SciencesPo
DEPARTMENT OF ECONOMICS

Applied Data Analysis for Public Policy Studies

Summarising, Visualizing and Tidying Data

Michele Fioretti
SciencesPo Paris
2020-08-24

Recap from last week

- *Causality* plays a central role in modern econometrics!

Recap from last week

- ***Causality*** plays a central role in modern econometrics!
- Using R is ***very*** valuable

Recap from last week

- ***Causality*** plays a central role in modern econometrics!
- Using R is ***very*** valuable
- Basic data wrangling:
    - `View`, `str`, `names`, `nrow`, `ncol`
    - *subsetting:* `murders[row condition, "column name"]`
    - *variable creation:* `murders$total_percap = (murders$total / murders$population) * 10000`

Recap from last week

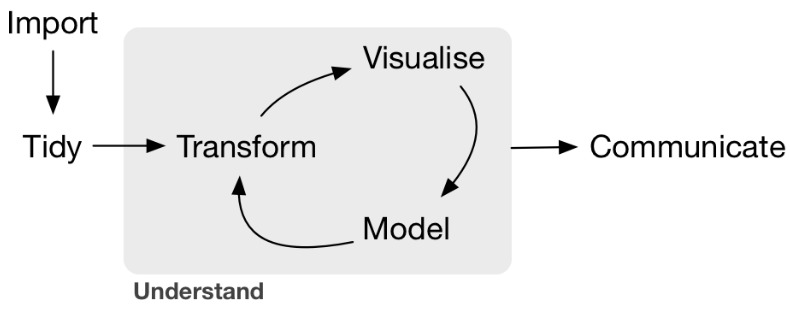
- ***Causality*** plays a central role in modern econometrics!
- Using R is ***very*** valuable
- Basic data wrangling:
  - `View`, `str`, `names`, `nrow`, `ncol`
  - *subsetting:* `murders[row condition, "column name"]`
  - *variable creation:* `murders$total_percap = (murders$total / murders$population) * 10000`

Today

- Deeper dive into data wrangling with `R`:
  - **summarizing** data,
  - **visualisation** data,
  - **tidying** data

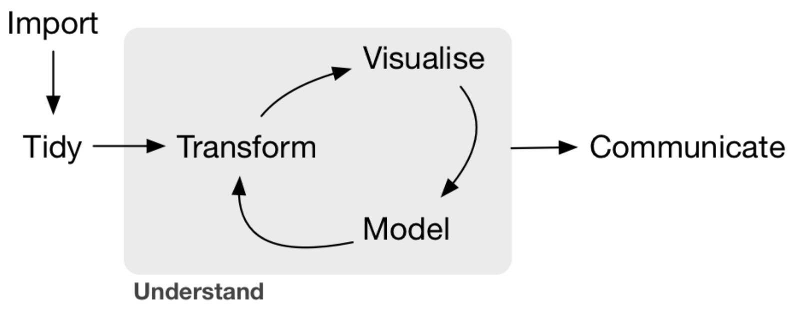
Working With Data

- Econometrics is about `data`.



Working With Data

- Econometrics is about `data`.



- According a to 2014 NYTimes article, "data scientists [...] spend from ***50 percent to 80 percent of their time*** mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets."

- In the next two lectures you will learn the **basics** of summarizing, visualising and tidying data

The `gapminder` dataset: Overview

- Let's first load a dataset with these commands:

```r
library(dslabs)
gapminder <- gapminder
```

- Here are the first 3 rows

```r
head(gapminder, n = 3)
```

```
##   country year infant_mortality life_expectancy fertility population
## 1 Albania 1960            115.4           62.87      6.19    1636054
## 2 Algeria 1960            148.2           47.50      7.65   11124892
## 3  Angola 1960            208.0           35.98      7.32    5270844
##           gdp continent          region
## 1          NA    Europe Southern Europe
## 2 13828152297    Africa Northern Africa
## 3          NA    Africa   Middle Africa
```

The `gapminder` dataset: Overview

- What variables does this dataset contain?

```
names(gapminder)
```

```
## [1] "country"          "year"             "infant_mortality" "life_expectancy"
## [5] "fertility"        "population"       "gdp"              "continent"
## [9] "region"
```

- `tail` gives you the last (6) rows.

```
tail(gapminder)
```

The `gapminder` dataset: Datatypes

- It's important to know how the data is stored.
- We can use `str` for that:

```
str(gapminder)
```

```
## 'data.frame':    10545 obs. of  9 variables:
##  $ country         : Factor w/ 185 levels "Albania","Algeria",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ year            : int  1960 1960 1960 1960 1960 1960 1960 1960 1960 1960 ...
##  $ infant_mortality: num  115.4 148.2 208 NA 59.9 ...
##  $ life_expectancy : num  62.9 47.5 36 63 65.4 ...
##  $ fertility       : num  6.19 7.65 7.32 4.43 3.11 4.55 4.82 3.45 2.7 5.57 ...
##  $ population      : num  1636054 11124892 5270844 54681 20619075 ...
##  $ gdp             : num  NA 1.38e+10 NA NA 1.08e+11 ...
##  $ continent       : Factor w/ 5 levels "Africa","Americas",..: 4 1 1 2 2 3 2 5 4 3 ...
##  $ region          : Factor w/ 22 levels "Australia and New Zealand",..: 19 11 10 2 15 21 2 1 22 21 ...
```

Task 1 (7 minutes)

- Create a new variable called `gdppercap` corresponding to `gdp` divided by `population`
- Which countries had a 2011 GDP per capita greater than 30.000?
- Filter the dataset to only keep the year 2015: `gapminder_2015`
- How many countries have an infant mortality in 2015 greater than 90 (per 1000)?
- What is the average life expectancy in Africa in 2015?

Summarizing

Summarizing Data

- One can learn only a limited amount from **looking** at a `data.frame`. 🔍

Summarizing Data

- One can learn only a limited amount from **looking** at a `data.frame`. 🔍
- Even if you *could* see all rows of the dataset, you would not know very much **about it**.

Summarizing Data

- One can learn only a limited amount from **looking** at a `data.frame`. 🔍
- Even if you *could* see all rows of the dataset, you would not know very much **about it**.
- We need to **summarize** the data for us to learn from it.

Summarizing Data

- One can learn only a limited amount from **looking** at a `data.frame`. 🔍
- Even if you *could* see all rows of the dataset, you would not know very much **about it**.
- We need to **summarize** the data for us to learn from it.
- In general, we can compute summary statistics, or visualize the data with plots.

Summarizing Data

- One can learn only a limited amount from **looking** at a `data.frame`. 🔍
- Even if you *could* see all rows of the dataset, you would not know very much **about it**.
- We need to **summarize** the data for us to learn from it.
- In general, we can compute summary statistics, or visualize the data with plots.
- Let's start with some statistics first!

Summarizing Data

- One can learn only a limited amount from **looking** at a `data.frame`. 🔍
- Even if you *could* see all rows of the dataset, you would not know very much **about it**.
- We need to **summarize** the data for us to learn from it.
- In general, we can compute summary statistics, or visualize the data with plots.
- Let's start with some statistics first!
- Let's look at two features: *central tendency* and *spread*.

Central Tendency

1. `mean(x)`: the average of all values in `x`.

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

```
x <- c(1,2,2,2,2,100)
mean(x)
```

```
## [1] 18.16667
```

```
mean(x) == sum(x) / length(x)
```

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

Your turn: What's the mean of `infant_mortality` in 1960? Read the help for `mean` to remove `NA`s.

Central Tendency

1. `mean(x)`: the average of all values in `x`.

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

```
x <- c(1,2,2,2,2,100)
mean(x)
```

```
## [1] 18.16667
```

```
mean(x) == sum(x) / length(x)
```

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

Your turn: What's the mean of `infant_mortality` in 1960? Read the help for `mean` to remove `NA`s.

