Demographic exploration and discovery notes

Tim Riffe

April 24, 2019

Finishing up chapter 2 of Healy (2019)[https://socviz.co/gettingstarted.html#make-your-first-figure]

Let's take some time to make notes about what happens in this code chunk. 1) ggplot needs to be fed a data object — specifically, a `data.frame` that is in tidy format, i.e. kind of like *long* format data, meaning one observation per row, variables in columns. 2) we use `aes()` function to specify the *mapping*, which means that data are translated to graphical properties. So far, just x and y. 3) what geometric form shoud it render? use `geom_*`. A vanilla Preston curve is done with `geom_point()`

```
#p52 of print book
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------- tidyverse 1.2.1 --
```

```
## v tibble  2.1.1       v purrr   0.3.2
## v tidyr   0.8.3       v dplyr   0.8.0.1
## v readr   1.3.1       v stringr 1.4.0
## v tibble  2.1.1       v forcats 0.4.0
```

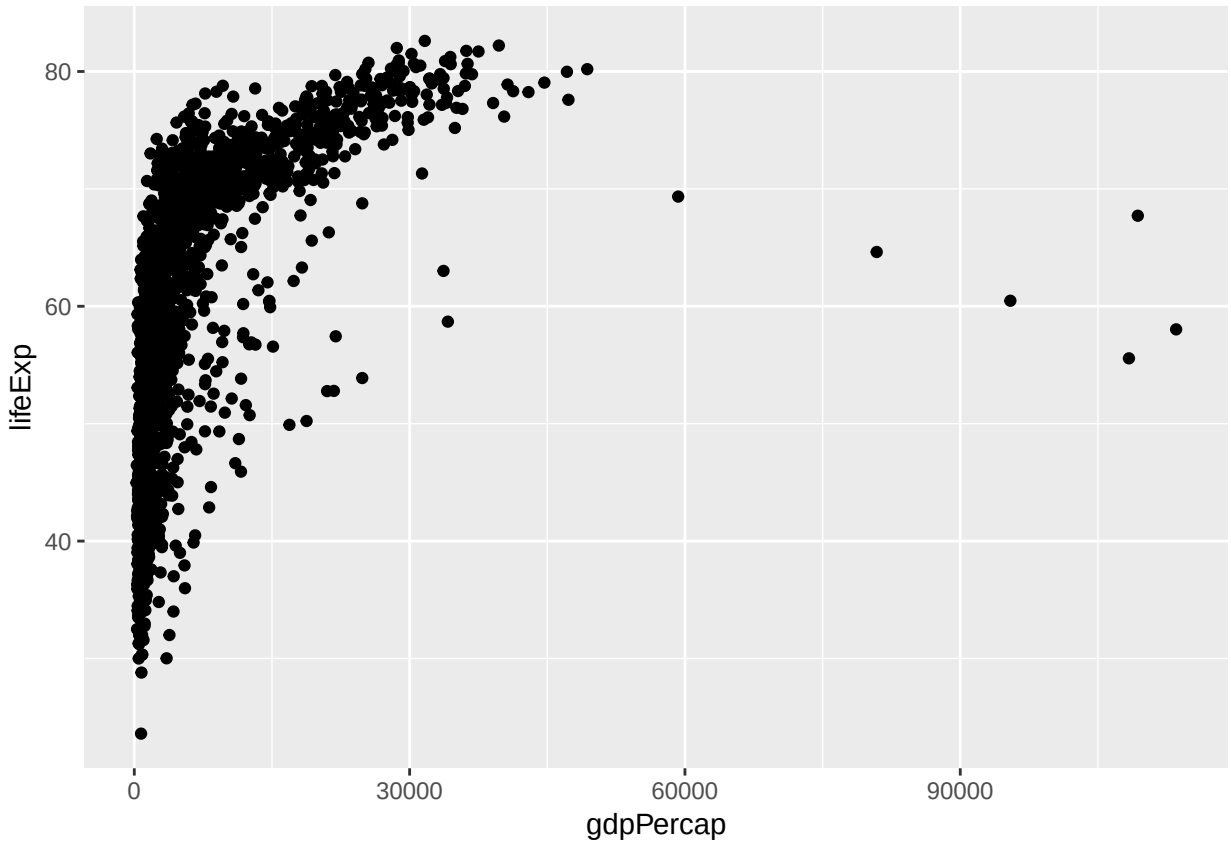
```
## -- Conflicts -------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(gapminder)

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 <- p + geom_point()
p
```



