Session 4 notes

Tim Riffe

7/29/2021

ggplot2 basics

```
# install.packages("gapminder")
library(gapminder)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1
```

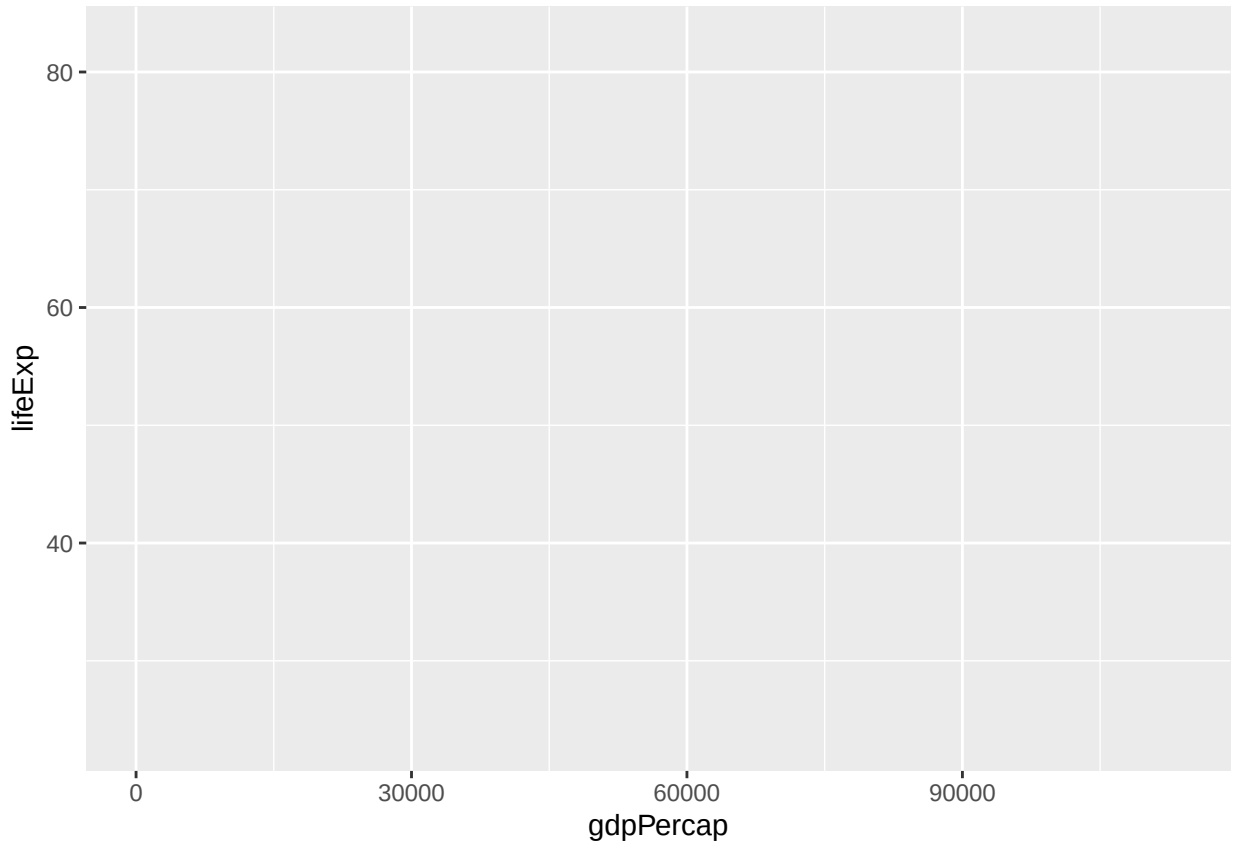
```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
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
```

