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
 Other type of plots:
+ geom_line ()# line chart
+ geom_density ()# density chart
+ geom_histogram() # histogram chart
```

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.

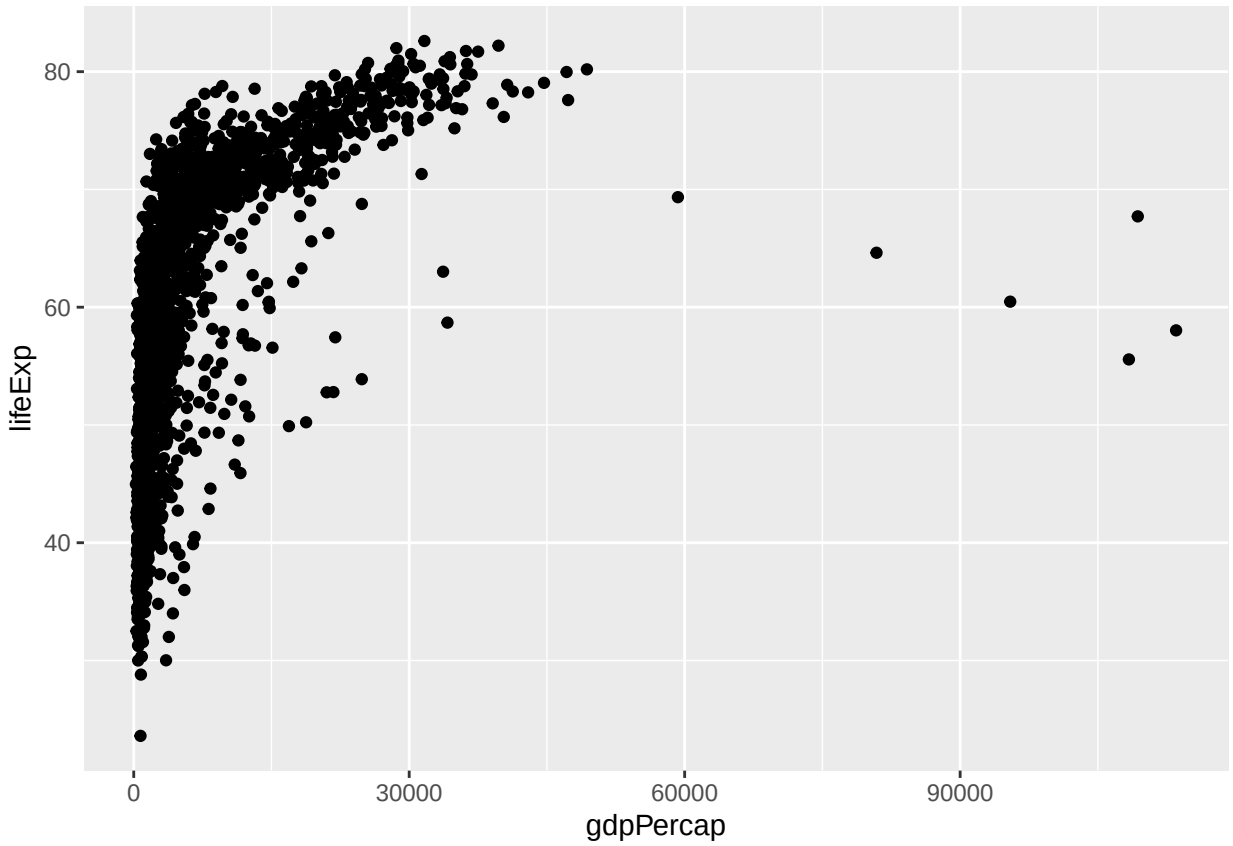
A: Other type of plots: geom_line # line chart geom_density # density chart geom_histogram # histogram chart

geom_col() # another type of bar chart geom_blank() # The blank geom draws nothing, but can be a useful way of ensuring common scales between different plots. geom_count() # count number geom_text() # add text directly to the plot ## 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

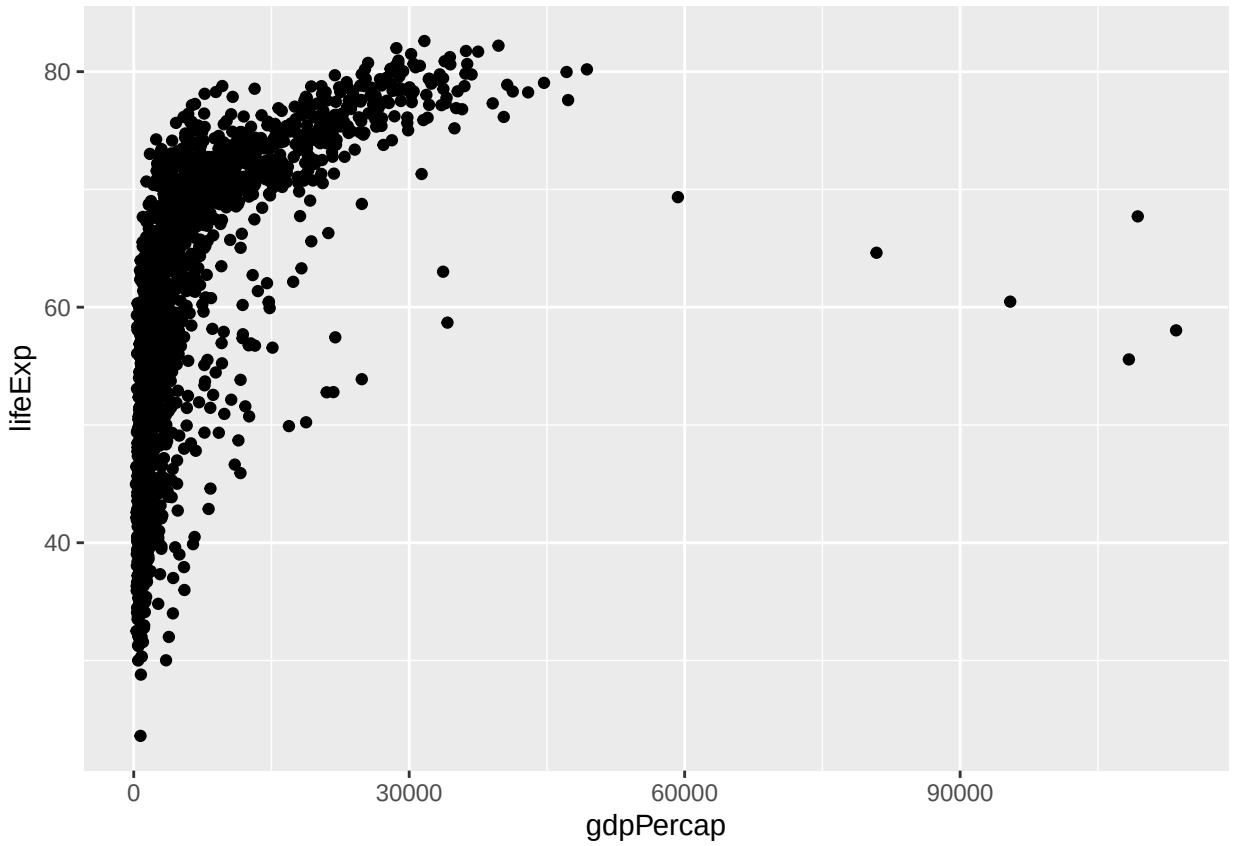
Importantly, mappings don't say *what* color or shape some variable will have – rather, it says that a given dataset will be mapped *to* the color or *to* the shape.

```
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.

