



ScPoEconometrics

Tidying, Visualising and Summarising Data

Mylène Feuillade, Gustave Kenedi, Florian Oswald and Pierre Villedieu
SciencesPo Paris
2022-02-01

Quick "Quiz" on Last Week's Material

1. From your *computer* ➦ connect to *www.wooclap.com/SCPOINTRO*

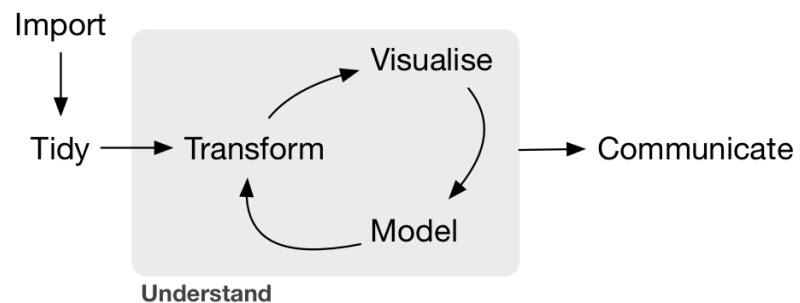
OR

2. From your *phone* ➦ flash QR code below



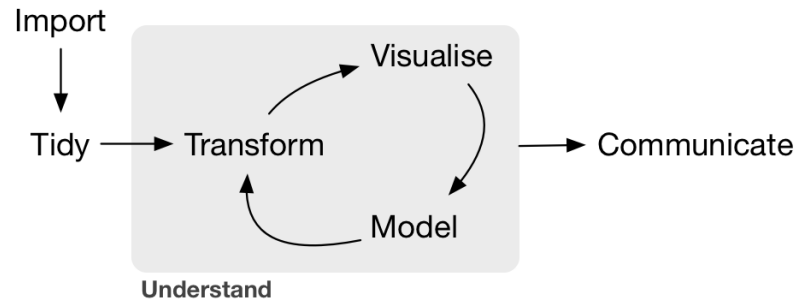
Working With Data

- 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 **tidying**, **visualising** and **summarising** data



Tidying Data

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 are *highly encouraged* to read through [Hadley Wickham's chapter](#). It's clear and concise.



dplyr Overview

- You are *highly encouraged* to read through [Hadley Wickham's chapter](#). It's clear and concise.
- Also check out this great "cheatsheet" [here](#).



dp1yr Overview

- You are *highly encouraged* to read through **Hadley Wickham's chapter**. It's clear and concise.
- Also check out this great "cheatsheet" **here**.
- 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.*)



dplyr Overview

- You are *highly encouraged* to read through **Hadley Wickham's chapter**. It's clear and concise.
- Also check out this great "cheatsheet" **here**.
- 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* work as follows:

verb(data.frame, what to do)
1st argument 2nd argument



dplyr Overview

- You are *highly encouraged* to read through **Hadley Wickham's chapter**. It's clear and concise.
- Also check out this great "cheatsheet" **here**.
- 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* work as follows:

$$\text{verb}(\underbrace{\text{data.frame}}_{\text{1st argument}}, \underbrace{\text{what to do}}_{\text{2nd argument}})$$

- Alternatively you can (should) use the `pipe` operator `%>%`:

$$\underbrace{\text{data.frame}}_{\text{1st argument}} \underbrace{\%>\%}_{\text{"pipe" operator}} \text{verb}(\underbrace{\text{what to do}}_{\text{2nd argument}})$$



Main `dplyr` Verbs

1. `filter()`: Choose observations based on a certain condition (i.e. subset)



Main `dplyr` Verbs

1. `filter()`: Choose observations based on a certain condition (i.e. subset)
2. `arrange()`: Reorder rows



Main `dplyr` Verbs

1. `filter()`: Choose observations based on a certain condition (i.e. subset)
2. `arrange()`: Reorder rows
3. `select()`: Select variables by name



Main `dplyr` Verbs

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



Main `dplyr` Verbs

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



Main `dplyr` Verbs

1. `filter()`: Choose observations based on a certain condition (i.e. subset)
2. `arrange()`: Reorder rows
3. `select()`: Select variables by name
4. `mutate()`: Create new variables out of existing ones
5. `summarise()`: Collapse data to a single summary
6. `group_by()`: All the above can be used in conjunction with `group_by()` to use function on groups rather than entire data



Data: 2016 US election polls from the `ds1abs` package

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

```
library(ds1abs)
library(tidyverse)
data(polls_us_election_2016, package = "ds1abs") # load data from package
polls_us_election_2016 <- as_tibble(polls_us_election_2016) # convert to a tibble
head(polls_us_election_2016[,1:6]) # 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`)



Data: 2016 US election polls from the `ds1abs` package

What variables does this dataset contain?

```
str(polls_us_election_2016)
```

```
## tibble [4,208 x 15] (S3: tbl_df/tbl/data.frame)
## $ state      : Factor w/ 57 levels "Alabama","Alaska",...: 50 50 50 50 50 50 50 50 37 50 ...
## $ startdate  : Date[1:4208], format: "2016-11-03" "2016-11-01" ...
## $ enddate    : Date[1:4208], format: "2016-11-06" "2016-11-07" ...
## $ pollster   : Factor w/ 196 levels "ABC News/Washington Post",...: 1 63 81 194 65 55 18 113 195 76 ...
## $ grade      : Factor w/ 10 levels "D","C-","C","C+",...: 10 6 8 6 5 9 8 8 NA 8 ...
## $ samplesize : int [1:4208] 2220 26574 2195 3677 16639 1295 1426 1282 8439 1107 ...
## $ population : chr [1:4208] "lv" "lv" "lv" "lv" ...
## $ rawpoll_clinton : num [1:4208] 47 38 42 45 47 ...
## $ rawpoll_trump   : num [1:4208] 43 35.7 39 41 43 ...
## $ rawpoll_johnson : num [1:4208] 4 5.46 6 5 3 3 5 6 6 7.1 ...
## $ rawpoll_mcmullin: num [1:4208] NA NA NA NA NA NA NA NA NA ...
## $ adjpoll_clinton : num [1:4208] 45.2 43.3 42 45.7 46.8 ...
## $ adjpoll_trump    : num [1:4208] 41.7 41.2 38.8 40.9 42.3 ...
## $ adjpoll_johnson  : num [1:4208] 4.63 5.18 6.84 6.07 3.73 ...
## $ adjpoll_mcmullin: num [1:4208] NA NA NA NA NA NA NA NA NA ...
```



dplyr Verbs

Filter observations

```
filter()
```

Example: Which polls had a sample size of at least 2,000 people?



dplyr Verbs

Filter observations

```
filter()
```

Example: Which polls had a sample size of at least 2,000 people?

