SOC6707: Intermediate Data Analysis

Monica Alexander

Week 2: Exploratory Data Analysis and Data Visualization

Overview

What we will cover today:

- What is EDA and why do we do it?
- Steps of EDA
- Data visualization
- Key types of graphs
    - another summary measure: correlation coefficient
- Doing EDA and data viz in R (intro to `ggplot`)

(Exploratory Data Analysis = EDA)

Exploratory Data Analysis (EDA)

What is EDA and why do we do it?

Before we even do any sort of statistical inference, we need to understand the main characteristics of our dataset.

- Helps to identify any potential issues or surprising things about our data
- Helps to check / explore / refine research questions

What is EDA and why do we do it?

EDA is all about asking:

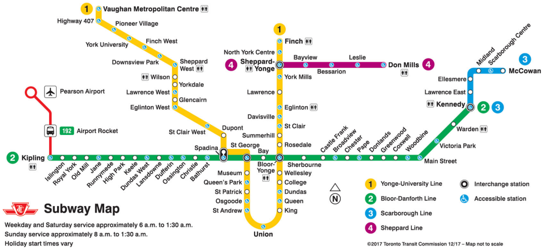
- What types of variables do we have?
- Do we have a complete dataset, or do we have missing data or observations?
- If we have missing data, is it missing equally across observations of different types or concentrated in particular groups?
- Are there any obvious outliers or strange data points?
- What do the data 'look' like?
  - summary measures, measures of centrality, spread
  - Visualizing the data through plots and tables

Steps of EDA

1. Become familiar with size of data set (number of observations and variables available)
2. What kinds of variables are available
3. For the variables that I'm interested in, are there any missing values or other issues?
4. What does the distribution/frequency of observations look like for the variables I'm interested in? (summary measures, tables and graphs)

Example: TTC subway delays in 2019

- Data on TTC subway delay times by station and day available from the Open Data Toronto website: https://open.toronto.ca/
- Accessed through using the `opendatatoronto` R package: https://sharlagelfand.github.io/opendatatoronto/



Get familiar with dataset

```
delay_2019
```

```
## # A tibble: 19,222 x 11
##    date                time  day   station code  min_delay min_gap bound line
##    <dttm>              <tim> <chr> <chr>   <chr>     <dbl>   <dbl> <chr> <chr>
##  1 2019-01-01 00:00:00 01:08 Tues~ YORK M~ PUSI          0       0 S     YU
##  2 2019-01-01 00:00:00 02:14 Tues~ ST AND~ PUMST         0       0 <NA>  YU
##  3 2019-01-01 00:00:00 02:16 Tues~ JANE    TUSC          0       0 W     BD
##  4 2019-01-01 00:00:00 02:27 Tues~ BLOOR   SUO           0       0 N     YU
##  5 2019-01-01 00:00:00 03:03 Tues~ DUPONT  MUATC        11      16 N     YU
##  6 2019-01-01 00:00:00 03:08 Tues~ EGLINT~ EUATC        11      16 S     YU
##  7 2019-01-01 00:00:00 03:09 Tues~ DUPONT  EUATC         6      11 N     YU
##  8 2019-01-01 00:00:00 03:26 Tues~ ST CLA~ EUATC         4       9 N     YU
##  9 2019-01-01 00:00:00 03:37 Tues~ KENNED~ TUMVS         0       0 E     BD
## 10 2019-01-01 00:00:00 08:04 Tues~ DAVISV~ MUNOA         5      10 S     YU
## # ... with 19,212 more rows, and 2 more variables: vehicle <dbl>,
## #   code_desc <chr>
```

Get familiar with dataset

Dimensions (number of rows x number of columns)

```r
dim(delay_2019)
```

```
## [1] 19222    11
```

Variable names

```r
colnames(delay_2019)
```

```
##  [1] "date"      "time"      "day"       "station"   "code"      "min_delay"
##  [7] "min_gap"   "bound"     "line"      "vehicle"   "code_desc"
```

The summary function is useful for a quick overview

```r
summary(delay_2019)
```

```
##       date                          time               day
##  Min.   :2019-01-01 00:00:00   Length:19222      Length:19222
##  1st Qu.:2019-03-28 00:00:00   Class1:hms        Class :character
##  Median :2019-06-27 00:00:00   Class2:difftime   Mode  :character
##  Mean   :2019-06-27 16:58:00   Mode  :numeric
##  3rd Qu.:2019-09-25 00:00:00
##  Max.   :2019-12-31 00:00:00
##    station              code             min_delay          min_gap
##  Length:19222       Length:19222       Min.   :  0.000   Min.   :  0.000
##  Class :character   Class :character   1st Qu.:  0.000   1st Qu.:  0.000
##  Mode  :character   Mode  :character   Median :  0.000   Median :  0.000
##                                        Mean   :  2.406   Mean   :  3.536
##                                        3rd Qu.:  3.000   3rd Qu.:  6.000
##                                        Max.   :455.000   Max.   :460.000
##     bound               line              vehicle       code_desc
##  Length:19222       Length:19222       Min.   :   0   Length:19222
##  Class :character   Class :character   1st Qu.:   0   Class :character
##  Mode  :character   Mode  :character   Median :5239   Mode  :character
##                                        Mean   :3974
##                                        3rd Qu.:5671
##                                        Max.   :9206
```

Research question?

- What are some good potential research questions with this dataset?

Sanity checks

We need to check variables should be what they say they are. If they aren't, the natural next question is to what to do with issues (recode? remove?)

E.g. check days of week make sense with the `unique` function

```
delay_2019 %>%
  select(day) %>%
  unique()
```

```
## # A tibble: 7 x 1
##   day
##   <chr>
## 1 Tuesday
## 2 Wednesday
## 3 Thursday
## 4 Friday
## 5 Saturday
## 6 Sunday
## 7 Monday
```

Sanity checks

Check lines: oh no. some issues here. Some have obvious recodes, others, not so much.

```r
delay_2019 %>%
  select(line) %>%
  unique() %>%
  pull() # turn into a vector for better display
```

```
##  [1] "YU"                    "BD"                    "YU/BD"
##  [4] "SHP"                   "SRT"                   NA
##  [7] "YUS"                   "B/D"                   "BD LINE"
## [10] "999"                   "YU/ BD"                "YU & BD"
## [13] "BD/YU"                 "YU\\BD"                "46 MARTIN GROVE"
## [16] "RT"                    "BLOOR-DANFORTH"        "YU / BD"
## [19] "134 PROGRESS"          "YU - BD"               "985 SHEPPARD EAST EXPR"
## [22] "22 COXWELL"            "100 FLEMINGDON PARK"   "YU LINE"
```

Data issues

How bad is the mislabeling of lines? look at frequency of cases

NOTE! New very important function: `group_by`

```
delay_2019 %>%
  group_by(line) %>% # group by line label
  tally() %>% # count the number of occurrences
  arrange(-n) # arrange in descending order
```

```
## # A tibble: 24 x 2
##    line        n
##    <chr>   <int>
##  1 YU       9275
##  2 BD       8200
##  3 SRT       699
##  4 SHP       600
##  5 YU/BD     356
##  6 <NA>       50
##  7 YU / BD    16
##  8 YUS         6
##  9 YU/ BD      3
## 10 999         2
## # ... with 14 more rows
```

Missing values

```r
delay_2019 %>%
  summarise_all(.funs = funs(sum(is.na(.))))
```

```
## # A tibble: 1 x 11
##    date  time   day station  code min_delay min_gap bound  line vehicle
##   <int> <int> <int>   <int> <int>     <int>   <int> <int> <int>   <int>
## 1     0     0     0       0     0         0       0  4380    50       0
## # ... with 1 more variable: code_desc <int>
```