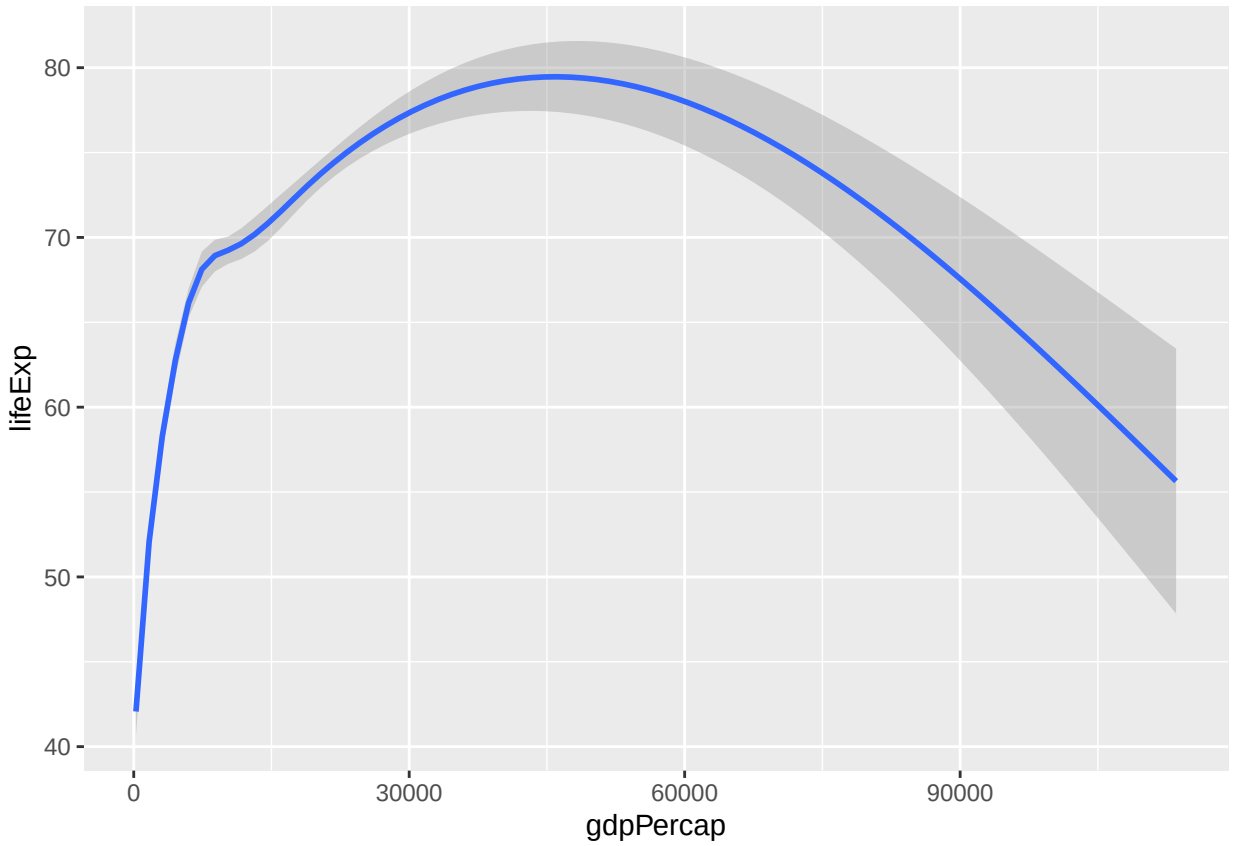
Finally, we add a *layer*. This says how some data gets turned into concrete visual aspects.

```
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.

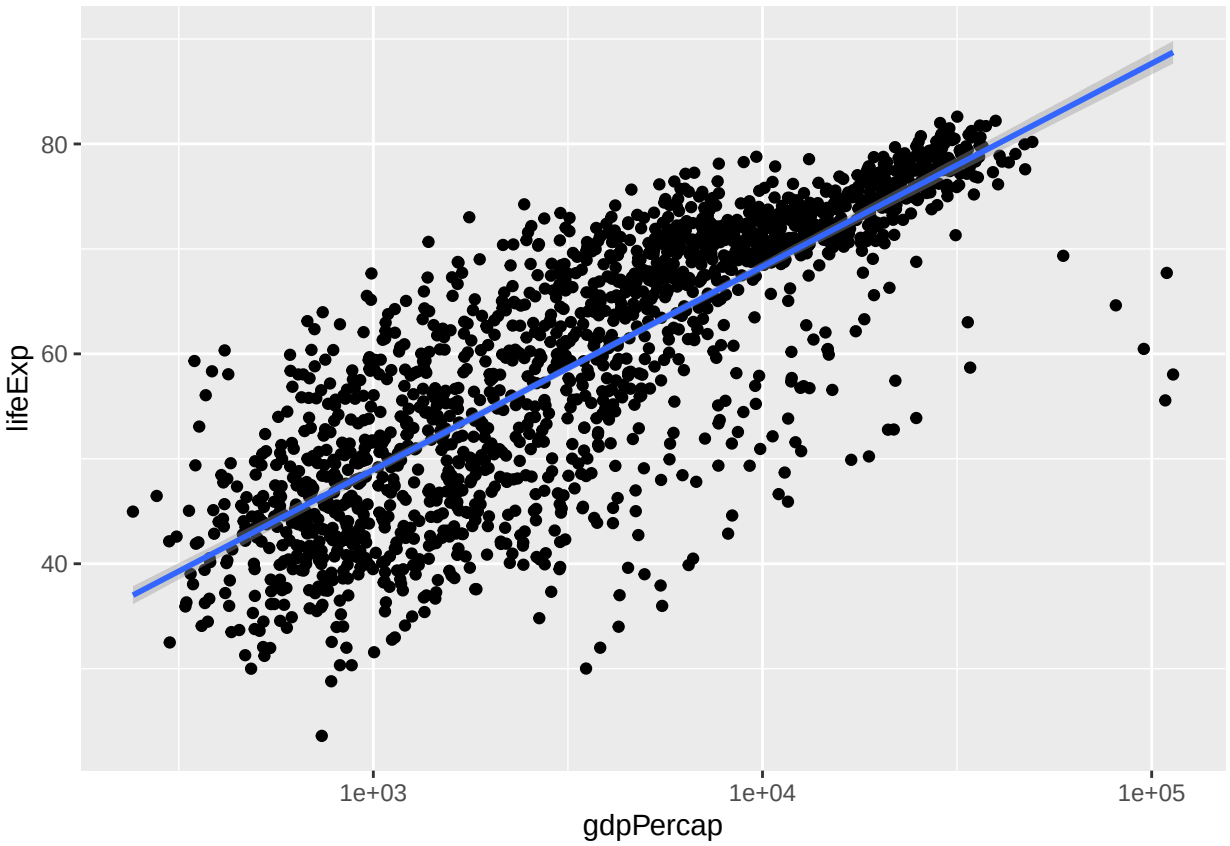
We get a message that tells us that `geom_smooth()` is using the method = 'gam', so presumably we can use other methods. Let's see if we can figure out which other methods there are.

```
?geom_smooth
p + geom_point() + geom_smooth() + geom_smooth(method = 'lm') + geom_smooth(method = 'glm')
p + geom_point() + geom_smooth() + geom_smooth(method = 'auto') + geom_smooth(method = loess, color = "
```

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.) A: `scale_x_log10()`: Logarithmic conversion of the x-axis of the function. Because log function could shrink the distance between number in x-axis, which will make the data display more uniform and more significant.

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

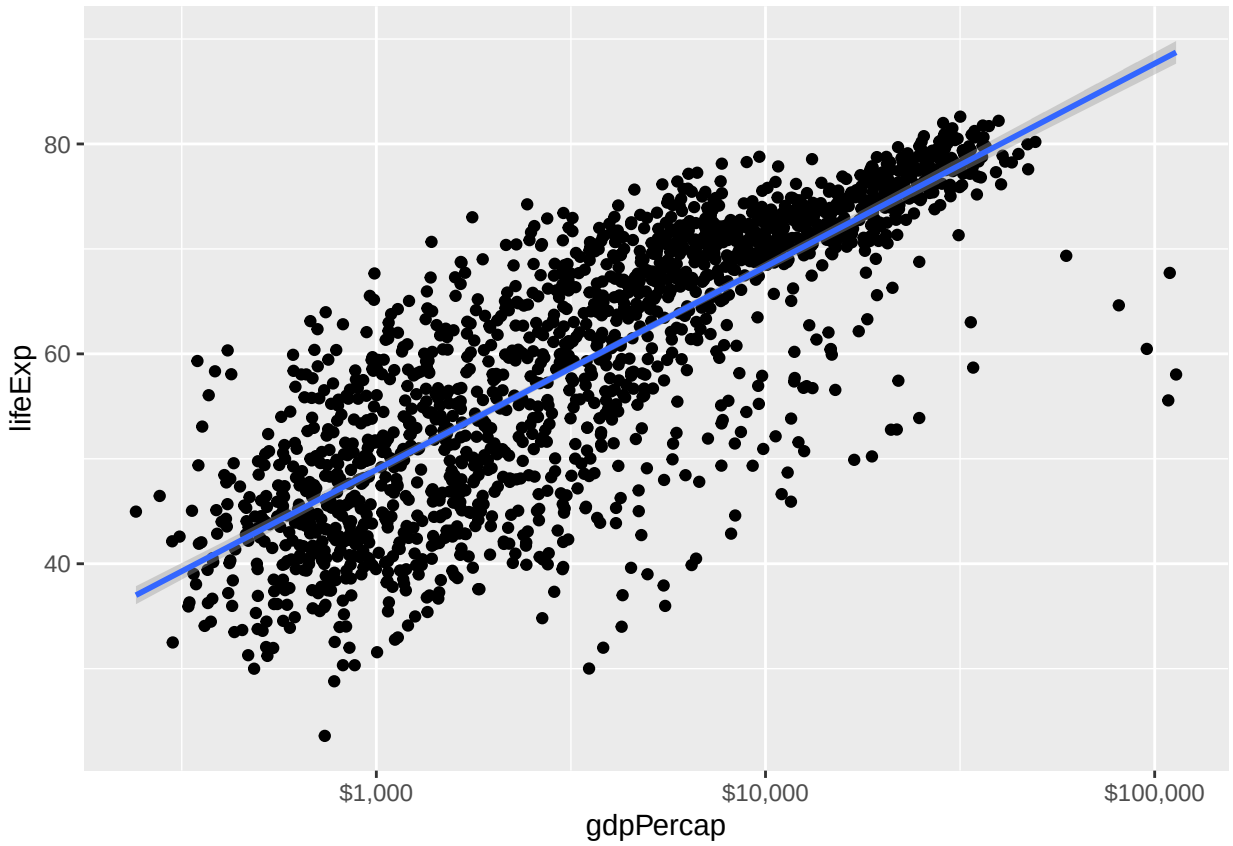
```
library(scales)
```

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

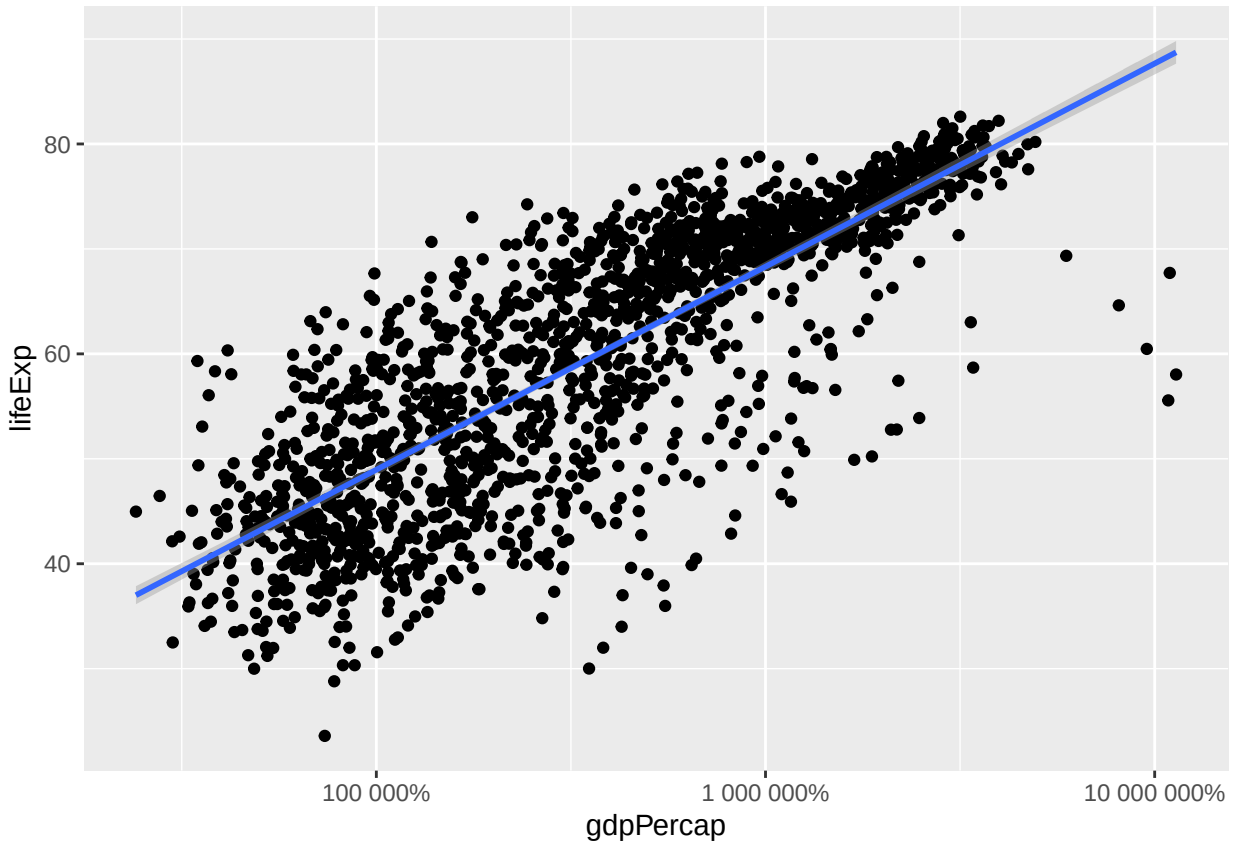
```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## 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::percent)
```



Exercise: What does the `dollar()` call do? A: Currency formatter: round to nearest cent and display dollar sign.