```
polls_us_election_2016
```

```
## # A tibble: 4,208 x 15
##   state      startdate   enddate   pollster      grade samplesize population
##   <fct>      <date>     <date>     <fct>      <fct>      <int> <chr>
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Washin~ A+         2220 lv
## 2 U.S.      2016-11-01 2016-11-07 Google Consumer~ B          26574 lv
## 3 U.S.      2016-11-02 2016-11-06 Ipsos         A-         2195 lv
## 4 U.S.      2016-11-04 2016-11-07 YouGov        B          3677 lv
## 5 U.S.      2016-11-03 2016-11-06 Gravis Marketing B-        16639 rv
## 6 U.S.      2016-11-03 2016-11-06 Fox News/Anders~ A          1295 lv
## 7 U.S.      2016-11-02 2016-11-06 CBS News/New Yo~ A-         1426 lv
## 8 U.S.      2016-11-03 2016-11-05 NBC News/Wall S~ A-         1282 lv
## 9 New Mexico 2016-11-06 2016-11-06 Zia Poll      <NA>       8439 lv
## 10 U.S.      2016-11-04 2016-11-07 IBD/TIPP      A-         1107 lv
## # ... with 4,198 more rows, and 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>
```



dplyr Verbs

Filter observations

```
filter()
```

Example: Which polls had a sample size of at least 2,000 people?

```
polls_us_election_2016 %>%  
  filter(samplesize > 2000)
```

```
## # A tibble: 403 x 15  
##   state      startdate   enddate   pollster      grade samplesize population  
##   <fct>      <date>      <date>      <fct>      <fct>      <int> <chr>  
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Washin~ A+         2220 lv  
## 2 U.S.      2016-11-01 2016-11-07 Google Consumer~ B          26574 lv  
## 3 U.S.      2016-11-02 2016-11-06 Ipsos         A-         2195 lv  
## 4 U.S.      2016-11-04 2016-11-07 YouGov        B          3677 lv  
## 5 U.S.      2016-11-03 2016-11-06 Gravis Marketing B-         16639 rv  
## 6 New Mexico 2016-11-06 2016-11-06 Zia Poll      <NA>       8439 lv  
## 7 U.S.      2016-11-05 2016-11-07 The Times-Picay~ <NA>       2521 lv  
## 8 U.S.      2016-11-01 2016-11-07 USC Dornsife/LA~ <NA>       2972 lv  
## 9 Georgia    2016-11-03 2016-11-06 Gravis Marketing B-         2002 rv  
## 10 Virginia  2016-11-01 2016-11-02 Remington     <NA>       3076 lv  
## # ... with 393 more rows, and 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>
```



dplyr Verbs

Filter observations

```
filter()
```

Standard suite of comparison operators:

- `>`: greater than,
- `<`: smaller than,
- `>=`: greater than or equal to,
- `<=`: smaller than or equal to,
- `!=`: not equal to,
- `==`: equal to.

Logical operators:

1. `x & y`: `x and y`
2. `x | y`: `x or y`
3. `!y`: `not y`



dplyr Verbs

Filter observations

```
filter()
```

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 strings
```

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



dplyr Verbs

Filter observations

```
filter()
```

Example: Which A graded poll with at least 2,000 people had Trump win at least 45% of the vote?



dplyr Verbs

Filter observations

```
filter()
```

Example: Which A graded poll with at least 2,000 people had Trump win at least 45% of the vote?

```
polls_us_election_2016
```

```
## # A tibble: 4,208 x 15
##   state      startdate   enddate   pollster      grade samplesize population
##   <fct>      <date>      <date>      <fct>      <fct>      <int> <chr>
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Washin~ A+         2220 lv
## 2 U.S.      2016-11-01 2016-11-07 Google Consumer~ B          26574 lv
## 3 U.S.      2016-11-02 2016-11-06 Ipsos         A-         2195 lv
## 4 U.S.      2016-11-04 2016-11-07 YouGov        B          3677 lv
## 5 U.S.      2016-11-03 2016-11-06 Gravis Marketing B-         16639 rv
## 6 U.S.      2016-11-03 2016-11-06 Fox News/Anders~ A          1295 lv
## 7 U.S.      2016-11-02 2016-11-06 CBS News/New Yo~ A-         1426 lv
## 8 U.S.      2016-11-03 2016-11-05 NBC News/Wall S~ A-         1282 lv
## 9 New Mexico 2016-11-06 2016-11-06 Zia Poll      <NA>       8439 lv
## 10 U.S.      2016-11-04 2016-11-07 IBD/TIPP       A-         1107 lv
## # ... with 4,198 more rows, and 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>
```



dplyr Verbs

Filter observations

```
filter()
```

Example: Which A graded poll with at least 2,000 people had Trump win at least 45% of the vote?

```
polls_us_election_2016 %>%  
  filter(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 Indiana 2016-04-26 2016-04-28 Marist College 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>
```



dplyr Verbs

Filter

Create new variable(s)

```
mutate()
```

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?



dplyr Verbs

Filter

Create new variable(s)

