006 - ggplot2 package for plots

EPIB 607

Sahir Rai Bhatnagar
Department of Epidemiology, Biostatistics, and Occupational Health
McGill University

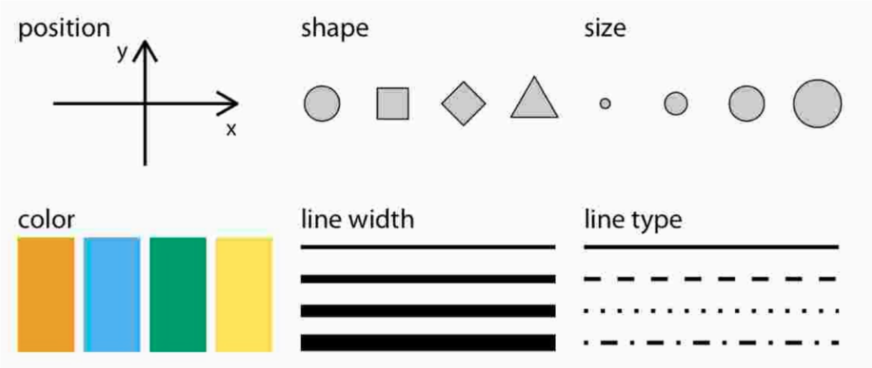
sahir.bhatnagar@mcgill.ca
https://sahirbhatnagar.com/EPIB607/ggplot2-package-for-plots.html

slides compiled on September 14, 2021

Objective

- Understand the how to map data onto aesthetics with ggplot2
- Use `ggplot2` core functions to produce a series of scatterplots

Commonly Used Aesthetics



How ggplot2 works

Tidy data

Mappings link data to things you see

Build your plots layer by layer

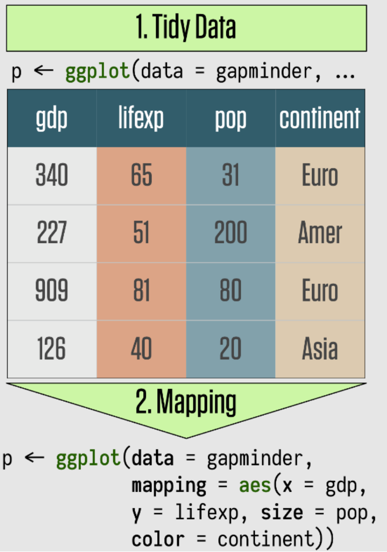
Mapping aesthetics vs setting them

Aesthetics can be mapped per geom

ggplot2 to make plots

- ggplot provides you with a set of tools to map data
  1. to visual elements on your plot
  2. to specify the kind of plot you want, and
  3. then subsquently to control the fine details of how it will be displayed.

Aesthetic mappings



1. Tidy Data

```
p ← ggplot(data = gapminder, ...
```

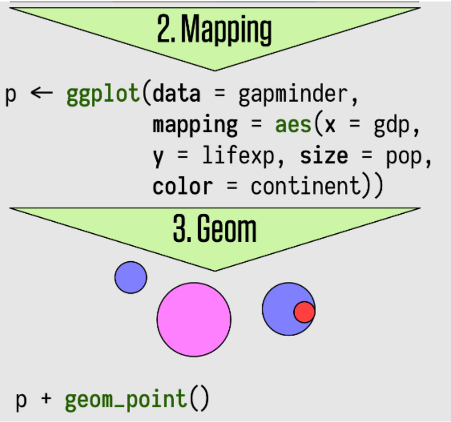
gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

2. Mapping

```
p ← ggplot(data = gapminder,
           mapping = aes(x = gdp,
           y = lifexp, size = pop,
           color = continent))
```

- The code you write specifies the connections between the variables in your data, and the colors, points, and shapes you see on the screen.
- In ggplot, these logical connections between your data and the plot elements are called *aesthetic mappings* or just *aesthetics*.
- You begin every plot by telling the `ggplot()` function what your data is, and then how the variables in this data logically map onto the plot's aesthetics.