```
?dollar()
```

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

A: Use percentage (percent) to relabel the scales

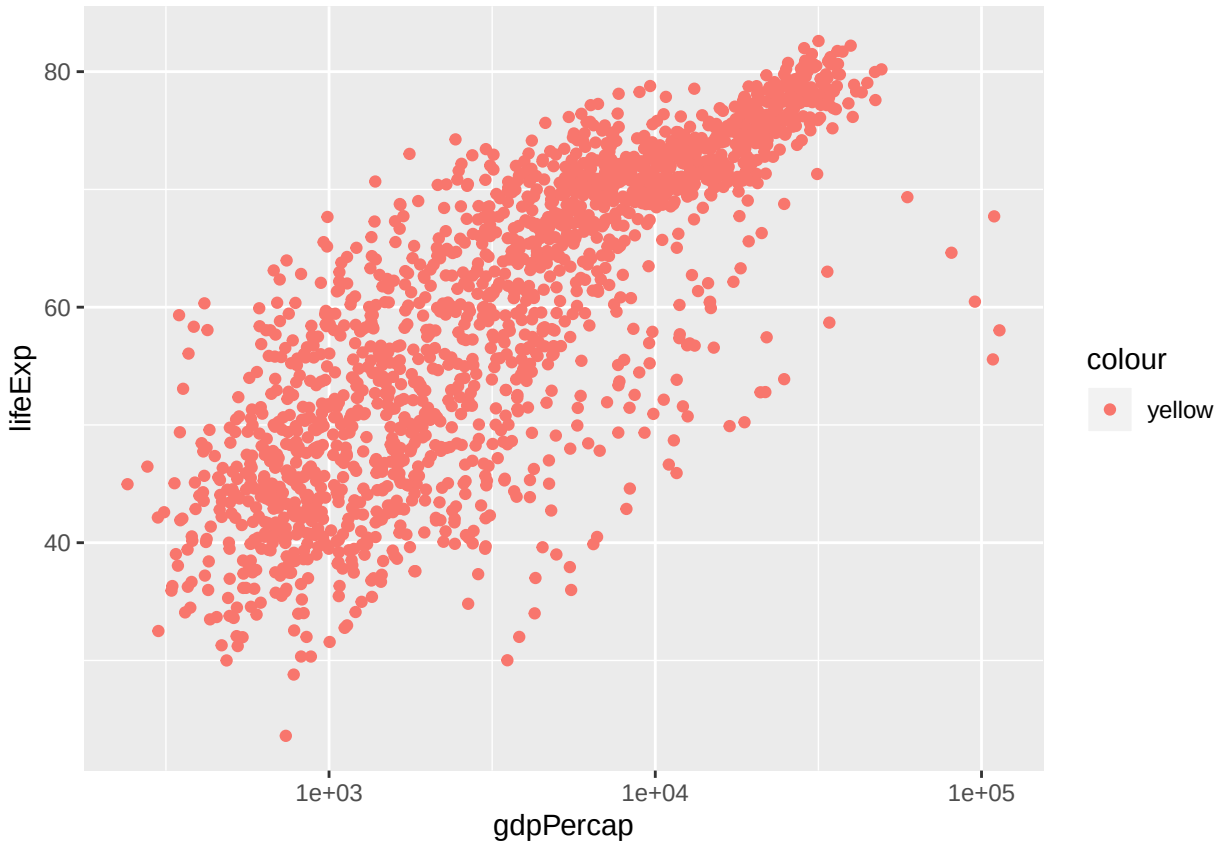
The Ggplot Recipe

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, color='red'))
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. A: In the above programming statement, color is in aes(), aes(color=yellow)just defines color, and yellow is treated as data. Therefore, the color of the scatter plot did not change as expected. If we want to change the color, the color=yellow should be placed outside the brackets and in the right place.

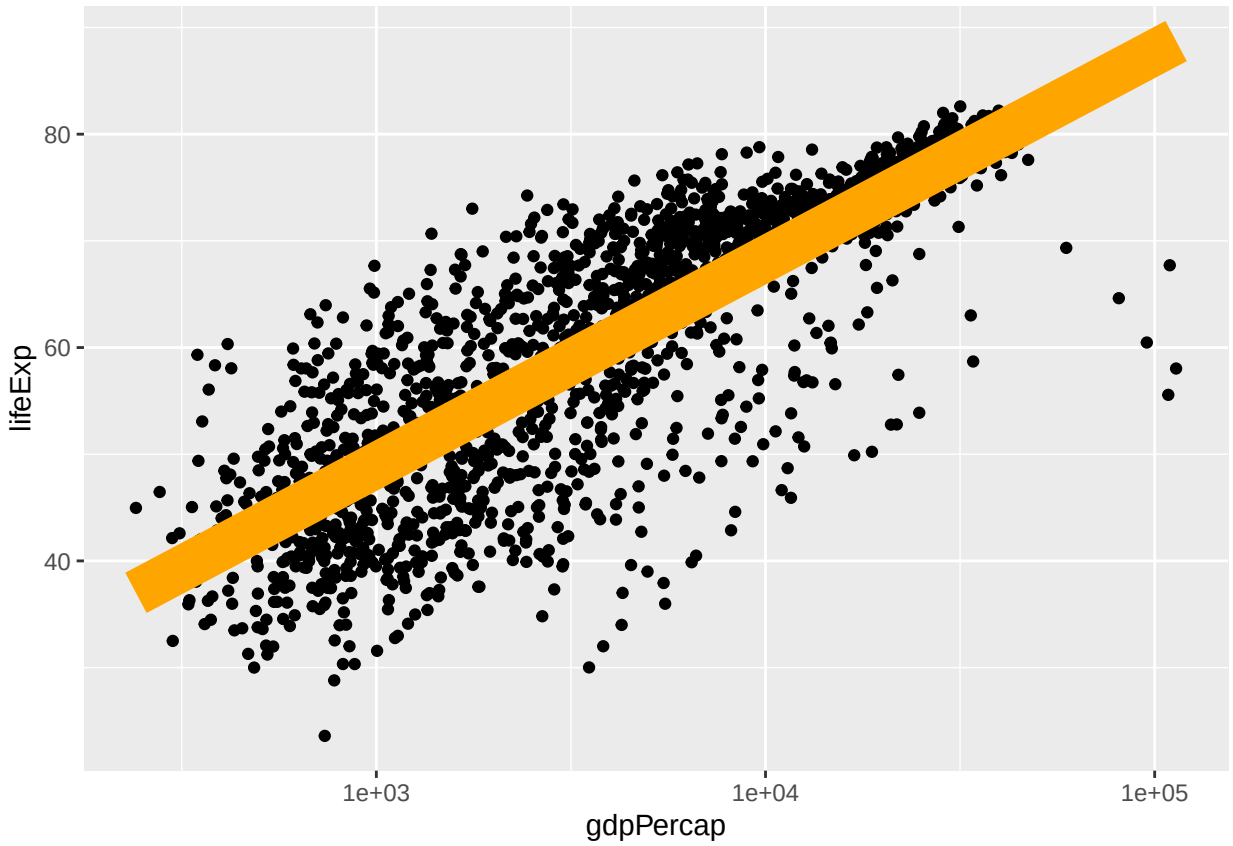
One way to avoid such mistakes is to read arguments inside `aes(<property> = <variable>)`as *the property in the graph is determined by the data in* .

Exercise: Write the above sentence for the original call `aes(x = gdpPercap, y = lifeExp, color = 'yellow')`. A: Shown as the figure and function above.

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.

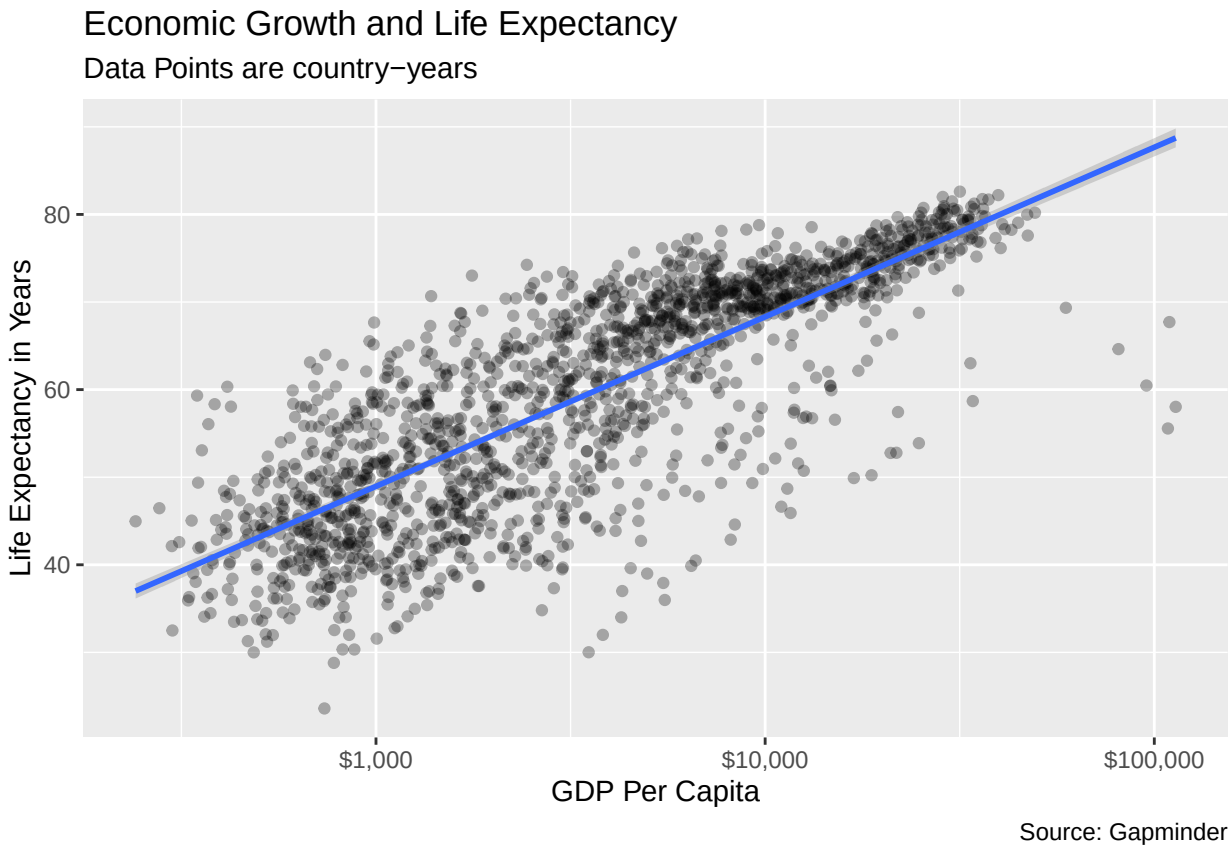
Remember: `color = 'yellow'` and `aes(color = 'yellow')` are very different, and the second makes usually no sense, as `'yellow'` is treated as *data*.