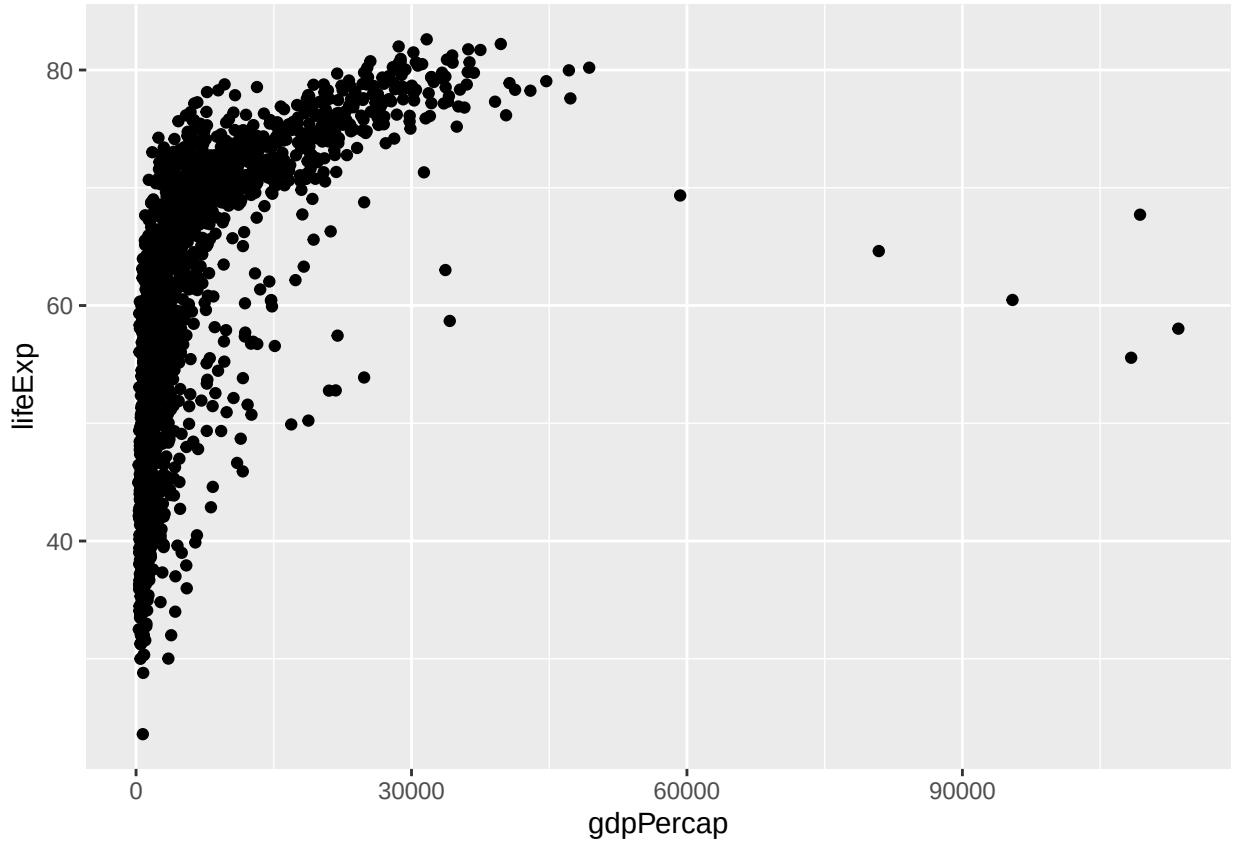
The fundamentals, `ggplot()` is necessary but not sufficient to build the plot. here we declare, at minimum, the coordinate mapping. Executing this opens a blank plot area, which is however aware of the data dimensions.

```
gapminder %>%
  ggplot(mapping = aes(x = gdpPercap, y = lifeExp))
```



To actually *draw* something (sorry, map something) to the plot, we need to specify a geometric element, using `geom_*`