1. `median`: the value x_j below and above which 50% of the values in `x` lie. m is the median if

$$\Pr(X \leq m) \geq 0.5 \text{ and } \Pr(X \geq m) \geq 0.5$$

2. The median is robust against *outliers*. 🤔? (later).

```
median(x)
```

```
## [1] 2
```

Your turn: What's the median of `infant_mortality` in 1960?

Missing Values: `NA`

- Whenever a value is *missing*, we code it as `NA`.

```
x <- NA
```

- `R` propagates `NA` through operations:

```
NA > 5
```

```
## [1] NA
```

```
NA + 10
```

```
## [1] NA
```

- the function `is.na(x)` returns `TRUE` if `x` is an `NA`.

```
is.na(x)
```

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

Missing Values: NA

- Whenever a value is *missing*, we code it as NA.

```
x <- NA
```

- R propagates NA through operations:

```
NA > 5
```

```
## [1] NA
```

```
NA + 10
```

```
## [1] NA
```

- the function is.na(x) returns TRUE if x is an NA.

```
is.na(x)
```

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

- What is confusing is that

```
NA == NA
```

```
## [1] NA
```

- It's easy to illustrate like that:

```
# Let x be Mary's age. We don't know how old she
x <- NA

# Let y be John's age. We don't know how old he i
y <- NA

# Are John and Mary the same age?
x == y
```

```
## [1] NA
```

```
#> [1] NA
# We don't know!
```

Spread

- Another interesting feature is how much a variable is *spread out* about it's center (the mean in this case).

- The *variance* is such a measure.

$$Var(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

- Consider two `normal distributions` with equal mean at `0`:

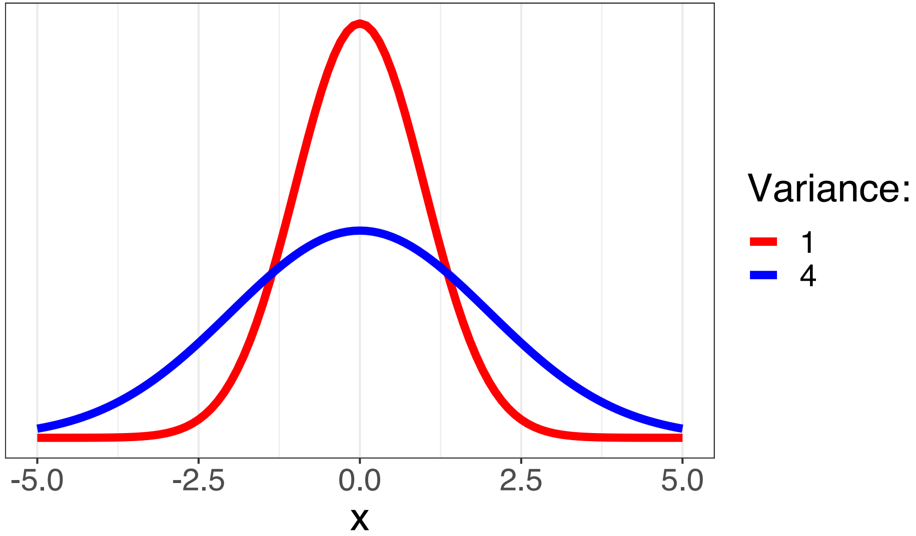
Spread

- Another interesting feature is how much a variable is *spread out* about it's center (the mean in this case).

- The *variance* is such a measure.

$$Var(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

- Consider two `normal distributions` with equal mean at `0`:



- Compute with:

```
var(x)
range(x)    # range
```

Example: the Weight of Women and Men[1]

- Men weight 65 kg and women 55 kg on average. The variance is 25.

```
# Generate a random dataset
set.seed(1234) # `set.seed` allows replicating ra
df <- data.frame(
    sex=factor(rep(c("F", "M"), each=200)),
    weight=round(c(rnorm(200, mean=55, sd=5), # ?
                   rnorm(200, mean=65, sd=5))))
```

- Plot the overall density

```
ggplot(df, aes(x=weight)) + geom_density() + them
```

- Plot separated densities

```
# Change density plot line colors by groups
ggplot(df, aes(x=weight, color=sex)) + geom_densi
```

Example: the Weight of Women and Men[1]

- Men weight 65 kg and women 55 kg on average. The variance is 25.

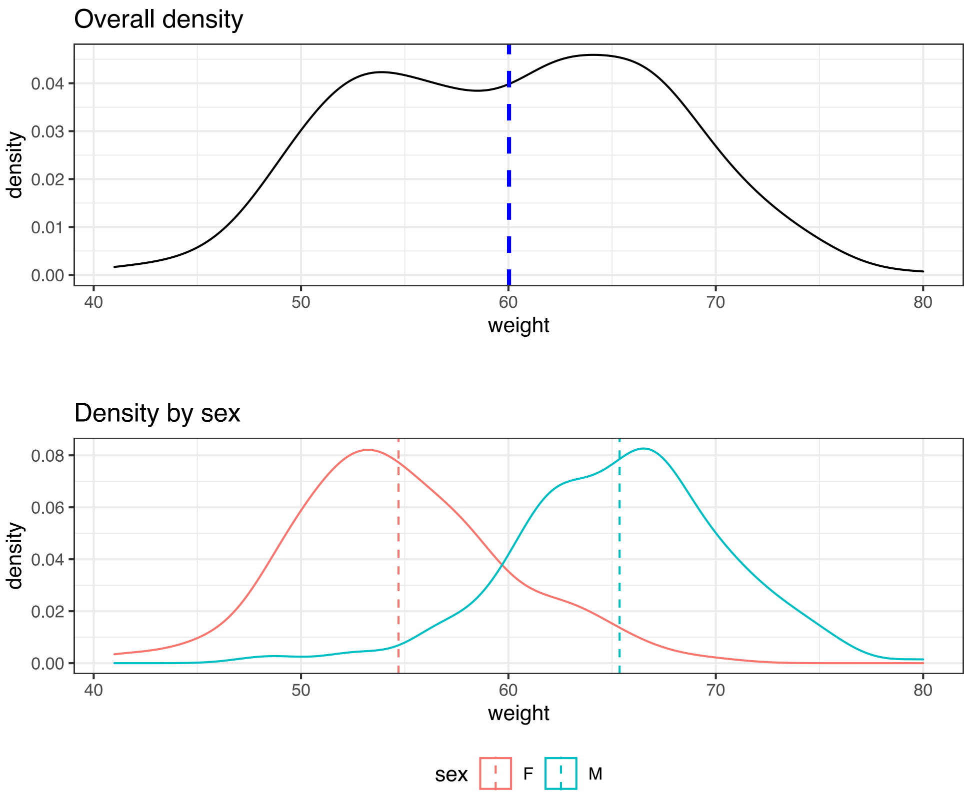
```
# Generate a random dataset
set.seed(1234) # `set.seed` allows replicating ra
df <- data.frame(
    sex=factor(rep(c("F", "M"), each=200)),
    weight=round(c(rnorm(200, mean=55, sd=5), # ?
                   rnorm(200, mean=65, sd=5))))
```

- Plot the overall density

```
ggplot(df, aes(x=weight)) + geom_density() + them
```

- Plot separated densities

```
# Change density plot line colors by groups
ggplot(df, aes(x=weight, color=sex)) + geom_densi
```



[1]: This example is taken from sthda.com.