Or use the skimr package

```r
library(skimr) # install using install.packages("skimr")
skim(delay_2019)
```

(Show this in lecture)

Summary statistics

Most interested in delay minutes, which is the `min_delay` variable

```
delay_2019 %>%
  summarize(n_obs = n(),
            mean_delay = mean(min_delay),
            median_delay = median(min_delay),
            range_delay = max(min_delay) - min(min_delay),
            iqr_delay = IQR(min_delay))
```

```
## # A tibble: 1 x 5
##   n_obs mean_delay median_delay range_delay iqr_delay
##   <int>      <dbl>        <dbl>       <dbl>     <dbl>
## 1 18697       2.43            0         455         3
```

Summary statistics

Probably more interesting to do these summaries by line (**stratify** by line); easy extension with the `group_by` function

```
delay_2019 %>%
  group_by(line) %>%
  summarize(n_obs = n(),
            mean_delay = mean(min_delay),
            median_delay = median(min_delay),
            range_delay = max(min_delay) - min(min_delay),
            iqr_delay = IQR(min_delay))
```

```
## # A tibble: 4 x 6
##   line  n_obs mean_delay median_delay range_delay iqr_delay
##   <chr> <int>      <dbl>        <dbl>       <dbl>     <dbl>
## 1 BD     8197       2.11            0         180       3
## 2 SHP     598       2.20            0         165       3
## 3 SRT     631       5.79            3         284       5.5
## 4 YU     9271       2.50            0         455       3
```

Summaries

Could also stratify by reason for delay

```r
delay_2019 %>%
  group_by(code_desc) %>%
  summarize(n_obs = n(),
            mean_delay = mean(min_delay),
            median_delay = median(min_delay),
            range_delay = max(min_delay) - min(min_delay),
            iqr_delay = IQR(min_delay)) %>%
  arrange(-n_obs)
```

```
## # A tibble: 119 x 6
##    code_desc                n_obs mean_delay median_delay range_delay iqr_delay
##    <chr>                    <int>      <dbl>        <dbl>       <dbl>     <dbl>
##  1 Miscellaneous Speed Cont~  1997    0.186             0          19         0
##  2 Injured or ill Customer ~  1747    0.151             0          54         0
##  3 Operator Overspeeding      1379    0.114             0           8         0
##  4 Passenger Assistance Ala~  1353    0.800             0          12         0
##  5 Disorderly Patron          1147    3.02              3          23         4
##  6 <NA>                        931    4.19              0         284         5
##  7 Injured or ill Customer ~   671    3.92              3          50         5
##  8 Escalator/Elevator Incid~   605    0.00826           0           5         0
##  9 Speed Control Equipment     527    0.436             0          30         0
## 10 ATC Project                 514    3.88              3          28         5
## # ... with 109 more rows
```

Summaries

Arrange by mean delay time

```
delay_2019 %>%
  group_by(code_desc) %>%
  summarize(n_obs = n(),
            mean_delay = mean(min_delay),
            median_delay = median(min_delay),
            range_delay = max(min_delay) - min(min_delay),
            iqr_delay = IQR(min_delay)) %>%
  arrange(-mean_delay)
```

```
## # A tibble: 119 x 6
##    code_desc                n_obs mean_delay median_delay range_delay iqr_delay
##    <chr>                    <int>      <dbl>        <dbl>       <dbl>     <dbl>
##  1 Traction Power Rail Rela~     1      145          145            0       0
##  2 Priority One - Train in ~    24       78.8         80          193      70.2
##  3 Structure Related Problem     4       70.5         27          228      97.5
##  4 Rail Related Problem          8       58.6          3          455       4
##  5 Fire/Smoke Plan A             6       50           11.5        250      17.5
##  6 Bomb Threat                  12       36.7         20          130      32
##  7 Fire/Smoke Plan B - Sour~    84       19.4         11          180      16.2
##  8 Doors Open in Error          11       18.7         16           40       7.5
##  9 Fire/Smoke Plan B - Sour~     2       13.5         13.5         19       9.5
## 10 Suspicious Package           14       13            3.5         67      22
## # ... with 109 more rows
```

EDA: summary so far

- There's no one checklist of things to looks at, depends on your data and research question
- Get familiar with your dataset
- Check for missing values, and that existing values make sense
- Summary statistics depend on your research question of interest
  - stratifying (`group_by`) by important characteristics often useful

Data visualization

Plot your data!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

- We started to compute some summary statistics above, and showed how summaries can be calculated by group and arranged in different ways to get a sense of differences across groups
- However, graphing/plotting your data is usually the best way to visualize patterns, trends, outliers, issues and other surprising points
- The most appropriate types of graph for your data depends on:
  - the type of variable you are interested in (quantitative or qualitative/categorical)
  - your research questions

Plot your data!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

- Before you start to do any statistical analysis, you should always plot your data
- Data visualization is a key part of EDA and essential in understanding the assumptions and outcomes of your eventual statistical analysis

Important types of graphs

Important types of graphs

- Histograms
- Bar charts
- Boxplots
- Line plots
- Scatter plots

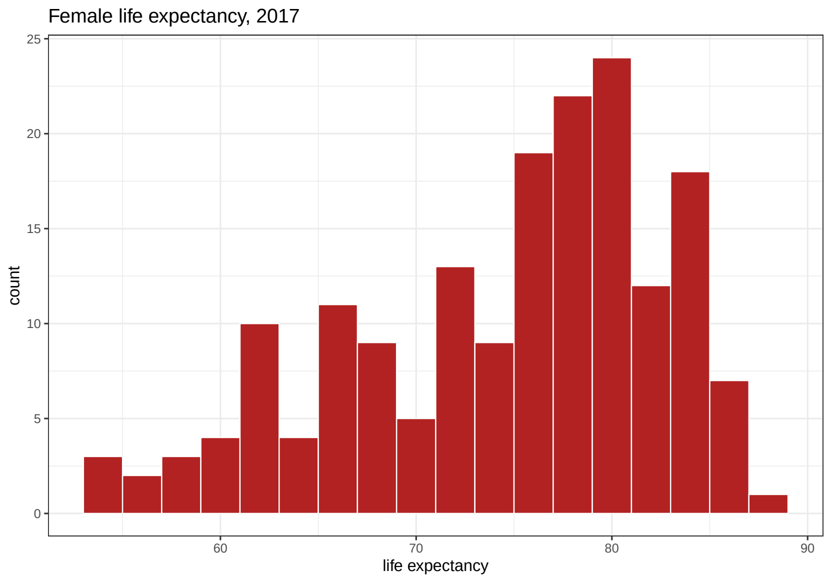
Example datasets used here

1. TTC subway delays (from above)
2. Country-level indicators, 2009-2017
    - Uploaded onto Quercus
    - TFR = total fertility rate
    - GDP = gross domestics product
    - dataset also has life expectancy (females), child mortality, maternal mortality