Geometry

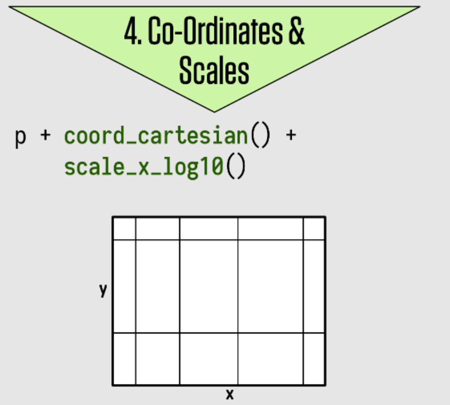


```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdp,
            y = lifexp, size = pop,
            color = continent))
```

```
p + geom_point()
```

- Then you take the result and say what general sort of plot you want, such as a scatterplot, a boxplot, or a bar chart. In `ggplot`, the overall type of plot is called a `geom`.
- Each geom has a function that creates it. For example, `geom_point()` makes scatterplots, `geom_bar()` makes barplots, `geom_boxplot()` makes boxplots, and so on.
- You combine these two pieces, the `ggplot()` object and the `geom`, by literally adding them together in an expression, using the “+” symbol.

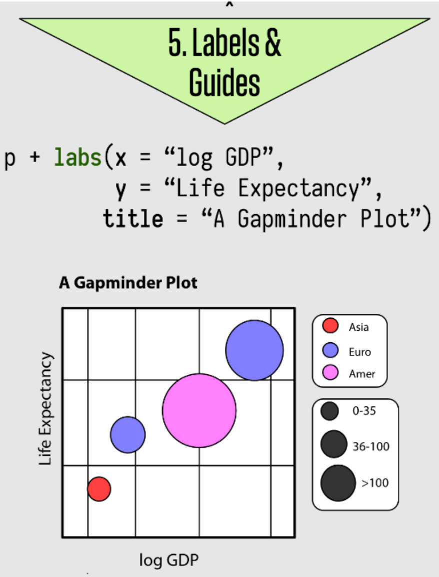
Customization



```
p + coord_cartesian() +
    scale_x_log10()
```

- At this point, ggplot will have enough information to be able to draw a plot for you. ggplot will use a set of defaults that try to be sensible about what gets drawn.
- But more often, you will want to specify exactly what you want, including information about the scales, the labels of legends and axes, and other guides that help people to read the plot.
- Each component has it own function, you provide arguments to it specifying what to do, and you literally add it to the sequence of instructions.
- In this way you systematically build your plot piece by piece.

Customization



```
p + labs(x = “log GDP”,
         y = “Life Expectancy”,
     title = “A Gapminder Plot”)
```

How ggplot2 works

Tidy data

Mappings link data to things you see

Build your plots layer by layer

Mapping aesthetics vs setting them

Aesthetics can be mapped per geom

Gapminder dataset in format 1

```r
library(gapminder)
gapminder::gapminder %>%
  dplyr::select(country, year, lifeExp) %>%
  tidyr::pivot_wider(names_from = "year", values_from = "lifeExp") %>%
  dplyr::slice_head(n = 10) %>%
  knitr::kable(caption = "Life Expectancy data from gapminder dataset for
  the first 10 countries.", digits = 0, booktabs=TRUE)
```

Table: Life Expectancy data from gapminder dataset for the first 10 countries.

country	1952	1957	1962	1967	1972	1977	1982	1987	1992	1997	2002	2007
Afghanistan	29	30	32	34	36	38	40	41	42	42	42	44
Albania	55	59	65	66	68	69	70	72	72	73	76	76
Algeria	43	46	48	51	55	58	61	66	68	69	71	72
Angola	30	32	34	36	38	39	40	40	41	41	41	43
Argentina	62	64	65	66	67	68	70	71	72	73	74	75
Australia	69	70	71	71	72	73	75	76	78	79	80	81
Austria	67	67	70	70	71	72	73	75	76	78	79	80
Bahrain	51	54	57	60	63	66	69	71	73	74	75	76
Bangladesh	37	39	41	43	45	47	50	53	56	59	62	64
Belgium	68	69	70	71	71	73	74	75	76	78	78	79

Gapminder dataset in format 2

```r
gapminder::gapminder %>%
        dplyr::select(country, year, lifeExp) %>%
        dplyr::filter(country %in% c("Afghanistan","Albania")) %>%
        knitr::kable(caption = "gapminder dataset for Afghanistan and Albania", digits = 0, booktabs=TRUE)
```