Code for chapter 3

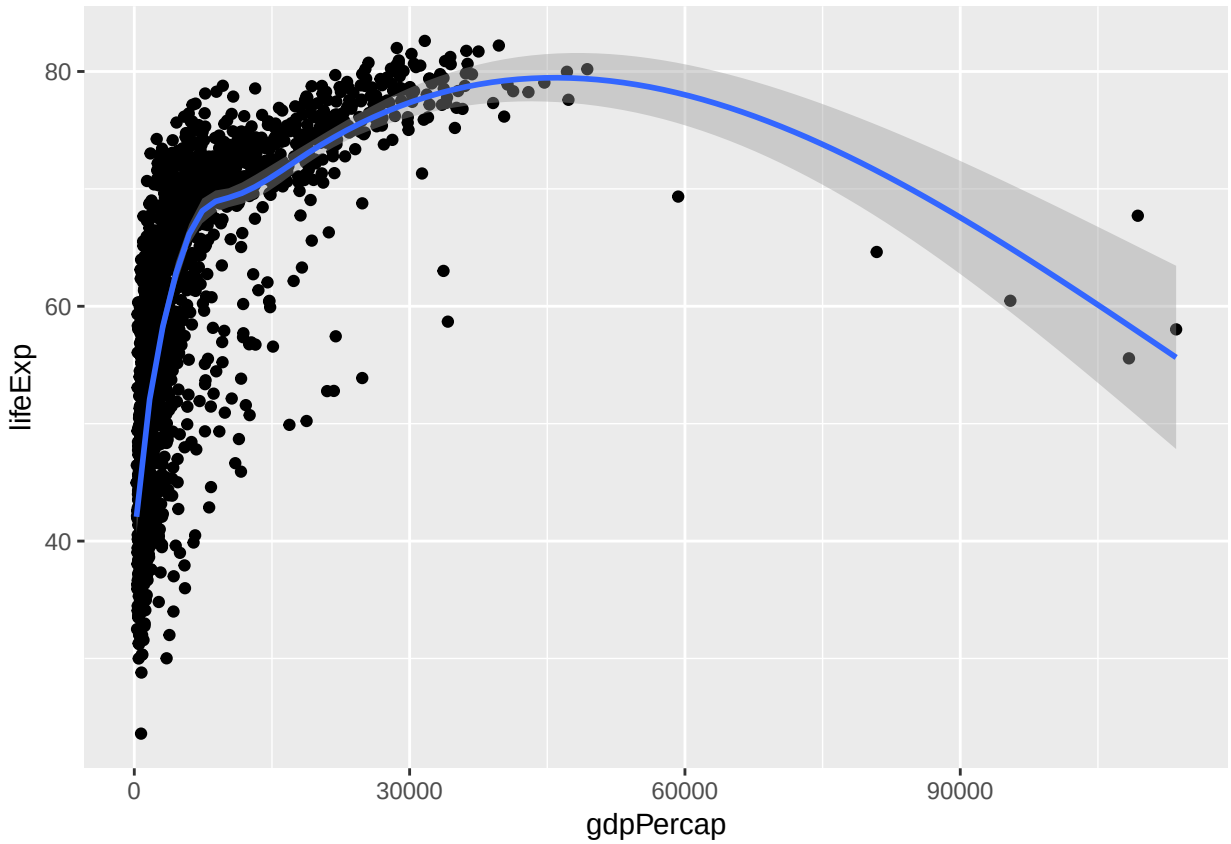
You can find it online here [https://socviz.co/makeplot.html#makeplot]

Seriously, type stuff in, no copy-pasting please!

The `geom_point()` addition step added a *layer* to the plot. We can add more layers, and they are literally just sitting on top of one another. So, like, we could add a smoother. Note when you execute this, it tells us that a GAM was used, and a bit more, and also selects some decent (but modifiable) graphical way of showing it.