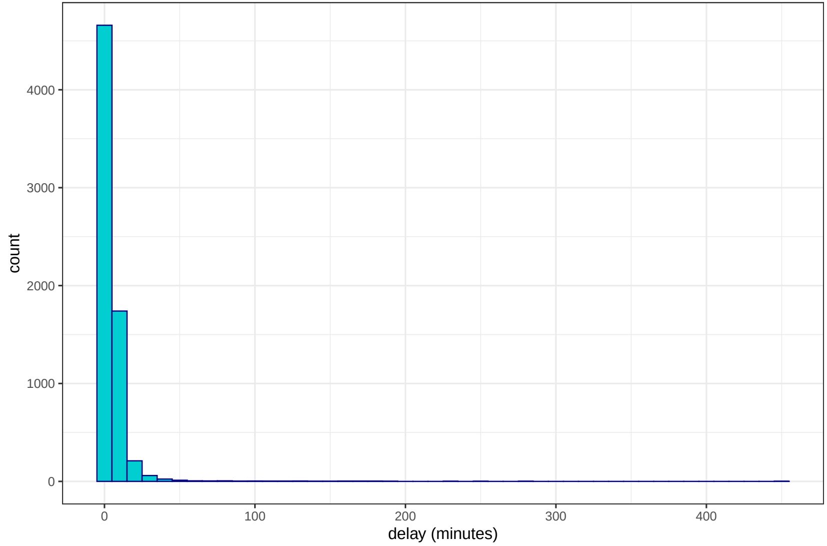
Histograms

Shows the distribution of a **quantitative** variable

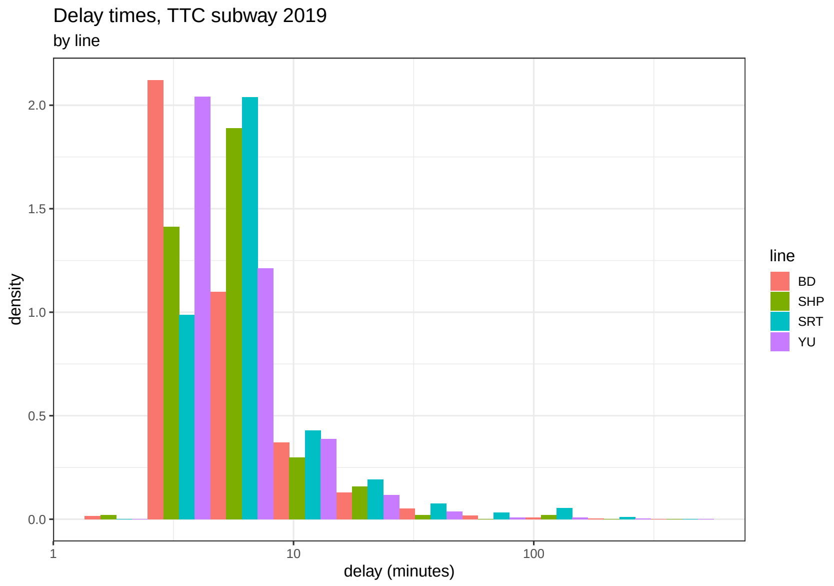
- Histograms show the frequency (count) of observations by value
- The range of values of a variables is divided into intervals (‘bins’) and then the number of observations in each bin is tabulated
- A histogram shows the count of observations in each bin with a rectangle of height equal to the count
- The x axis is the value bins, the y axis is the count/frequency (or proportion)


Female life expectancy, 2017
count
life expectancy
0
5
10
15
20
25
60
70
80
90

Delay times, TTC subway 2019



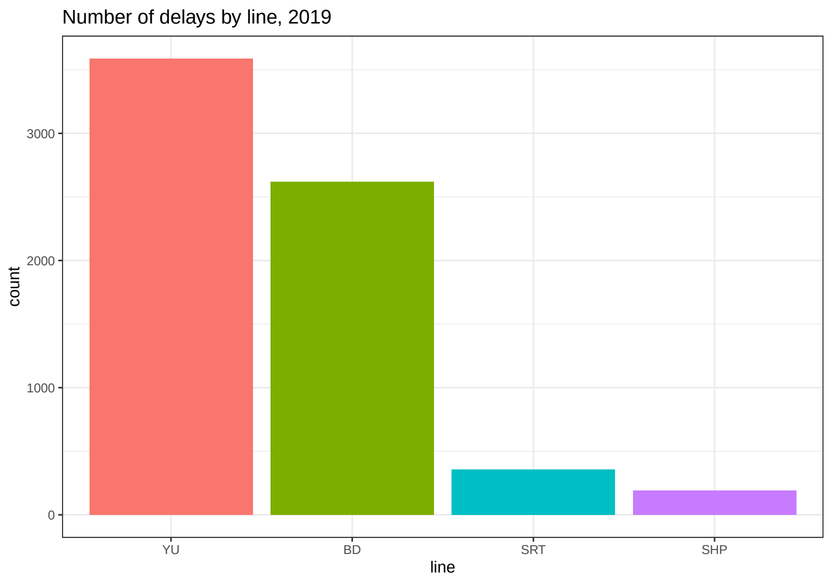
Making the histogram more informative



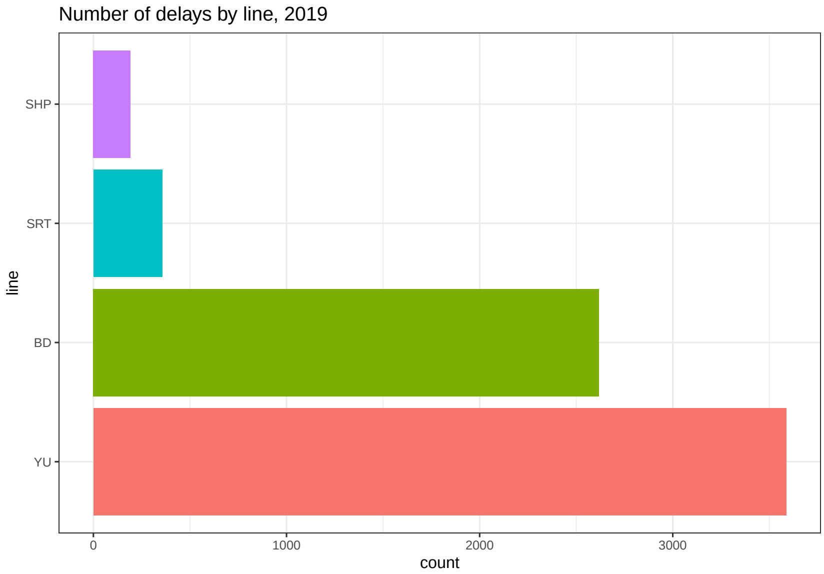
Bar charts

Shows summary measures across values of a **categorical** (qualitative) variable

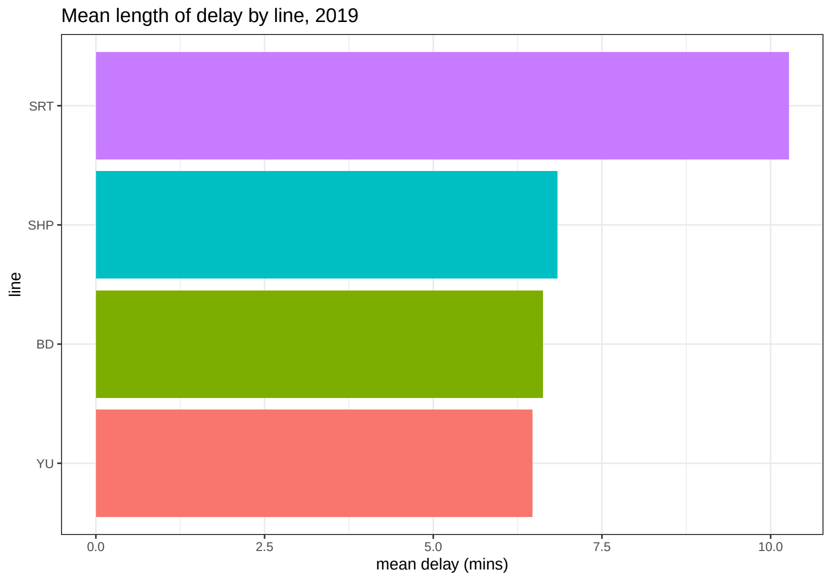
- Illustrate the value of a particular outcome in a particular category
- The 'value' can be counts, but could also be a summary measure (e.g. mean)
- The value is again shown by a rectangle of height equal to the value
- Bar carts can be plotted vertically or horizontally
- In the vertical setting, the x axis is the categories and the y axis is the value of the quantitative variable