```
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()
```



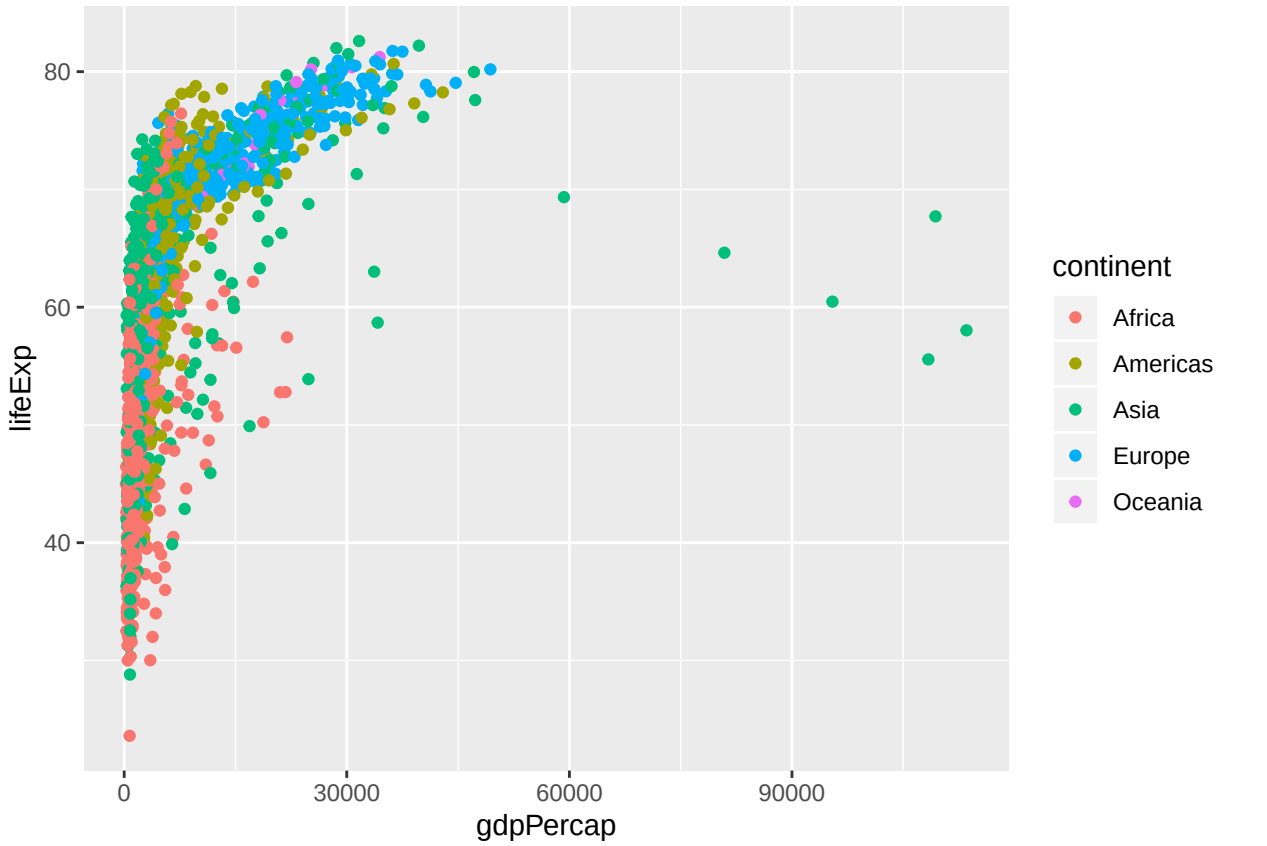
Exercise: Write down what all those arguments in `geom_smooth(...)` do. A: color: the color of the line se: Display confidence interval around smooth? (TRUE by default, see level to control.) size: the size of the line method: smoothing method

```
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")
```

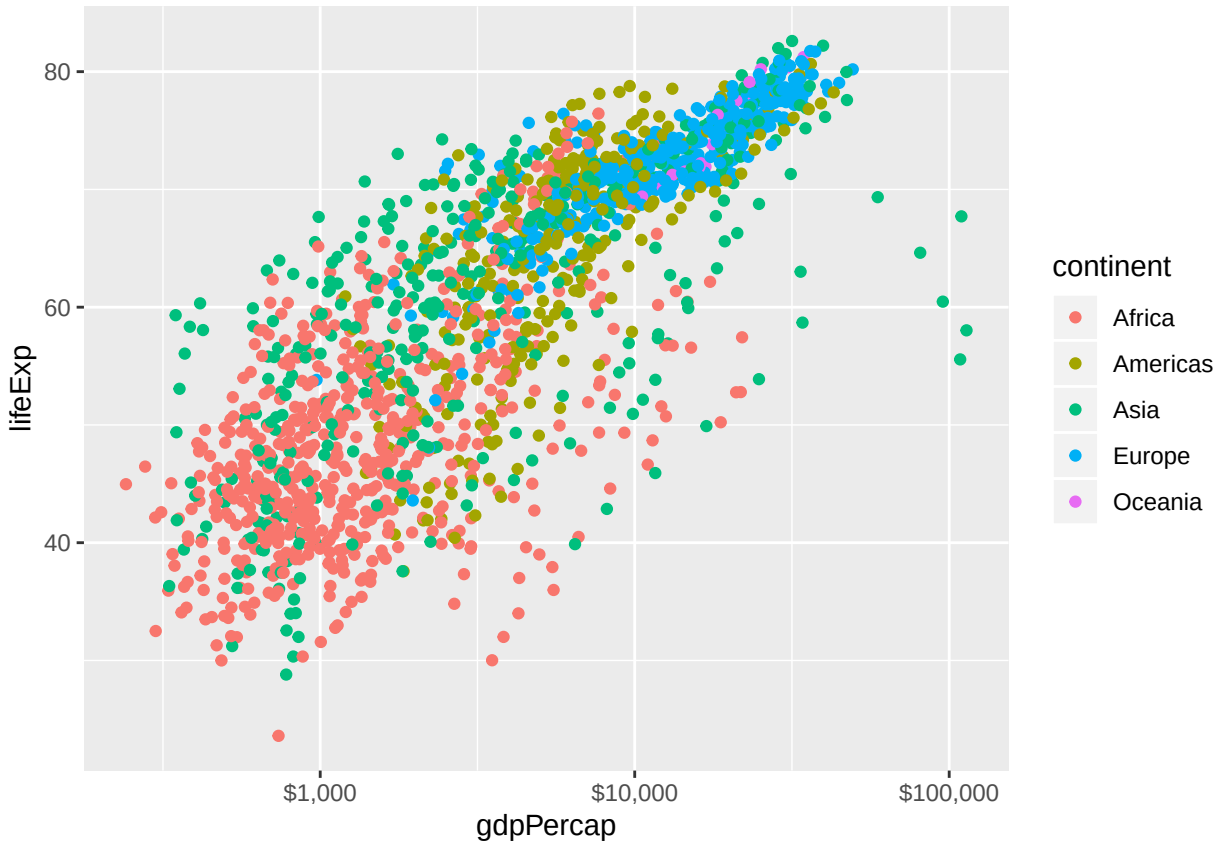


Coloring by continent:

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

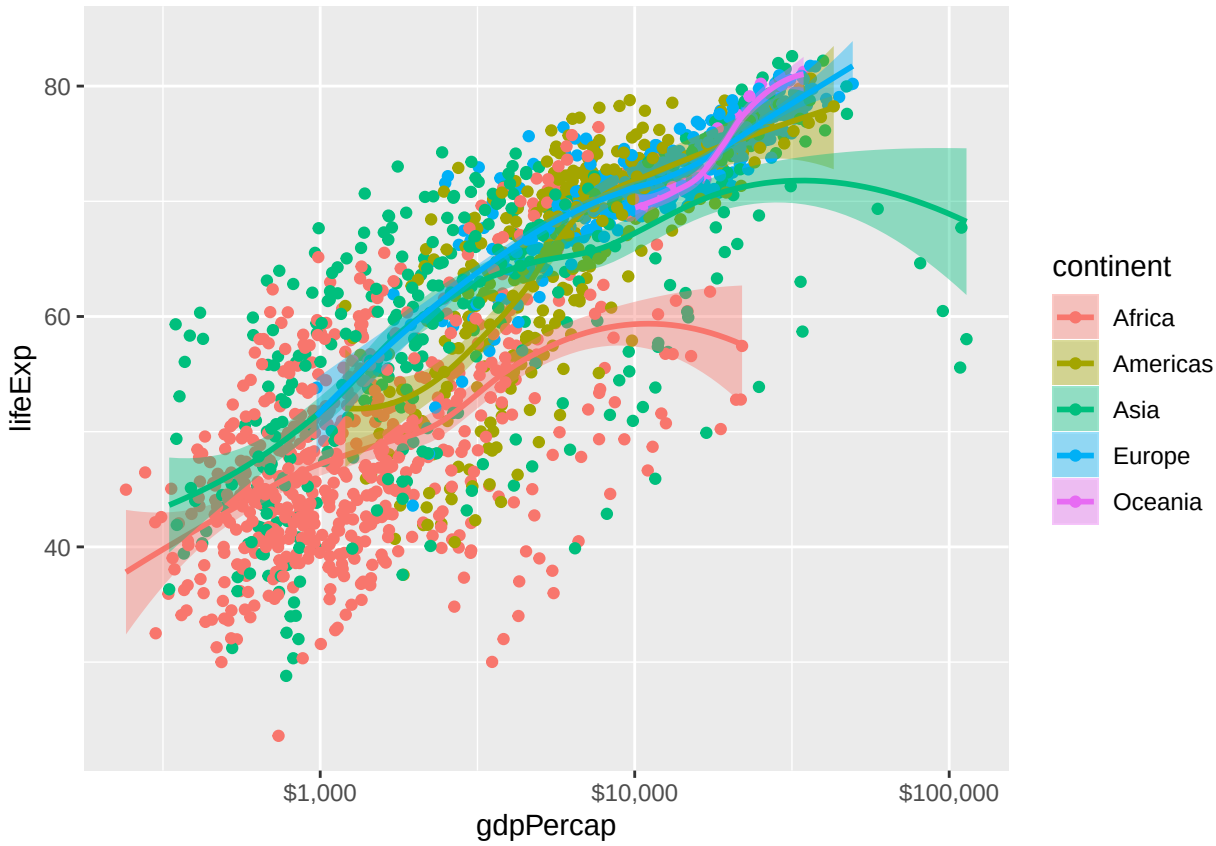