The `table` function

- `table(x)` is a useful function that counts the occurence of each unique value in `x`:

```
table(gapminder$continent)
```

```
## 
##   Africa Americas     Asia   Europe  Oceania 
##     2907     2052     2679     2223      684 
```

```
table(gapminder$region)
```

```
## 
## Australia and New Zealand                 Caribbean           Central America 
##                       114                       741                       456 
##              Central Asia            Eastern Africa              Eastern Asia 
##                       285                       912                       342 
##            Eastern Europe                 Melanesia                Micronesia 
##                       570                       285                       114 
##             Middle Africa           Northern Africa          Northern America 
##                       456                       342                       171 
##           Northern Europe                 Polynesia             South America 
##                       570                       171                       684 
##        South-Eastern Asia           Southern Africa             Southern Asia 
##                       570                       285                       456 
##           Southern Europe            Western Africa              Western Asia 
##                       684                       912                      1026 
##            Western Europe 
##                       399 
```

Crosstables

- Given two vectors, `table` produces a contingency table:

```
gapminder_2015 <- subset(gapminder, year == 2015)
gapminder_2015$fertility_above_2 = (gapminder_2015$fertility > 2.1) # dummy variable for fertility rate abov
table(gapminder_2015$fertility_above_2,gapminder_2015$continent)
```

```
##
##         Africa Americas Asia Europe Oceania
##   FALSE      2       15   20     39       4
##   TRUE      49       20   27      0       8
```

Crosstables

- Given two vectors, `table` produces a contingency table:

```
gapminder_2015 <- subset(gapminder, year == 2015)
gapminder_2015$fertility_above_2 = (gapminder_2015$fertility > 2.1) # dummy variable for fertility rate abov
table(gapminder_2015$fertility_above_2,gapminder_2015$continent)
```

```
##
##         Africa Americas Asia Europe Oceania
##   FALSE      2       15   20     39       4
##   TRUE      49       20   27      0       8
```

- With `prop.table`, we can get proportions:

```
# proportions by row
prop.table(table(gapminder_2015$fertility_above_2,gapminder_2015$continent), margin = 1)
# proportions by column
prop.table(table(gapminder_2015$fertility_above_2,gapminder_2015$continent), margin = 2)
```

- ⚠️ To obtain `table`s with `NA`s, use the `useNA = "always"` or `useNA = "ifany"`

Plotting

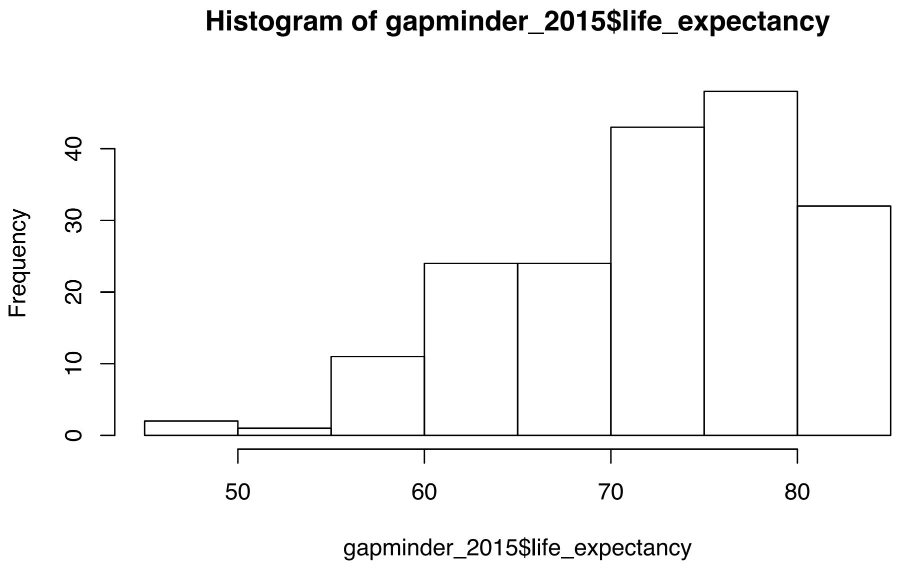
Plotting

- `R` base plotting is fairly good.
- There is an extremely powerful alternative in package `ggplot2`. We'll see both.
- First example: *histograms*. A histogram counts how many observations fall within a certain bin.

Plotting

- `R` base plotting is fairly good.
- There is an extremely powerful alternative in package `ggplot2`. We'll see both.
- First example: *histograms*. A histogram counts how many observations fall within a certain bin.

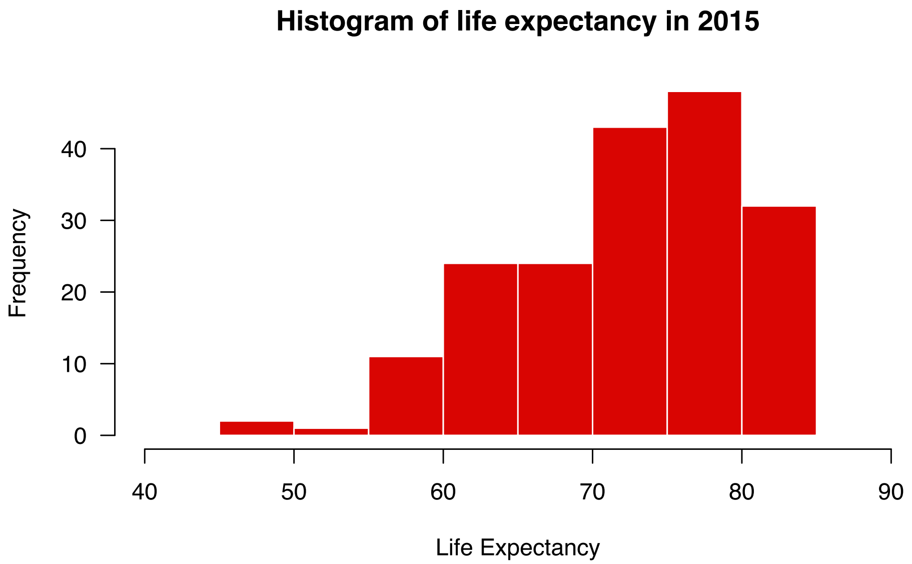
```
gapminder_2015 <- gapminder[gapminder$year == 2015,]
hist(gapminder_2015$life_expectancy)
```



A Nicer Histogram

- We can give additional arguments to `hist`.
- Look at `?hist` for more.

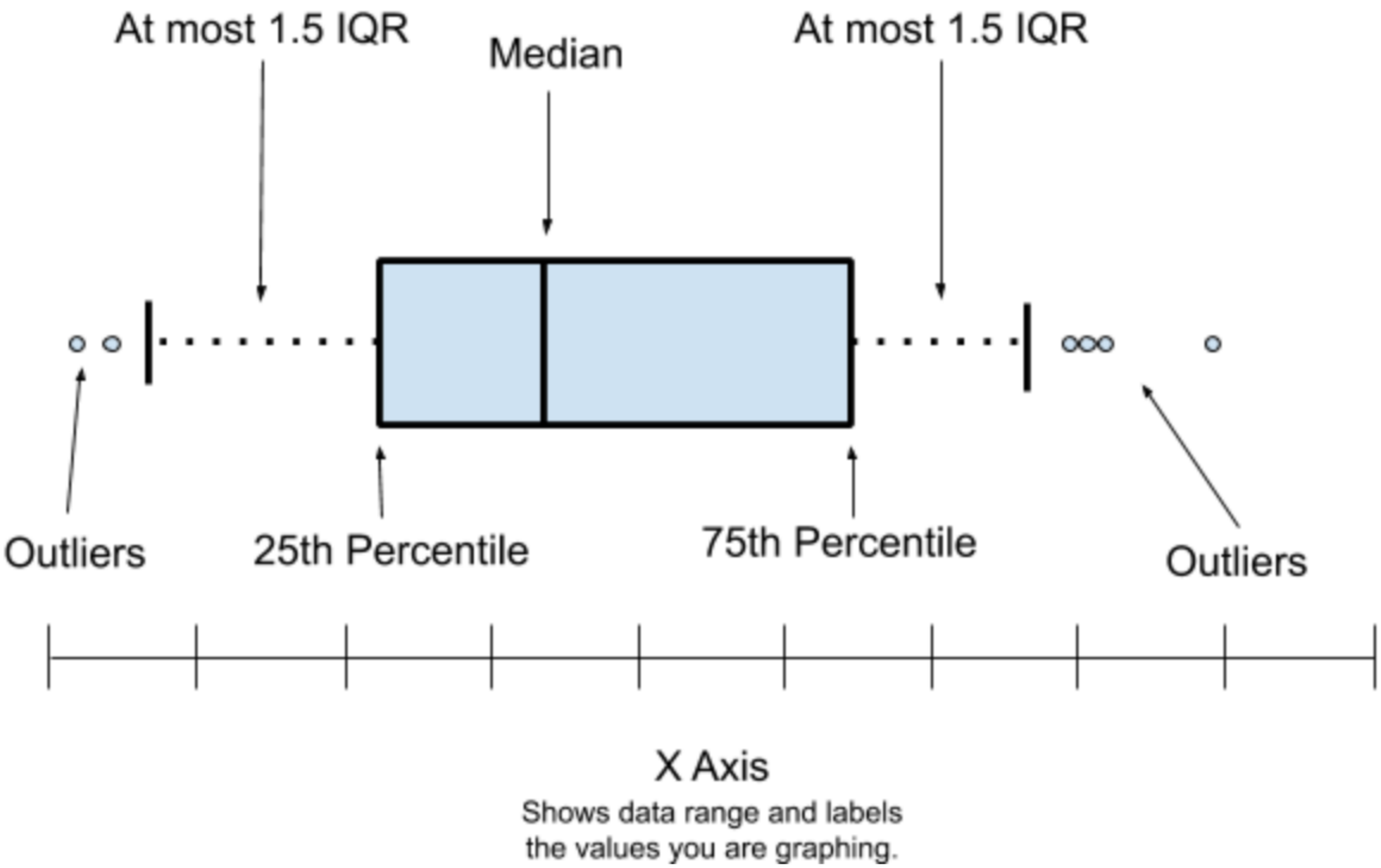
```
hist(gapminder_2015$life_expectancy,
     xlab   = "Life Expectancy",
     main   = "Histogram of life expectancy in 2015",
     breaks = seq(from = 40, to = 90, by = 5),
     las = 1, # horizontal y-axis values
     col  = "#d90502",
     border = "white")
```