Table: gapminder dataset for Afghanistan and Albania

country	year	lifeExp
Afghanistan	1952	29
Afghanistan	1957	30
Afghanistan	1962	32
Afghanistan	1967	34
Afghanistan	1972	36
Afghanistan	1977	38
Afghanistan	1982	40
Afghanistan	1987	41
Afghanistan	1992	42
Afghanistan	1997	42
Afghanistan	2002	42
Afghanistan	2007	44
Albania	1952	55
Albania	1957	59
Albania	1962	65
Albania	1967	66
Albania	1972	68
Albania	1977	69
Albania	1982	70
Albania	1987	72
Albania	1992	72
Albania	1997	73
Albania	2002	76
Albania	2007	76

Tidy data is usually not compact

- If you compare the two previous tables, it is clear that a tidy table does not present data in its most compact form.
- In fact, it is usually not how you would choose to present your data if you wanted to just show people the numbers.
- Neither is untidy data “messy” or the “wrong” way to lay out data in some generic sense.
- It’s just that, even if its long-form shape makes tables larger, tidy data is much more straightforward to work with when it comes to specifying the mappings that you need to coherently describe plots.

How ggplot2 works

Tidy data

Mappings link data to things you see

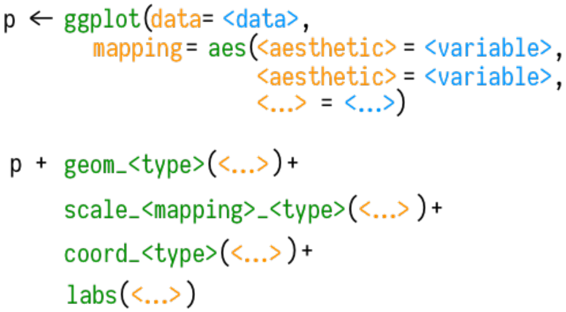
Build your plots layer by layer

Mapping aesthetics vs setting them

Aesthetics can be mapped per geom

Mappings

- It’s useful to think of a recipe or template that we start from each time we want to make a plot:



```
p ← ggplot(data= <data>,
             mapping= aes(<aesthetic> = <variable>,
                          <aesthetic> = <variable>,
                          <...> = <...>)

p + geom_<type>(<...>)+
    scale_<mapping>_<type>(<...> )+
    coord_<type>(<...> )+
    labs(<...> )
```

Gapminder example - the data

- Let’s say we want to plot Life Expectancy against per capita GDP for all country-years in the data.
- We’ll do this by creating an object that has some of the necessary information in it, and build it up from there.
- First, we must tell the `ggplot()` function what data we are using:

```
library(ggplot2)
library(cowplot)
ggplot2::theme_set(cowplot::theme_cowplot())
p <- ggplot(data = gapminder)
```

Gapminder example - mappings

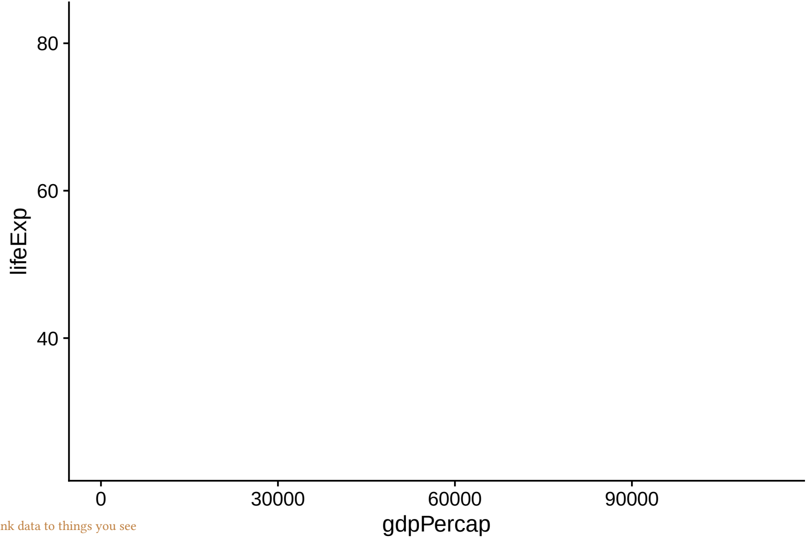
- At this point ggplot knows our data, but not what the mapping.
- That is, we need to tell it which variables in the data should be represented by which visual elements in the plot.
- It also doesn't know what sort of plot we want.
- In ggplot, mappings are specified using the `aes()` function. Like this:

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

Gapminder example

- What happens if we just type `p` at the console at this point and hit return?