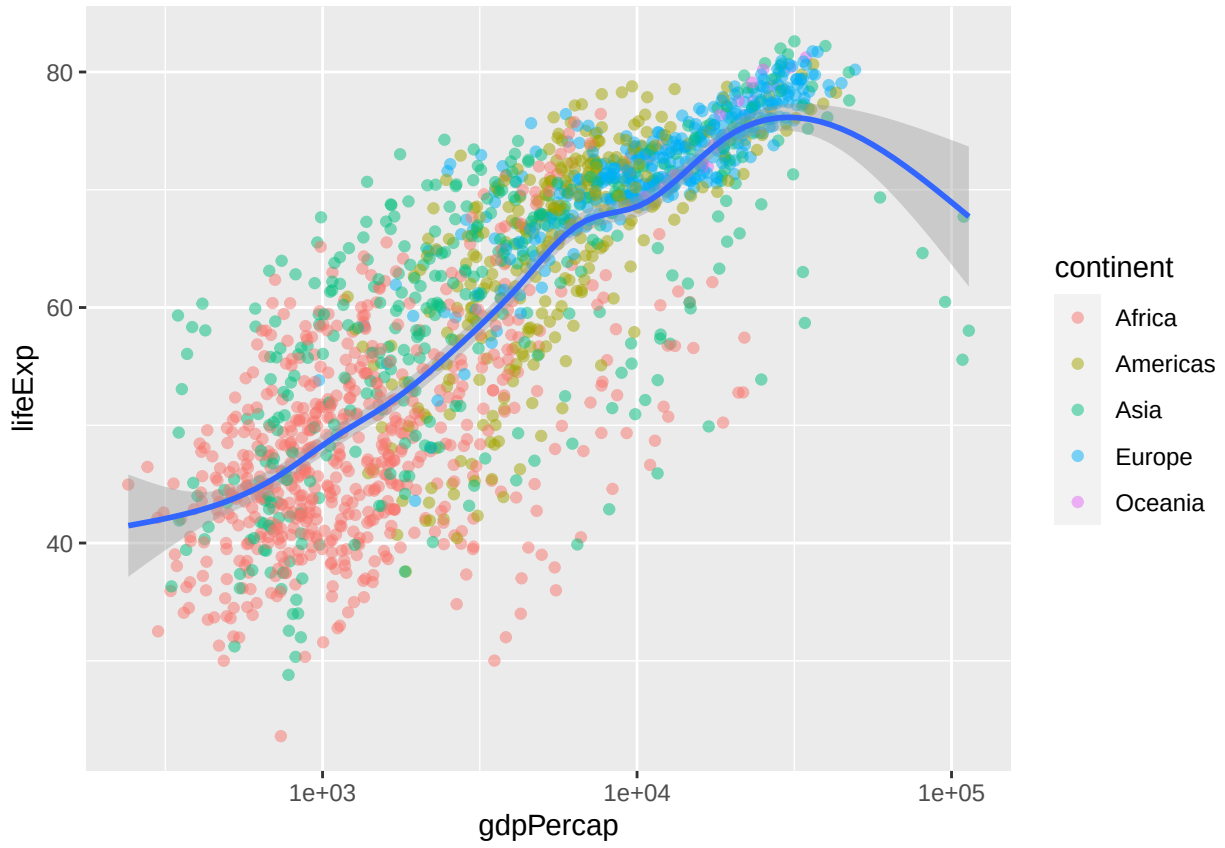
```
gapminder %>%
  ggplot(mapping = aes(x = gdpPercap, y = lifeExp)) +

  # this declare the geometric mapping
  geom_point()
```



If we instead specify `geom_line()` it renders the observations as a single path, because it doesn't know where each country begins and ends.

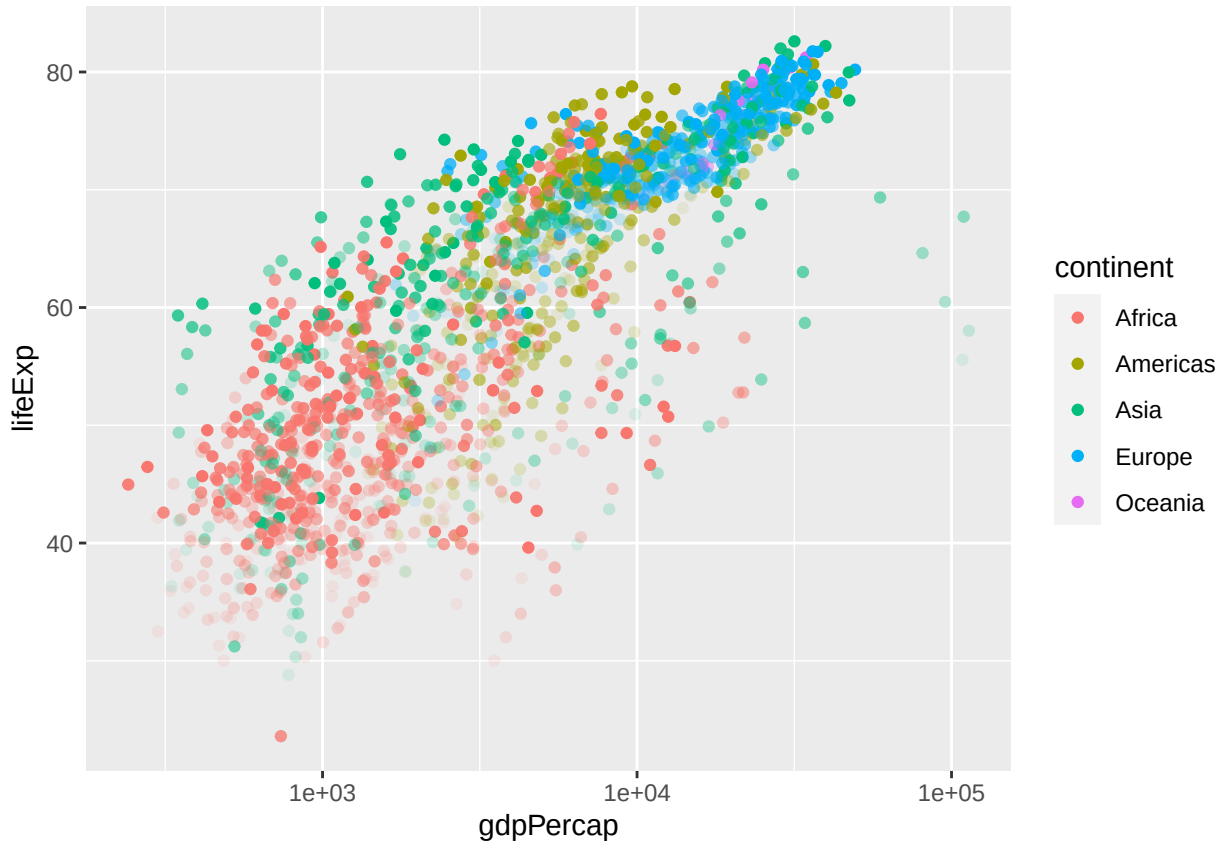
```
gapminder %>%
  ggplot(aes(x = gdpPercap,
             y = lifeExp)) +
  #geom_line()
  geom_point(mapping = aes(color = continent),
             alpha = .5) +
  scale_x_log10() +
  #scale_y_log10() +
  geom_smooth() # ?geom_smooth
```