Looking for Outliers: Boxplots

- An *outlier* is a datapoint far removed from the center of a distribution.
- Boxplots are an effective way to visualise the distribution of a variable.
- The *box* typically denotes the **interquartile range** (observations between 25th pctile and 75th pctile).
- The *thick line* corresponds to the **median**.
- The *dots* are **outliers** (⚠️ no universally accepted definition).

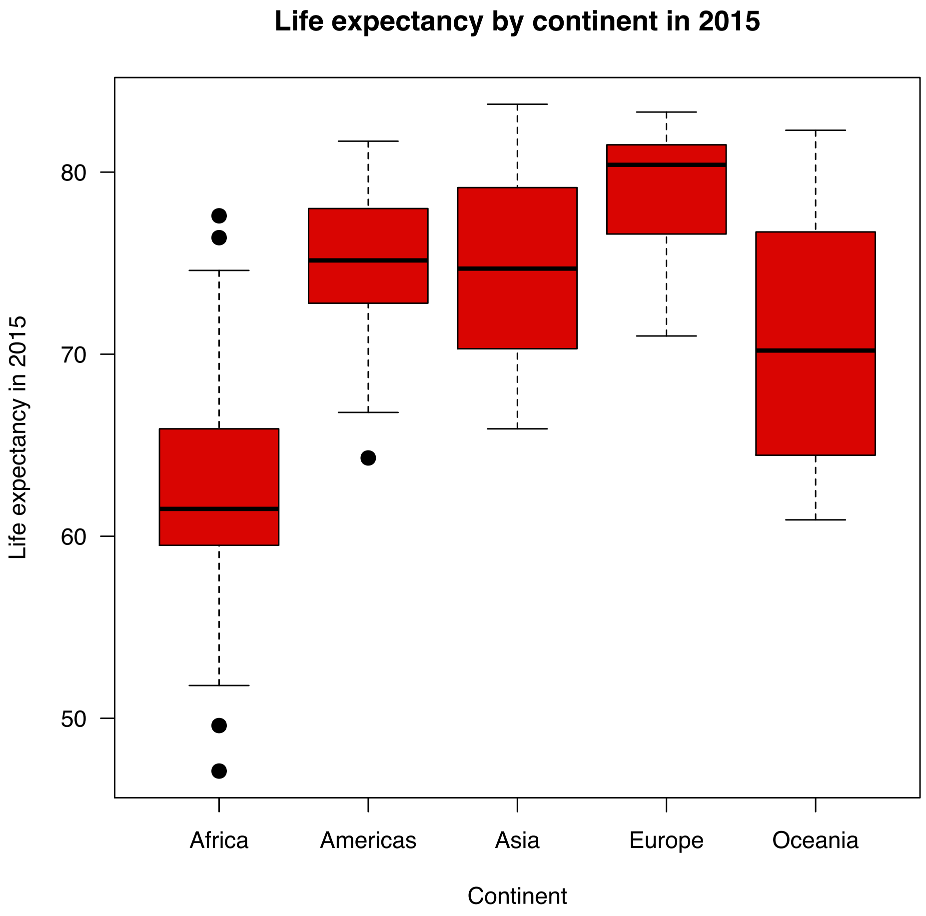
Looking for Outliers: Boxplots



Looking for Outliers: Boxplots

```
boxplot(life_expectancy ~ continent,
    data = gapminder_2015,
    xlab   = "Continent",
    ylab   = "Life expectancy in 2015",
    main   = "Life expectancy by continent in 2015",
    pch = 20, cex = 2, # colour and size of outliers
    col ="#d90502",border = "black", las = 1)
```

- see `?boxplot` for more options



Scatter Plots

- Two variables x and y

Scatter Plots

- Two variables x and y
- Natural to ask: How often do certain pairs of (x_i, y_i) occur?

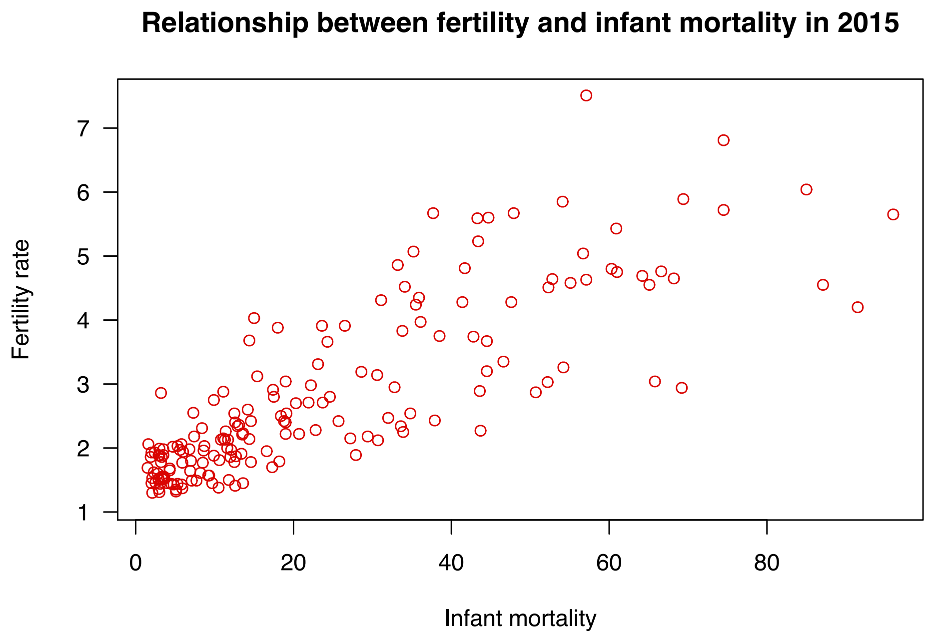
```
head(gapminder_2015[,c("fertility","infant_mortality")])
```

```
##       fertility infant_mortality
## 10176      1.78             12.5
## 10177      2.71             21.9
## 10178      5.65             96.0
## 10179      2.06              5.8
## 10180      2.15             11.1
## 10181      1.41             12.6
```

- That's what a scatter plots shows.