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
```

```
## # A tibble: 4,208 x 15
##   state      startdate   enddate   pollster      grade samplesize population
##   <fct>      <date>      <date>      <fct>      <fct>      <int> <chr>
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Washin~ A+         2220 lv
## 2 U.S.      2016-11-01 2016-11-07 Google Consumer~ B          26574 lv
## 3 U.S.      2016-11-02 2016-11-06 Ipsos         A-         2195 lv
## 4 U.S.      2016-11-04 2016-11-07 YouGov        B          3677 lv
## 5 U.S.      2016-11-03 2016-11-06 Gravis Marketing B-        16639 rv
## 6 U.S.      2016-11-03 2016-11-06 Fox News/Anders~ A          1295 lv
## 7 U.S.      2016-11-02 2016-11-06 CBS News/New Yo~ A-         1426 lv
## 8 U.S.      2016-11-03 2016-11-05 NBC News/Wall S~ A-         1282 lv
## 9 New Mexico 2016-11-06 2016-11-06 Zia Poll      <NA>       8439 lv
## 10 U.S.      2016-11-04 2016-11-07 IBD/TIPP       A-         1107 lv
## # ... with 4,198 more rows, and 8 more variables: rawpoll_clinton <dbl>,
## #   rawpoll_trump <dbl>, rawpoll_johnson <dbl>, rawpoll_mcmullin <dbl>,
## #   adjpoll_clinton <dbl>, adjpoll_trump <dbl>, adjpoll_johnson <dbl>,
## #   adipoll_mcmullin <dbl>
```



dplyr Verbs

Filter

Create new variable(s)

```
mutate()
```

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?

```
polls_us_election_2016 %>%
```

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

```
## # A tibble: 4,208 x 17
```

```
##   state      startdate   enddate   pollster      grade samplesize population  
##   <fct>      <date>      <date>      <fct>      <fct>      <int> <chr>  
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Washin~ A+          2220 lv  
## 2 U.S.      2016-11-01 2016-11-07 Google Consumer~ B           26574 lv  
## 3 U.S.      2016-11-02 2016-11-06 Ipsos         A-          2195 lv  
## 4 U.S.      2016-11-04 2016-11-07 YouGov        B           3677 lv  
## 5 U.S.      2016-11-03 2016-11-06 Gravis Marketing B-          16639 rv  
## 6 U.S.      2016-11-03 2016-11-06 Fox News/Anders~ A           1295 lv  
## 7 U.S.      2016-11-02 2016-11-06 CBS News/New Yo~ A-          1426 lv  
## 8 U.S.      2016-11-03 2016-11-05 NBC News/Wall S~ A-          1282 lv  
## 9 New Mexico 2016-11-06 2016-11-06 Zia Poll      <NA>         8439 lv  
## 10 U.S.      2016-11-04 2016-11-07 IBD/TIPP       A-          1107 lv  
## # ... with 4,198 more rows, and 10 more variables: rawpoll_clinton <dbl>,  
## #   rawpoll_trump <dbl>, rawpoll_iohnson <dbl>, rawpoll_mcmullin <dbl>.
```



dplyr Verbs

Filter

Create new variable(s)

```
mutate()
```

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?

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

```
## [1] "state"           "startdate"       "enddate"  
## [4] "pollster"       "grade"          "samplesize"  
## [7] "population"     "rawpoll_clinton" "rawpoll_trump"  
## [10] "rawpoll_johnson" "rawpoll_mcmullin" "adjpoll_clinton"  
## [13] "adjpoll_trump"   "adjpoll_johnson" "adjpoll_mcmullin"  
## [16] "trump_clinton_tot" "trump_raw_adj_diff"
```



dplyr Verbs

Filter

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

Mutate

Keep some variable(s)

```
select()
```



dplyr Verbs

Filter

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

Mutate

Keep some variable(s)

`select()`

```
polls_us_election_2016
```

```
## # A tibble: 4,208 x 15
##   state      startdate  enddate  pollster      grade samplesize population
##   <fct>      <date>    <date>    <fct>      <fct>      <int> <chr>
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Washin~ A+         2220 lv
## 2 U.S.      2016-11-01 2016-11-07 Google Consumer~ B          26574 lv
## 3 U.S.      2016-11-02 2016-11-06 Ipsos        A-         2195 lv
## 4 U.S.      2016-11-04 2016-11-07 YouGov       B          3677 lv
## 5 U.S.      2016-11-03 2016-11-06 Gravis Marketing B-        16639 rv
## 6 U.S.      2016-11-03 2016-11-06 Fox News/Anders~ A          1295 lv
## 7 U.S.      2016-11-02 2016-11-06 CBS News/New Yo~ A-         1426 lv
## 8 U.S.      2016-11-03 2016-11-05 NBC News/Wall S~ A-         1282 lv
## 9 New Mexico 2016-11-06 2016-11-06 Zia Poll     <NA>       8439 lv
## 10 U.S.      2016-11-04 2016-11-07 IBD/TIPP     A-         1107 lv
## # ... with 4,198 more rows, and 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>
```



dplyr Verbs

Filter

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

Mutate

Keep some variable(s)

```
select()
```

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

```
## # A tibble: 4,208 x 6  
##   state      startdate   enddate   pollster      rawpoll_clinton rawpoll_trump  
##   <fct>      <date>     <date>     <fct>          <dbl>         <dbl>  
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Wash~    47            43  
## 2 U.S.      2016-11-01 2016-11-07 Google Consum~   38.0          35.7  
## 3 U.S.      2016-11-02 2016-11-06 Ipsos             42            39  
## 4 U.S.      2016-11-04 2016-11-07 YouGov           45            41  
## 5 U.S.      2016-11-03 2016-11-06 Gravis Market~   47            43  
## 6 U.S.      2016-11-03 2016-11-06 Fox News/Ande~   48            44  
## 7 U.S.      2016-11-02 2016-11-06 CBS News/New ~   45            41  
## 8 U.S.      2016-11-03 2016-11-05 NBC News/Wall~   44            40  
## 9 New Mexico 2016-11-06 2016-11-06 Zia Poll         46            44  
## 10 U.S.      2016-11-04 2016-11-07 IBD/TIPP         41.2          42.7  
## # ... with 4,198 more rows
```



dplyr Verbs

Filter

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

Mutate

Keep some variable(s)

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

```
## [1] "state"          "startdate"      "enddate"       "pollster"  
## [5] "rawpoll_clinton" "rawpoll_trump"
```

```
select()
```



dp1yr Verbs

Filter

Example: What is the maximum vote share for Trump?

Mutate

Select

Compute statistic

```
summarise()
```



dplyr Verbs

Filter

Mutate

Select

Compute statistic

```
summarise()
```

Example: What is the maximum vote share for Trump?