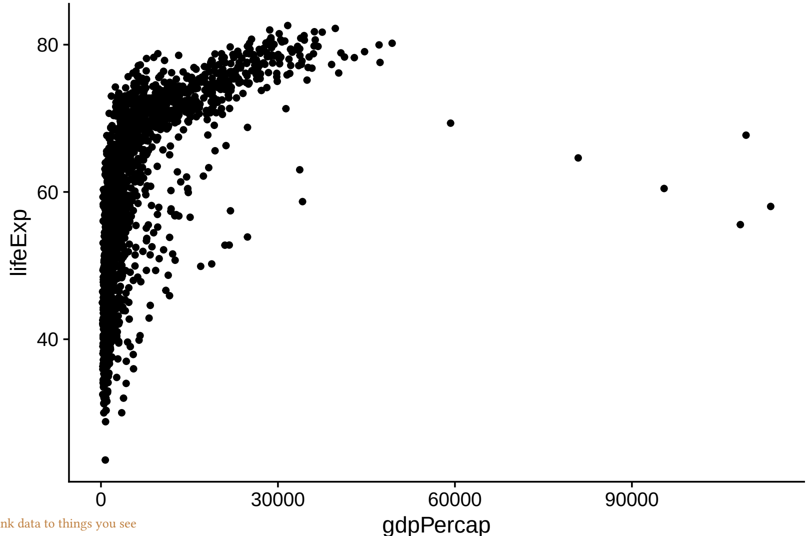
```
p
```



Gapminder example - scatterplot

- We need to add a `layer` to the plot. This means picking a `geom_` function.

```
p + geom_point()
```



How ggplot2 works

Tidy data

Mappings link data to things you see

Build your plots layer by layer

Mapping aesthetics vs setting them

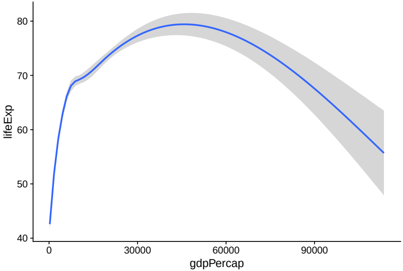
Aesthetics can be mapped per geom

Five steps to build a plot

1. Tell the `ggplot` function what our data is. The `data = …` step.
2. Tell `ggplot` *what* relationships we want to see. The `mapping = aes(…)` step. 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, i.e. choose a geom.
4. Layer on geoms as needed, by adding them to the `p` object one at a time.
5. Use some additional functions to adjust scales, labels, tick marks, titles using the `scale_`, family, `labs()` and `guides()` functions. We'll learn more about some of these functions shortly.

Add a GAM smoother

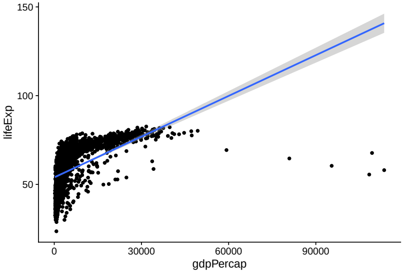
```
p <- ggplot(data = gapminder,
            mapping = aes(x = gdpPercap,
                          y=lifeExp))
p + geom_smooth()

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



- The coordinate system used in plots is most often cartesian, i.e., a plane defined by an x axis and a y axis. This is what ggplot assumes, unless you tell it otherwise.
- Usually in R, functions cannot simply be added to objects. Rather, they take objects as inputs and produce objects as outputs.
- But the objects created by `ggplot()` are special. This makes it easier to assemble plots one piece at a time, and to inspect how they look at every step.

Add a linear smoother with data points

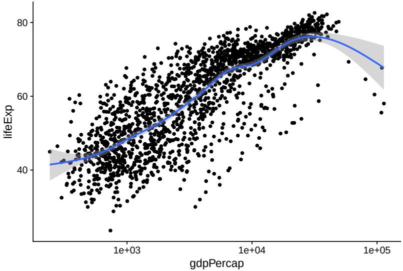
```
p <- ggplot(data = gapminder,
mapping = aes(x = gdpPercap,
y=lifeExp))
p + geom_point() + geom_smooth(method = "lm")