Number of delays by line, 2019
count
3000
2000
1000
0
YU
BD
SRT
SHP
line

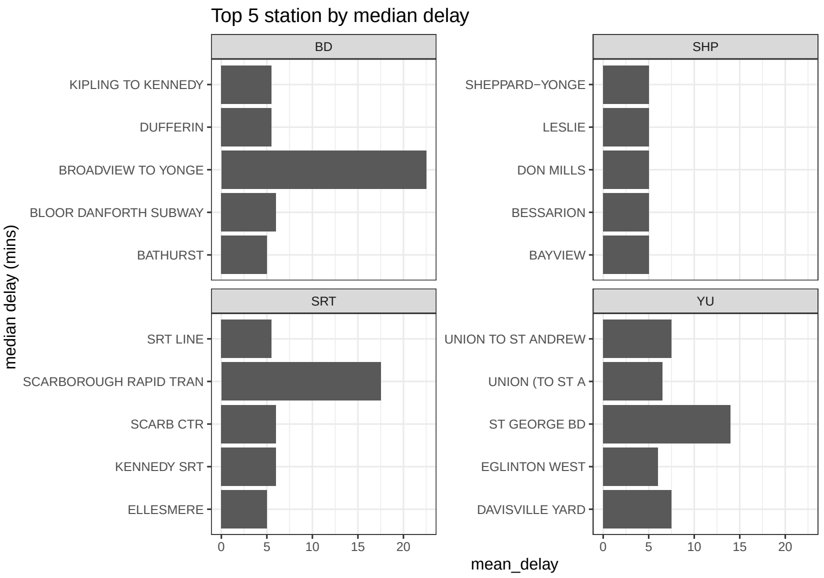
Same but horizontal



Showing mean delay time

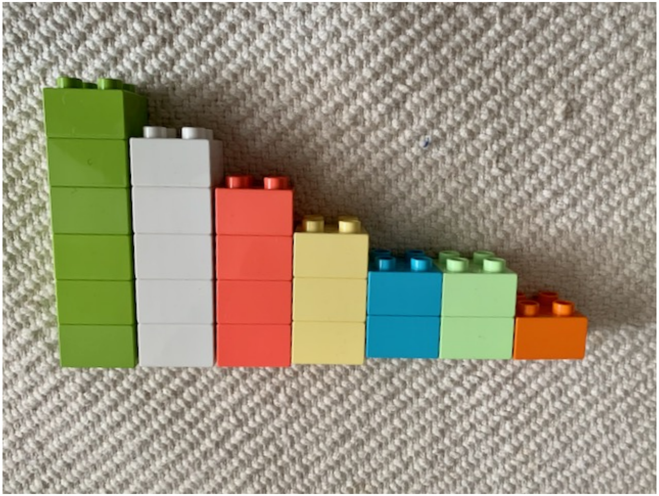


More complicated example

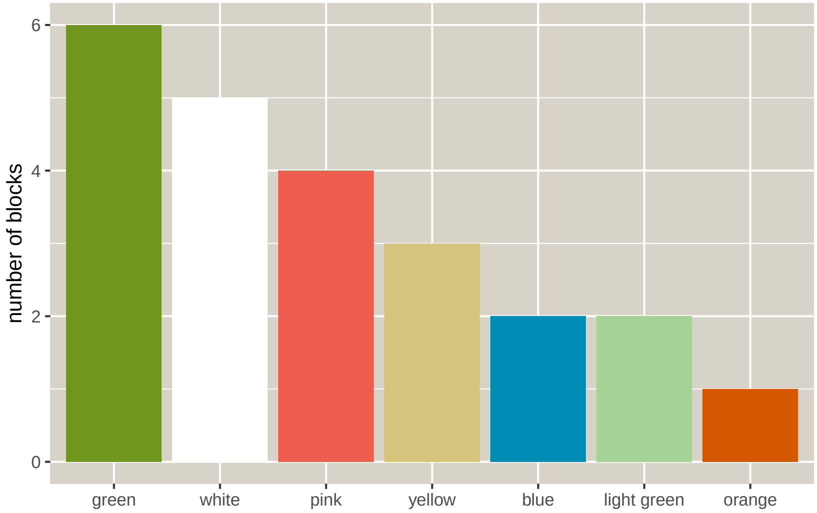


A lego example

```r
lego <- tibble(color = c("green", "white", "pink",
                         "yellow", "blue", "light green", "orange"),
               count = c(6,5,4,3,2,2,1))
```



```
ggplot(lego, aes(color, count, fill = color)) +
  geom_bar(stat="identity") +
  scale_fill_manual(values = c("#70961c", "white",
                               "#ee5e4f", "#d5c47c", "#008db3", "#a5d395", "#d35800")) +
  theme(legend.position = 'none',
        panel.background = element_rect(fill = "#d7d3c9",
                                        colour = "#d7d3c9",
                                        size = 0.5, linetype = "solid")) +
  ylab("number of blocks")+xlab("")
```

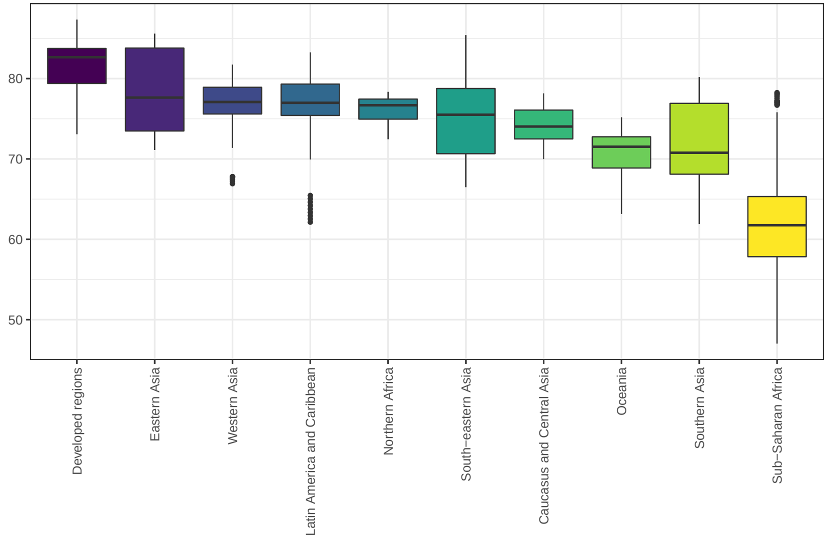


Box plots

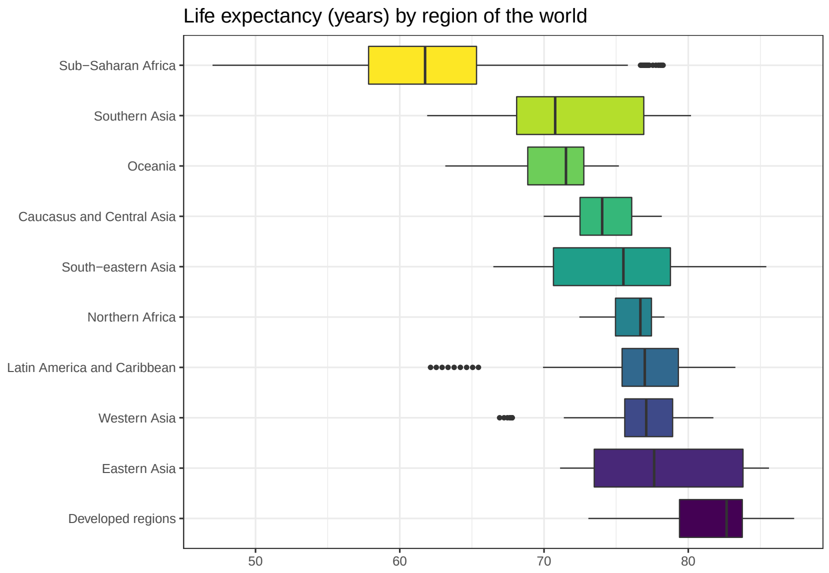
Good for showing summaries of **quantitative** variables across different **categorical** groups.