Scatter Plots

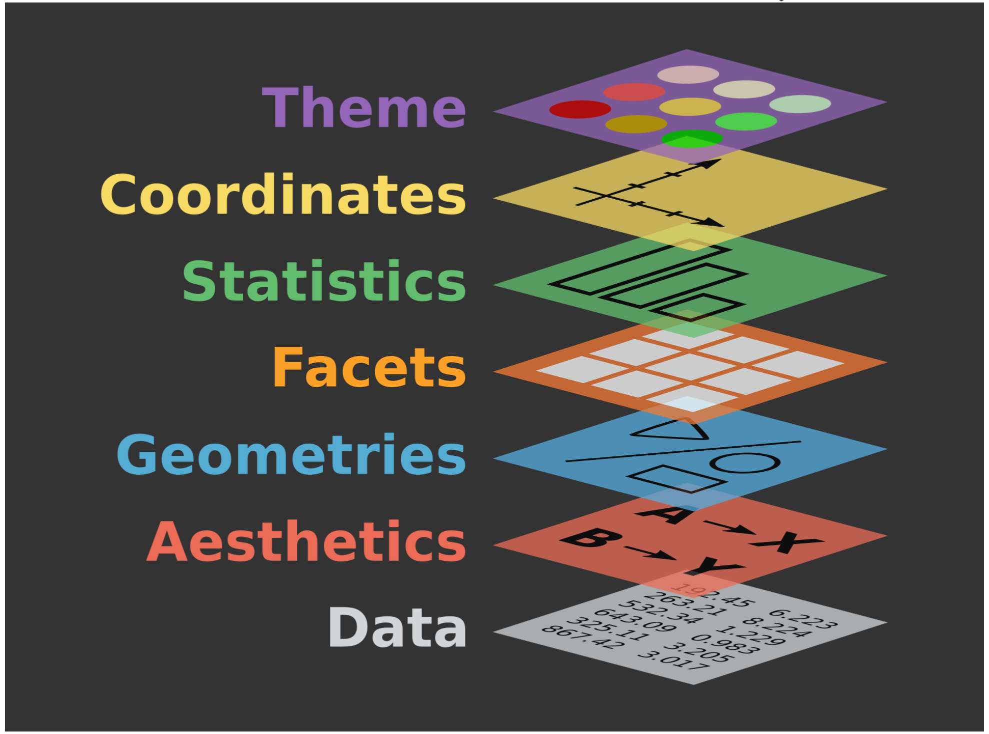
```
plot(fertility ~ infant_mortality,
    data = gapminder_2015,
    xlab   = "Infant mortality",
    ylab   = "Fertility rate",
    main   = "Relationship between fertility and infar
    col = "#d90502",
    las = 1)
```



- Each dot is one pair (x_i, y_i).
- We often call it one *observation*.
- Corresponding to one *row* of the `data.frame`.
- Why do some dots appear *darker* than others here?

Quick `ggplot2` Intro

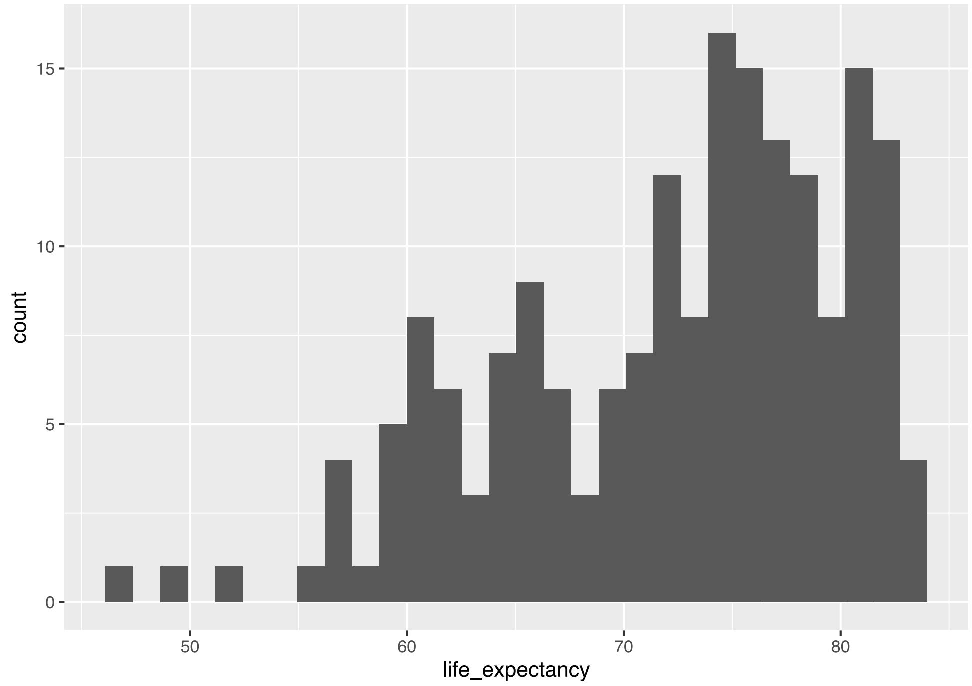
- Excellent cheatsheet on project website.
- Great intro to `ggplot2` here.
- Based on *The **G**rammar of **G**raphics* (hence **gg**plot).
- More powerful than base `R` plotting
- Let's reproduce the previous graphs in ggplot



source: BloggoType

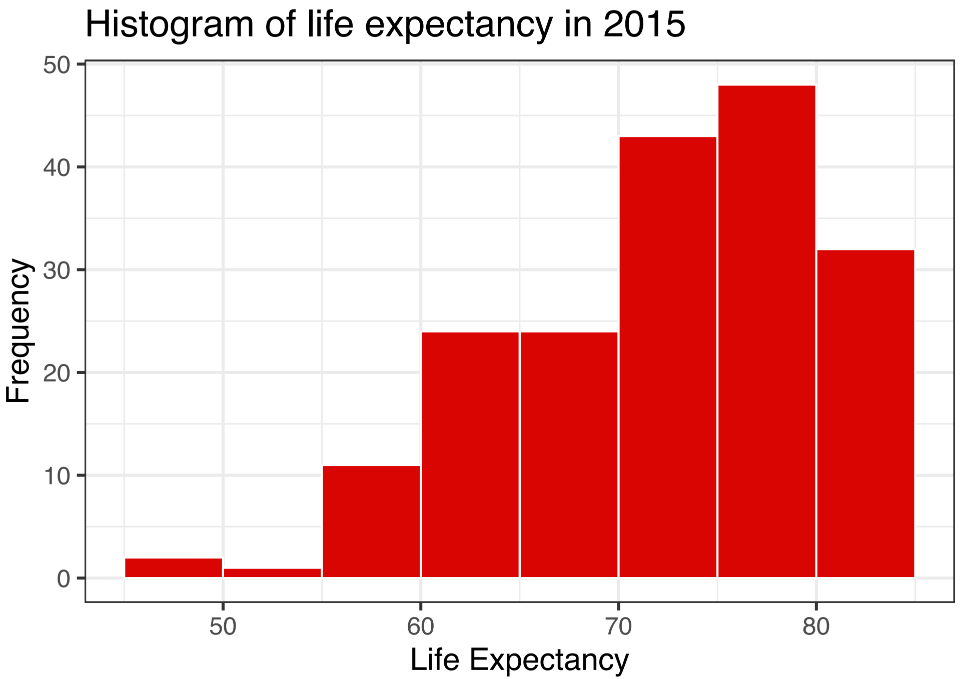
`ggplot2`: Basic Histogram

```
library(ggplot2)

ggplot(gapminder_2015,
       aes(x = life_expectancy)) +
  geom_histogram()
```