## `geom_smooth()` using formula 'y ~ x'
```



- We did not have to tell `geom_point()` or `geom_smooth()` where their data was coming from, or what mappings they should use.
- They *inherit* this information from the original `p` object.
- As we'll see later, it's possible to give geoms separate instructions that they will follow instead. But in the absence of any other information, the geoms will look for the instructions it needs in the `ggplot()` function, or the object created by it.

Change scales

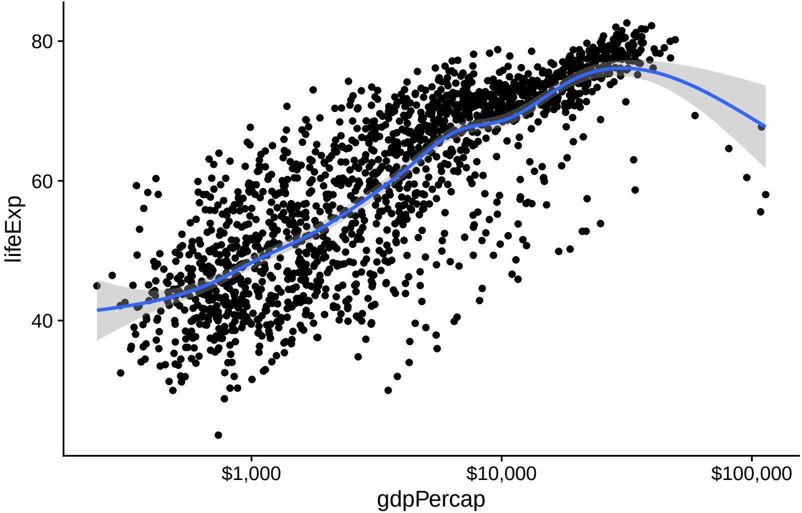
```
p <- ggplot(data = gapminder,
        mapping = aes(x = gdpPercap,
                y=lifeExp))
p + geom_point() +
        geom_smooth(method = "gam") +
        scale_x_log10()

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



- The data is quite bunched up against the left side.
- Gross Domestic Product per capita is not normally distributed across our country years.
- The x-axis scale would probably look better if it were transformed from a linear scale to a log scale. For this we can use a function called `scale_x_log10()`.
- As you might expect this function scales the x-axis of a plot to a log 10 basis.

Formatting labels with the `scales` package

```
library(scales)
p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y=lifeExp))
p + geom_point() +
        geom_smooth(method = "gam") +
        scale_x_log10(labels = scales::dollar)
```



How ggplot2 works

Tidy data

Mappings link data to things you see

Build your plots layer by layer

Mapping aesthetics vs setting them

Aesthetics can be mapped per geom

Color aesthetic

- An *aesthetic mapping* specifies that a variable will be expressed by one of the available visual elements, such as size, or color, or shape, and so on. As we've seen, we map variables to aesthetics like this:

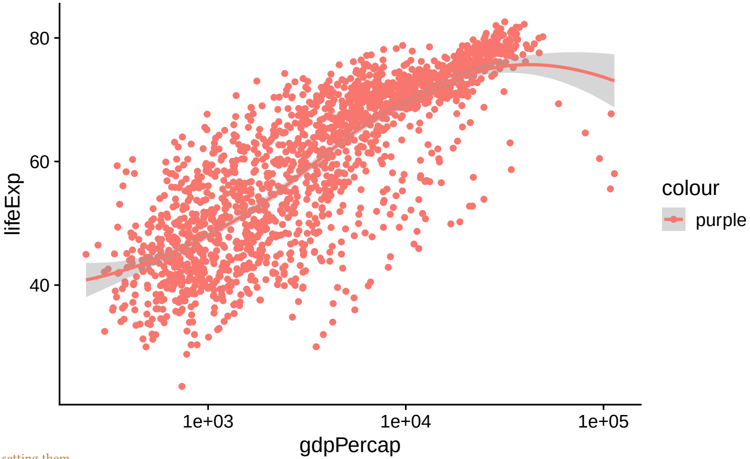
```r
p <-  ggplot(data = gapminder,
                mapping = aes(x = gdpPercap,
                                y = lifeExp,
                                color = continent))
```

- This code does *not* give a direct instruction like “color the points purple”.
- Instead it says, “the variable continent will map onto the color aesthetic”.