```
p + geom_point() + scale_x_log10(labels = dollar)
```




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

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

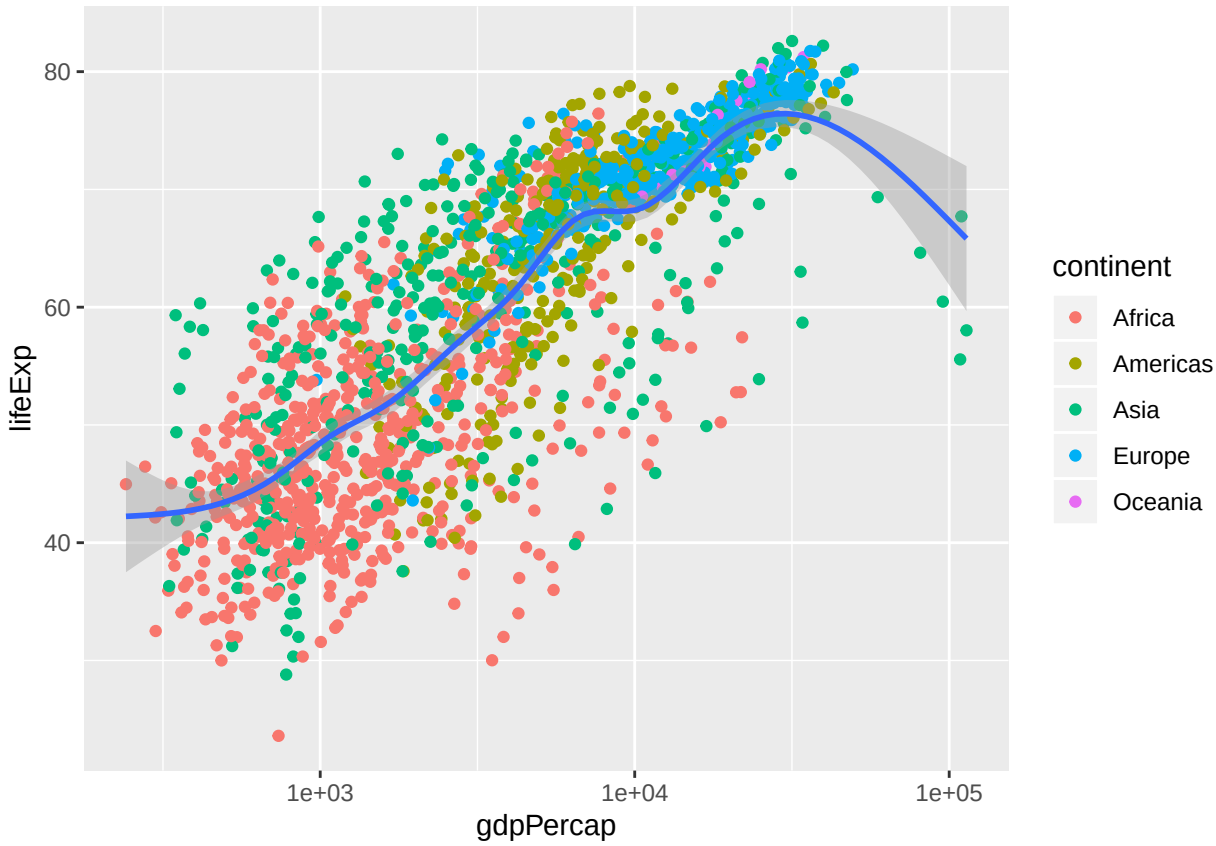


Exercise: What does `fill = continent` do? What do you think about the match of colors between lines and error bands?

A: Use fill to classify existing data, and continent is a classified tag.

```
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")'
```

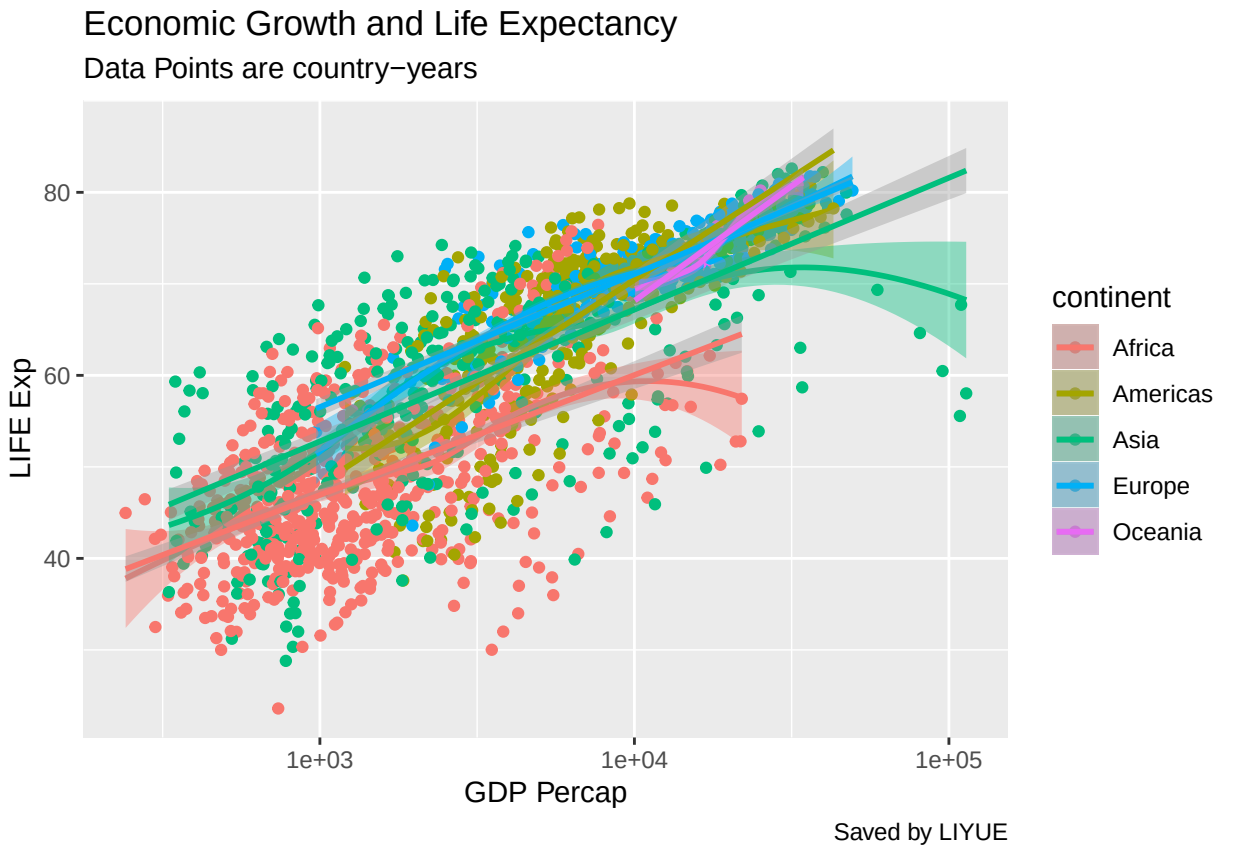


Exercise: Notice how the above code leads to a single smooth line, not one per continent. Why? A: Because, there is no classification of color in ggplot(). The scatter plots are color categorized only in the geom-point, so this operation does not affect the color categorization of the smooth.

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. A: I think we just need to set the color and classification in ggplot, like ggplot(color = continent,fill = continent), and do not need set them in every individual geometry level.

