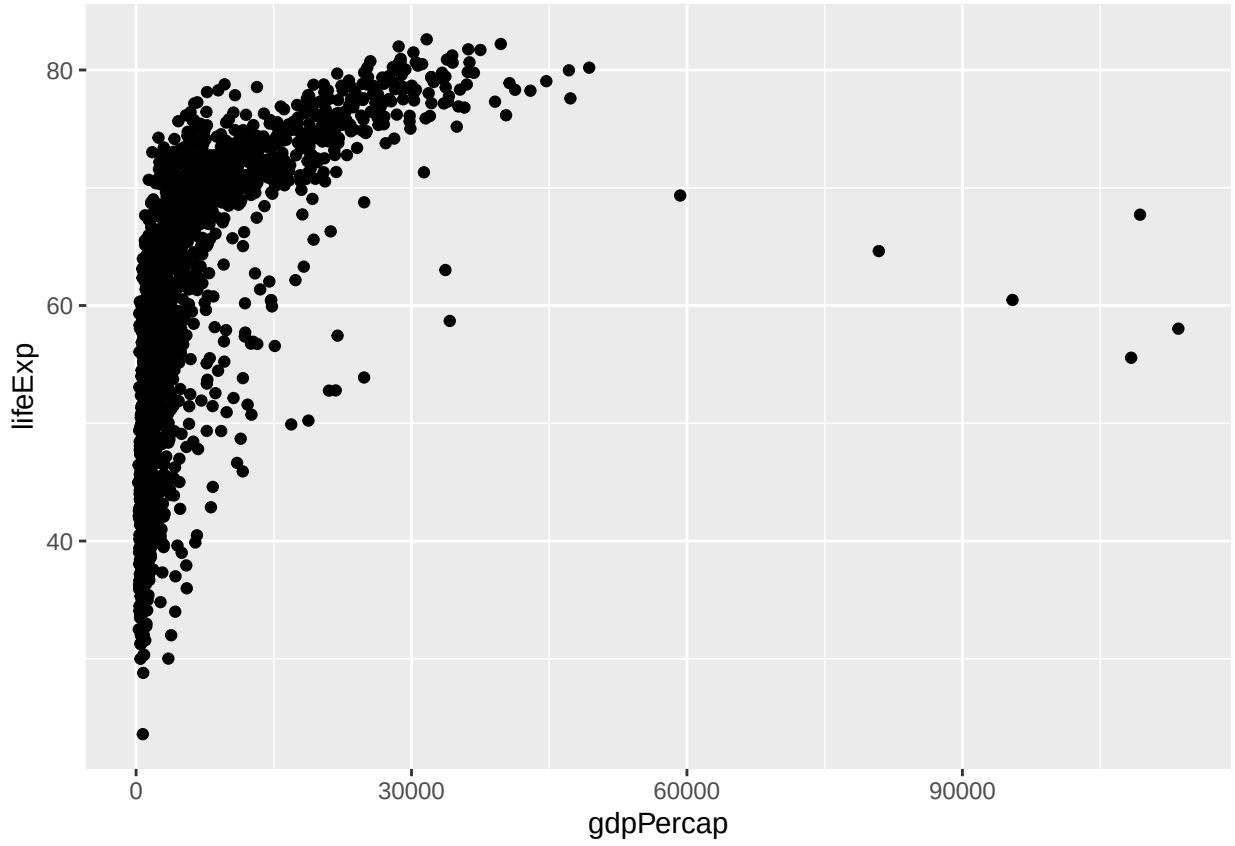
Answer: There are several types such as geom_density(), geom_bar(), geom_map(). Full list to be found in the official documentation of the package: https://ggplot2.tidyverse.org/reference/

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.

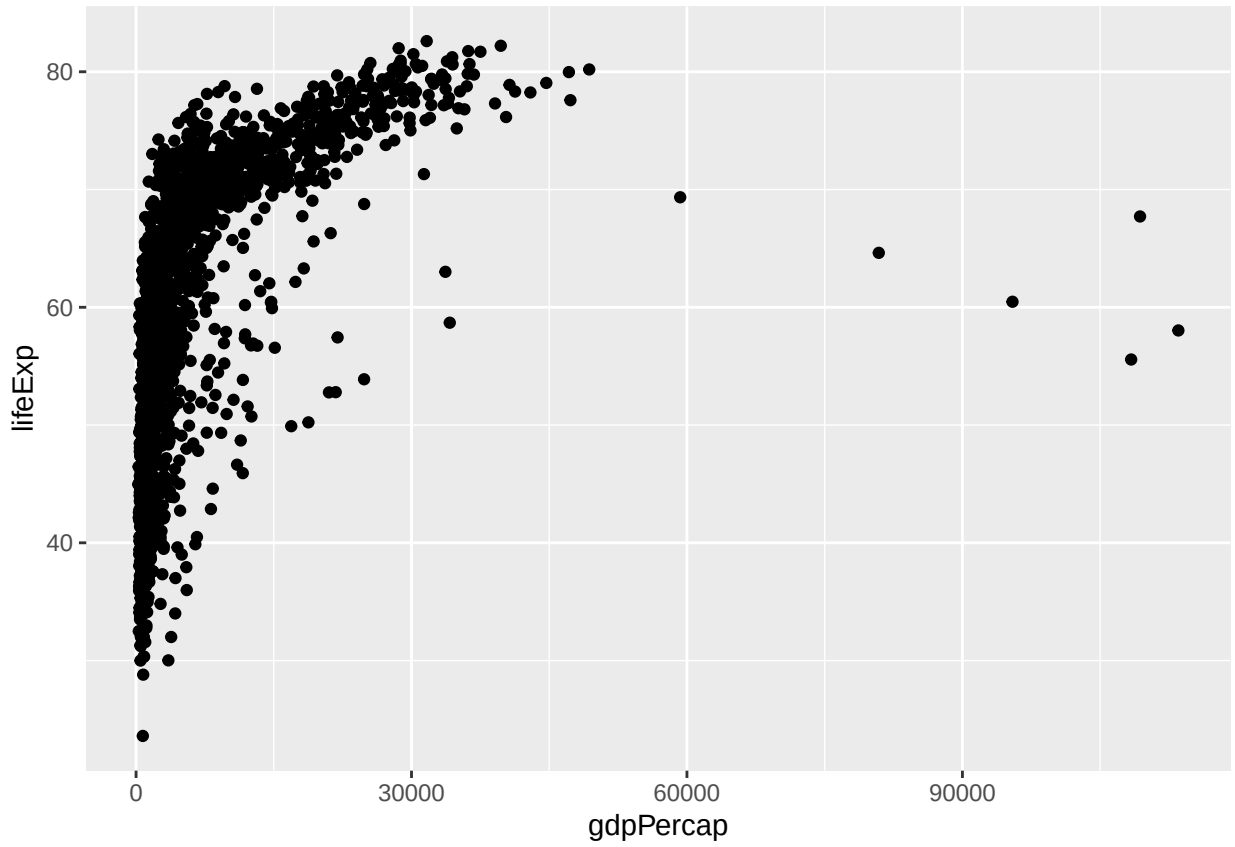
```
str(p)
str(p + geom_point())
```

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

Answer: Sure.