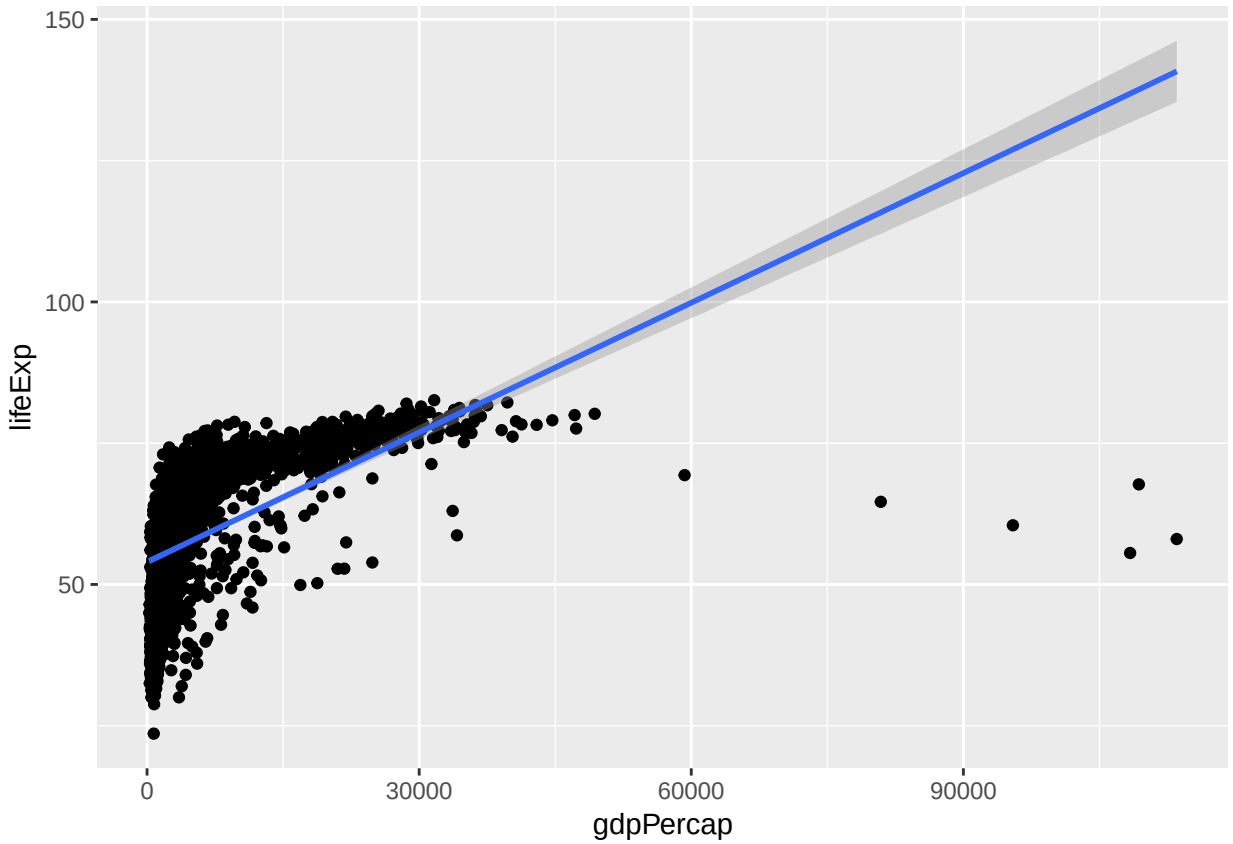
```
p + geom_smooth()
```

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



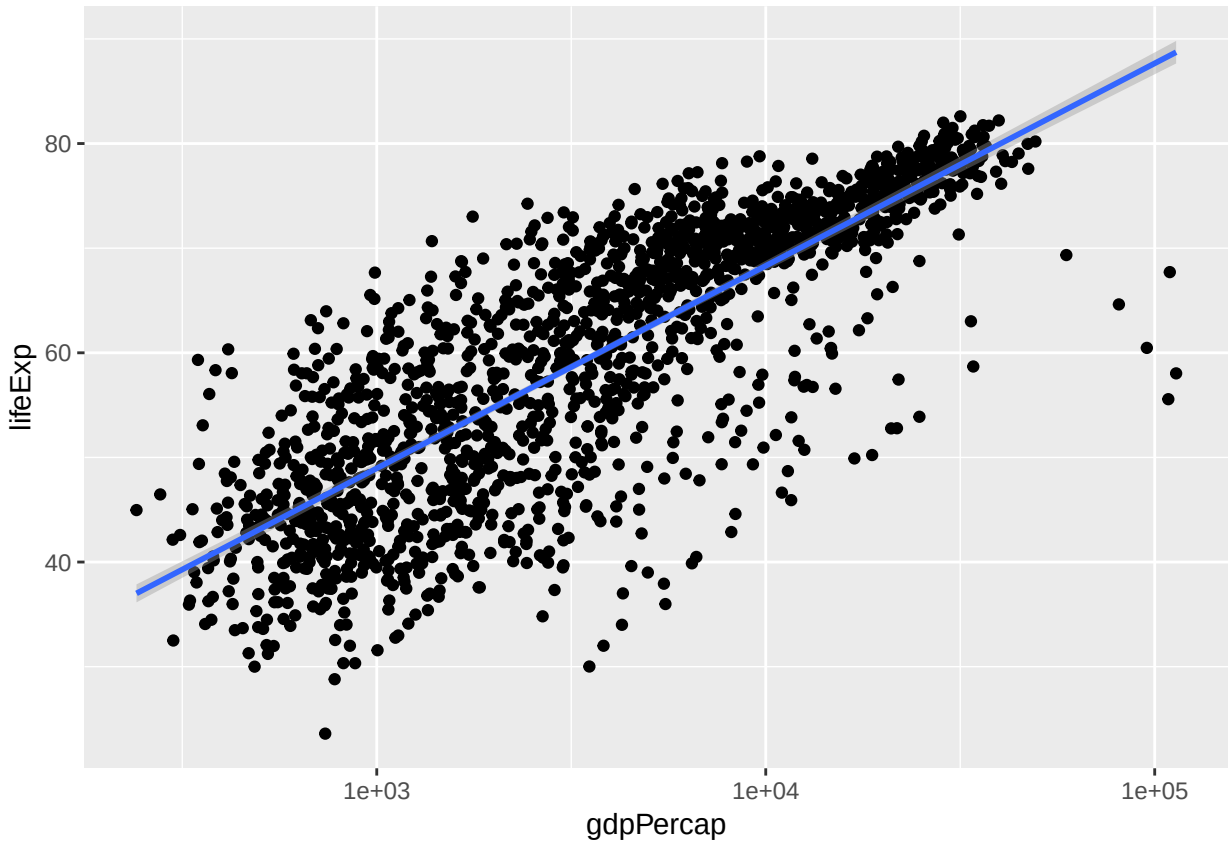
What if we want to change the smoother?

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




```
#But, it might make more sense if the axes were transformed.
```

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



At this point, we've seen mapping, we've seen geom, and we've seen an axis transformation. There are already many options that we've basically ignored. So let's mess with some to get a taste.