```
polls_us_election_2016
```

```
## # A tibble: 4,208 x 15
##   state      startdate   enddate   pollster      grade samplesize population
##   <fct>      <date>     <date>     <fct>      <fct>      <int> <chr>
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Washin~ A+         2220 lv
## 2 U.S.      2016-11-01 2016-11-07 Google Consumer~ B          26574 lv
## 3 U.S.      2016-11-02 2016-11-06 Ipsos         A-         2195 lv
## 4 U.S.      2016-11-04 2016-11-07 YouGov        B          3677 lv
## 5 U.S.      2016-11-03 2016-11-06 Gravis Marketing B-        16639 rv
## 6 U.S.      2016-11-03 2016-11-06 Fox News/Anders~ A          1295 lv
## 7 U.S.      2016-11-02 2016-11-06 CBS News/New Yo~ A-         1426 lv
## 8 U.S.      2016-11-03 2016-11-05 NBC News/Wall S~ A-         1282 lv
## 9 New Mexico 2016-11-06 2016-11-06 Zia Poll      <NA>       8439 lv
## 10 U.S.      2016-11-04 2016-11-07 IBD/TIPP      A-         1107 lv
## # ... with 4,198 more rows, and 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>
```



dp1yr Verbs

Filter

Mutate

Select

Compute statistic

Example: What is the maximum vote share for Trump?

```
polls_us_election_2016 %>%  
  summarise(max_trump = max(rawpoll_trump))
```

```
## # A tibble: 1 x 1  
##   max_trump  
##       <dbl>  
## 1         68
```

```
summarise()
```



dplyr Verbs

Filter

Example: What is the average vote share for Clinton by poll grade?

Mutate

Select

Summarise

Apply function by
group

```
group_by()
```



dp1yr Verbs

Filter

Example: What is the average vote share for Clinton by poll grade?

Mutate

Select

Summarise

Apply function by group

```
polls_us_election_2016
```

```
## # A tibble: 4,208 x 15
##   state      startdate   enddate   pollster      grade samplesize population
##   <fct>      <date>      <date>      <fct>      <fct>      <int> <chr>
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Washin~ A+         2220 lv
## 2 U.S.      2016-11-01 2016-11-07 Google Consumer~ B          26574 lv
## 3 U.S.      2016-11-02 2016-11-06 Ipsos         A-         2195 lv
## 4 U.S.      2016-11-04 2016-11-07 YouGov        B          3677 lv
## 5 U.S.      2016-11-03 2016-11-06 Gravis Marketing B-        16639 rv
## 6 U.S.      2016-11-03 2016-11-06 Fox News/Anders~ A          1295 lv
## 7 U.S.      2016-11-02 2016-11-06 CBS News/New Yo~ A-         1426 lv
## 8 U.S.      2016-11-03 2016-11-05 NBC News/Wall S~ A-         1282 lv
## 9 New Mexico 2016-11-06 2016-11-06 Zia Poll      <NA>       8439 lv
## 10 U.S.      2016-11-04 2016-11-07 IBD/TIPP      A-         1107 lv
## # ... with 4,198 more rows, and 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>
```

```
group_by()
```



dp1yr Verbs

Filter

Example: What is the average vote share for Clinton by poll grade?

Mutate

```
polls_us_election_2016 %>%  
  group_by(grade)
```

Select

Summarise

Apply function by
group

```
group_by()
```

```
## # A tibble: 4,208 x 15  
## # Groups:   grade [11]  
##   state      startdate   enddate   pollster      grade samplesize population  
##   <fct>      <date>      <date>      <fct>      <fct>      <int> <chr>  
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Washin~ A+         2220 lv  
## 2 U.S.      2016-11-01 2016-11-07 Google Consumer~ B          26574 lv  
## 3 U.S.      2016-11-02 2016-11-06 Ipsos         A-         2195 lv  
## 4 U.S.      2016-11-04 2016-11-07 YouGov        B          3677 lv  
## 5 U.S.      2016-11-03 2016-11-06 Gravis Marketing B-         16639 rv  
## 6 U.S.      2016-11-03 2016-11-06 Fox News/Anders~ A          1295 lv  
## 7 U.S.      2016-11-02 2016-11-06 CBS News/New Yo~ A-         1426 lv  
## 8 U.S.      2016-11-03 2016-11-05 NBC News/Wall S~ A-         1282 lv  
## 9 New Mexico 2016-11-06 2016-11-06 Zia Poll      <NA>       8439 lv  
## 10 U.S.      2016-11-04 2016-11-07 IBD/TIPP       A-         1107 lv  
## # ... with 4,198 more rows, and 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>
```



dplyr Verbs

Filter

Example: What is the average vote share for Clinton by poll grade?

Mutate

```
polls_us_election_2016 %>%  
  group_by(grade) %>%  
  class()
```

Select

```
## [1] "grouped_df" "tbl_df"      "tbl"        "data.frame"
```

Summarise

Apply function by
group

```
group_by()
```



dp1yr Verbs

Filter

Example: What is the average vote share for Clinton by poll grade?

Mutate

```
polls_us_election_2016 %>%  
  group_by(grade)
```

Select

Summarise

Apply function by
group