```
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 Percap', y = 'LIFE Exp',
       title= 'Economic Growth and Life Expectancy',
       subtitle = 'Data Points are country-years',
       caption = 'Saved by LIYUE')
```

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




```
ggsave(p, filename = 'Saved by LiYue.png', width = 12, height = 9 )
```

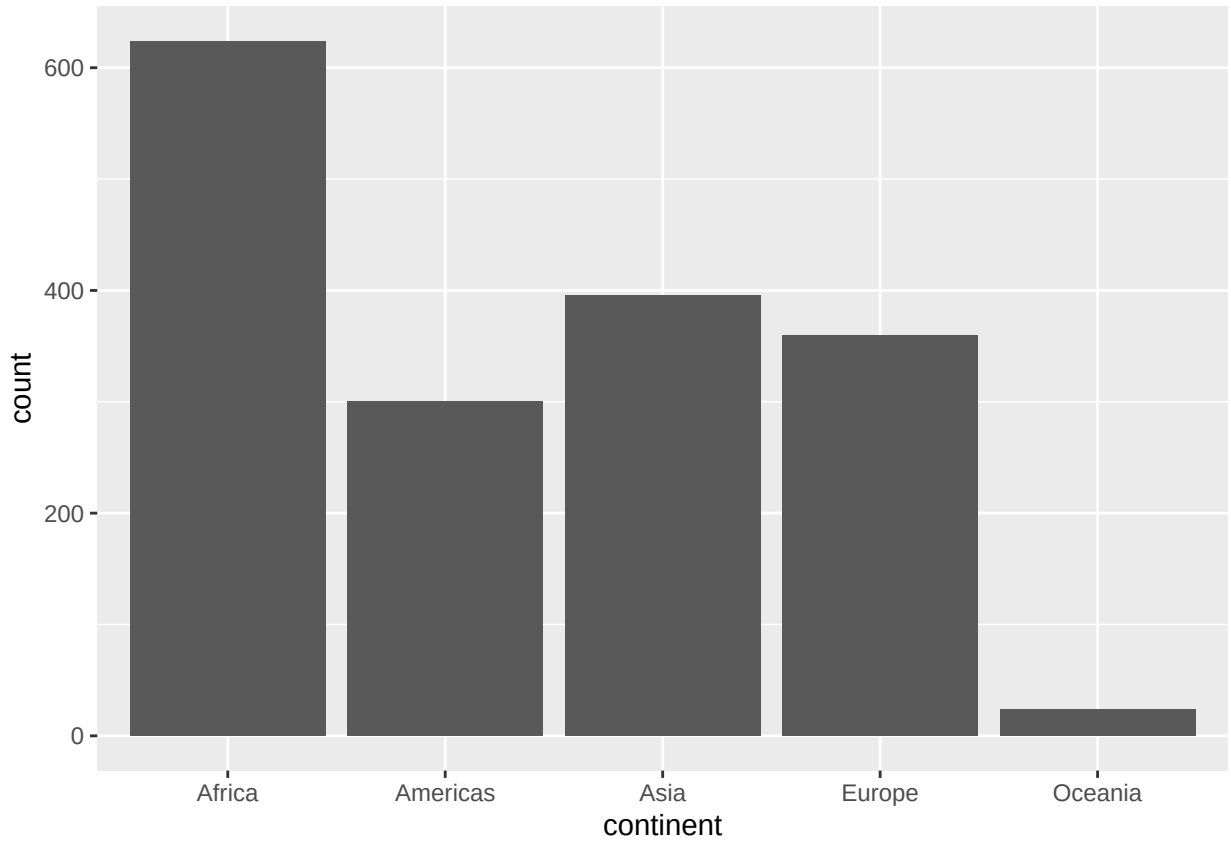
Additional (not Optional) Exercises

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.

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.

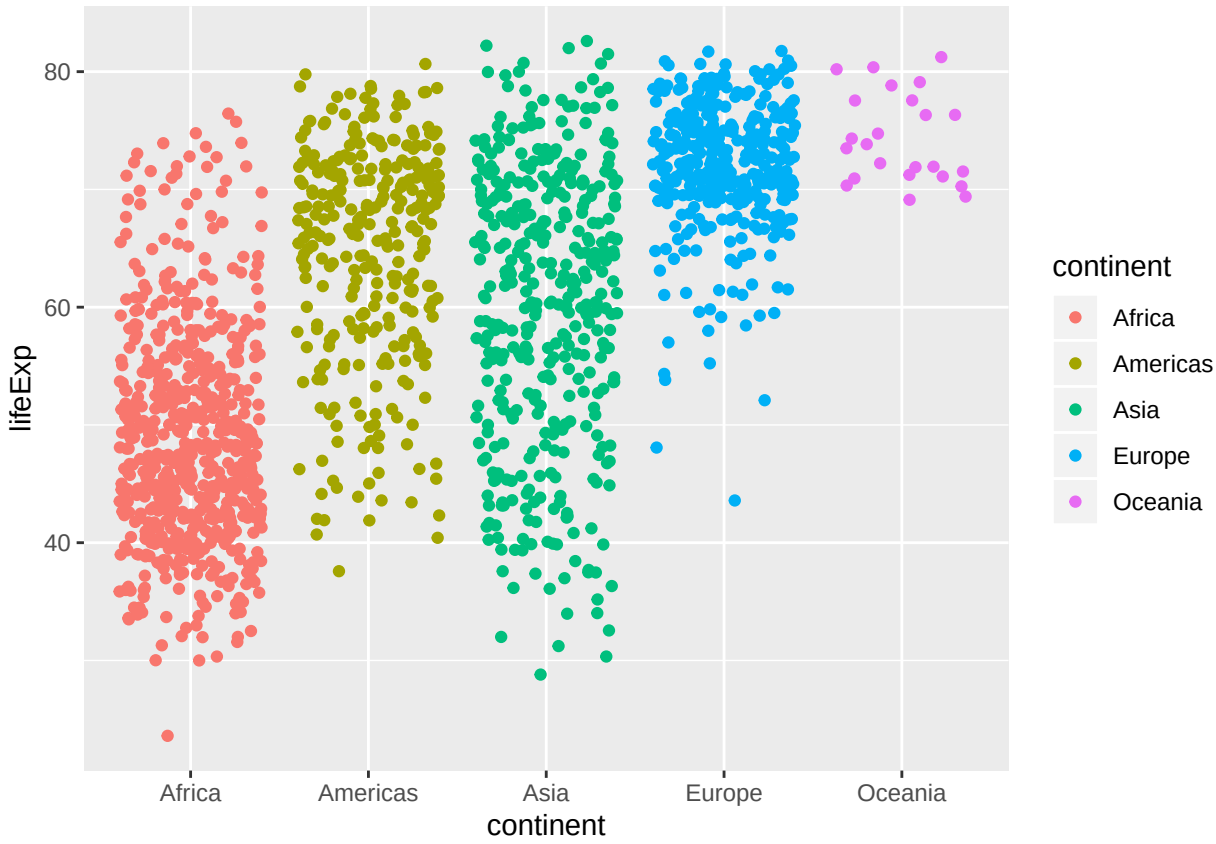
A: (1)Frequency of occurrence in different continents in the data.(shown as below)

```
ggplot(data = gapminder) +
  geom_bar(mapping = aes(x =continent))
```



(2) LifeExp in different continent.(shown as below)

```
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = continent, y = lifeExp, color = continent, fill = continent), position =
```



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.

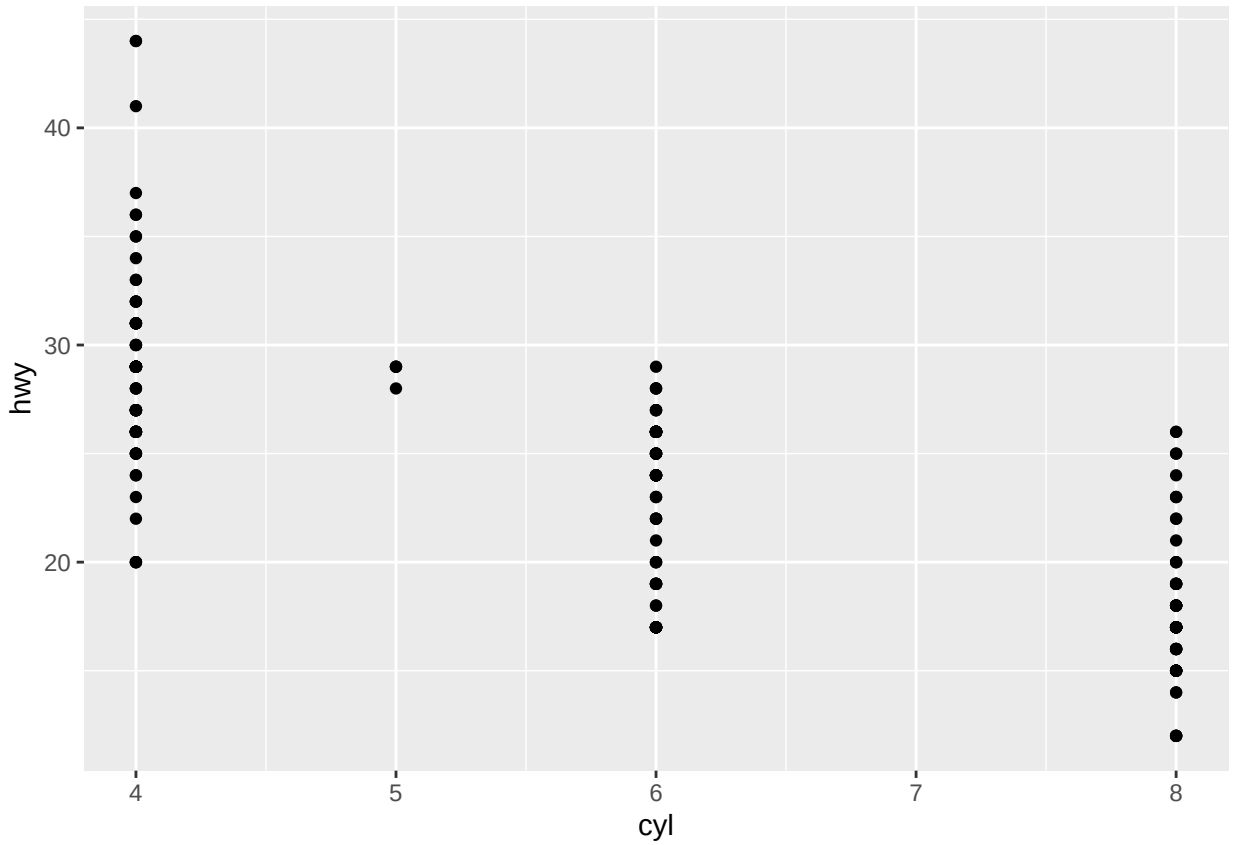
Exercise: Do exercise 3.2.4 from R for Data Science. Include your code in chunks, describe the output and code (where necessary) in the surrounding text.