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



Time to map `alpha`, but without legend

```
gapminder %>%
  ggplot(aes(x = gdpPercap,
             y = lifeExp)) +
  geom_point(mapping = aes(color = continent,
                           alpha = year)) +
  scale_x_log10() +
  # turn off alpha legend
  guides(alpha = "none")
```

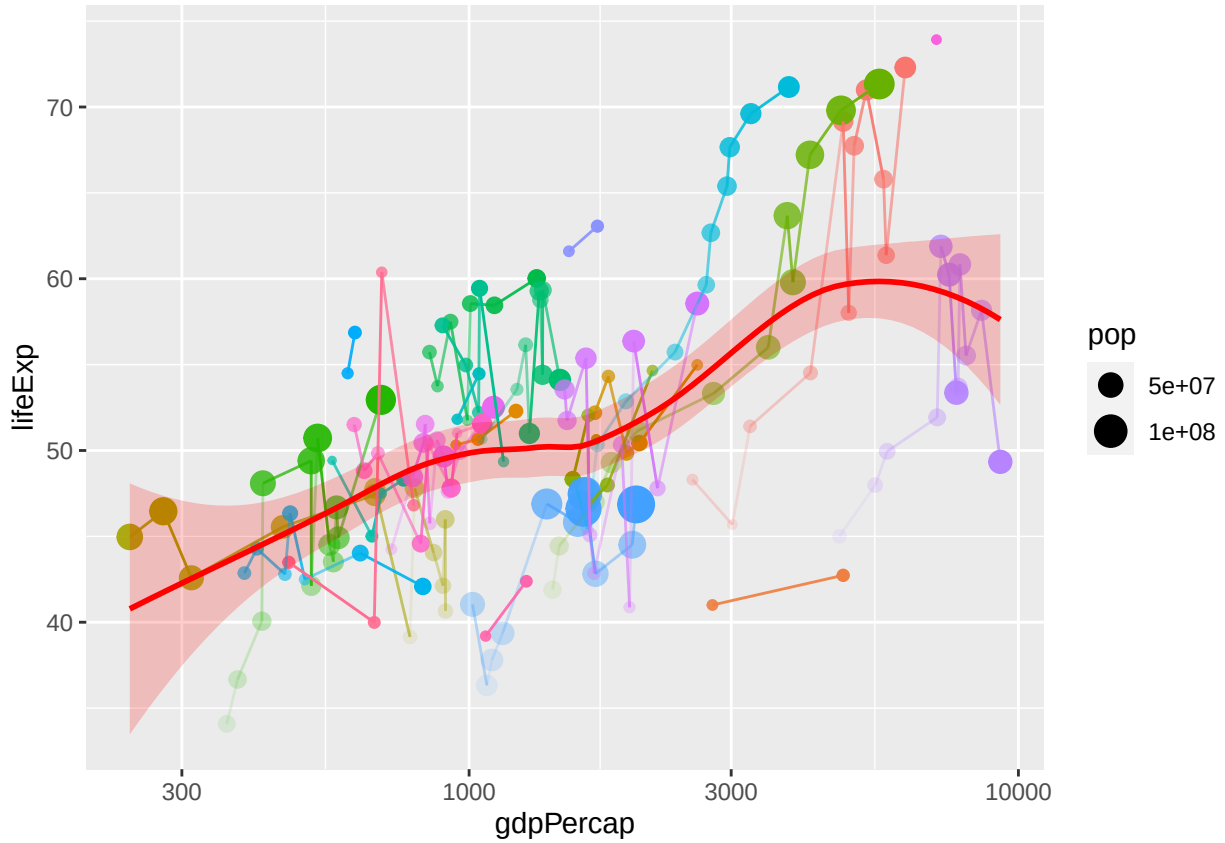


Add some filters to look at just one subset:

```
gapminder %>%
  # filter down to large countries in Africa
  filter(continent == "Africa",
         pop > 1e7) %>%

  # start the plot, declaring only coordinates
  # so that we can have aesthetic control over each geom
  ggplot(aes(x = gdpPercap,
             y = lifeExp)) +

  # x and y map to points, whose color *depends on* the country
  # and transparency on year, and size on population
  geom_point(mapping = aes(color = country,
                           alpha = year,