- Visualizing quartiles (25/50/75 percentiles) of quantitative data
- Boxes show the IQR and median
- Whiskers show values outside IQR (in R/`ggplot`, default is 1.5*IQR)
- Outliers may be shown with individual dots
- In the vertical case, the x axis is the categories and the y axis is the quantitative variable

Life expectancy (years) by region of the world



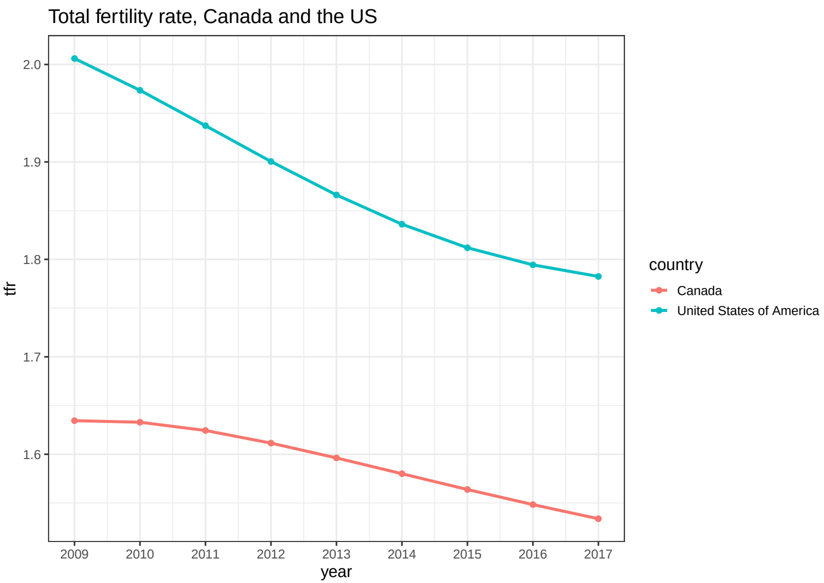
Could also do horizontal



Line plots

Best used to describe values of a **quantitative** variable (on y axis) across sequential values of another **quantitative** variable on the x axis

- Plots a series of values of a quantitative variable connected together by a line
- Useful to visualize trends over time



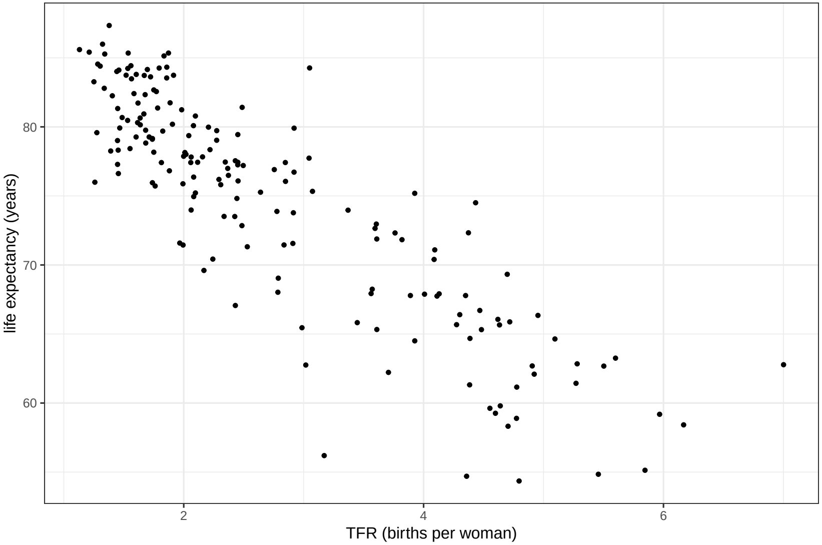
Total fertility rate, Canada and the US
2.0
1.9
1.8
1.7
1.6
tfr
2009
2010
2011
2012
2013
2014
2015
2016
2017
year
country
Canada
United States of America

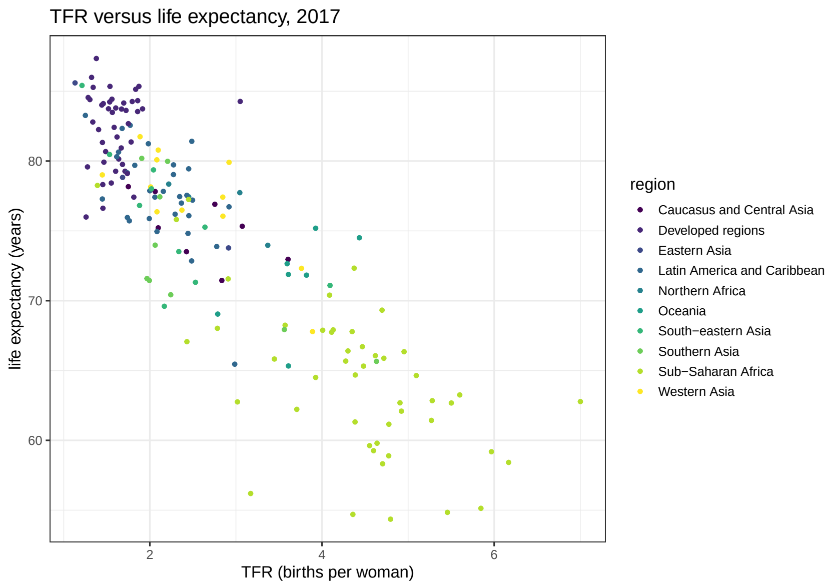
Scatter plots

Shows relationship between two different **quantitative** variables

- Uses dots to represent values for two different **quantitative** values
- The position of each dot on the x and y axis indicates values for an individual data point
- Extremely useful in visualizing the relationship between two quantitative variables

TFR versus life expectancy, 2017




TFR versus life expectancy, 2017
life expectancy (years)
80
70
60
2
4
6
TFR (births per woman)
region
Caucasus and Central Asia
Developed regions
Eastern Asia
Latin America and Caribbean
Northern Africa
Oceania
South−eastern Asia
Southern Asia
Sub−Saharan Africa
Western Asia

Aside: another summary measure

Based on the previous graphs, evidence to suggest a relationship between TFR and life expectancy

- as TFR goes up, life expectancy tends to go down
- life expectancy is **negatively correlated** with TFR

Correlation is the statistical measure of the relationship between two variables. **Pearson's correlation coefficient**, r_{xy} summarizes this relationship into one number. For an observation sample of two random variables $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$,

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Correlation coefficients in R