```
ggplot(data = mpg) # a blank picture
```

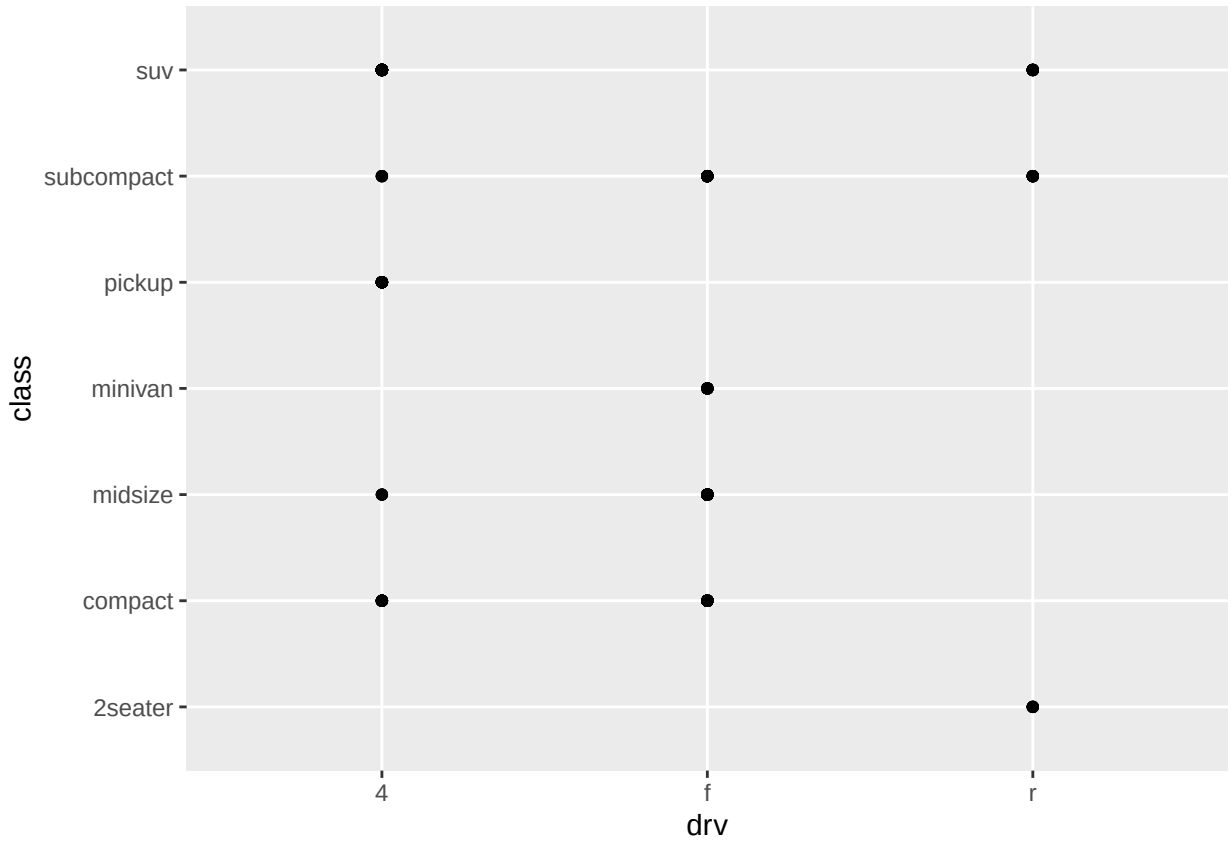
```
dim(mpg) # 234 11
```

```
## [1] 234  11
```

```
?mpg # drv: f = front-wheel drive, r = rear wheel drive, 4 = 4wd
ggplot(data = mpg)+
  geom_point(aes(x = cyl, y = hwy))
```



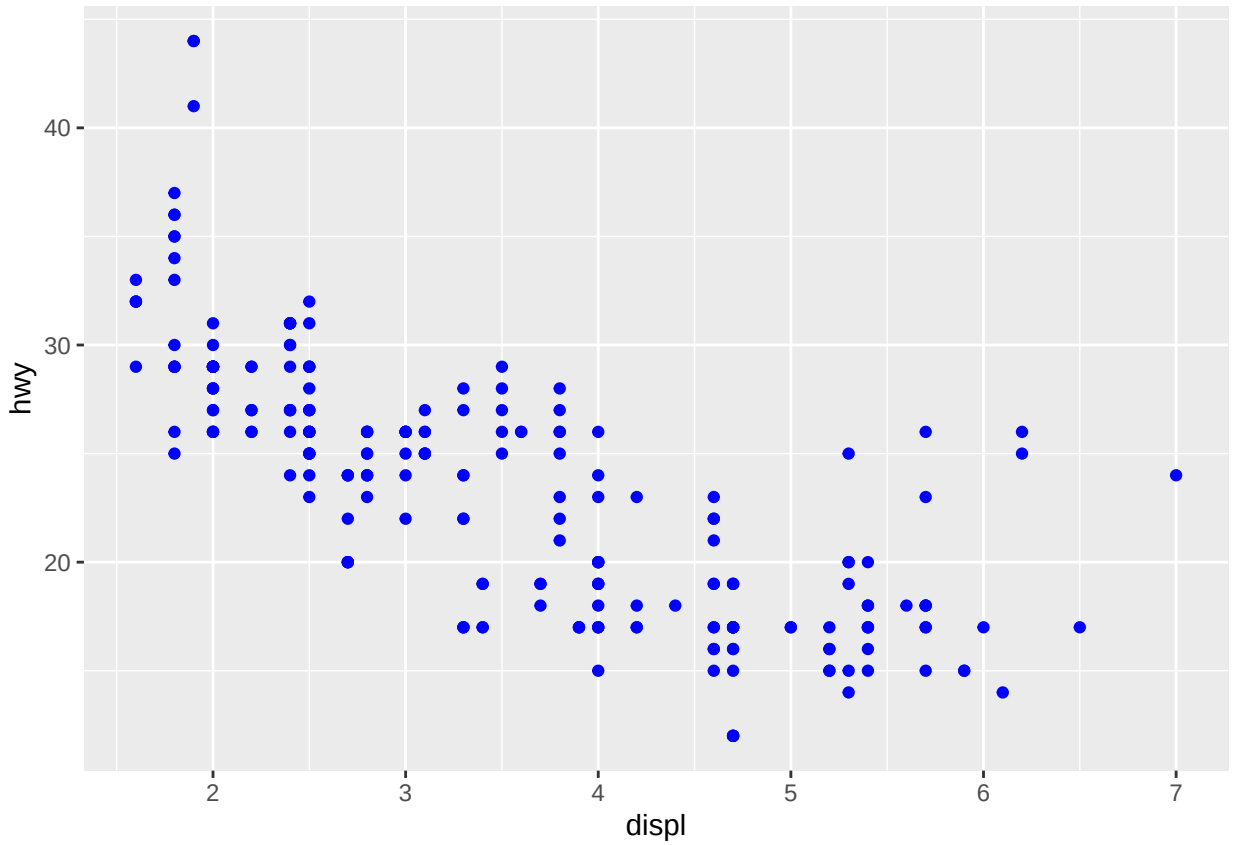

```
ggplot(data = mpg)+
  geom_point(aes(x = drv, y = class))
```



(1)I saw a blank picture. (2)dim(mpg) # 234 11, so it has 234 row. (3)?mpg # drv: f = front-wheel drive, r = rear wheel drive, 4 = 4wd. (4)As shown in the figure above. (5)The x-axis has no meaning for it has numbers and characters toghter.

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.

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



(1). A: The color argument should be out of aes(), then the color could be blue.