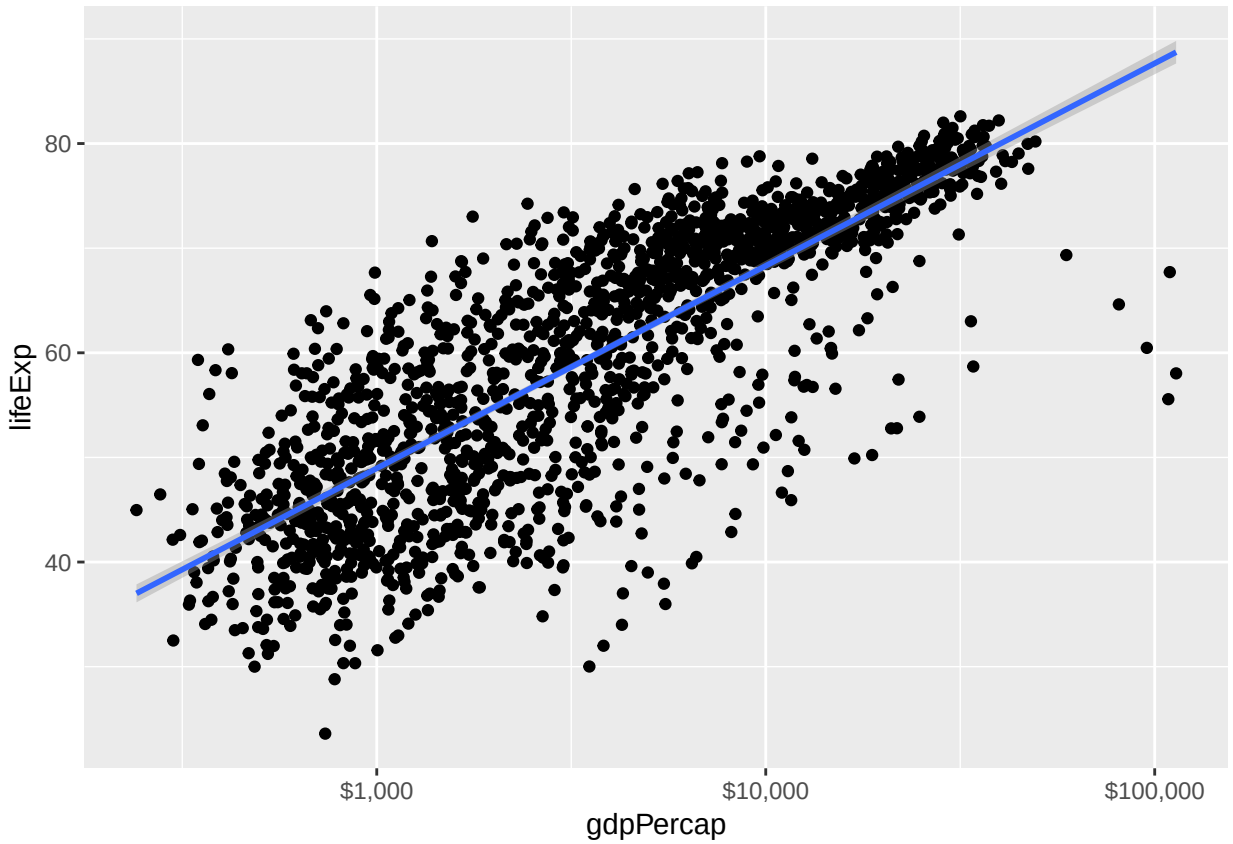
```
library(scales)
```

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

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

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

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



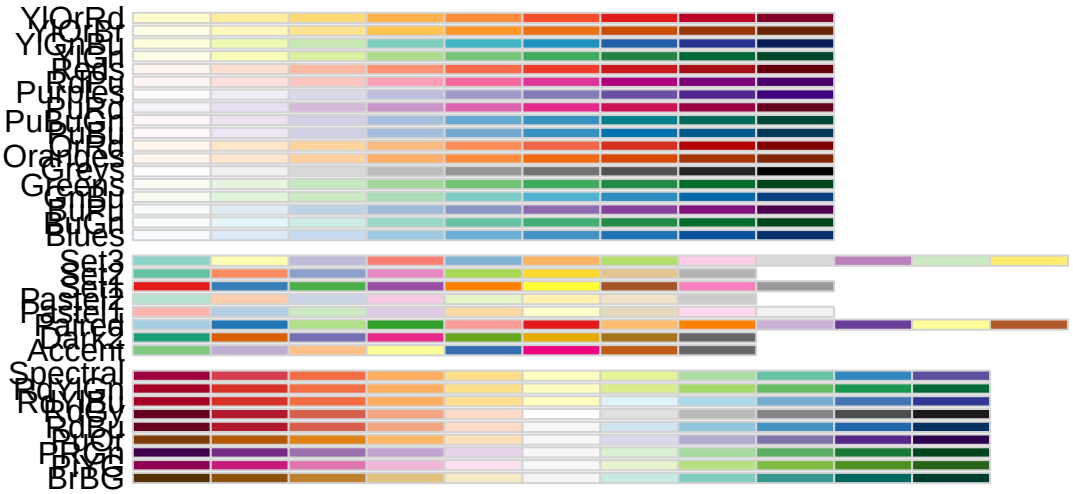
A nice trick to clean up your axes: the `scales` package takes care of some of that formatting. Here it swaps out (unitless) scientific notation for a more human readable dollar format.