```
group_by()
```

```
## # A tibble: 4,208 x 15  
## # Groups:   grade [11]  
##   state      startdate   enddate   pollster      grade samplesize population  
##   <fct>      <date>      <date>      <fct>      <fct>      <int> <chr>  
## 1 U.S.      2016-11-03 2016-11-06 ABC News/Washin~ A+          2220 lv  
## 2 U.S.      2016-11-01 2016-11-07 Google Consumer~ B           26574 lv  
## 3 U.S.      2016-11-02 2016-11-06 Ipsos         A-          2195 lv  
## 4 U.S.      2016-11-04 2016-11-07 YouGov        B           3677 lv  
## 5 U.S.      2016-11-03 2016-11-06 Gravis Marketing B-          16639 rv  
## 6 U.S.      2016-11-03 2016-11-06 Fox News/Anders~ A           1295 lv  
## 7 U.S.      2016-11-02 2016-11-06 CBS News/New Yo~ A-          1426 lv  
## 8 U.S.      2016-11-03 2016-11-05 NBC News/Wall S~ A-          1282 lv  
## 9 New Mexico 2016-11-06 2016-11-06 Zia Poll      <NA>        8439 lv  
## 10 U.S.      2016-11-04 2016-11-07 IBD/TIPP       A-          1107 lv  
## # ... with 4,198 more rows, and 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>
```



dplyr Verbs

Filter

Example: What is the average vote share for Clinton by poll grade?

Mutate

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

Select

Summarise

Apply function by
group

```
## # A tibble: 11 x 2  
##   grade mean_vote_clinton  
##   <fct>          <dbl>  
## 1 D              46.7  
## 2 C-             43.2  
## 3 C              41.8  
## 4 C+            44.2  
## 5 B-            43.9  
## 6 B              37.3  
## 7 B+            44.1  
## 8 A-            43.0  
## 9 A             45.3  
## 10 A+           45.8  
## 11 <NA>         43.2
```

```
group_by()
```



Chaining Commands Together

Works for all `dplyr` verbs:

```
polls_us_election_2016
```

```
## # A tibble: 4,208 x 15
##   state      startdate   enddate   pollster      grade sar
##   <fct>      <date>      <date>      <fct>          <fct>
## 1 U.S.      2016-11-03  2016-11-06 ABC News/Washin~ A+
## 2 U.S.      2016-11-01  2016-11-07 Google Consumer~ B
## 3 U.S.      2016-11-02  2016-11-06 Ipsos           A-
## 4 U.S.      2016-11-04  2016-11-07 YouGov          B
## 5 U.S.      2016-11-03  2016-11-06 Gravis Marketing B-
## 6 U.S.      2016-11-03  2016-11-06 Fox News/Anders~ A
## 7 U.S.      2016-11-02  2016-11-06 CBS News/New Yo~ A-
## 8 U.S.      2016-11-03  2016-11-05 NBC News/Wall S~ A-
## 9 New Mexico 2016-11-06  2016-11-06 Zia Poll        <NA>
## 10 U.S.      2016-11-04  2016-11-07 IBD/TIPP        A-
## # ... with 4,198 more rows, and 8 more variables: rawpoll_clin
## #   rawpoll_trump <dbl>, rawpoll_johnson <dbl>, rawpoll_mcmull
## #   adjpoll_clinton <dbl>, adjpoll_trump <dbl>, adjpoll_johnso
## #   adjpoll_mcmullin <dbl>
```



Chaining Commands Together

Works for all `dplyr` verbs:

```
polls_us_election_2016 %>%  
  mutate(trump_clinton_diff = rawpoll_trump-rawpoll_
```

```
## # A tibble: 4,208 x 16  
##   state      startdate   enddate   pollster      grade sar  
##   <fct>      <date>      <date>      <fct>      <fct>  
## 1 U.S.      2016-11-03  2016-11-06  ABC News/Washin~ A+  
## 2 U.S.      2016-11-01  2016-11-07  Google Consumer~ B  
## 3 U.S.      2016-11-02  2016-11-06  Ipsos          A-  
## 4 U.S.      2016-11-04  2016-11-07  YouGov         B  
## 5 U.S.      2016-11-03  2016-11-06  Gravis Marketing B-  
## 6 U.S.      2016-11-03  2016-11-06  Fox News/Anders~ A  
## 7 U.S.      2016-11-02  2016-11-06  CBS News/New Yo~ A-  
## 8 U.S.      2016-11-03  2016-11-05  NBC News/Wall S~ A-  
## 9 New Mexico 2016-11-06  2016-11-06  Zia Poll       <NA>  
## 10 U.S.      2016-11-04  2016-11-07  IBD/TIPP       A-  
## # ... with 4,198 more rows, and 9 more variables: rawpoll_clin  
## #   rawpoll_trump <dbl>, rawpoll_johnson <dbl>, rawpoll_mcmull  
## #   adjpoll_clinton <dbl>, adjpoll_trump <dbl>, adjpoll_johnso  
## #   adjpoll_mcmullin <dbl>, trump_clinton_diff <dbl>
```



Chaining Commands Together

Works for all `dplyr` verbs:

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

```
## # A tibble: 3 x 16  
##   state startdate   enddate   pollster grade samplesize popula  
##   <fct> <date>       <date>       <fct>    <fct>      <int> <chr>  
## 1 Iowa  2016-09-09 2016-09-29 Ipsos     A-           343 lv  
## 2 Iowa  2016-09-02 2016-09-22 Ipsos     A-           344 lv  
## 3 Iowa  2016-08-26 2016-09-15 Ipsos     A-           347 lv  
## # ... with 9 more variables: rawpoll_clinton <dbl>, rawpoll_tr  
## #   rawpoll_johnson <dbl>, rawpoll_mcmullin <dbl>, adjpoll_cl  
## #   adjpoll_trump <dbl>, adjpoll_johnson <dbl>, adjpoll_mcmul  
## #   trump_clinton_diff <dbl>
```



Chaining Commands Together

Works for all `dplyr` verbs:

```
polls_us_election_2016 %>%  
  mutate(trump_clinton_diff = rawpoll_trump-rawpoll_clinton,  
         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
```



Chaining Commands Together

But also with other  commands:

```
polls_us_election_2016$samplesize %>% mean(na.rm = T)
```

```
## [1] 1148.216
```



Chaining Commands Together

But also with other  commands:

```
polls_us_election_2016$samplesize %>% mean(na.rm = T)
```

```
## [1] 1148.216
```

```
polls_us_election_2016 %>% count()
```

```
## # A tibble: 1 x 1  
##       n  
##   <int>  
## 1  4208
```



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
```

- is.na(x) function 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
```

- is.na(x) function 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 is  
x <- NA
```

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

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

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

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



Task 1: Data wrangling

10:00

Load the data by running the following code:

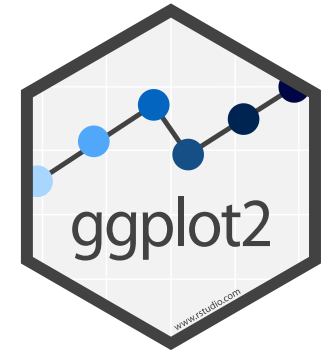
```
library(dslabs)
data(polls_us_election_2016)
```

1. Which polls had a missing `grade`?
2. Which polls were (i) polled by American Strategies, GfK Group or Merrill Poll, (ii) had a sample size greater than 1,000, *and* (iii) started on October 20th, 2016? (*Hint: for (i) `%in%` might come in handy. Recall that vectors are created using the `c()` function. For (iii) make sure to check the format of the variable containing the poll's start date.*)
3. 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) were done in the state of Ohio? (*Hint: it might be practical to first create a variable containing the combined raw poll vote share for Trump and Clinton and then filter.*)
4. Which state had the highest average Trump vote share for polls which had at least a sample size of 2,000? (*Hint: you'll have to use `filter`, `group_by`, `summarise` and `arrange`. To obtain ranking in descending order check `arrange`'s help page.*)



Visualising Data

Base R and ggplot2



- Base R plotting is fairly good.
- There is an extremely powerful alternative: ggplot2 (part of the tidyverse suite) → what we'll be using
- Let's go back to the gapminder dataset to run the examples.



The `gapminder` dataset: Overview

- Let's first load the `gapminder` dataset with these commands:

```
library(dslabs)  
data(gapminder, package = "dslabs")
```



The gapminder dataset: Overview

- Let's first load the gapminder dataset with these commands:

```
library(dslabs)
data(gapminder, package = "dslabs")
```

- Here are the first 3 rows and last 2 rows.

```
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
```

```
tail(gapminder, n = 2)
```

```
##           country year infant_mortality life_expectancy fertility population gdp
## 10544   Zambia 2016           NA           57.10           NA           NA NA
## 10545  Zimbabwe 2016           NA           61.69           NA           NA NA
##           continent      region
## 10544   Africa Eastern Africa
## 10545   Africa Eastern Africa
```



Task 2: Understanding the data

05:00

Load the data by running the following code:

```
library(dslabs)
data(gapminder, package = "dslabs")
```

1. Compute the average population per continent per year, `mean_pop`, and assign the output to a new object `gapminder_mean`. (*Hint: you should have one observation (row) per continent for each year. You'll have to use `group_by` and `summarise`.*)



gg is for Grammar of Graphics¹



[1]: The following slides are taken from [Garrick Aden-Buie's](#) wonderful [Gentle Guide to the Grammar of Graphics with `ggplot2`](#)

gg is for Grammar of Graphics

Data

```
data %>%  
  ggplot()
```

or

```
ggplot(data)
```



gg is for Grammar of Graphics

Data

```
data %>%  
  ggplot()
```

or

```
ggplot(data)
```

Tidy Data

1. Each variable forms a *column*
2. Each observation forms a *row*
3. Each observational unit forms a table



gg is for Grammar of Graphics

Data

```
data %>%  
  ggplot()
```

or

```
ggplot(data)
```

Tidy Data

1. Each variable forms a *column*
2. Each observation forms a *row*
3. Each observational unit forms a table

Start by asking

1. What information do I want to use in my visualization?
2. Is that data contained in *one column/row* for a given data point?



gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

- year
- population
- country



gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

- year → **x**
- population → **y**
- country → *shape, color*, etc.



gg is for Grammar of Graphics

Data

Aesthetics

+ aes()

Map data to visual elements or parameters

```
aes(  
  x = year,  
  y = population,  
  color = country  
)
```



gg is for Grammar of Graphics

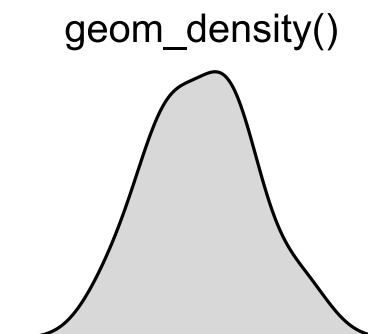
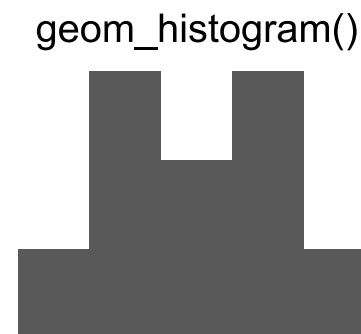
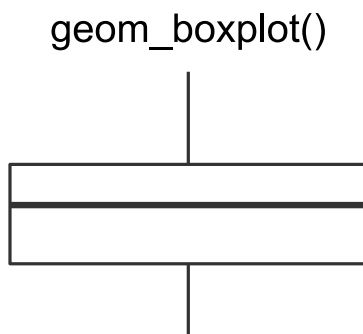
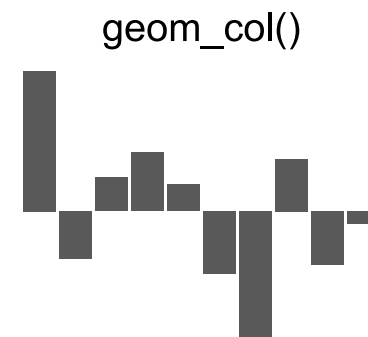
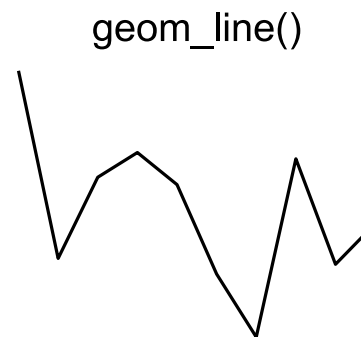
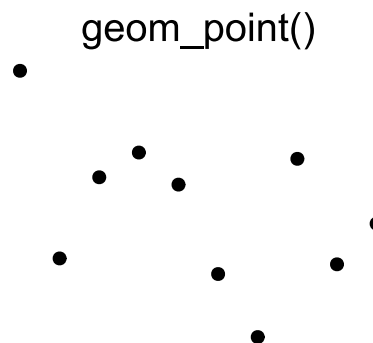
Data