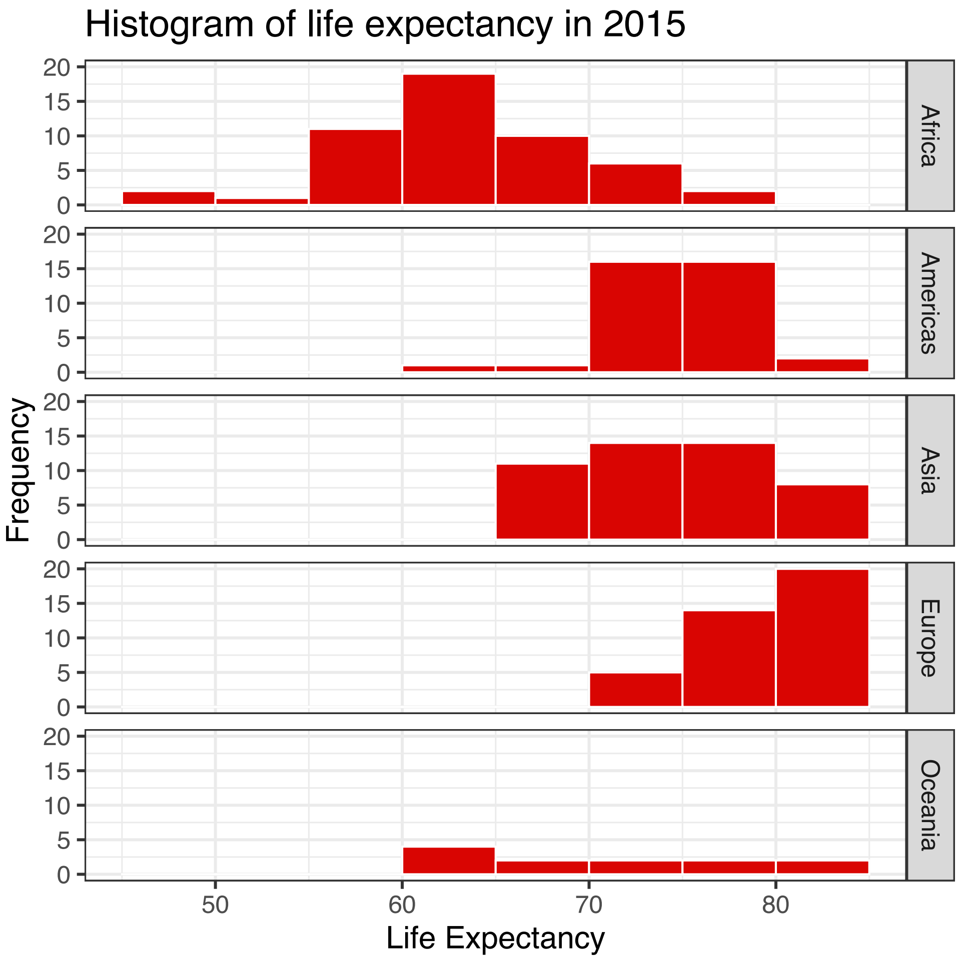
ggplot2: Fancy Histogram

```
library(ggplot2)

ggplot(gapminder_2015,
       aes(x = life_expectancy)) +
  geom_histogram(binwidth = 5,
                 boundary = 45,
                 colour = "white",
                 fill = "#d90502") +
  labs(x = "Life Expectancy",
       y = "Frequency",
       title = "Histogram of life expectancy in 2015")
  theme_bw(base_size = 16)
```



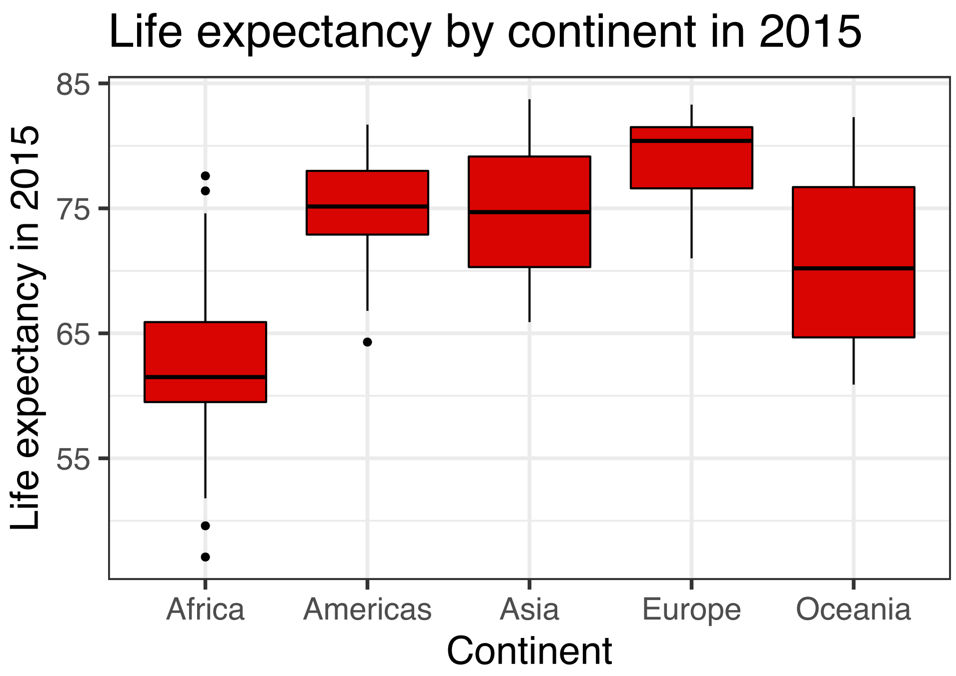
ggplot2: Fancy Histogram with facet_grid()

```
library(ggplot2)

ggplot(gapminder_2015,
       aes(x = life_expectancy)) +
  geom_histogram(binwidth = 5,
                 boundary = 45,
                 colour = "white",
                 fill = "#d90502") +
  labs(x = "Life Expectancy",
       y = "Frequency",
       title = "Histogram of life expectancy in 2015")
  theme_bw(base_size = 16) +
  facet_grid(rows = vars(continent))
```



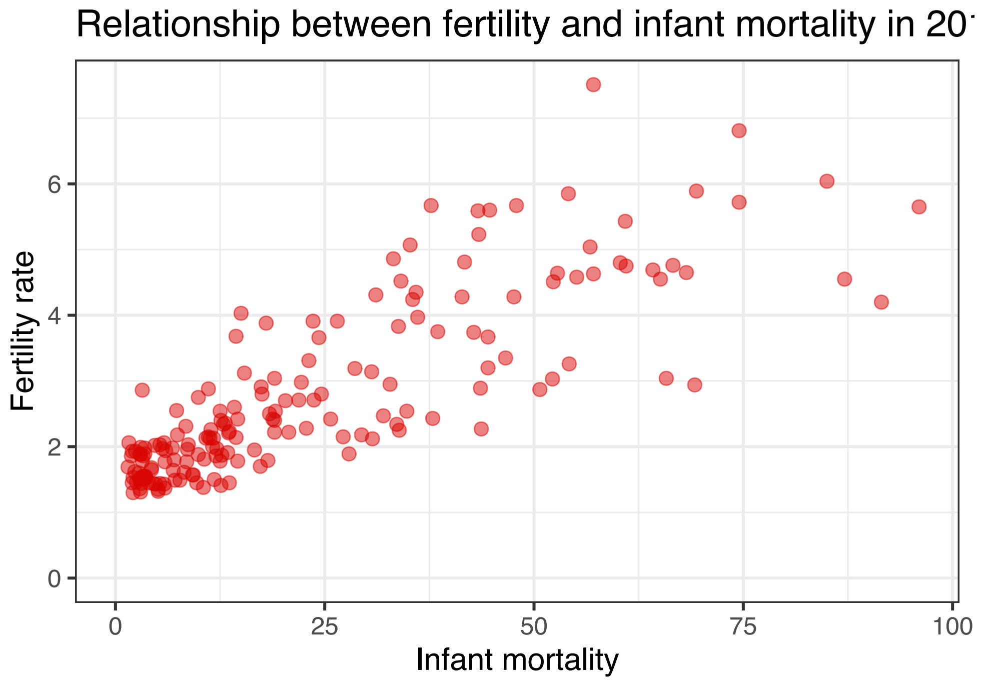
ggplot2: Boxplots

```
ggplot(gapminder_2015,
       aes(x = continent, y = life_expectancy)) +
  geom_boxplot(colour = "black",
               fill = "#d90502") +
  labs(x = "Continent",
       y = "Life expectancy in 2015",
       title = "Life expectancy by continent in 2015")
  theme_bw(base_size = 20)
```