                           size = pop)) +

  # same mapping for lines, except we are explicit that
  # countries are distinct groups, so each line should
  # be separate
  geom_line(mapping = aes(group = country,
                          color = country,
                          alpha = year)) +

  # turn off color and transparency legends
  guides(alpha = "none",
```

```
        color = "none") +

  # log the x scale
  scale_x_log10() +

  # add overall smoother with explicit aesthetic settings
  geom_smooth(color = "red",
              fill = "red",
              alpha = .2)
```

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



I'm removing the smoother because it's distracting in a bad way: it doesn't summarize the within-country trend the way we'd like.

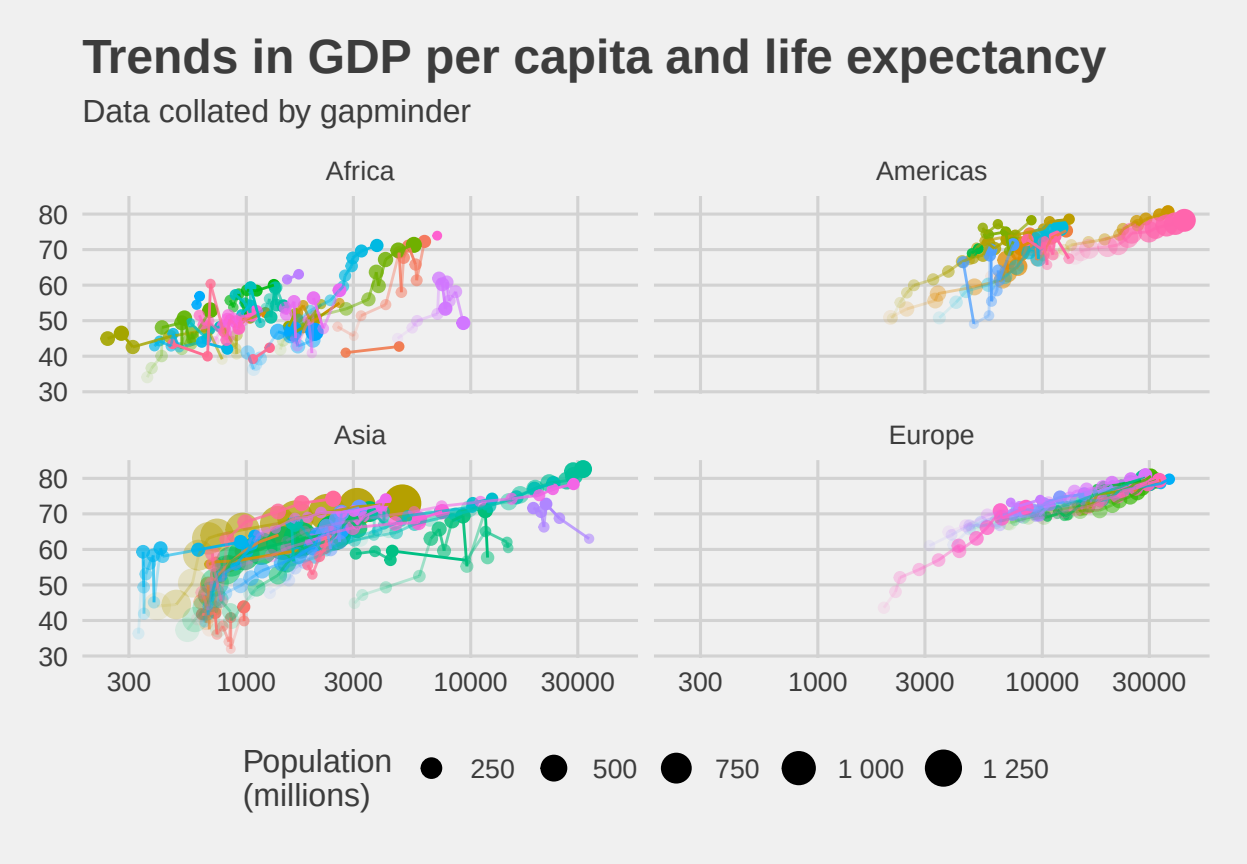
```
# install.packages("scales")
library(scales)
```