Misunderstanding of an aesthetic

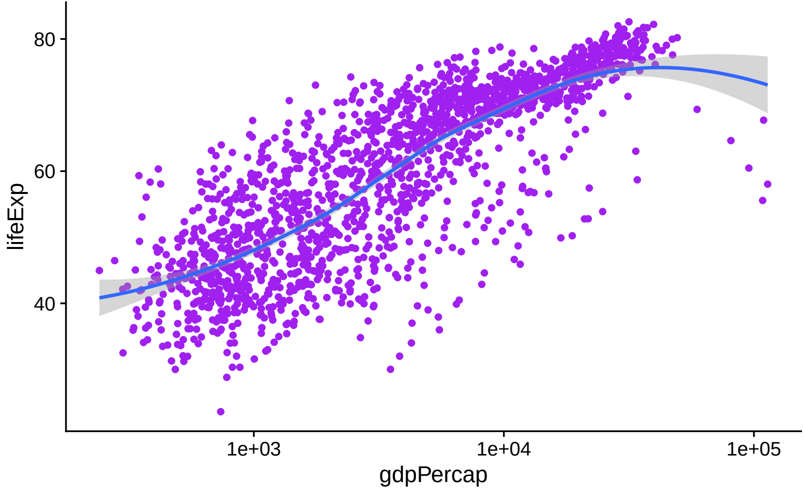
- If we want to turn all the points in the figure purple, we do *not* do it through the mapping function. Look at what happens when we try:

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



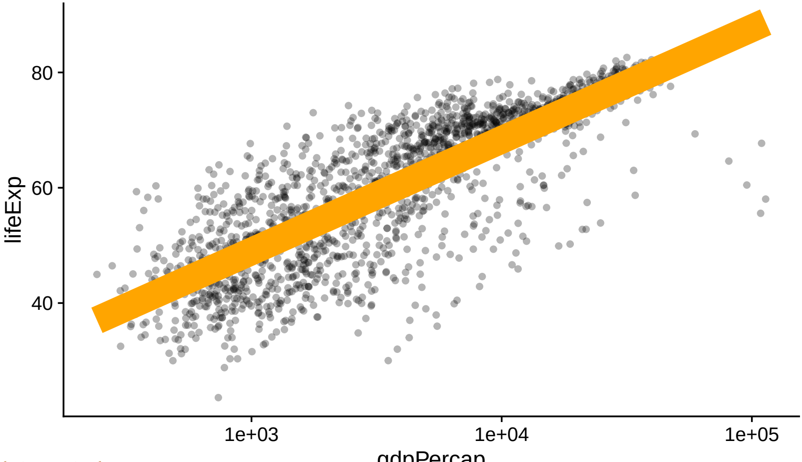
The correct way of specifying point colors

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



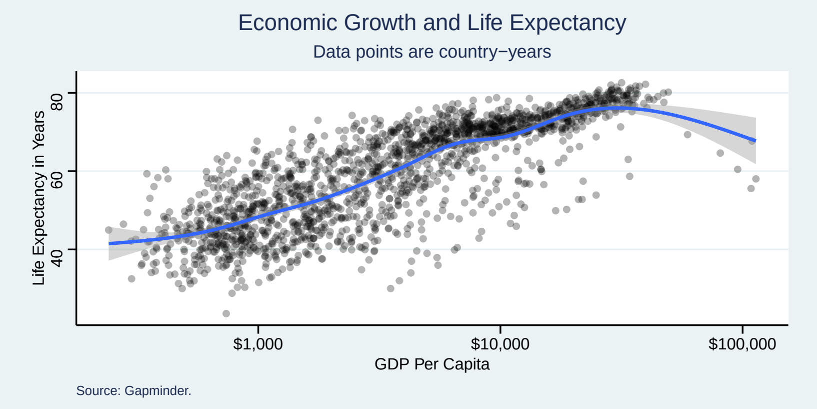
geom_ functions can take many other arguments that will affect how the plot looks

```
p <- ggplot(data = gapminder,
                mapping = aes(x = gdpPercap,
                                        y = lifeExp))
p + geom_point(alpha = 0.3) +
geom_smooth(color = "orange", se = FALSE, size = 8, method = "lm") +
scale_x_log10()
```