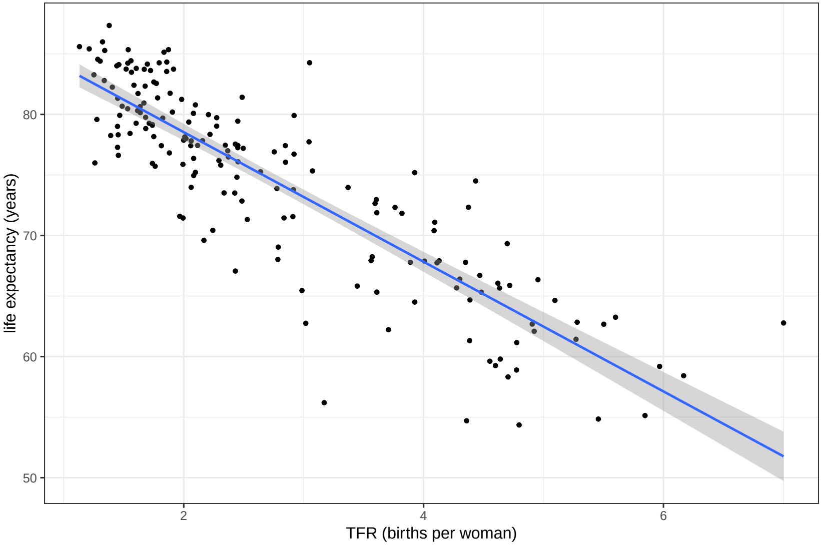
Easiest to use the function `cor`

```
country_ind_2017 <- country_ind %>% filter(year==2017)
country_ind_2017 %>%
  select(tfr, life_expectancy) %>%
  summarize(correlation = cor(tfr, life_expectancy))
```

```
## # A tibble: 1 x 1
##   correlation
##         <dbl>
## 1      -0.868
```

```
# alternative code
# cor(country_ind_2017$tfr, country_ind_2017$life_expectancy)
```

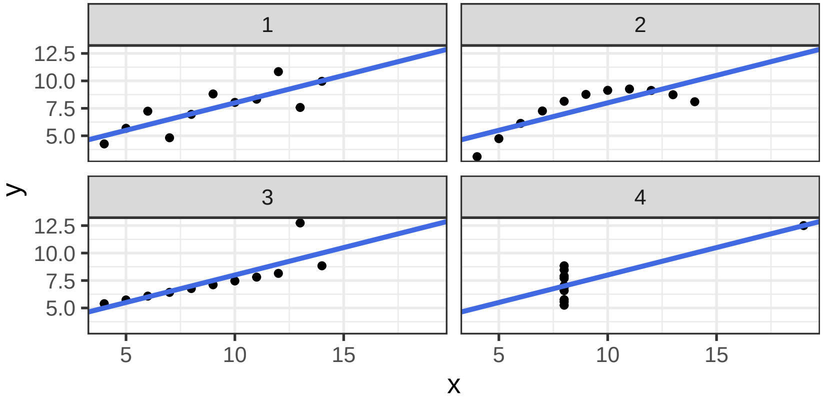
TFR versus life expectancy, 2017

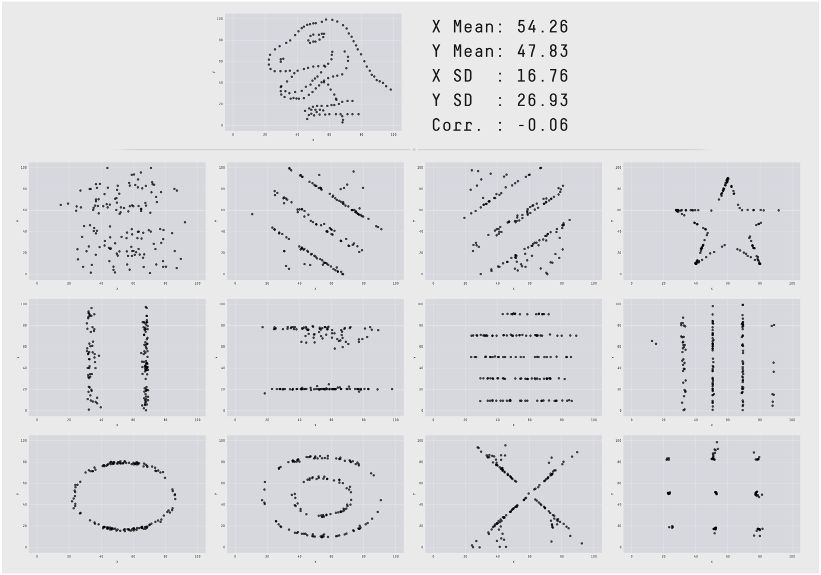


Plot your data!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

The correlation coefficient tells us in a single number that there is a negative relationship observed between TFR and life expectancy. So why bother plotting at all?

Anscombe's quartet

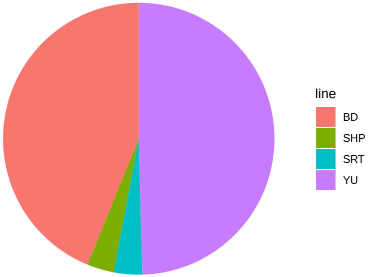


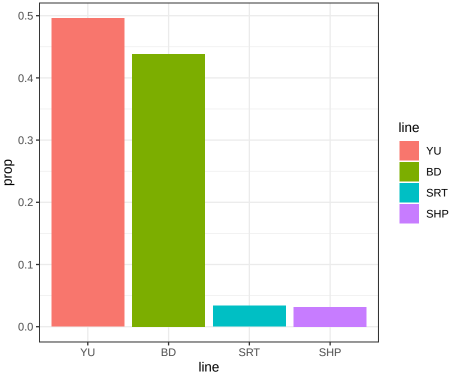


Source: https://www.autodeskresearch.com/publications/samestats

Where are the pie charts?

Don't use pie charts!, Humans are inherently bad at judging angles, which is what you have to do with a pie chart. Use a bar chart instead.





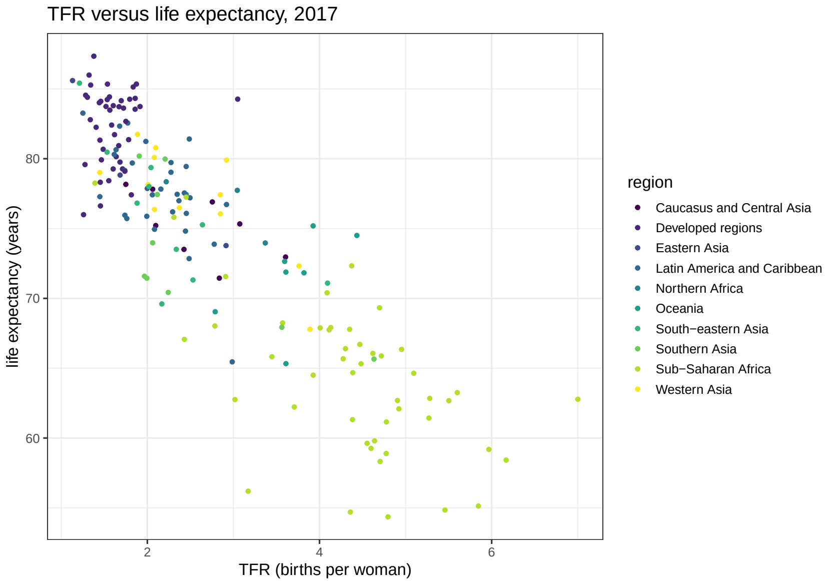
Introduction to `ggplot`

`ggplot`

- `ggplot` is the graphing package that goes with the `tidyverse` in R
- Very powerful to make a wide range of graphics
- Every graph so far this lecture was done in `ggplot`
- `ggplot` code works in layers, with each layer adding complexity
    - start with defining dataset and different variables
    - add on type of plot
    - scales
    - layout (facets)
    - themes, fonts, sizes. . .