Let's mess with colors. First, if you want to just tell the plot what color the points should be, do it in the geom function. Not in the mapping. Mapping, very literally translates variables in your data to graphical things. SO if you say `col="purple"` inside the `aes()` function, it will not do what you're expecting. Instead it creates a new column (temporarily) and maps it. To some other color!

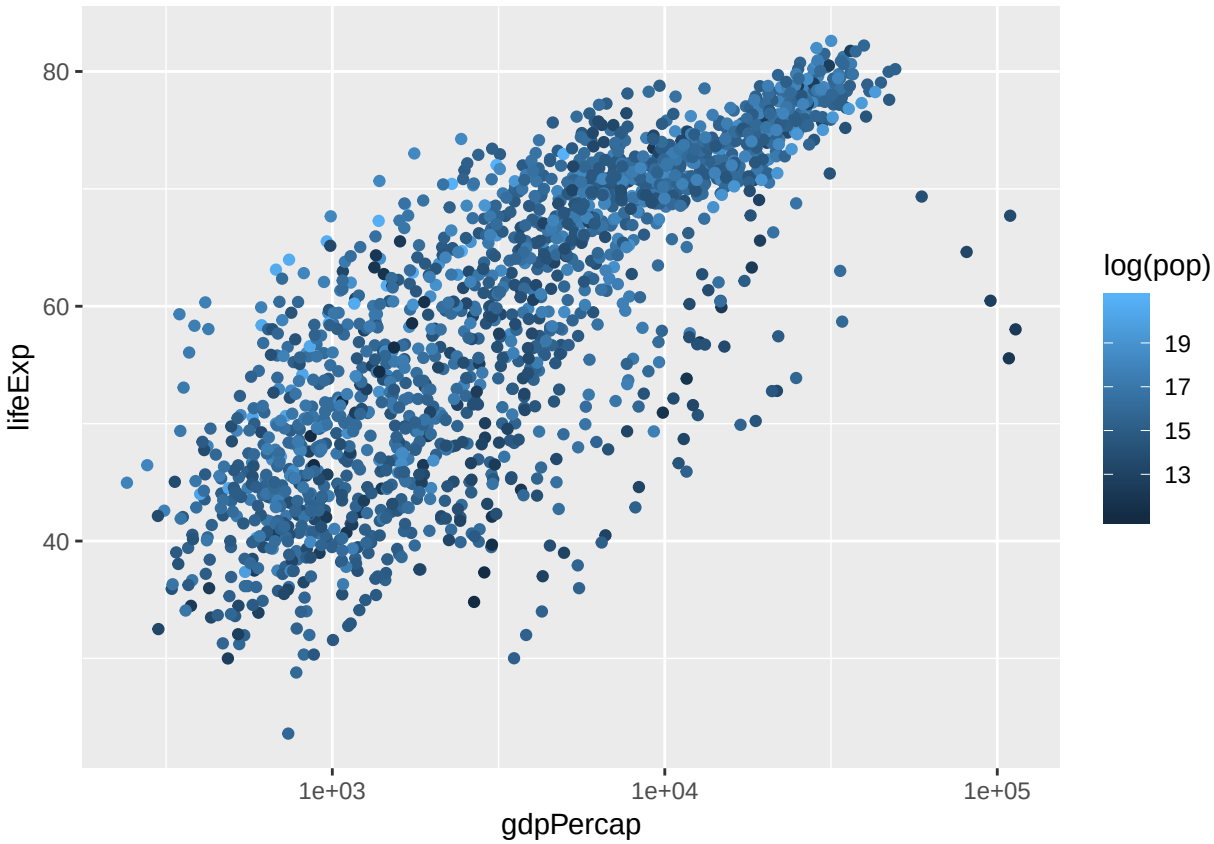
```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp, col = continent)) +
    geom_point() + scale_x_log10()
p
```

Instead, map actual variables to the colors. If we choose `continent` it will choose categorical colors. The reason it does this is because the column is in character/factor format. Factors are for categorical data, so it selects a categorical palette.

```
library(RColorBrewer)
display.brewer.all()
```




```
# let's try mapping a numeric column to color
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap, y = lifeExp, col = log(pop))) +
    geom_point() + scale_x_log10()
p
```



Here we mapped a numeric column to color, and since it's a value with a range, it chooses a continuous color ramp. By default blues.

We will start next session by narrating the more involved aspects of chapters 3 and 4.