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

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

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

```
library(ggthemes)
gapminder %>%
  filter(pop > 1e7,
         continent != "Oceania") %>%
  ggplot(aes(x = gdpPercap, y = lifeExp)) +
  geom_point(mapping = aes(color = country,
                           size = pop,
                           alpha = year))+
  geom_line(mapping = aes(color = country,
                          alpha = year,
                          group = country))+
  scale_x_log10() +
  scale_size_continuous(labels = number_format(scale = 1 / 1e6)) +
  guides(color = "none",
         alpha = "none") +
  facet_wrap(~continent) +
  labs(size = "Population\n(millions)",
       x = "GDP per capita",
       y = "Life expectancy at birth",
       title = "Trends in GDP per capita and life expectancy",
       subtitle = "Data collated by gapminder") +
  theme_fivethirtyeight()
```




```
# install.packages("ggthemes")
```

Visualize our lifetables from Wednesday

```
library(readr)
path <- "https://raw.githubusercontent.com/timriffe/KOSTAT_Workshop1/master/Data/AFR_LT.csv"
AFR_LT <- read_csv(path)
```

```
## Rows: 13395 Columns: 14
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): Country, Sex, ISO3
## dbl (11): Year, Age, nMx, nAx, n, nqx, lx, ndx, nLx, Tx, ex
```

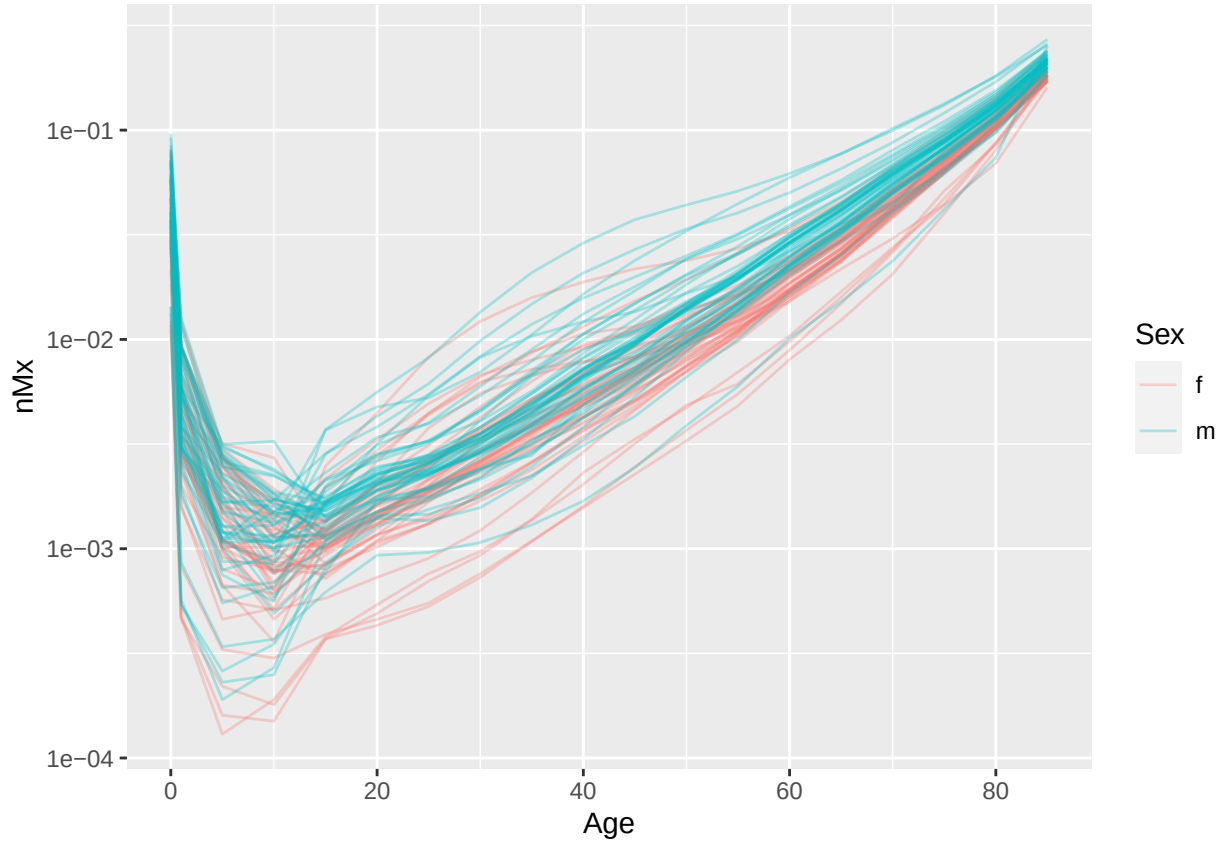
```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Examine the data, plot all M_x curves in a way that is legible

