Working in the tidyverse

Modern Research Methods

1/31/2020

A couple new functions

count()

`count()` is a useful shortcut for `group_by() %>% summarize(num = n())`. This code:

```
gapminder %>%
  group_by(country) %>%
  summarize(num_countries = n())
```

Does the same as this:

```
gapminder %>%
  count(country)
```

glimpse()

Glimpse is useful for getting the "big picture" view of your data frame.

```
glimpse(gapminder)
```

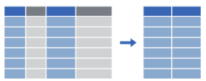
```
## Observations: 1,704
## Variables: 6
## $ country    <fct> Afghanistan, Afghanistan, Afghanistan, Afghanistan, Afghan…
## $ continent  <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia…
## $ year       <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997…
## $ life_exp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40…
## $ pop        <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, …
## $ gdp_percap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134…
```

summary() does something similiar:

```
summary(gapminder)
```

```
##         country        continent        year         life_exp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
##  Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
##  Australia  :  12                  Max.   :2007   Max.   :82.60
##  (Other)    :1632
```

And some old functions



Extract variables with **select()**

Extract cases with **filter()**

Arrange cases, with **arrange()**.

Make tables of summaries with **summarise()**.

Make new variables, with **mutate()**.

Some points of confusion

%in% vs. %>%

Even though these symbols are made up of three characters, you should think of them as a single symbol.

Despite their apparent similarity, these functions aren't really related to each other.

`%in%` checks whether something is a member of a set.

```
4 %in% c(1,2,3,4)
```

```
## [1] TRUE
```

```
5 %in% c(1,2,3,4)
```

```
## [1] FALSE
```

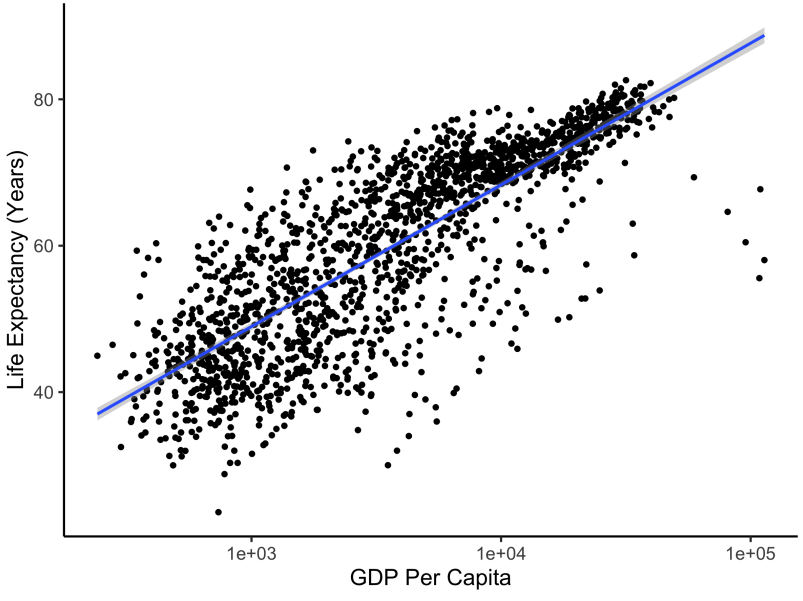
%>% ("the pipe") sends the output of one function to another function.

```
gapminder %>%
group_by(country) %>%
summarize(num_countries = n())
```

The scope of aes()

Remember this plot?

```
ggplot(data = gapminder, mapping = aes(x = gdp_percap,
                                       y = life_exp)) +
                    geom_point() +
                geom_smooth(method = "lm") +
                     scale_x_log10() +
            ylab("Life Expectancy (Years)") +
                 xlab("GDP Per Capita") +
              theme_classic(base_size = 16)
```

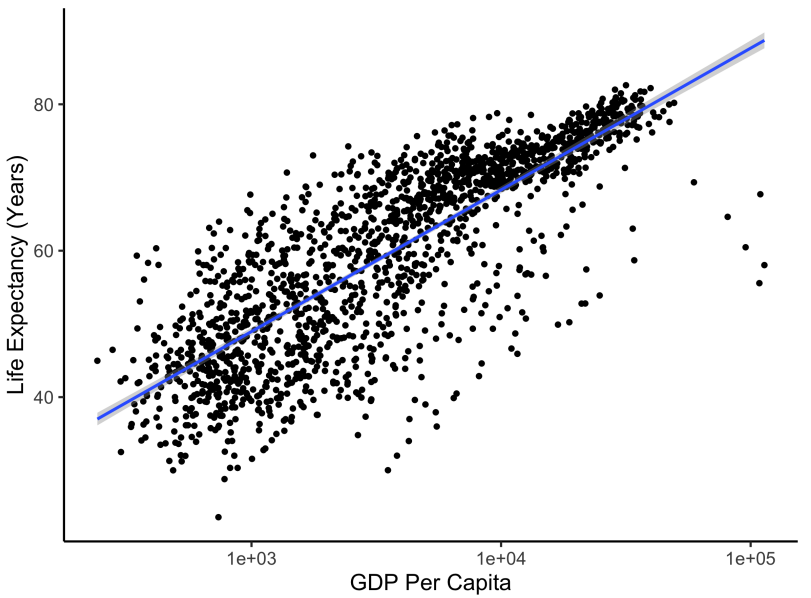


Another way to write this is by putting the aesthetics in the geom functions themselves

```
ggplot(data = gapminder) +
        geom_point(mapping = aes(x = gdp_percap,y = life_exp)) +
geom_smooth(mapping = aes(x = gdp_percap,  y = life_exp), method = "lm") +
                    scale_x_log10() +
                ylab("Life Expectancy (Years)") +
                    xlab("GDP Per Capita") +
                  theme_classic(base_size = 16)
```

But notice because `geom_point()` and `geom_smooth()` require both `x` and `y` aesthetics we have to include the mappings in both.