ggplot2: Scatter Plots

```
ggplot(gapminder_2015,
       aes(x = infant_mortality,
           y = fertility)) +
  geom_point(size = 3,
             alpha = 0.5,
             colour = "#d90502") +
  expand_limits(x = 0, y = 0) +
  labs(x = "Infant mortality",
       y = "Fertility rate",
       title = "Relationship between fertility and int
  theme_bw(base_size = 16)
```



It's Tutorial Time!

Tutorial 1 (10 minutes)

Time for our first tutorial!!

Type this into your `RStudio` console:

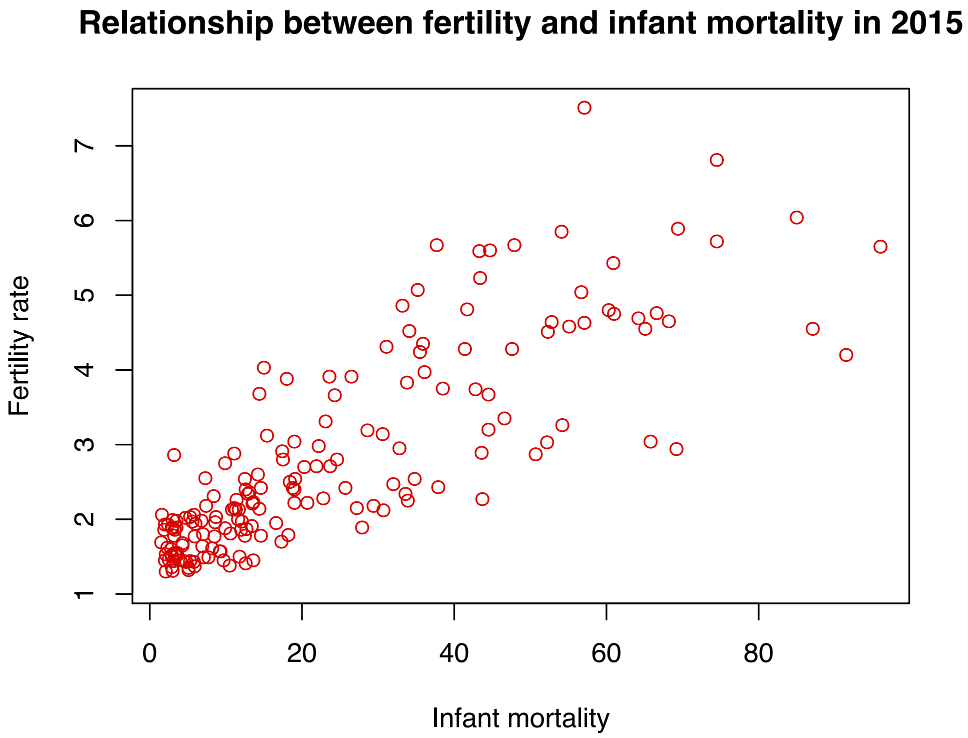
```
library(ScPoEconometrics)
runTutorial('chapter2')
```

If you have trouble with the interactive doc, try this version (no interactive content):

```
ScPoEconometrics::runTutorial('chapter2-script')
```

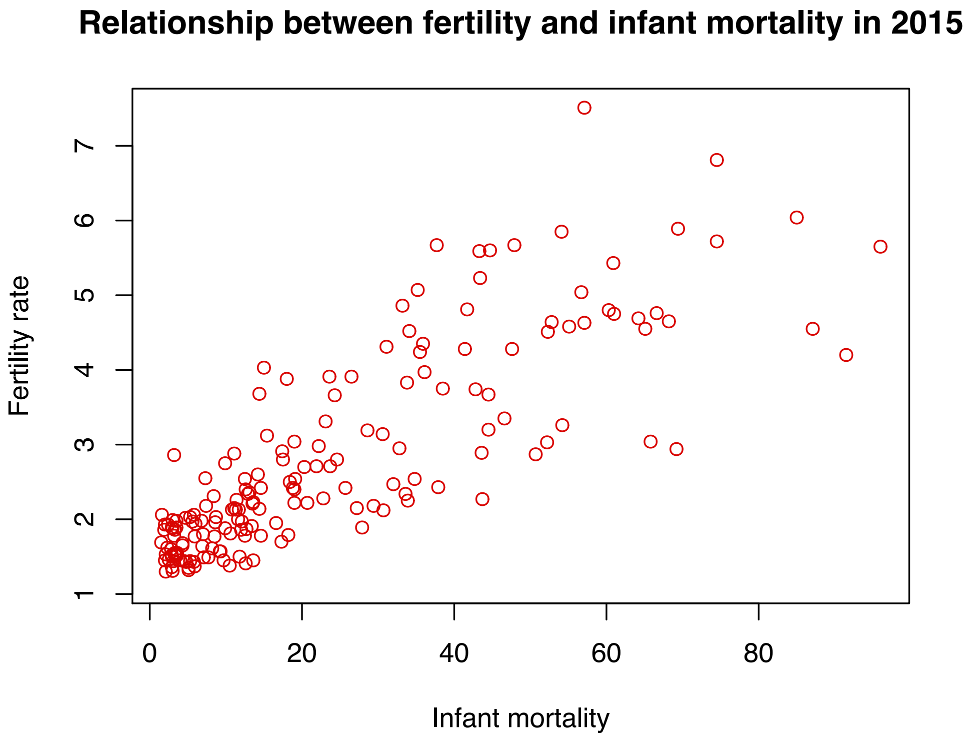
How are x and y related? Covariance and Correlation

- This is the relevant section in the book about Covariance.



How are x and y related? Covariance and Correlation

- This is the relevant section in the book about Covariance.



- The covariance is a measure of **joint variability** of two variables.

$$Cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

```
## [1] 24.21146
```

- The correlation is a measure of the strenght of the **linear association** between two variables.

$$Cor(x,y) = \frac{Cov(x,y)}{\sqrt(Var(x))\sqrt(Var(y))}$$

```
## [1] 0.8286402
```

Correlation App

```r
library(ScPoEconometrics)
runTutorial('correlation')
```

Wrangling

Intro to `dplyr`

- `dplyr` is part of the tidyverse package family.
- `data.table` is an alternative. Very fast but a bit more difficult.
- Both have pros and cons. We'll start you off with `dplyr`.

`dplyr` Overview

- You *must* read through Hadley Wickham's chapter. It's concise.
- The package is organized around a set of **verbs**, i.e. *actions* to be taken.
- We operate on `data.frames` or `tibbles` (*nicer looking* data.frames.)
- All *verbs*: First argument is a data.frame, subsequent arguments describe what to do, returns another data.frame.

`dplyr` Overview

- You *must* read through Hadley Wickham's chapter. It's concise.
- The package is organized around a set of **verbs**, i.e. *actions* to be taken.
- We operate on `data.frames` or `tibbles` (*nicer looking* data.frames.)
- All *verbs*: First argument is a data.frame, subsequent arguments describe what to do, returns another data.frame.