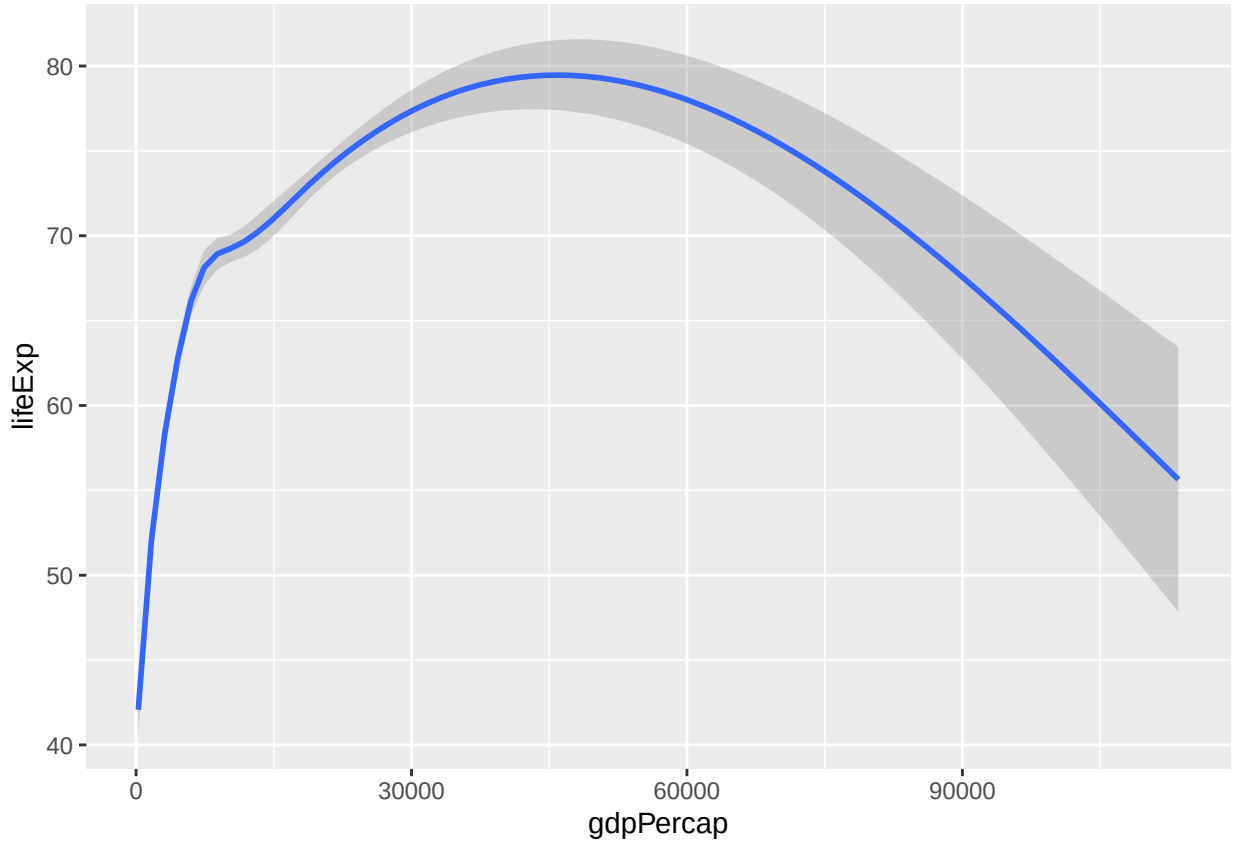
Finally, we add a *layer*. This says how some data gets turned into concrete visual aspects.

```
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.

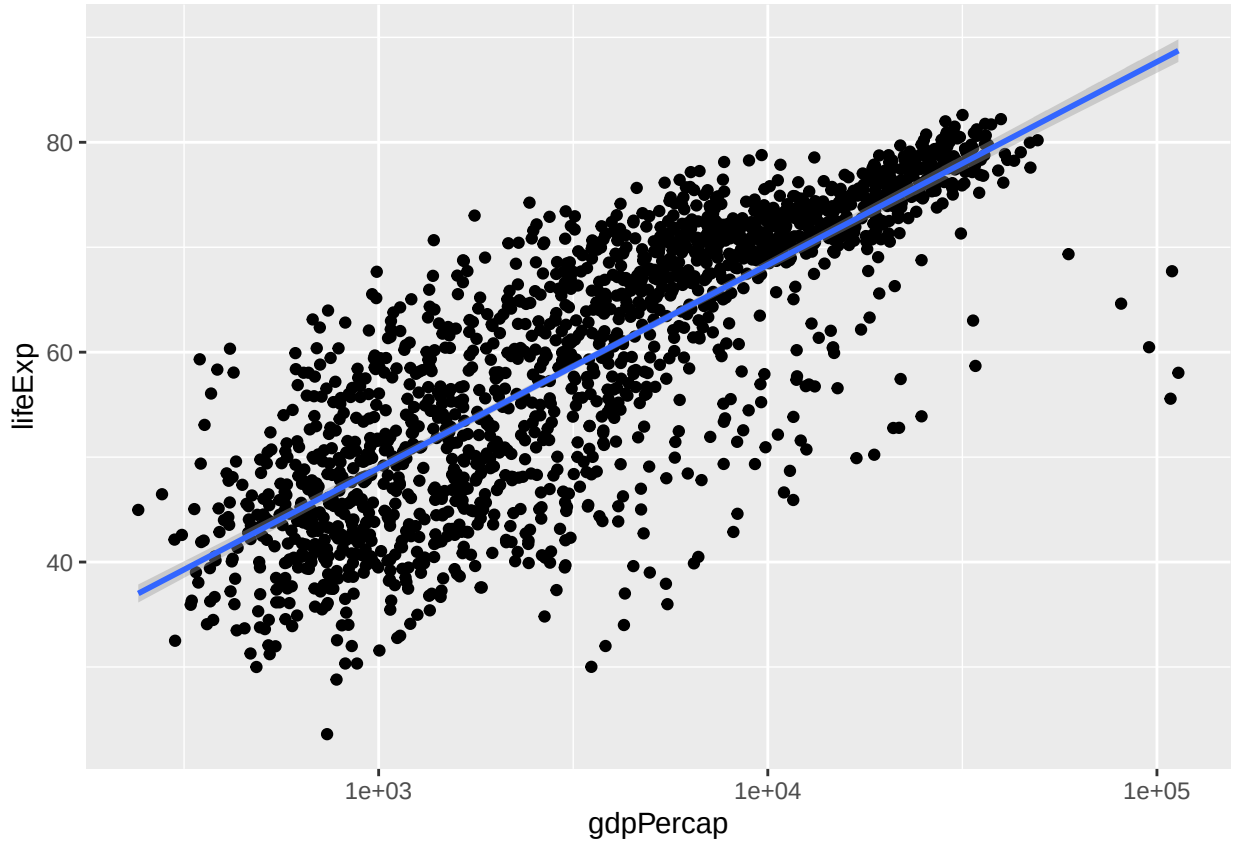
We get a message that tells us that `geom_smooth()` is using the method = 'gam', so presumably we can use other methods. Let's see if we can figure out which other methods there are.

```
?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: It transfroms the values of the x axis by taking the log of every single value. For example if values are 8 and 20 their difference is 12. Taking their logs and then their difference we only get log(20) - log(8) = 1.3 - 0.9 = 0.4. That is why it is more evenly distributed.

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

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

Answer: It labels the values on the selected axis as if they were money related values (dollars, to be precise).

```
?dollar()
```

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

Answer: Any other log function would do the work. Square root function is okay as well.