Aesthetics

Geoms

```
+ geom_*()
```

Geometric objects displayed on the plot



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

+ geom_*()

Here are the **some of the most widely used geoms**

Type	Function
Point	geom_point()
Line	geom_line()
Bar	geom_bar(), geom_col()
Histogram	geom_histogram()
Regression	geom_smooth()
Boxplot	geom_boxplot()
Text	geom_text()
Vert./Horiz. Line	geom_{vh}line()
Count	geom_count()
Density	geom_density()



gg is for Grammar of Graphics

Data

Just start typing `geom_` in `RStudio` to see all the options

Aesthetics

```
ggplot(df_geom) +  
  aes(x, y) +
```

Geoms

```
+ geom_*()
```



(Y)Our first plot!

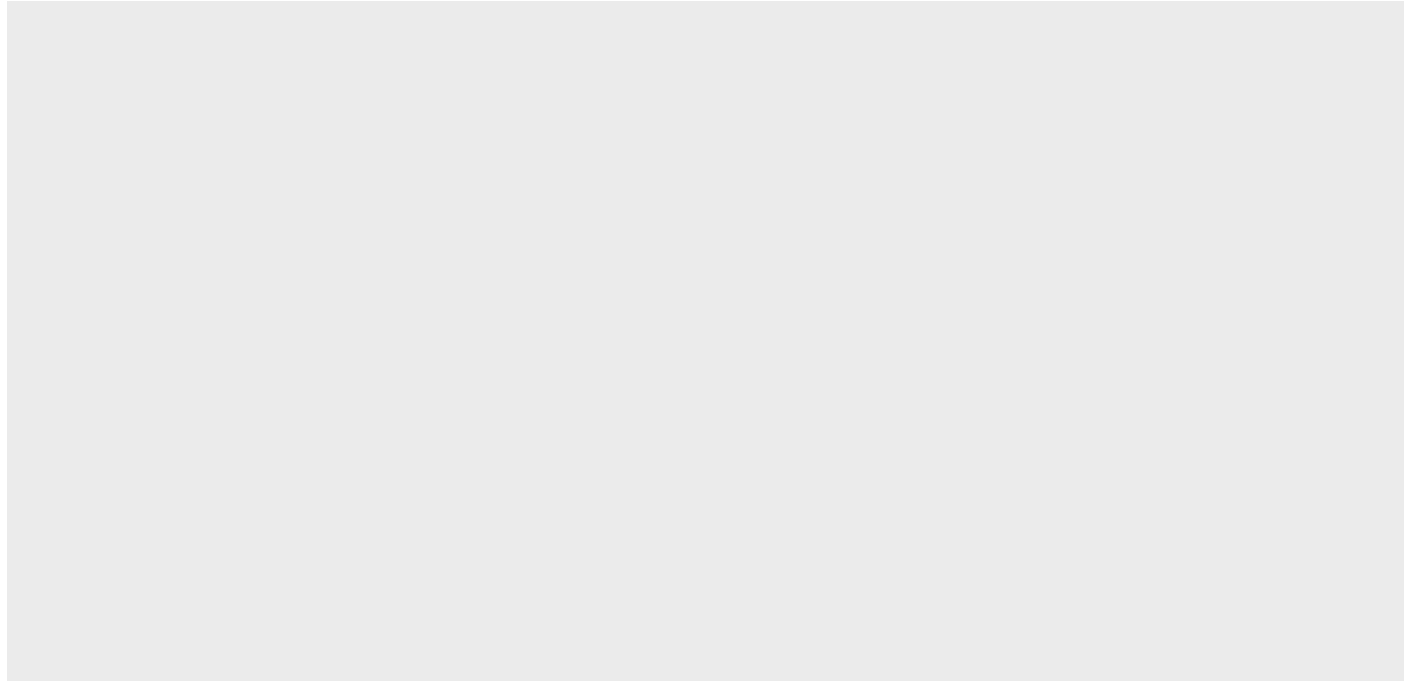
```
gapminder_mean
```

```
## # A tibble: 285 x 3
## # Groups:   continent [5]
##   continent year mean_pop
##   <fct>      <int>    <dbl>
## 1 Africa     1960  5464985.
## 2 Africa     1961  5598112.
## 3 Africa     1962  5736073.
## 4 Africa     1963  5878867.
## 5 Africa     1964  6026474.
## 6 Africa     1965  6178906.
## 7 Africa     1966  6336258.
## 8 Africa     1967  6498656.
## 9 Africa     1968  6666202.
## 10 Africa    1969  6839011.
## # ... with 275 more rows
```