Verbs

1. `filter()`: Choose observations based on a certain value (i.e. subset)
2. `arrange()`: Reorder rows
3. `select()`: Select variables by name
4. `mutate()`: Create new variables out of existing ones
5. `summarise()`: Summarise variables

Data on 2016 US election polls from the `dslabs` package

- This dataset contains **real** data on polls made during the 2016 US Presidential elections and compiled by fivethirtyeight

```
library(dslabs)
library(dplyr)
data(polls_us_election_2016) # this data is from fivethirtyeight.com
polls_us_election_2016 <- as_tibble(polls_us_election_2016)
head(polls_us_election_2016[,1:6], n = 6) # show first 6 lines of first 6 variables
```

```
## # A tibble: 6 x 6
##   state startdate  enddate    pollster                           grade samplesize
##   <fct> <date>     <date>     <fct>                              <fct>      <int>
## 1 U.S.  2016-11-03 2016-11-06 ABC News/Washington Post           A+          2220
## 2 U.S.  2016-11-01 2016-11-07 Google Consumer Surveys            B          26574
## 3 U.S.  2016-11-02 2016-11-06 Ipsos                              A-          2195
## 4 U.S.  2016-11-04 2016-11-07 YouGov                             B           3677
## 5 U.S.  2016-11-03 2016-11-06 Gravis Marketing                   B-         16639
## 6 U.S.  2016-11-03 2016-11-06 Fox News/Anderson Robbins Resear… A           1295
```

🚨 This is a `tibble` (more informative than `data.frame`)

What variables does this dataset contain?

`filter()`: subset a data.frame

- `filter` has the same purpose as `subset`
- Example: Which A graded poll with at least 2,000 people had Trump win at least 45% of the vote?

```
filter(polls_us_election_2016,
       grade == "A" & samplesize > 2000 & rawpoll_trump > 45)
```

`filter()`: subset a data.frame

- `filter` has the same purpose as `subset`
- Example: Which A graded poll with at least 2,000 people had Trump win at least 45% of the vote?

```
filter(polls_us_election_2016,
       grade == "A" & samplesize > 2000 & rawpoll_trump > 45)
```

```
## # A tibble: 1 x 15
##   state startdate  enddate    pollster grade samplesize population
##   <fct> <date>     <date>     <fct>    <fct>      <int> <chr>
## 1 Indi… 2016-04-26 2016-04-28 Marist … A           2149 rv
## # … with 8 more variables: rawpoll_clinton <dbl>, rawpoll_trump <dbl>,
## #   rawpoll_johnson <dbl>, rawpoll_mcmullin <dbl>, adjpoll_clinton <dbl>,
## #   adjpoll_trump <dbl>, adjpoll_johnson <dbl>, adjpoll_mcmullin <dbl>
```

Create a Filter: Comparisons and Logical Operators

- We have a standard suite of comparison operators:
  - `>`: greater than,
  - `<`: smaller than,
  - `>=`: greater than or equal to,
  - `<=`: smaller than or equal to,
  - `!=`: not equal to,
  - `==`: equal to.

- Construct more complex filters with logical operators
  1. `x & y`: `x` **and** `y`
  2. `x | y`: `x` **or** `y`
  3. `!y`: **not** `y`

- `R` has the convenient `x %in% y` operator (conversely `!(x %in% y)`), `TRUE` if `x` is *a member of* `y`.

```
3 %in% 1:3
```

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

```
c(2,5) %in% 2:10  # also vectorized
```

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

```
c("S","Po") %in% c("Sciences","Po")  # also strin
```

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

`mutate()`: create new variables

- *Example*: What was
  1. the combined vote share of Trump and Clinton for each poll?
  2. the difference between Trump's raw poll vote share and 538's adjusted vote share?

```
mutate(polls_us_election_2016,
       trump_clinton_tot = rawpoll_trump + rawpoll_clinton,
       trump_raw_adj_diff = rawpoll_trump - adjpoll_trump)
```

`select()`: only keep some variables

- *Example*: Only keep the variables `state,startdate,enddate,pollster,rawpoll_clinton,rawpoll_trump`

```
select(polls_us_election_2016,
       state,startdate,enddate,pollster,rawpoll_clinton,rawpoll_trump)
```

Task 2 (10 minutes)

1. Which polls had more vote intentions for Trump than for Clinton.
2. How many polls have a missing `grade`?
3. Which polls were (i) polled by American Strategies, GfK Group or Merrill Poll, *and* (ii) had a sample size greater than 1,000, *and* (iii) started on October 20th, 2016?

For the following questions you should use `filter` and `mutate`.

1. Which polls (i) did not have missing poll data for Johnson, (ii) had a combined raw poll vote share for Trump and Clinton greater than 95% *and* (iii) had a sample size greater than 1,000.?
2. Which polls (i) did not poll for vote intentions for Johnson, (ii) had a difference in raw poll vote shares between Trump and Clinton greater than 5, and (iii) were done in the state of Iowa?

Split-Apply-Combine

- Often we do *some* operation **by** some group in our dataset:
  - Mean vote share for Clinton by pollster grade.
  - Maximum vote share for Trump by poll month, etc
- For this, we need to
  1. Split the data **by** group
  2. Apply to each group the operation
  3. Recombine all groups into one table
- In `dplyr`, this is achieved with `group_by()` and `summarise`.

Split-Apply-Combine

- Often we do *some* operation **by** some group in our dataset:
    - Mean vote share for Clinton by pollster grade.
    - Maximum vote share for Trump by poll month, etc
- For this, we need to
    1. Split the data **by** group
    2. Apply to each group the operation
    3. Recombine all groups into one table
- In `dplyr`, this is achieved with `group_by()` and `summarise`.

1. `group_by(polls_us_election_2016, grade)` groups/splits `polls_us_election_2016` by pollster `grade`:

```
polls_grade = group_by(polls_us_election_2016, gr
```

2. `summarise` each chunk and re-combine

```
summarise(polls_grade, mean_vote_clinton = mean(r
```

```
##   mean_vote_clinton
## 1          41.99086
```

Chaining 🔗 Commands Together: The Pipe

- The `magrittr` package gives us the *pipe* `%>%`.
- `x %>% f(y)` becomes `f(x,y)`.
- With the *pipe* you construct data *pipelines*.

Chaining 🔗 Commands Together: The Pipe

- The `magrittr` package gives us the *pipe* `%>%`.
- `x %>% f(y)` becomes `f(x,y)`.
- With the *pipe* you construct data *pipelines*.

Our above example would become:

```
polls_us_election_2016 %>%
  group_by(grade) %>%
  summarise(
    mean_vote_clinton = mean(rawpoll_clinton)
    )
```

which is equivalent to, but nicer than:

```
summarise(
  group_by(polls_us_election_2016, grade),
  mean_vote_clinton = mean(rawpoll_clinton))
```

Chaining 🔗 Commands Together: The Pipe

- The `magrittr` package gives us the *pipe* `%>%`.
- `x %>% f(y)` becomes `f(x,y)`.
- With the *pipe* you construct data *pipelines*.

Our above example would become:

```
polls_us_election_2016 %>%
  group_by(grade) %>%
  summarise(
    mean_vote_clinton = mean(rawpoll_clinton)
    )
```

which is equivalent to, but nicer than:

```
summarise(
  group_by(polls_us_election_2016, grade),
  mean_vote_clinton = mean(rawpoll_clinton))
```

Works for all `dplyr` verbs:

```
polls_us_election_2016 %>%
  mutate(trump_clinton_diff = rawpoll_trump-rawpoll_c
  filter(trump_clinton_diff>5 &
         state == "Iowa" &
         is.na(rawpoll_johnson)) %>%
  select(pollster)
```

```
## # A tibble: 3 x 1
##   pollster
##   <fct>
## 1 Ipsos
## 2 Ipsos
## 3 Ipsos
```

SciencesPo
DEPARTMENT OF ECONOMICS

SEE YOU IN TWO WEEKS!

	michele.fioretti@sciencespo.fr
	Slides
	Book
	@ScPoEcon
	@ScPoEcon