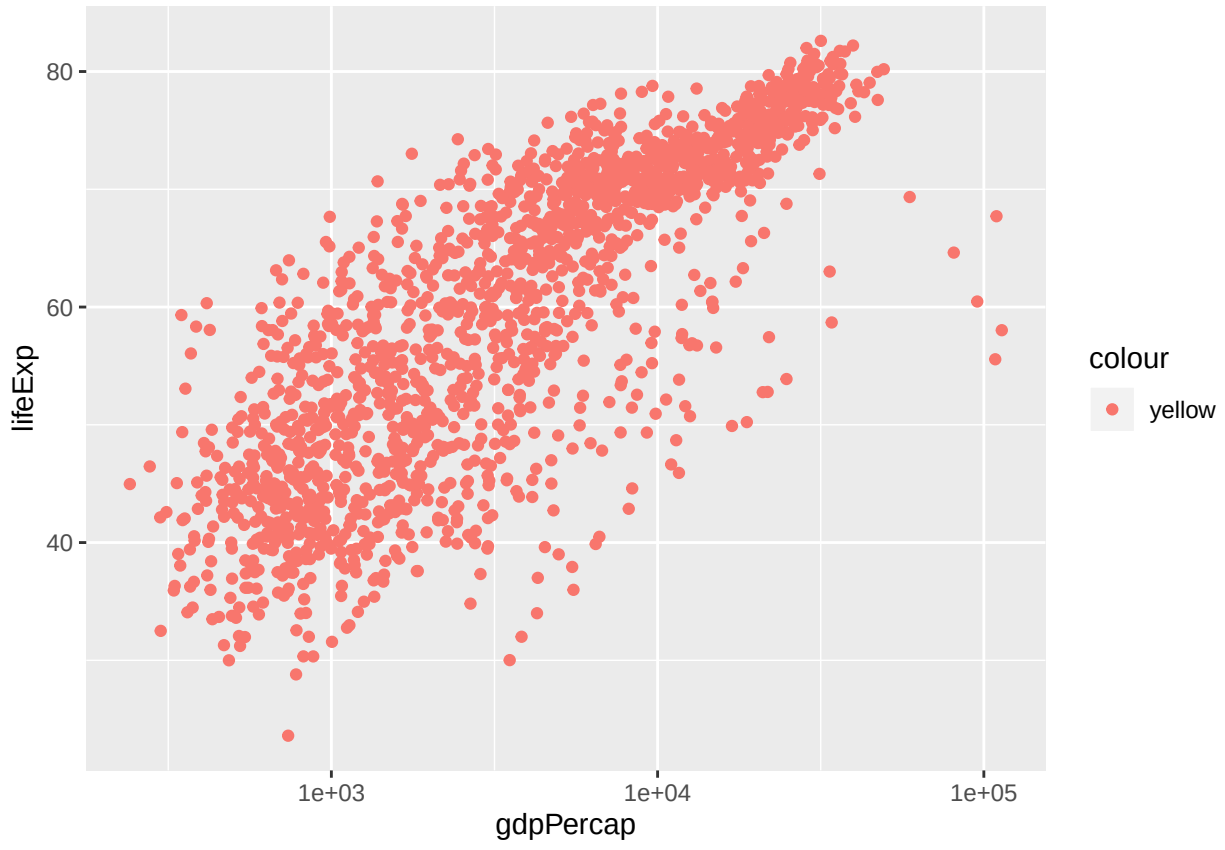
The Ggplot Recipe

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(...) + scale_x_log10()
```

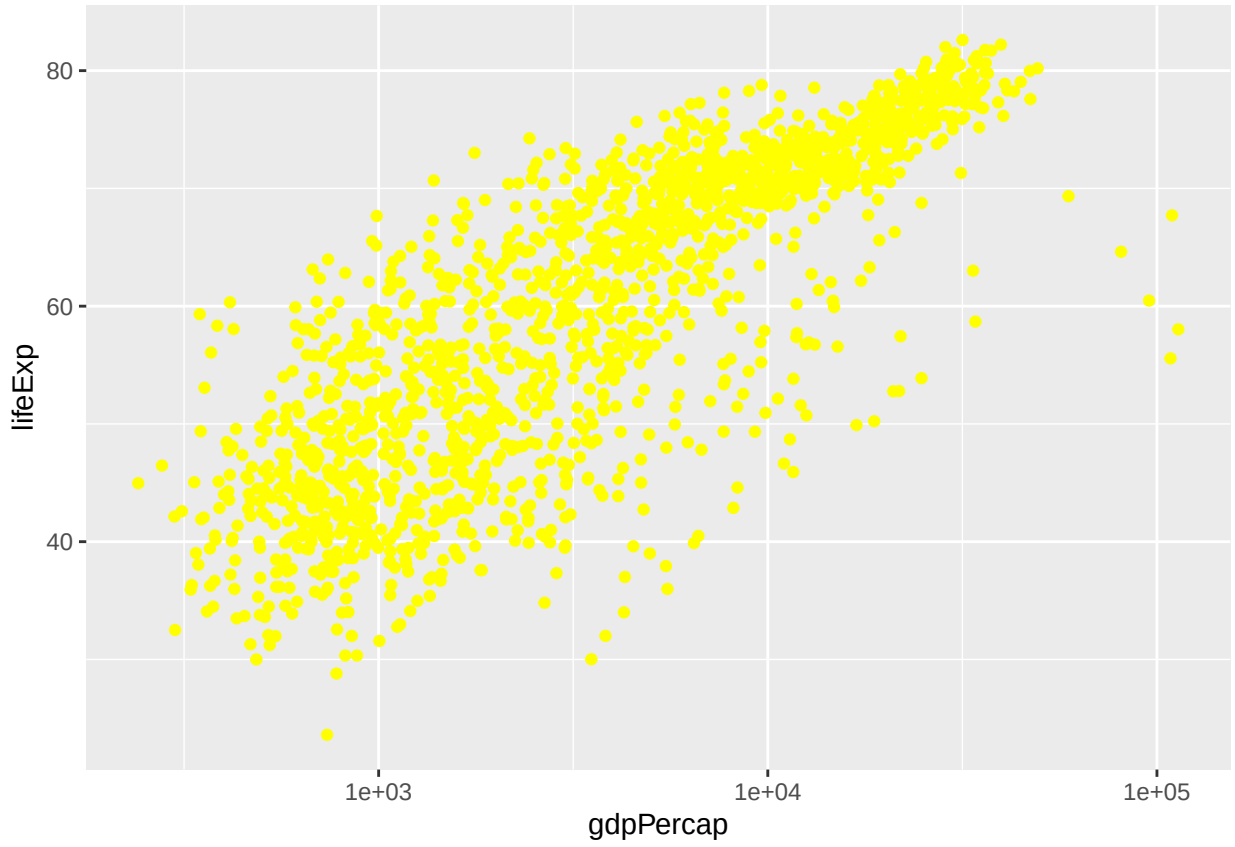
Exercise: Based on the discussion in Chapter 3 of *Data Visualization* (read it), describe in your words what is going on. One way to avoid such mistakes is to read arguments inside `aes(<property> = <variable>)`as *the property in the graph is determined by the data in* .

Answer: Colouring the dots to yellow is not related to the dataset, but related to the way you want to plot is. Accordingly, you have to set the colouring in the `geom_whaterver` as an argument.

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

Answer:

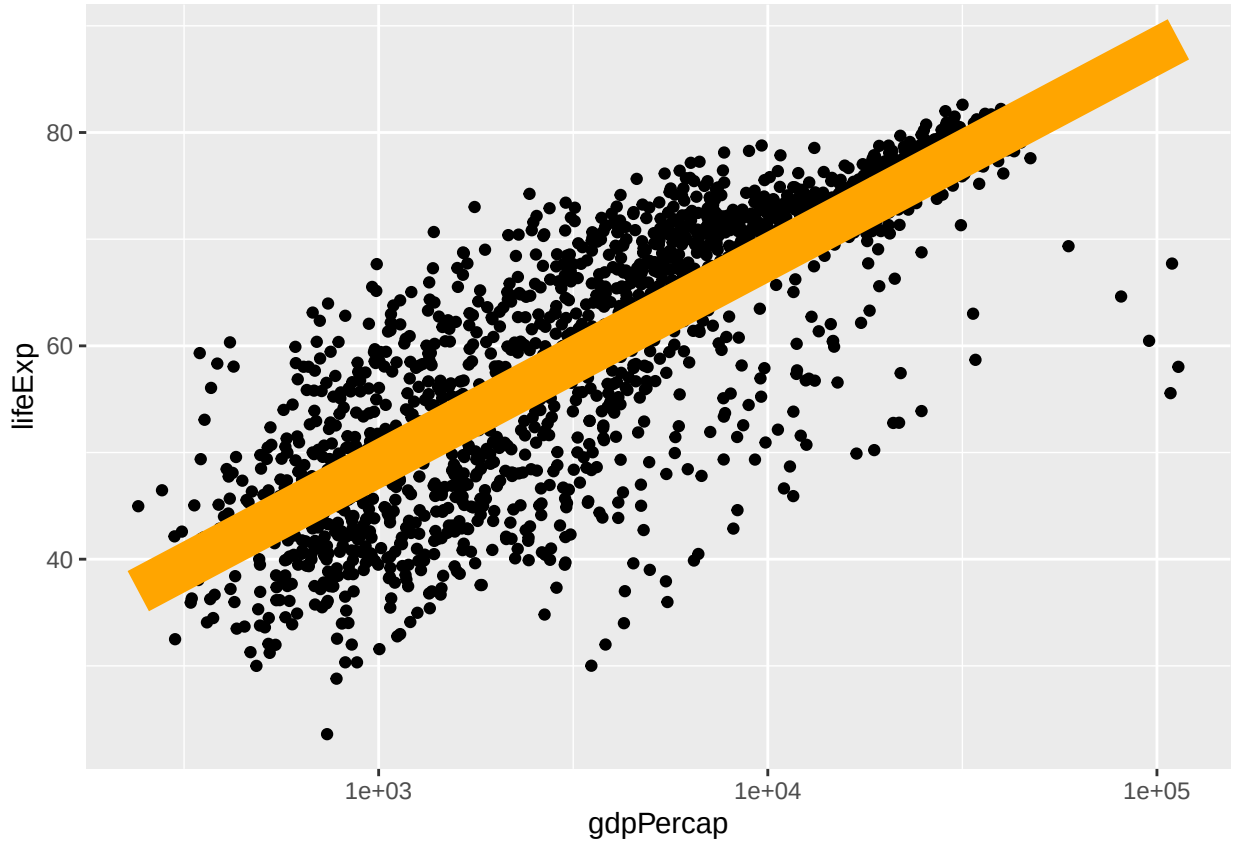
```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp))
p + geom_point(color = "yellow") + scale_x_log10()
```



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*.

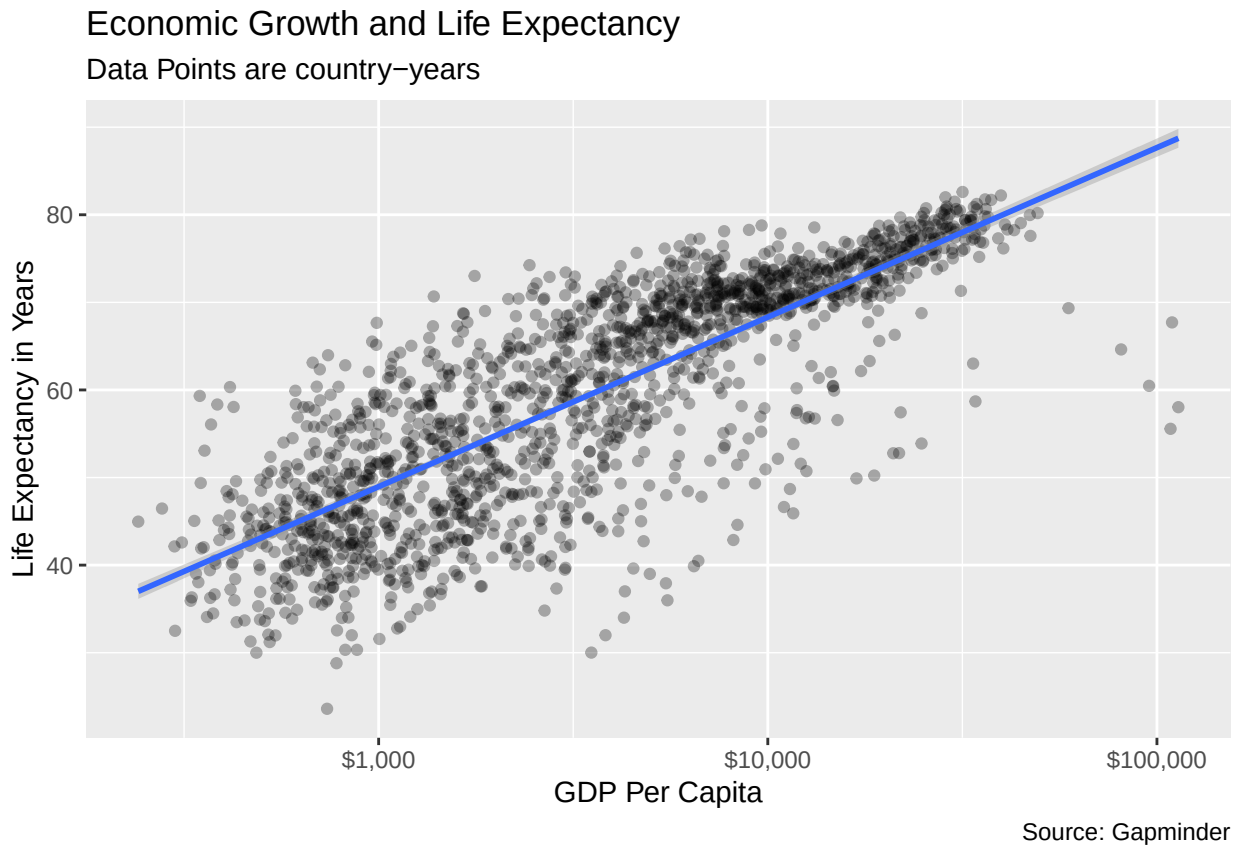
```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp))
p + geom_point() + geom_smooth(color = "orange", se = FALSE, size = 8, method = "lm") + scale_x_log10()
```



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

Answer: I think coloring goes without saying. It paints the smoothing line to orange. Size refers to the boldness of the line. SE refers to standard error, so whether a confidence intervall should be included or not. Method refers to the way the smoothing is done. LM probably stands for linear modelling.