```
AFR_LT %>% head()
```

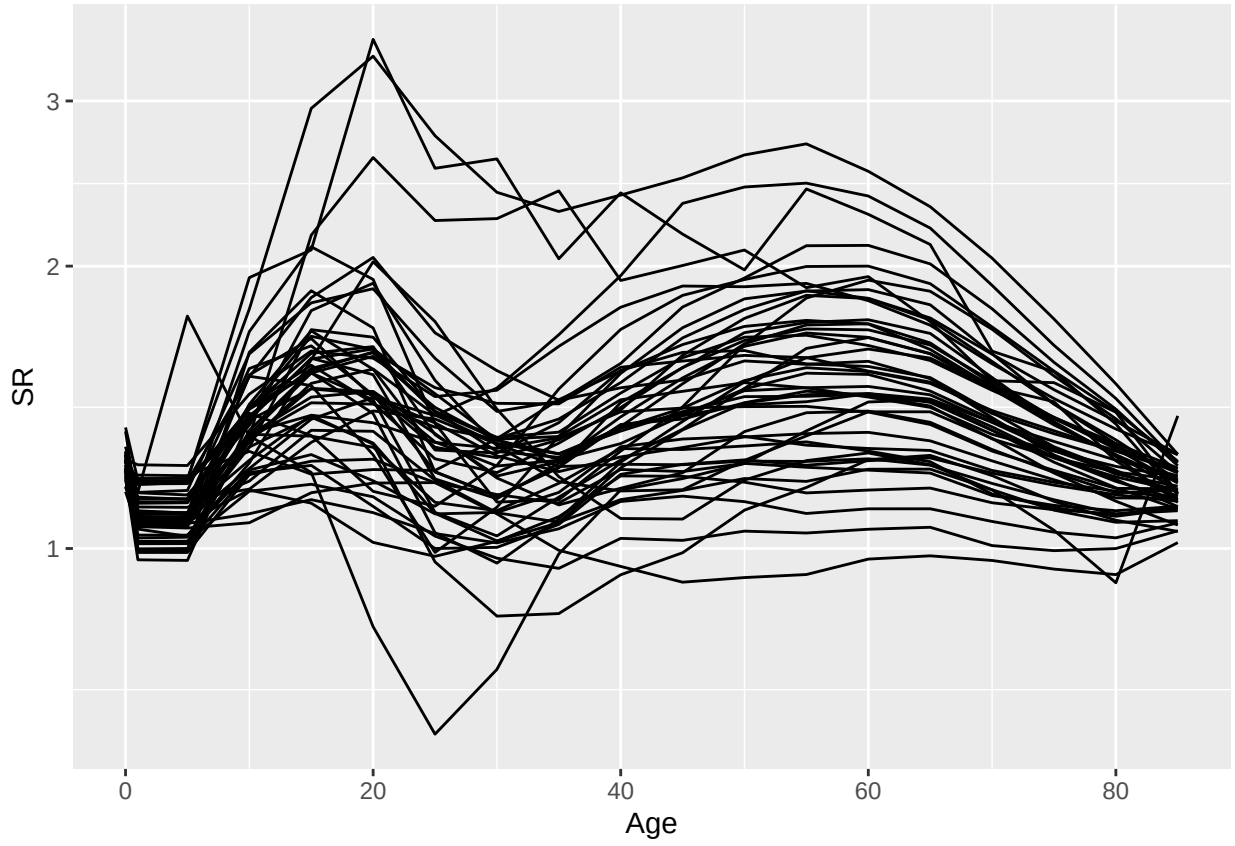
```
## # A tibble: 6 x 14
##   Country  Year Sex   ISO3    Age     nMx    nAx     n     nqx    lx     ndx
##   <chr>   <dbl> <chr> <chr> <dbl>   <dbl>  <dbl> <dbl>   <dbl> <dbl>   <dbl>
## 1 Algeria  2000 f     DZA       0 0.0344  0.0954     1 0.0334  1     0.0334
## 2 Algeria  2000 f     DZA       1 0.00145 2.81       4 0.00579 0.967 0.00560
## 3 Algeria  2000 f     DZA       5 0.0007  2.5        5 0.00349 0.961 0.00336
## 4 Algeria  2000 f     DZA      10 0.00046 2.5        5 0.00230 0.958 0.00220
## 5 Algeria  2000 f     DZA      15 0.00065 2.5        5 0.00324 0.955 0.00310
## 6 Algeria  2000 f     DZA      20 0.00078 2.5        5 0.00389 0.952 0.00371
## # ... with 3 more variables: nLx <dbl>, Tx <dbl>, ex <dbl>
```