Mappings put in the `ggplot()` function apply to all geoms.



A common error: Forgetting a pipe

```
gapminder %>%
group_by(country)
summarize(num_countries = n()) %>%
mutate(num_countries_round = round(num_countries))
```

```
Error: n() should only be called in a data context
Callrlang::last_error()to see a backtrace.
```

Error will depend on what exactly you're trying to do. But check this furst if you get an error you don't understand!

A common error: Forgetting the +

```
ggplot(data = gapminder, mapping = aes(x = gdp_percap,
                                       y = life_exp)) +
                      geom_point() +
                   geom_smooth(method = "lm")
                        scale_x_log10() +
                  ylab("Life Expectancy (Years)") +
                      xlab("GDP Per Capita") +
                    theme_classic(base_size = 16)
```

```
Error: Cannot add ggproto objects together. Did you
  forget to add this object to a ggplot object?
```

A common error: Forgetting to load packages

```
cool_data_frame <- read_csv("data/cool_data_frame.csv")
```

```
Error: object 'read_csv' not found
```

Solves the problem:

```
library(tidyverse)
cool_data_frame <- read_csv("data/cool_data_frame.csv")
```

You have to load packages before you can use their functions!

Notes on style

Style

- Why does style matter?
- Style doesn't matter to the computer, but it does matter to **humans** who produce, intepret and modify code.
- Having a code specific, consistent code style makes your own code easier to understand and debug, and it helps others do the same.
- In this class, **variable names** in data frames should be all lower case and descriptive. Separate multiple words with an underscore (_).
  - BAD: `NEWVARIABLE`, `thing`, `LIFEexpectancy`, `Time`
  - GOOD: `num_countries`, `age_years`, `life_expectancy`, `log_reaction_time_seconds`
- In this class, if you can use the pipe, **always use the pipe** (unless there's only a single function)

Line breaks

In the tidyverse, you should think of each **line** as doing **one** thing.

Like instructions in a recipe:

Data frame goes on own line, then each function (verb) on its own line after that (indent after first).

GREAT:

```
gapminder %>%
  group_by(country) %>%
  summarize(num_countries = n()) %>%
  mutate(num_countries_round = round(num_countries))
```

BAD:

```
gapminder %>% group_by(country)  %>% summarize(num_countries = n()) %>%
mutate(num_countries_round = round(num_countries))
```

Same for ggplot. Imagine your plot is a house and you're building it brick by brick.

Each "brick" of the plot goes on its own line.

Each layer of your plot goes on its own line.

GREAT:

```
ggplot(data = gapminder, mapping = aes(x = gdp_percap,
                                       y = life_exp)) +
                      geom_point() +
                 geom_smooth(method = "lm") +
                       scale_x_log10() +
                ylab("Life Expectancy (Years)") +
                    xlab("GDP Per Capita") +
                  theme_classic(base_size = 16)
```

BAD:

```
ggplot(data = gapminder, mapping = aes(x = gdp_percap,
                                       y = life_exp)) + geom_point() +
geom_smooth(method = "lm") + scale_x_log10() + ylab("Life Expectancy (Years)"
           xlab("GDP Per Capita") +theme_classic(base_size = 16)
```

Style for knitting

- No need to use `print()` function in .Rmd - will print output automatically.
- Make sure you look at your .html after you knit. Does it look as you expected? If not, go back to .Rmd.
- You can change size of plot output in .Rmd by specificy `fig.width` and `fig.height` in the relevant R chunk. In general, aim for the "plot" plot of your plot (i.e. excluding the legend) to be roughly square (or slightly wider than square).

```
{r CHUNKNAME, fig.width = 4.5, fig.height = 4}
```

"Literate Programming"

Plain text mixed with code.

````
## a

The following code selects all rows where name is "Garrett".

```{r}
filter(babynames, name == "Garrett")
```

## b
```{r}
filter(babynames, name == "Garrett")
```
````

a

The following code selects all rows where name is “Garrett”.

```
filter(babynames, name == "Garrett")
```

```
## # A tibble: 177 x 5
##    year sex   name        n      prop
##   <dbl> <chr> <chr>   <int>     <dbl>
## 1  1880 M     Garrett    13 0.000110
## 2  1881 M     Garrett     7 0.0000646
## 3  1882 M     Garrett    15 0.000123
## 4  1883 M     Garrett    13 0.000116
## 5  1884 M     Garrett    15 0.000122
## 6  1885 M     Garrett     9 0.0000776
## 7  1886 M     Garrett    16 0.000134
## 8  1887 M     Garrett    14 0.000128
## 9  1888 M     Garrett    10 0.0000770
## 10  1889 M     Garrett    16 0.000134
## # … with 167 more rows
```

b

```
filter(babynames, name == "Garrett")
```