```
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:

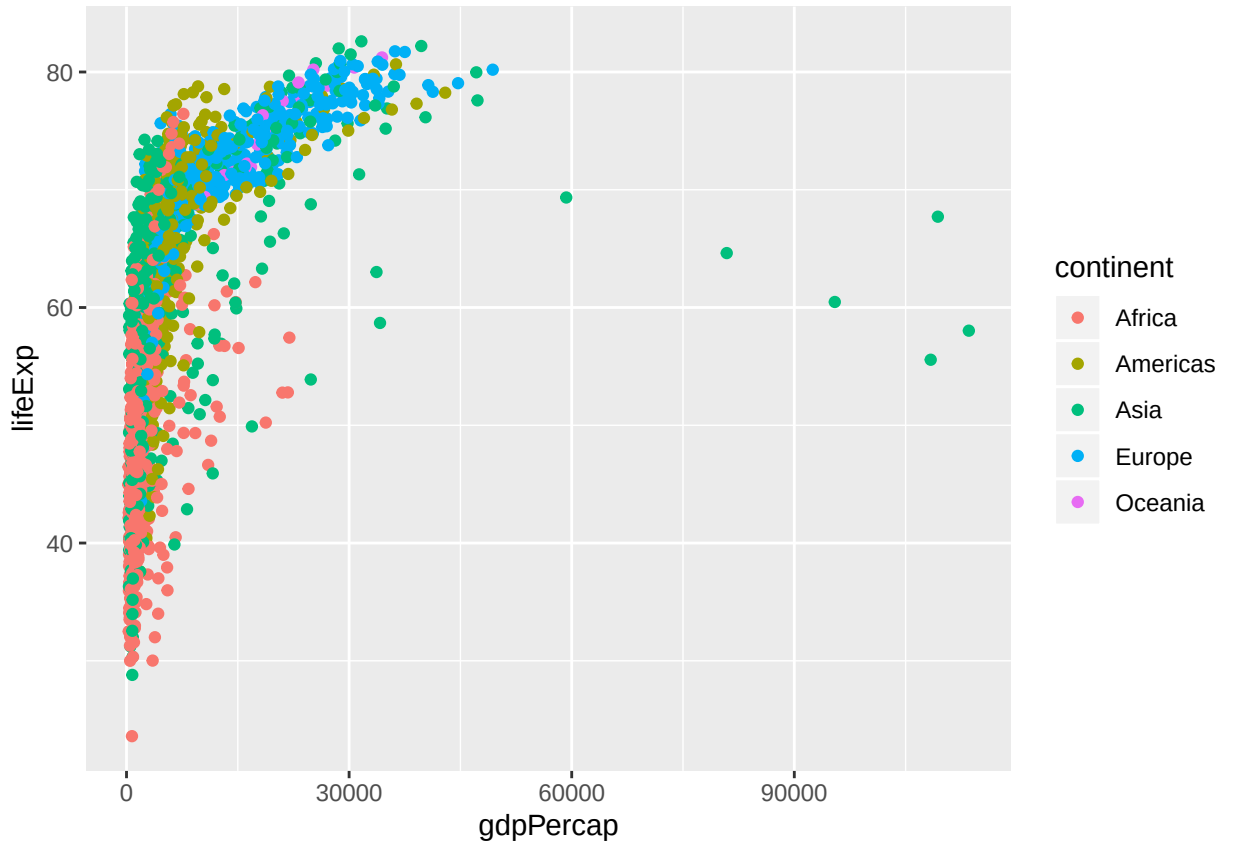
```
library(scales)
```

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

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

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

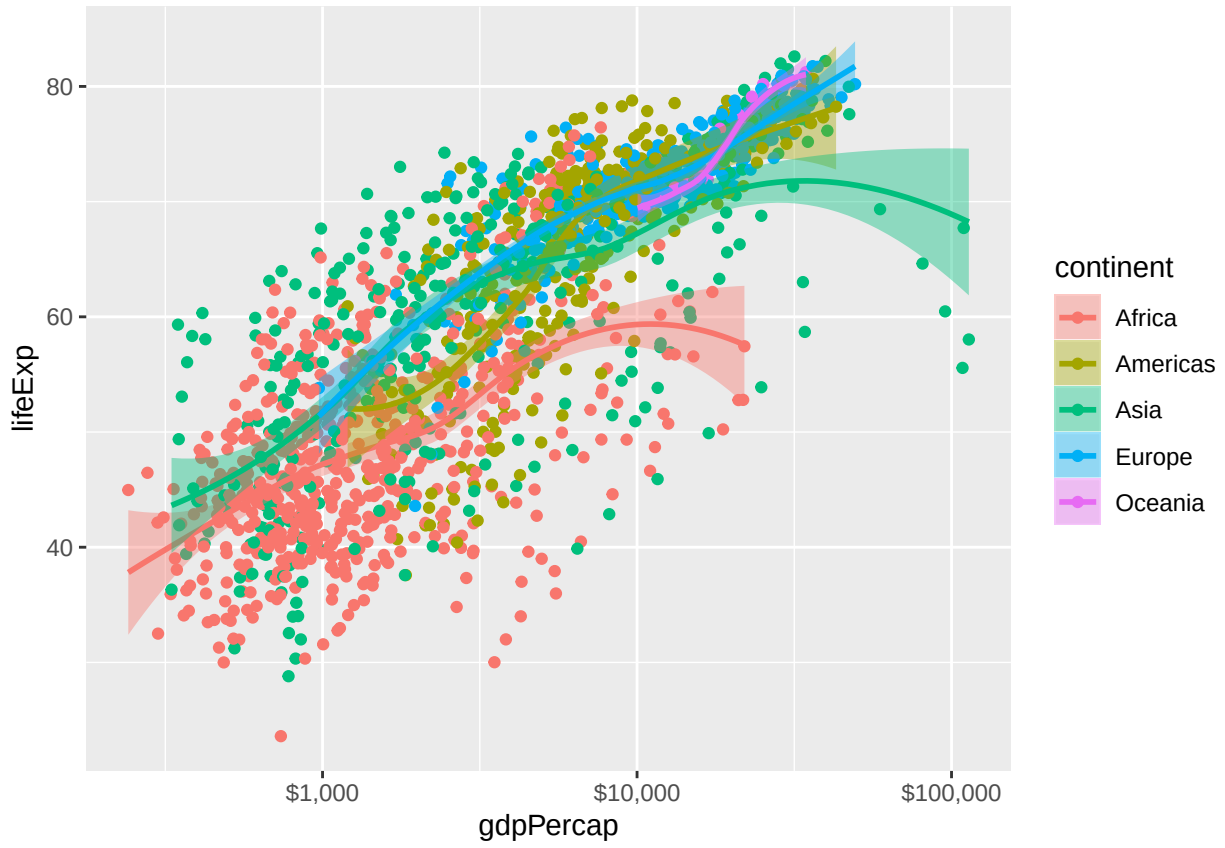
```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp, color = continent, fill = continent))
p + geom_point()
```




```
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?

Answer: By applying `fill = continent` the confidence interval of the smoothing line will have the same colour as the line and dots themselves.

```
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?