More practice in lab, but here's a starting example

Reproducing the TFR verus life expectancy chart, colored by region



Data

```
# read in the data
country_ind <- read_csv("../../data/country_indicators.csv")
country_ind
```

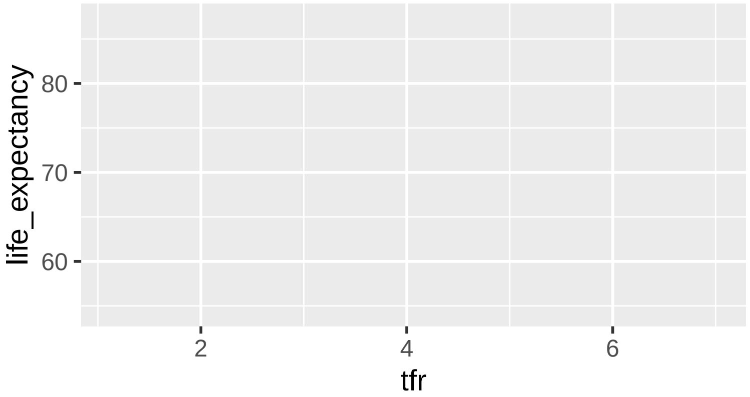
```
## # A tibble: 1,584 x 9
##    country_code country region  year   tfr life_expectancy child_mort
##    <chr>        <chr>   <chr>  <dbl> <dbl>           <dbl>      <dbl>
##  1 AFG          Afghan~ South~  2009  6.18            61.9       93.9
##  2 AFG          Afghan~ South~  2010  5.98            62.5       90.0
##  3 AFG          Afghan~ South~  2011  5.77            63         86.3
##  4 AFG          Afghan~ South~  2012  5.56            63.5       82.9
##  5 AFG          Afghan~ South~  2013  5.36            64.0       79.6
##  6 AFG          Afghan~ South~  2014  5.16            64.5       76.6
##  7 AFG          Afghan~ South~  2015  4.98            64.9       73.8
##  8 AFG          Afghan~ South~  2016  4.80            65.3       71.2
##  9 AFG          Afghan~ South~  2017  4.63            65.7       68.8
## 10 ALB          Albania Devel~  2009  1.65            79.0       16.7
## # ... with 1,574 more rows, and 2 more variables: maternal_mort <dbl>,
## #   gdp <dbl>
```

```
# filter to just be 2017
country_ind_2017 <- country_ind %>% filter(year==2017)
```

A blank canvas

aes stands for aesthetic and tells ggplot the main characteristics of your plot (x, y, and if the color or fill vary by group)

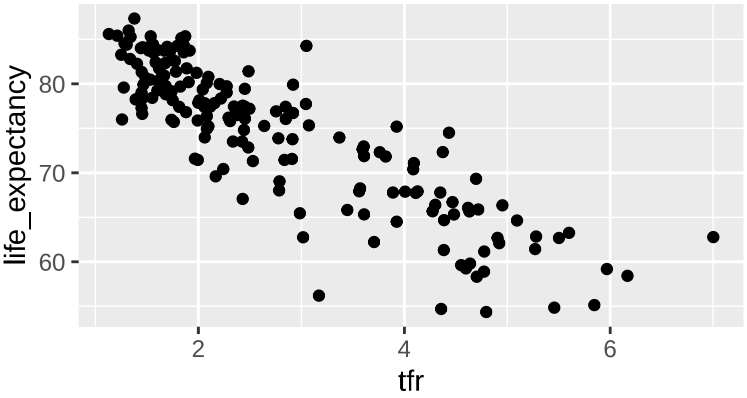
```
plot1 <- ggplot(data = country_ind_2017, aes(x = tfr, y = life_expectancy))

#print
plot1
```



Add the points

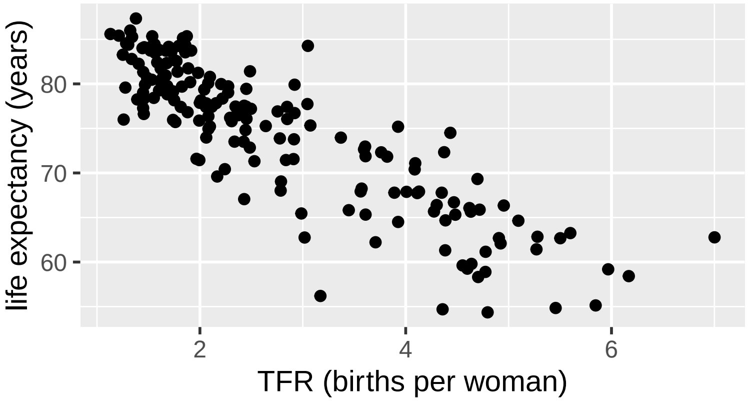
Add layers with ggplot using the +

```
plot1 <- ggplot(data = country_ind_2017, aes(x = tfr, y = life_expectancy)) +
  geom_point()

plot1
```



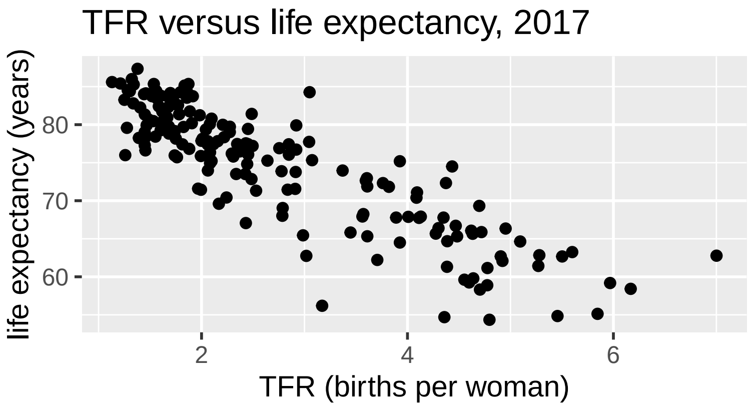
Tidy up labels

```
plot1 <- ggplot(data = country_ind_2017, aes(x = tfr, y = life_expectancy)) +
  geom_point()+
  xlab("TFR (births per woman)")+
  ylab("life expectancy (years)")

plot1
```



Title

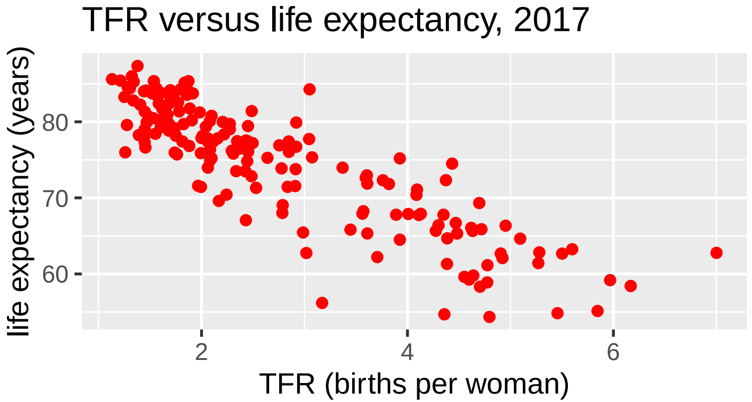
```r
plot1 <- ggplot(data = country_ind_2017, aes(x = tfr, y = life_expectancy)) +
  geom_point()+
  xlab("TFR (births per woman)")+
  ylab("life expectancy (years)")+
  ggtitle("TFR versus life expectancy, 2017")

plot1
```