```
AFR_LT %>%
  filter(Year == max(Year),
         Sex != "t") %>%
 ggplot(mapping = aes(x = Age, y = nMx, group = interaction(Country, Sex))) +
 geom_line(aes(color = Sex), alpha = .3) +
 scale_y_log10()
```



Make a sex ratios plot, this requires pre-processing

```
AFR_LT %>%
  # for sex ratios throw out total,
  # keep only most recent year
  filter(Year == max(Year),
         Sex != "t") %>%

  # select down to only-needed columns before pivot wider,
  # just because I was anticipating a very wide dataset otherwise
  # or at least not the anticipated dimensions
  select(Country, Age, Sex, nMx) %>%

  # move males and females side by side
  pivot_wider(names_from = Sex, values_from = nMx) %>%

  # calculate the measure
  mutate(SR = m / f) %>%

  # here begins the plot
  ggplot(aes(x = Age, y = SR, group = Country)) +
  geom_line() +
  scale_y_log10()
```



Population pyramids using `geom_bar()`

Our strategy for making a population pyramid will look a lot like the same steps you'd take anywhere else. Namely, to get males on the left, just make them negative.

```
countries <- AFR_LT$Country %>% unique()
# sample(countries, 1)
AFR_LT %>%
  filter(Country == "Nigeria",
         Year == 2019,
         Sex != "t") %>%
  mutate(SRB = 1.05,
         PF = 1 / (1 + SRB),
         `Stationary Population` = ifelse(Sex == "m",
                                          nLx * (1 - PF),
                                          nLx * PF),
         `Stationary Structure` = 100 * `Stationary Population` / sum(`Stationary Population`),
         `Stationary Structure` = ifelse(Sex == "m",
                                         -`Stationary Structure`,
                                         `Stationary Structure`) / 5,
         Age = Age - Age %% 5
         ) %>%
  group_by(Sex,Age) %>%
  summarize(`Stationary Structure` = sum(`Stationary Structure`),
            .groups = "drop") %>%
```

```
ggplot(aes(x = Age,
           fill = Sex,
           color = Sex)) +
geom_bar(stat = "identity") +
scale_y_continuous(breaks = seq(-.7, .7, by = .2),
                   labels = paste0(c("0.7","0.5","0.3","0.1","0.1","0.3","0.5","0.7"),"%")) +
coord_flip() +
scale_fill_brewer(palette = "Set1") +
scale_color_brewer(palette = "Set1") +
labs(title = "Stationary population structure",
     subitle = "Nigeria, 2019",
     caption = "Data: based on data from GHO")
```