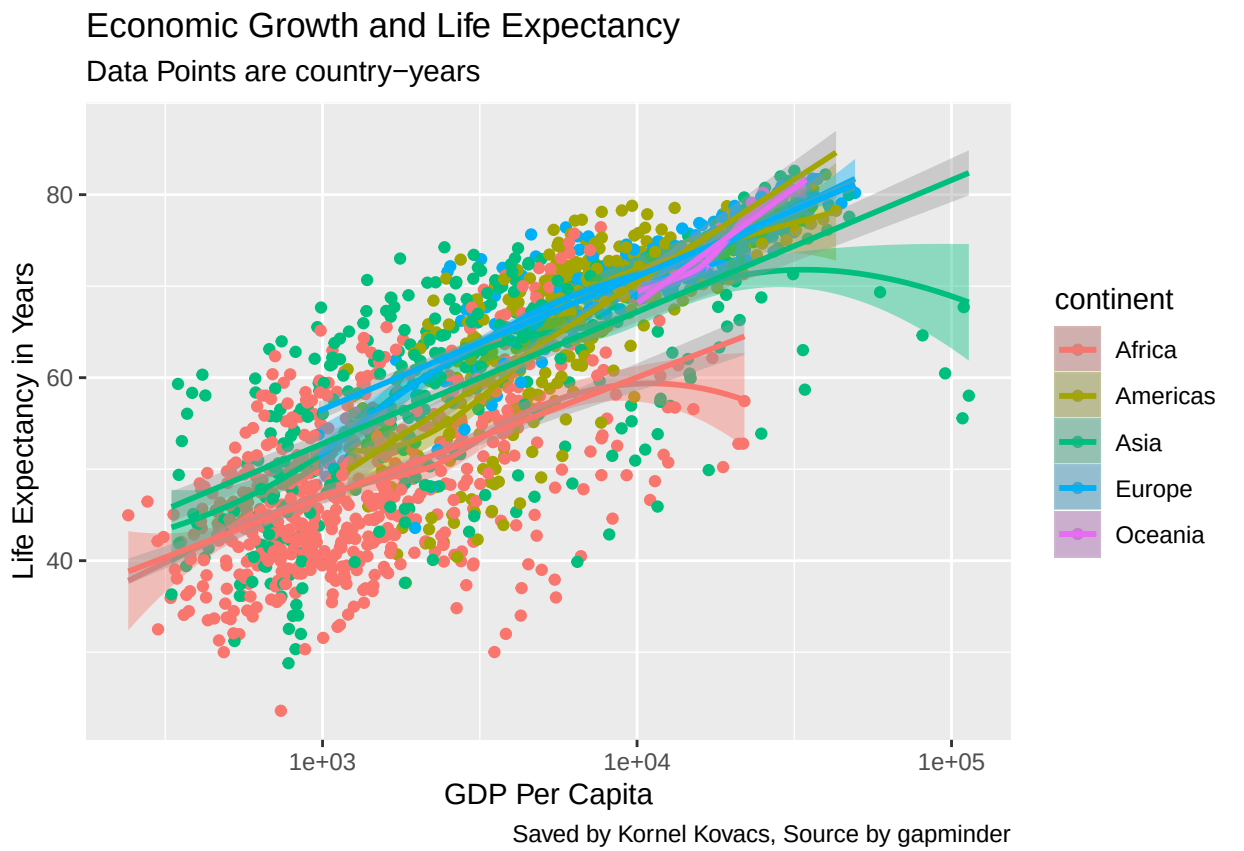
Answer: Because we only applied `aes(color = continent)` to `geom_point()` and did not apply it for `geom_smooth()`. The latter is responsible for the smoothing line(s).

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: We doubled `geom_smooth()` which is why there are two types of smoothing lines for each country. One with the default method and one with `gam`. If that is what you want, okay, go for it. But it is more likely that 2 different graphs would give you a better understanding of the data. In case you stick to this graph, do label which smoothing line (per country) is created by the `gam` method and which one belongs to the default smoothing method otherwise it cannot be determined.

```
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") +
  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 Kornel Kovacs, Source by gapminder")
```

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




```
ggsave("./data/growth.png", plot = last_plot())
```

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

Answer (Discourse): See on Discourse.

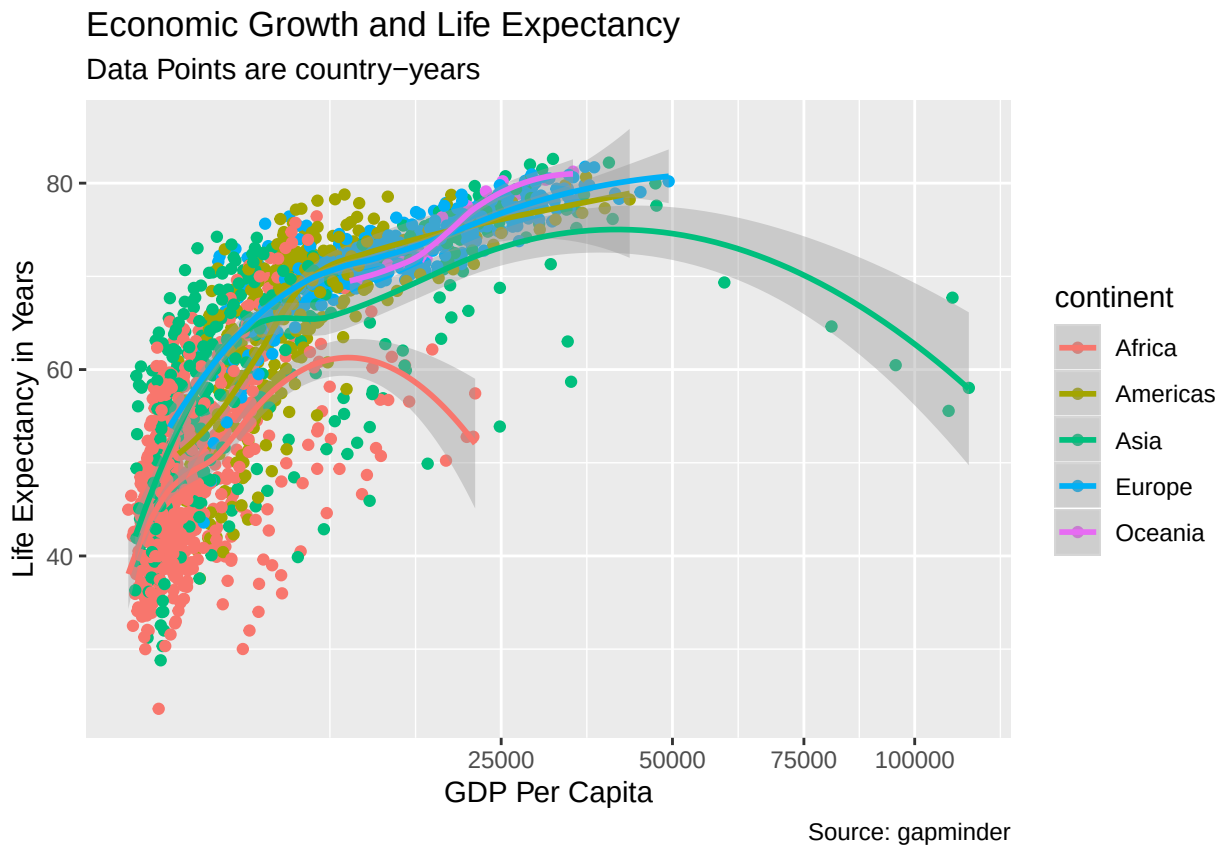
Exercise: Read section 3.8 "Where to go next" from DV. Based on those ideas, experiment and create two different graphs with the gapminder data. Describe each briefly in one sentence.

Answer : Now, instead of using log transformation, we use a square root transformation. It is more evenly distributed than the normal plot, but less then the log one.

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

```
  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")
```

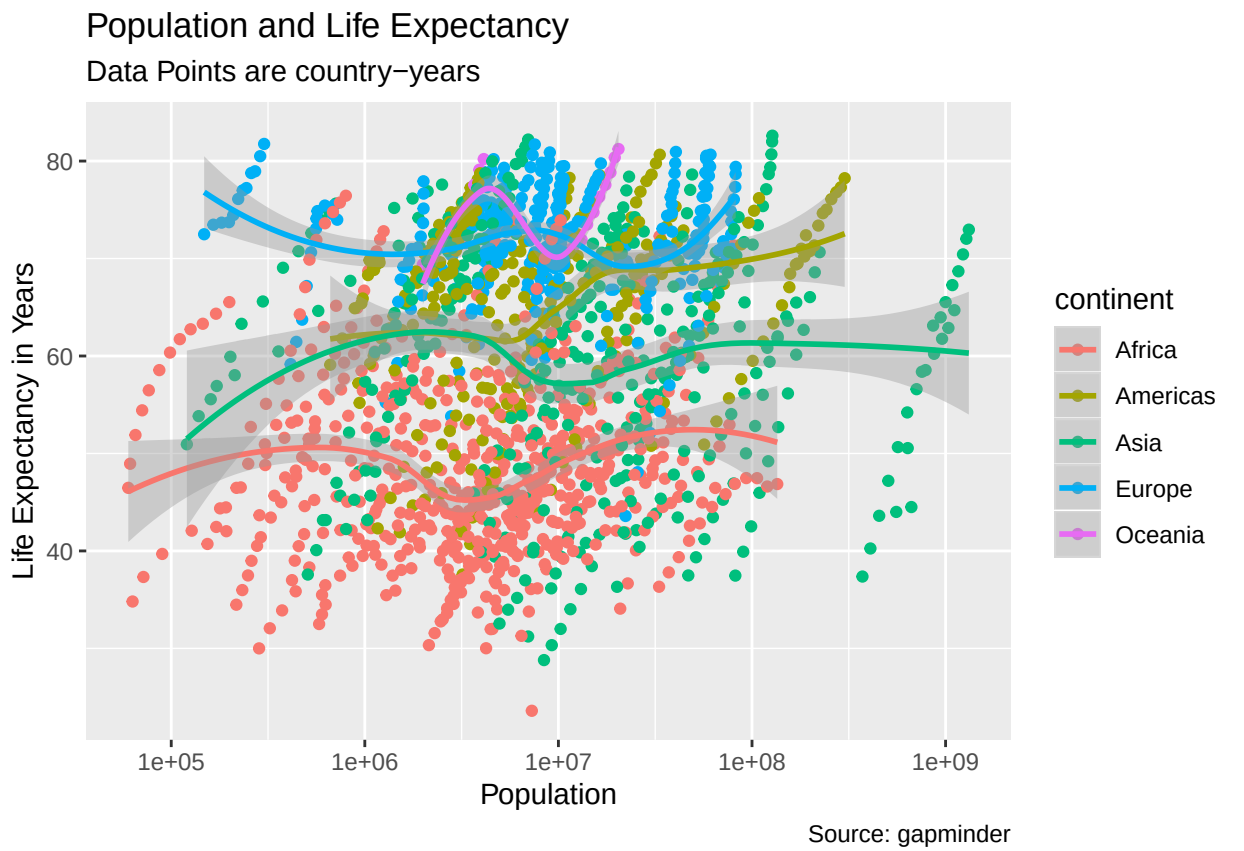
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Here, we use population instead of GDP per capita. We try to shed light on the connection of life expectancy and population. We see a strong distinction between developing continets (Africa) and developed continents (e.g. Europe).