Change color of points

to see all colors, type `colors()`

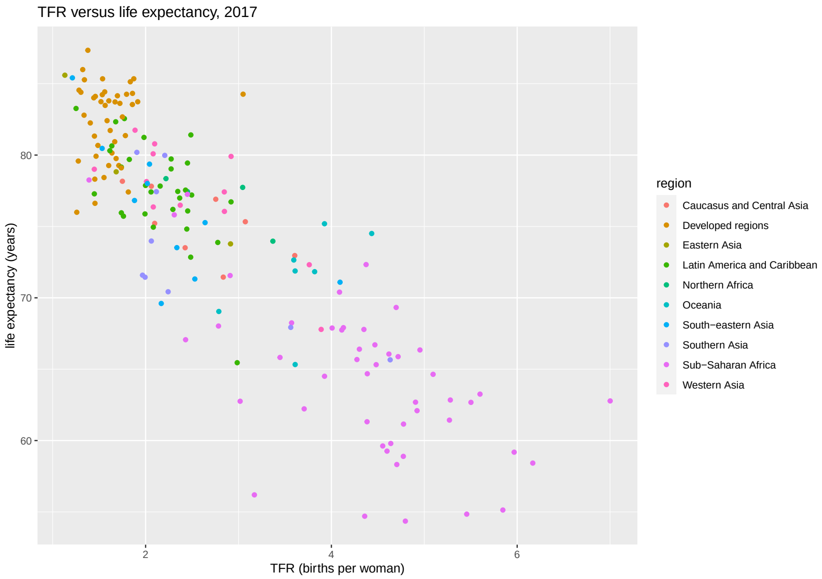
```
plot1 <- ggplot(data = country_ind_2017, aes(x = tfr, y = life_expectancy)) +
  geom_point(color = "red")+
  xlab("TFR (births per woman)")+
  ylab("life expectancy (years)")+
  ggtitle("TFR versus life expectancy, 2017")

plot1
```



Coloring by group

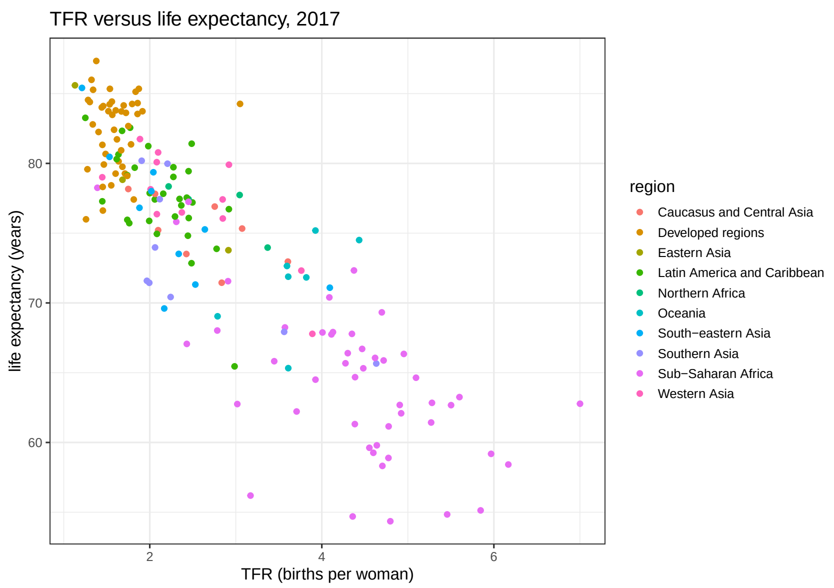
This goes in the `aes()` because it **depends on the data**

```
plot1 <- ggplot(data = country_ind_2017, aes(x = tfr, y = life_expectancy, color = region)) +
  geom_point()+
  xlab("TFR (births per woman)")+
  ylab("life expectancy (years)")+
  ggtitle("TFR versus life expectancy, 2017")

plot1
```


TFR versus life expectancy, 2017
life expectancy (years)
80
70
60
2
4
6
TFR (births per woman)
region
Caucasus and Central Asia
Developed regions
Eastern Asia
Latin America and Caribbean
Northern Africa
Oceania
South−eastern Asia
Southern Asia
Sub−Saharan Africa
Western Asia

Change theme (optional) and size of points

```r
plot1 <- ggplot(data = country_ind_2017, aes(x = tfr, y = life_expectancy, color = region)) +
  geom_point(size =2)+
  xlab("TFR (births per woman)")+
  ylab("life expectancy (years)")+
  ggtitle("TFR versus life expectancy, 2017")+
  theme_bw(base_size = 14)

plot1
```

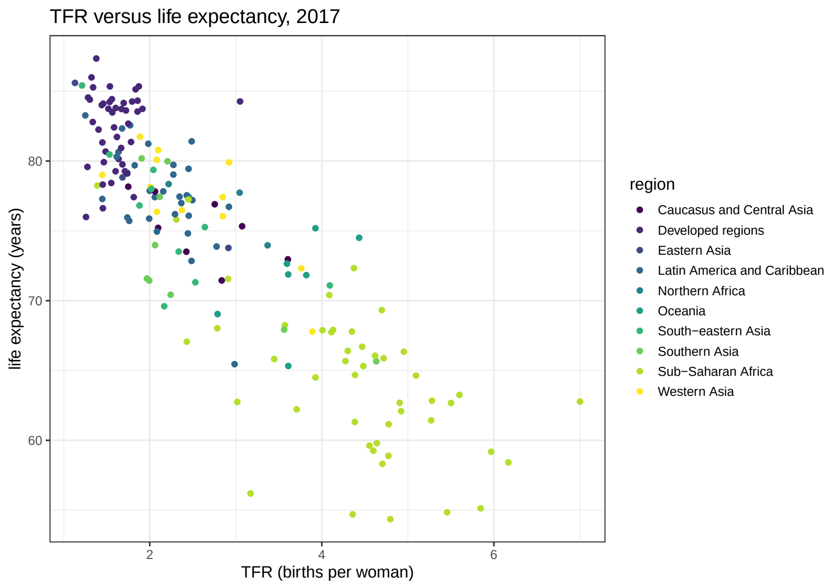


TFR versus life expectancy, 2017
life expectancy (years)
80
70
60
2
4
6
TFR (births per woman)
region
Caucasus and Central Asia
Developed regions
Eastern Asia
Latin America and Caribbean
Northern Africa
Oceania
South−eastern Asia
Southern Asia
Sub−Saharan Africa
Western Asia

Change color scheme

viridis and brewer both good options

```r
plot1 <- ggplot(data = country_ind_2017, aes(x = tfr, y = life_expectancy, color = region)) +
  geom_point(size =2)+
  xlab("TFR (births per woman)")+
  ylab("life expectancy (years)")+
  ggtitle("TFR versus life expectancy, 2017")+
  theme_bw(base_size = 14)+
  scale_color_viridis_d()

plot1
```


TFR versus life expectancy, 2017
life expectancy (years)
80
70
60
2
4
6
TFR (births per woman)
region
Caucasus and Central Asia
Developed regions
Eastern Asia
Latin America and Caribbean
Northern Africa
Oceania
South−eastern Asia
Southern Asia
Sub−Saharan Africa
Western Asia

Summary

- Ask well-defined and well-scoped research questions that can be answered based on the data you have
- EDA and data visualization is often just as informative and important as statistical analysis
- It is essential to understand the structure of your data, missing-ness, any outliers/issues, and the raw patterns in your data before deciding on your statistical analysis
- Plot, plot, plot
- Practice, practice, practice

Summary

Plots:

- Bar charts for categorical/qualitative variables
- Histograms, boxplots for one quantitative variable (potentially across multiple categories)
- Line plots and scatter plots for two quantitative variables (line plot when one is sequential)

Lab

- Finishing off last week's lab
- Practice with `ggplot` and how to graph important types of plots in R.