Add lables and change themes

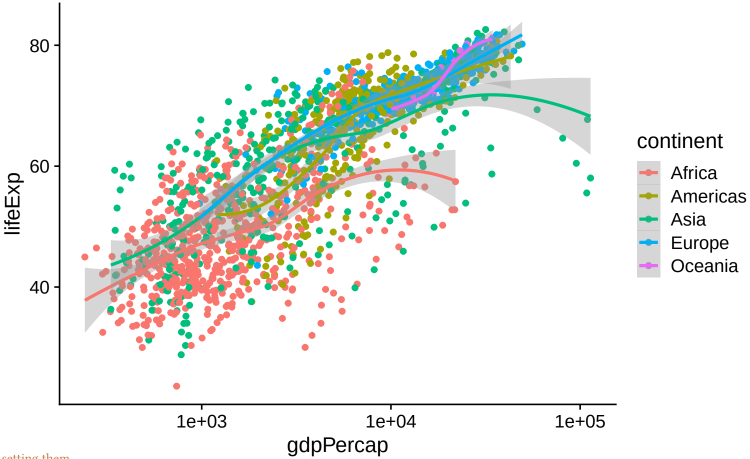
```
p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y=lifeExp))
p + geom_point(alpha = 0.3) + geom_smooth(method = "gam") +
    scale_x_log10(labels = scales::dollar) +
    ggthemes::theme_stata() +
    labs(x = "GDP Per Capita", y = "Life Expectancy in Years",
        title = "Economic Growth and Life Expectancy",
        subtitle = "Data points are country-years",
        caption = "Source: Gapminder.")
```



Smooth lines for each continent

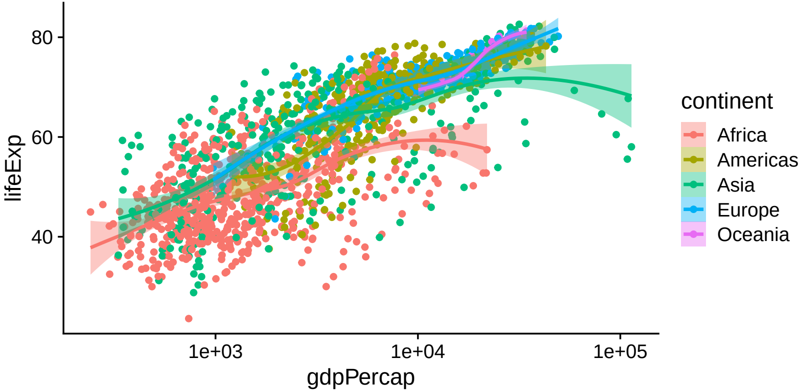
- Along with x and y, the color aesthetic mapping is set in the call to ggplot that we used to create the p object

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



Standard errors bars with matching color

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



How ggplot2 works

Tidy data

Mappings link data to things you see

Build your plots layer by layer

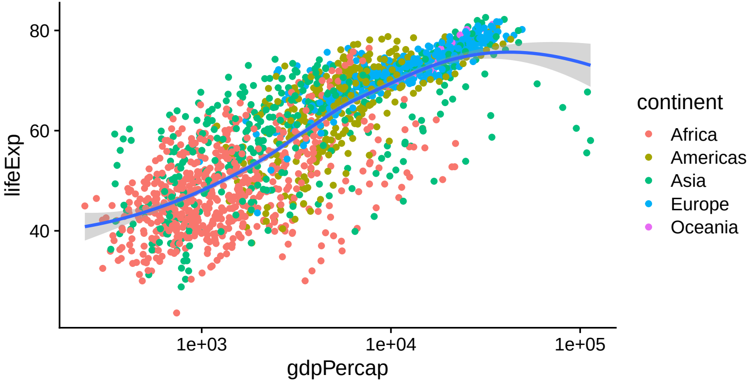
Mapping aesthetics vs setting them

Aesthetics can be mapped per geom

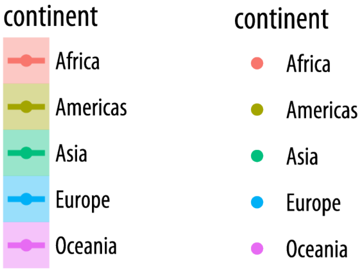
Specify different mapping for each geom

- Perhaps five separate smoothers is too many, and we just want one line. But we still would like to have the points color-coded by continent. By default, geoms inherit their mappings from the `ggplot()` function
- We can change this by mapping the aesthetics we want only the `geom_` functions that we want them to apply to.