```
str(mpg)
```

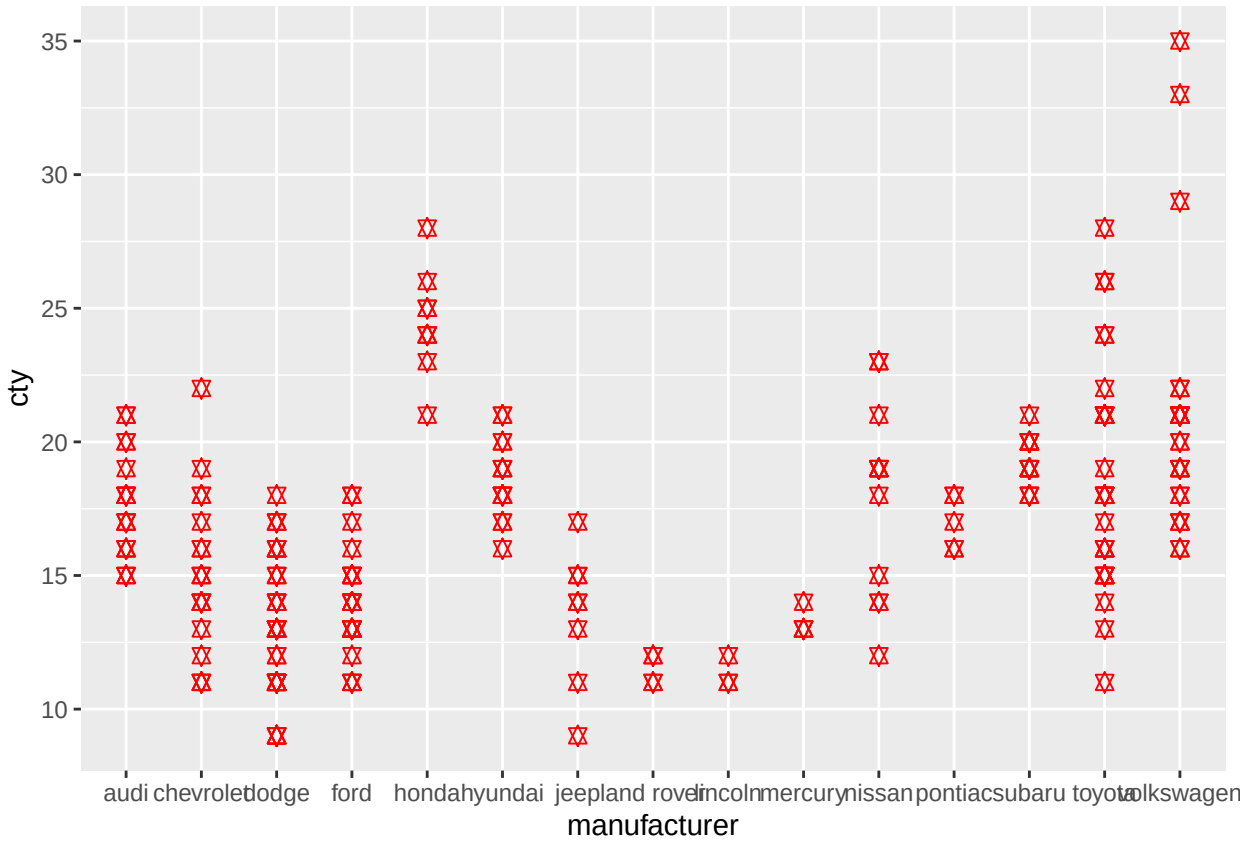
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    234 obs. of  11 variables:
##  $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
##  $ model       : chr  "a4" "a4" "a4" "a4" ...
##  $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr  "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr  "f" "f" "f" "f" ...
##  $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr  "p" "p" "p" "p" ...
##  $ class       : chr  "compact" "compact" "compact" "compact" ...
```

```
?mpg
```

(2). A: categorical: class, fl, manufacturer, trans,drv, displ, cyl continuous: cty, hwy

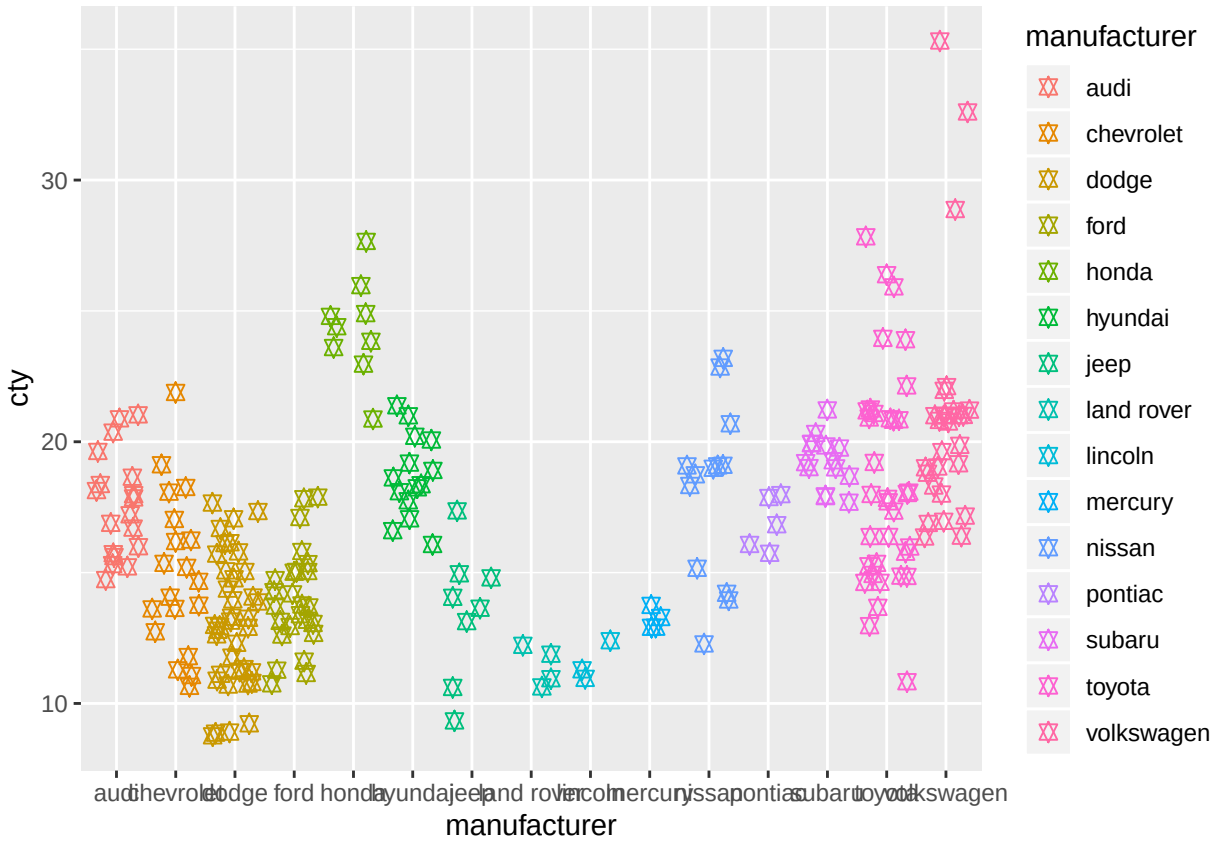
(3). A: shown as below

```
ggplot(data = mpg) +
  geom_point(aes( y=cty, x=manufacturer), color= 'red', size=2, shape= 'star')
```



(4). A: shown as bellow

```
ggplot(data = mpg) +
  geom_point(aes( y=cty, x=manufacturer,fill= manufacturer,color= manufacturer),  size = 2, shape= 'sta
```

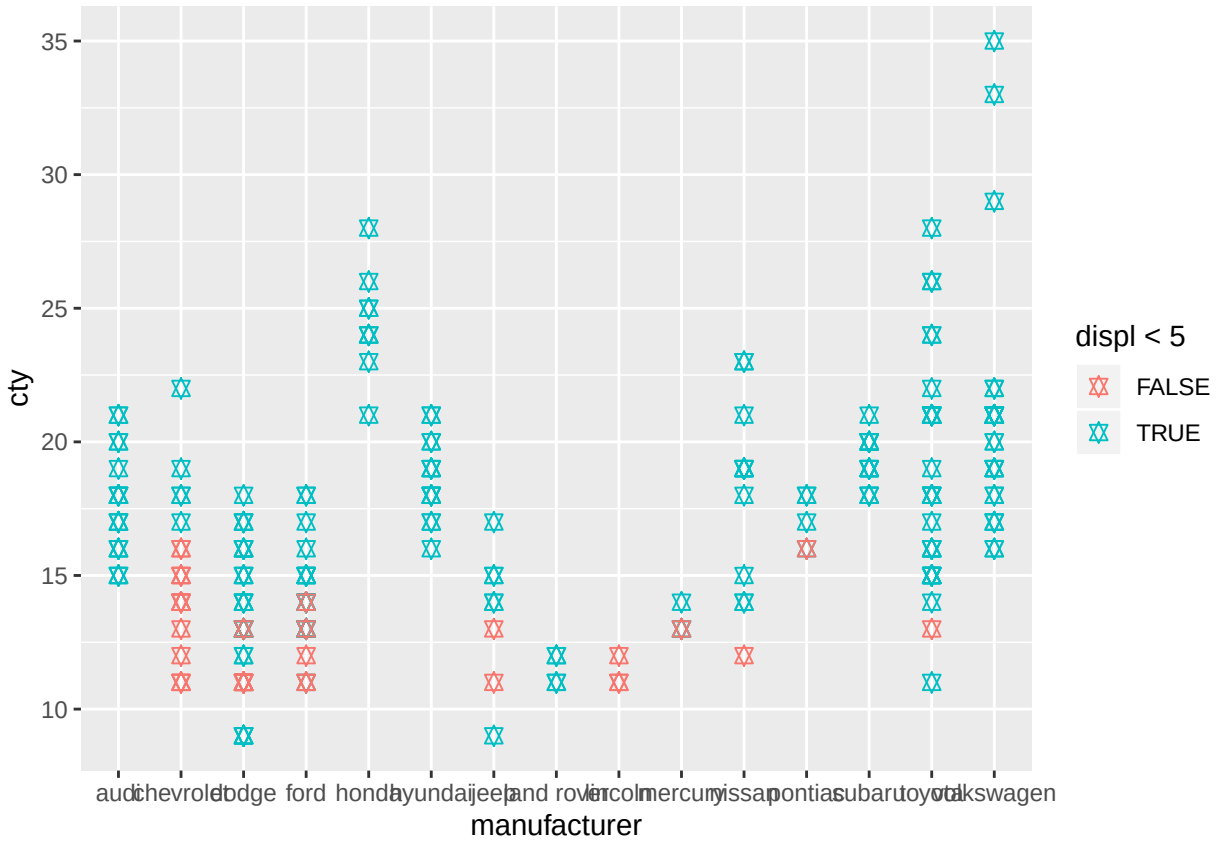


(5). A: Use the stroke aesthetic to modify the width of the border

```
?geom_point
```

(6). A:shown as below

```
ggplot(data = mpg) +
  geom_point(aes( y=cty, x=manufacturer, color = displ < 5), size=2, shape= 'star')
```



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

```
my_variable <- 10
my_variable # used to be a wrong i in 'my_variable'
```

```
## [1] 10
```

```
#> Error in eval(expr, envir, enclos): object 'my_variable' not found
```

4.4.1 A: Because he typied a wrong letter 'i' in 'my_variable'.

Bonus Exercise: Why did I load the `scales` library twice via `library(scales)` to knit? A: I guess, because the first time you run `library(scales)`, the parameters in the package may have been reset. When the function is run for the second time, it is reloaded once in order to prevent the first setting from affecting the existing situation.

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