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

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



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 : Posted on Discourse.

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 :

1, An emptpy plot, beacause only data is defined, but there is no mapping or geom_.

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

2, 234 rows and 11 cols

```
ncol(mpg)
```

```
## [1] 11
```

```
nrow(mpg)
```

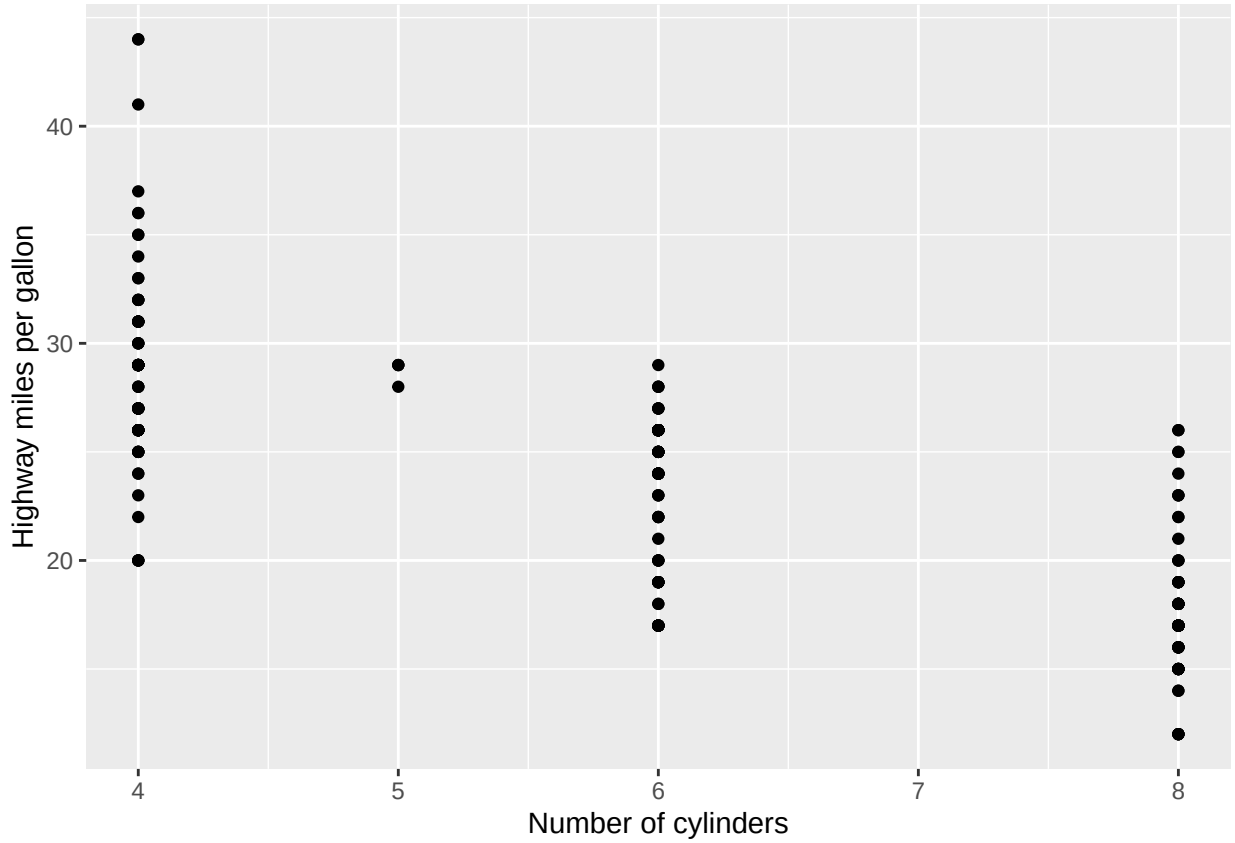
```
## [1] 234
```

3, drv: f = front-wheel drive, r = rear wheel drive, 4 = 4wd

```
?mpg
```

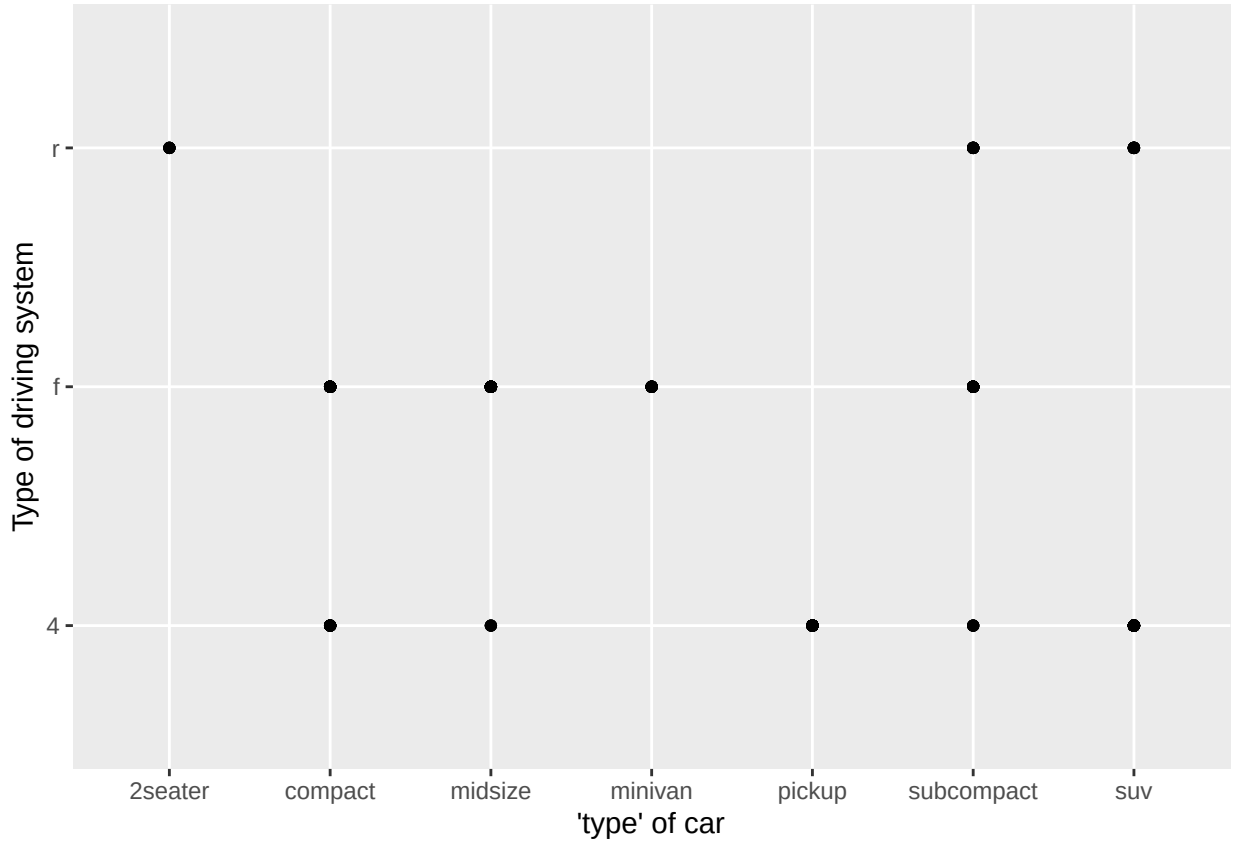
4, See graph below

```
p <- ggplot(data = mpg,
            mapping = aes(x = cyl , y = hwy))
p + geom_point() +
labs(x = "Number of cylinders", y = "Highway miles per gallon")
```