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

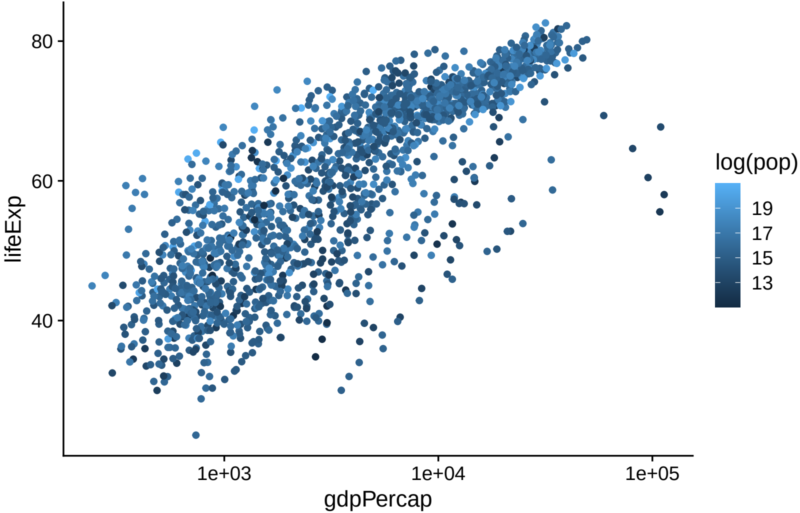


A closer look at the legends



Mapping a continuous variable to color

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



Session Info

```
R version 4.0.2 (2020-06-22)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Pop!_OS 20.10

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.10.so

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods  
[8] base     

other attached packages:
 [1] scales_1.1.1     cowplot_1.1.0    DT_0.16          kableExtra_1.2.1
 [5] socviz_1.2       gapminder_0.3.0  here_0.1         NCStats_0.4.7   
 [9] FSA_0.8.30       forcats_0.5.1    stringr_1.4.0    dplyr_1.0.7     
[13] purrr_0.3.4      readr_1.4.0      tidyr_1.1.3      tibble_3.1.3    
[17] ggplot2_3.3.5    tidyverse_1.3.0  knitr_1.33      

loaded via a namespace (and not attached):
 [1] nlme_3.1-149       fs_1.5.0           lubridate_1.7.9    webshot_0.5.2     
 [5] httr_1.4.2         rprojroot_2.0.2    backports_1.2.1    utf8_1.2.2        
 [9] R6_2.5.1           mgcv_1.8-33        DBI_1.1.1          colorspace_2.0-2  
[13] withr_2.4.2        tidyselect_1.1.1   gridExtra_2.3      leaflet_2.0.3     
[17] curl_4.3.2         compiler_4.0.2     cli_3.0.1          rvest_1.0.0       
[21] pacman_0.5.1       xml2_1.3.2         ggdendro_0.1.22    labeling_0.4.2    
[25] mosaicCore_0.8.0   digest_0.6.27      ggformula_0.9.4    foreign_0.8-80    
[29] rmarkdown_2.9.7    rio_0.5.16         pkgconfig_2.0.3    htmltools_0.5.2   
[33] highr_0.9          dbplyr_1.4.4       fastmap_1.1.0      ggthemes_4.2.4    
[37] htmlwidgets_1.5.3  rlang_0.4.11       readxl_1.3.1       rstudioapi_0.13   
[41] farver_2.1.0       generics_0.1.0     jsonlite_1.7.2     crosstalk_1.1.1   
[45] zip_2.2.0          car_3.0-9          magrittr_2.0.1     mosaicData_0.20.1 
[49] Matrix_1.2-18      Rcpp_1.0.7         munsell_0.5.0      fansi_0.5.0       
[53] abind_1.4-5        lifecycle_1.0.0    stringi_1.7.3      carData_3.0-4     
[57] MASS_7.3-53        plyr_1.8.6         ggstance_0.3.4     grid_4.0.2        
[61] blob_1.2.1         ggrepel_0.8.2      crayon_1.4.1       lattice_0.20-41   
[65] haven_2.3.1        splines_4.0.2      hms_1.0.0          pillar_1.6.2      
[69] reprex_0.3.0       glue_1.4.2         evaluate_0.14      data.table_1.14.0 
[73] modelr_0.1.8       vctrs_0.3.8        tweenr_1.0.1       cellranger_1.1.0  
[77] gtable_0.3.0       polyclip_1.10-0    assertthat_0.2.1   TeachingDemos_2.12
[81] xfun_0.25          ggforce_0.3.2      openxlsx_4.1.5     broom_0.7.2       
```