5, These are two categorical variables, doesn't really makes sense to use a scatterplot here.

```
p <- ggplot(data = mpg,
            mapping = aes(x = class , y = drv))
p + geom_point() +
labs(x = "'type' of car", y = "Type of driving system")
```

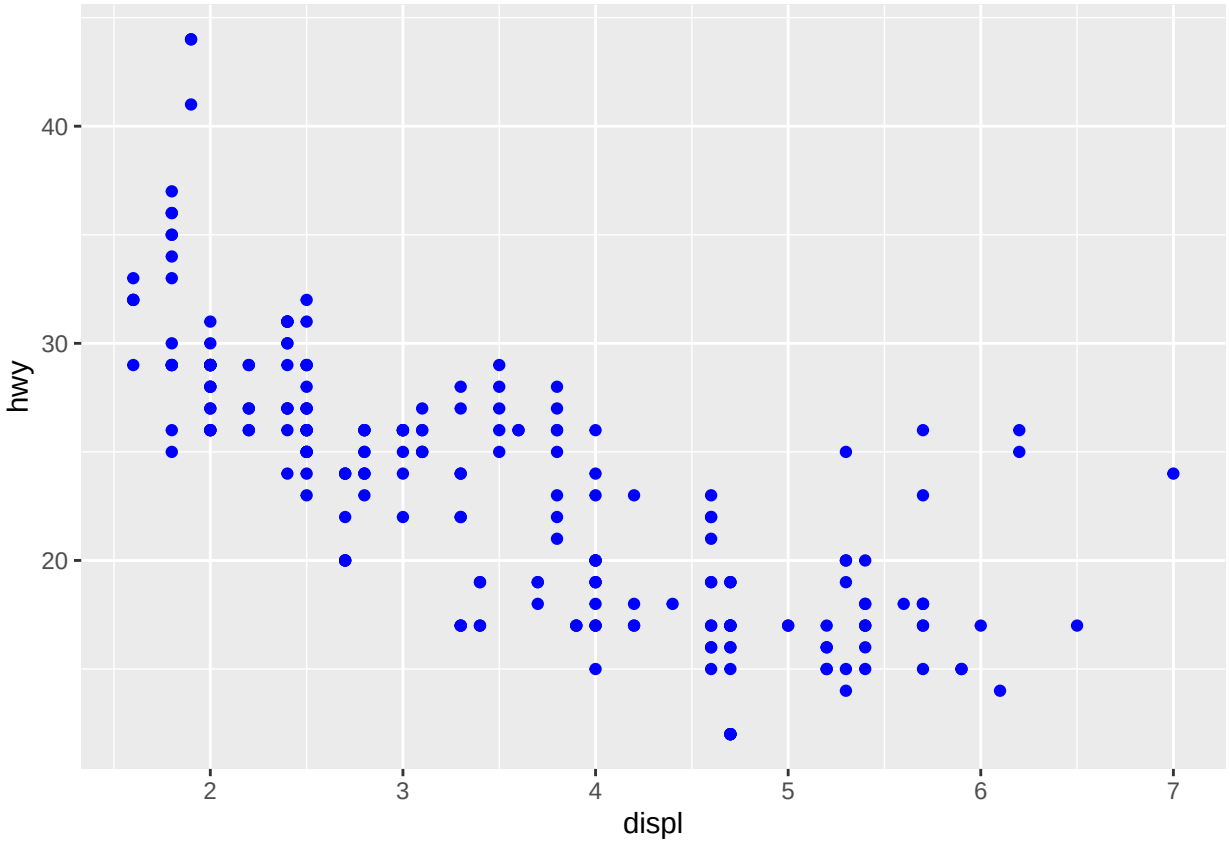


Exercise: Go through Exercises in 3.3.1. If an exercise does not make sense immediately (such as you don't know what a categorical variable is), replace the question by a question that addresses that point (in the case of the caregorical variable "What are categorical and continuous variables and how 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 :

1, Should have place the 'color' argument outside of aes, like this:

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



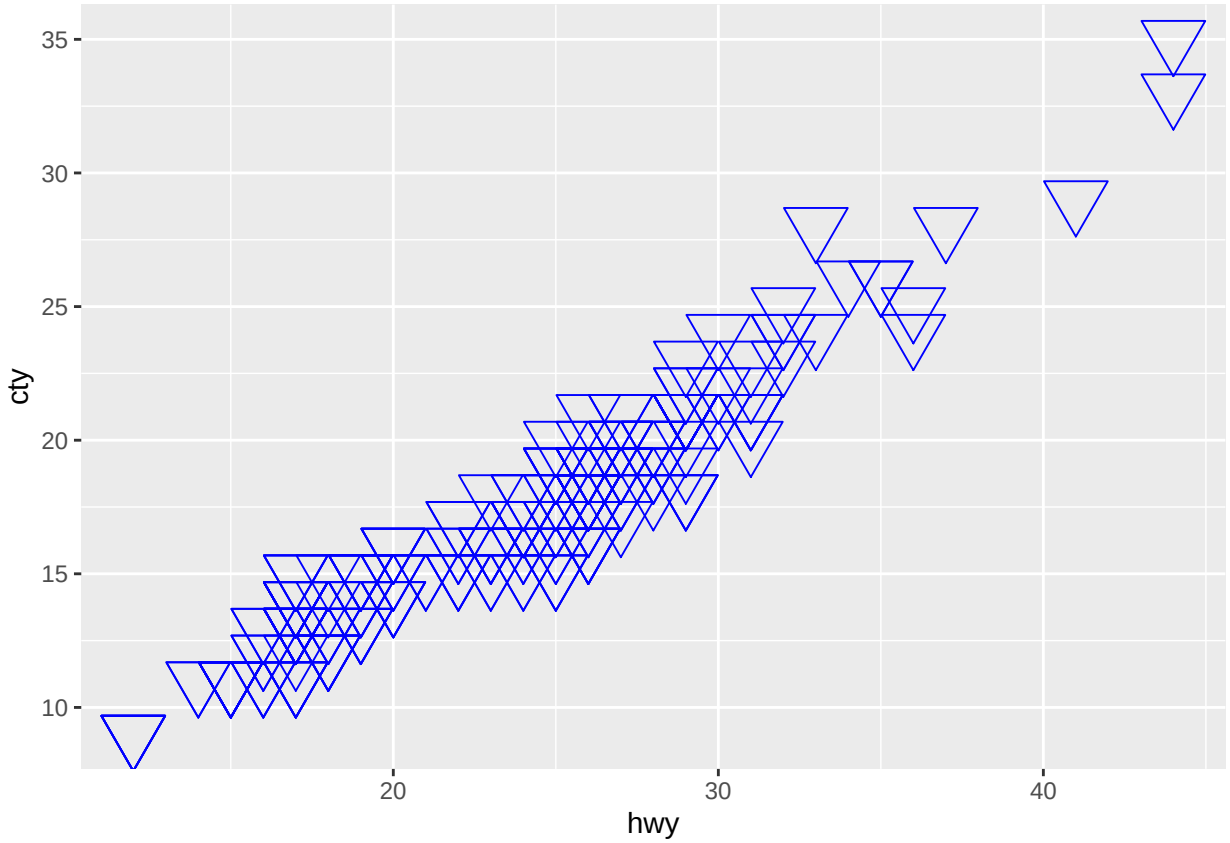
2, Categorical: manufacturer, model, cyl, trans, drv, fl, class

Continuous: disp, year, cty, hwy

Note: Year can actually be both types, it is a matter of interpretation.

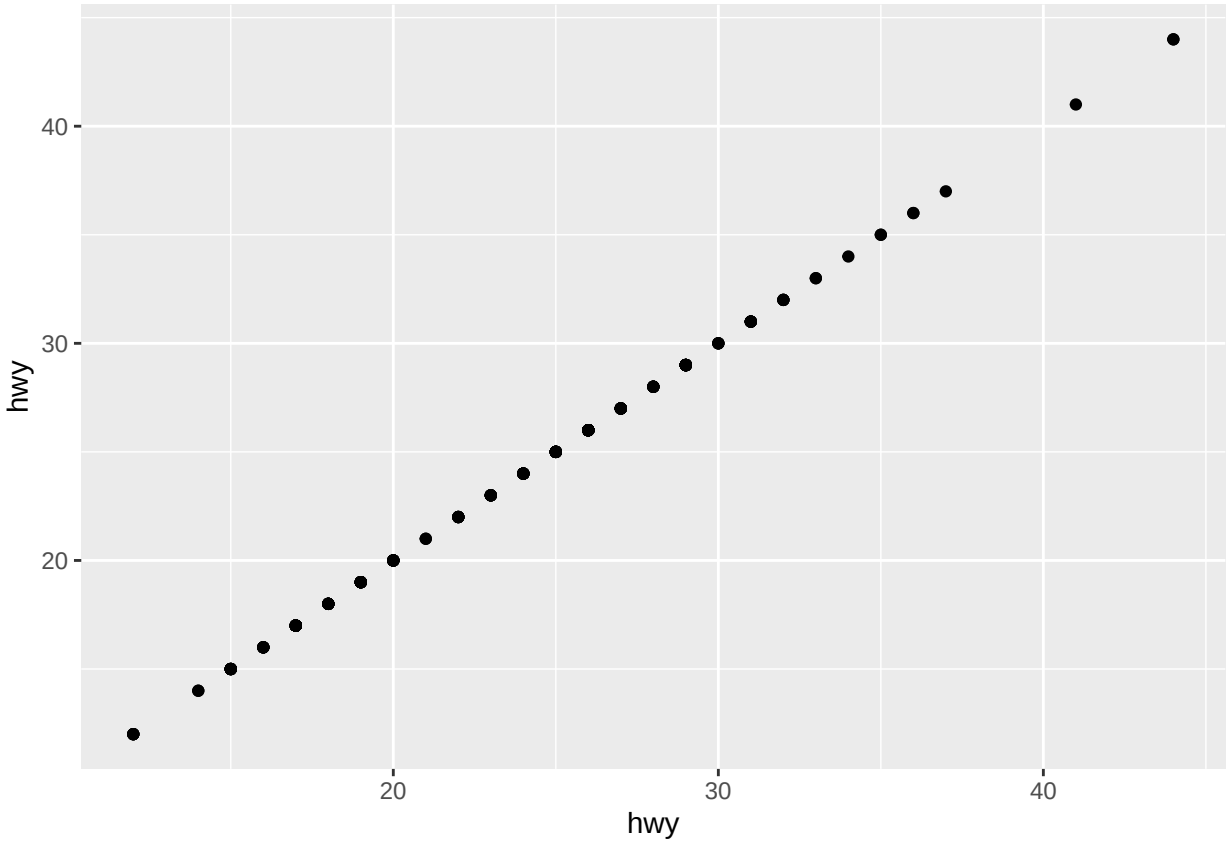
3, Horrible graph, I know. It is not really a good idea to use categorical variables with scatter plot on both axes.Every single category will be displayed as there is not really a between of two categorical vars.

```
ggplot(data = mpg) + geom_point(mapping = aes(x = hwy, y = cty), size = 8, color = "blue", shape = 25)
```



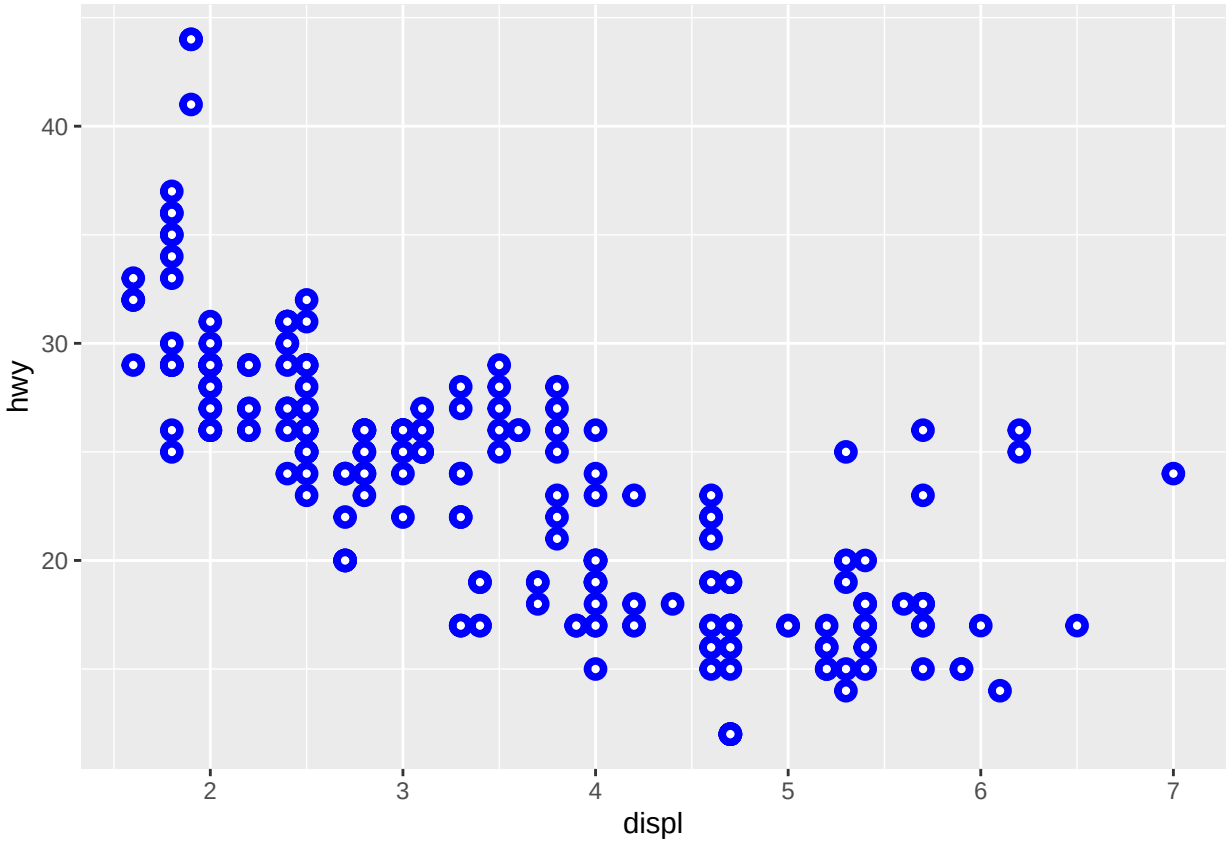
4, Perfect correlation can be seen here. Of course, it is possible, but doesn't really make sense.

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



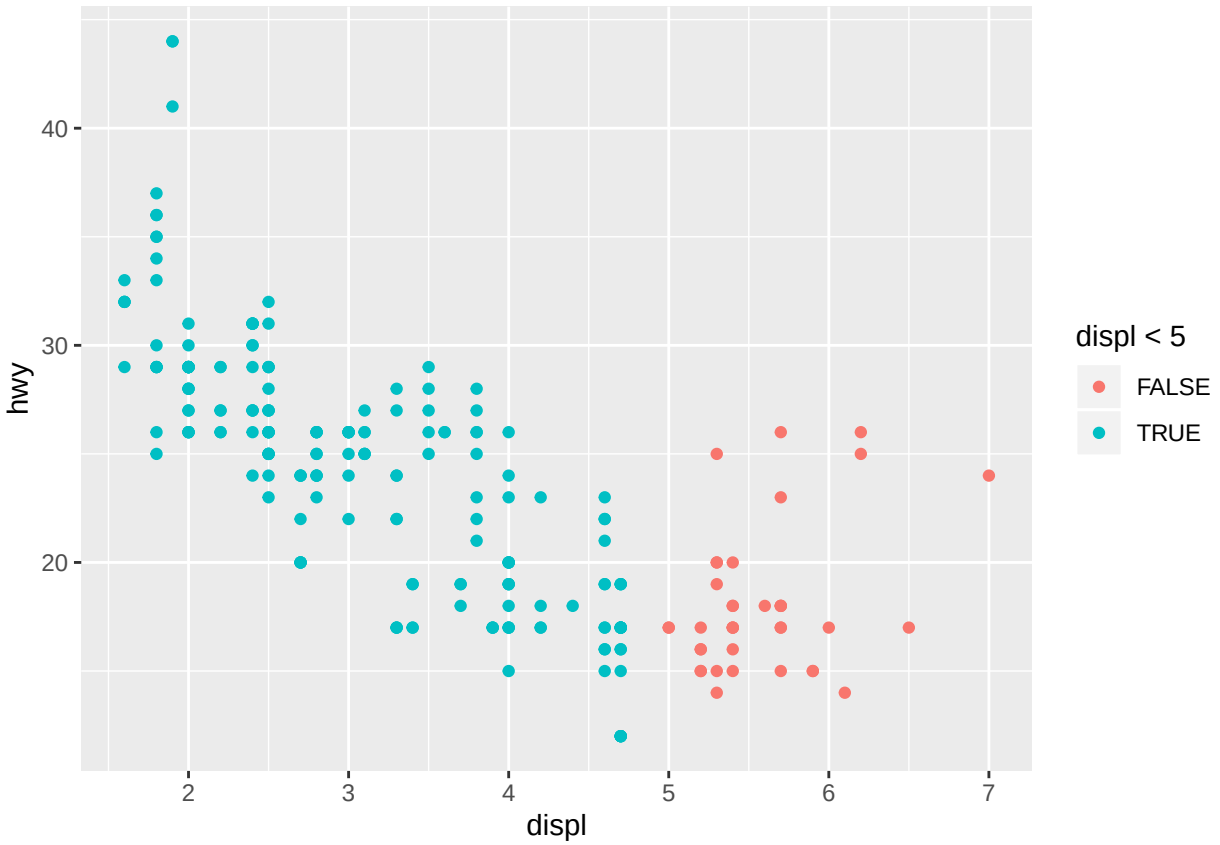
5, When dots have an inside and an outside part, it is possible to colour them differently. Stroke is used to set the width of the outside part, also called the border.

```
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy), shape = 21, color = "blue", fill = "
```



6, It is going to colour the dots according to whether the corresponding 'disp' value is greater then 5 or not.

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



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

Answer : In the second line, there is a typo or a wrong character. "i" and "ı" are not the same, so there is no variable to be found, accordingly we are given an error message.

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

Bonus Answer : In the first appearance, the `eval` argument is set to `FALSE`, therefore when knitting it is not compiled. It is just basically a text, not a line of code to be run.

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.