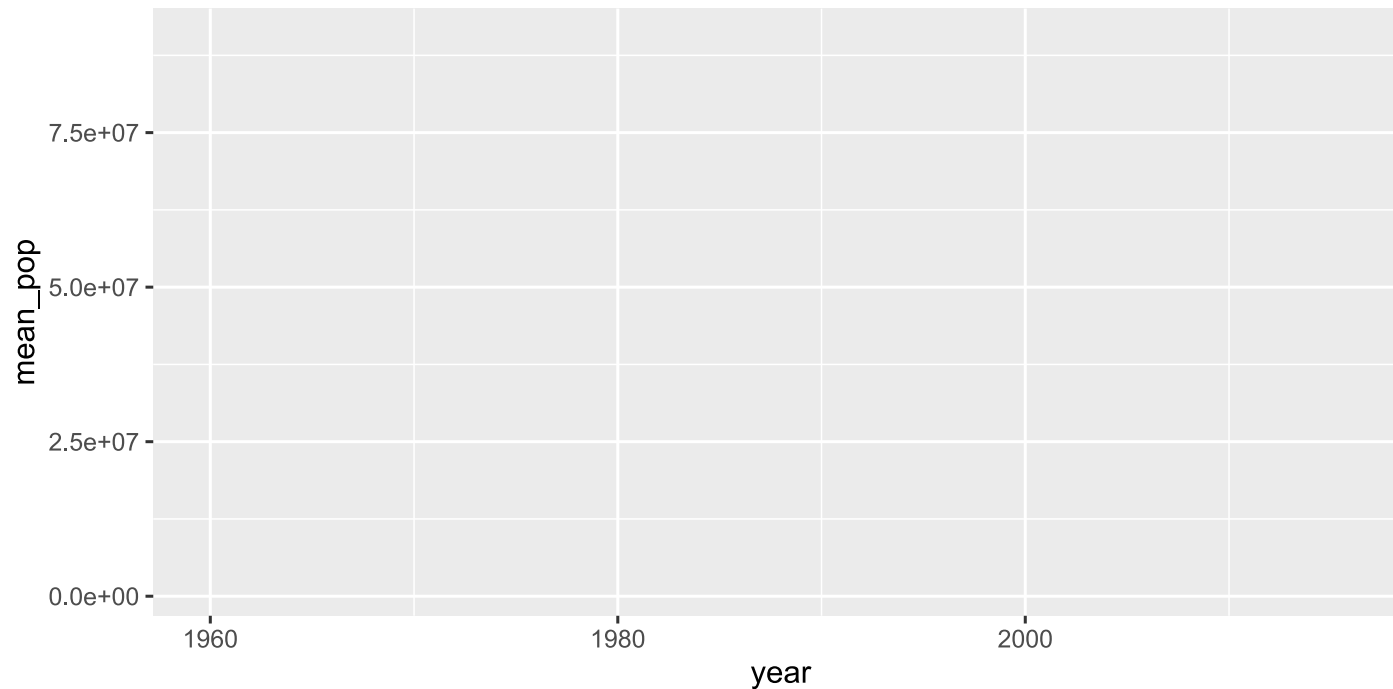
(Y)Our first plot!

```
gapminder_mean %>%  
  ggplot()
```



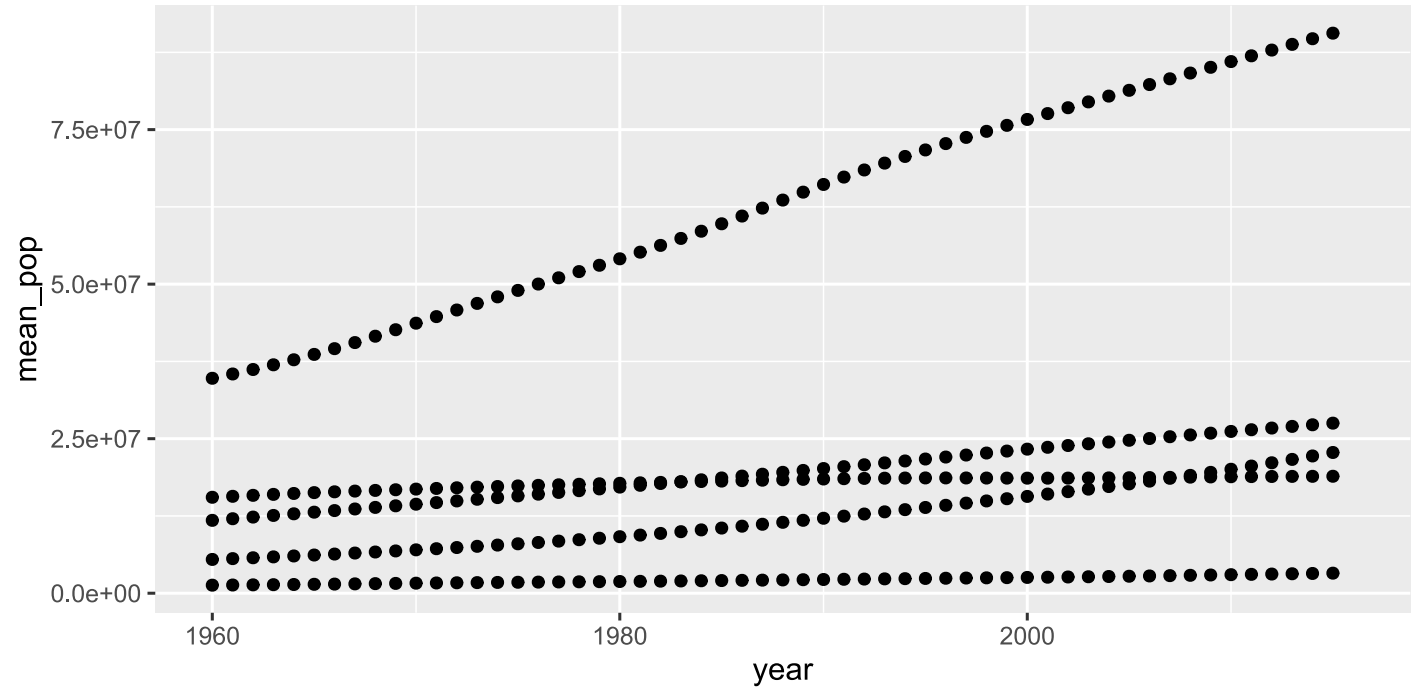
(Y)Our first plot!

```
gapminder_mean %>%  
  ggplot() +  
  aes(x = year,  
       y = mean_pop)
```



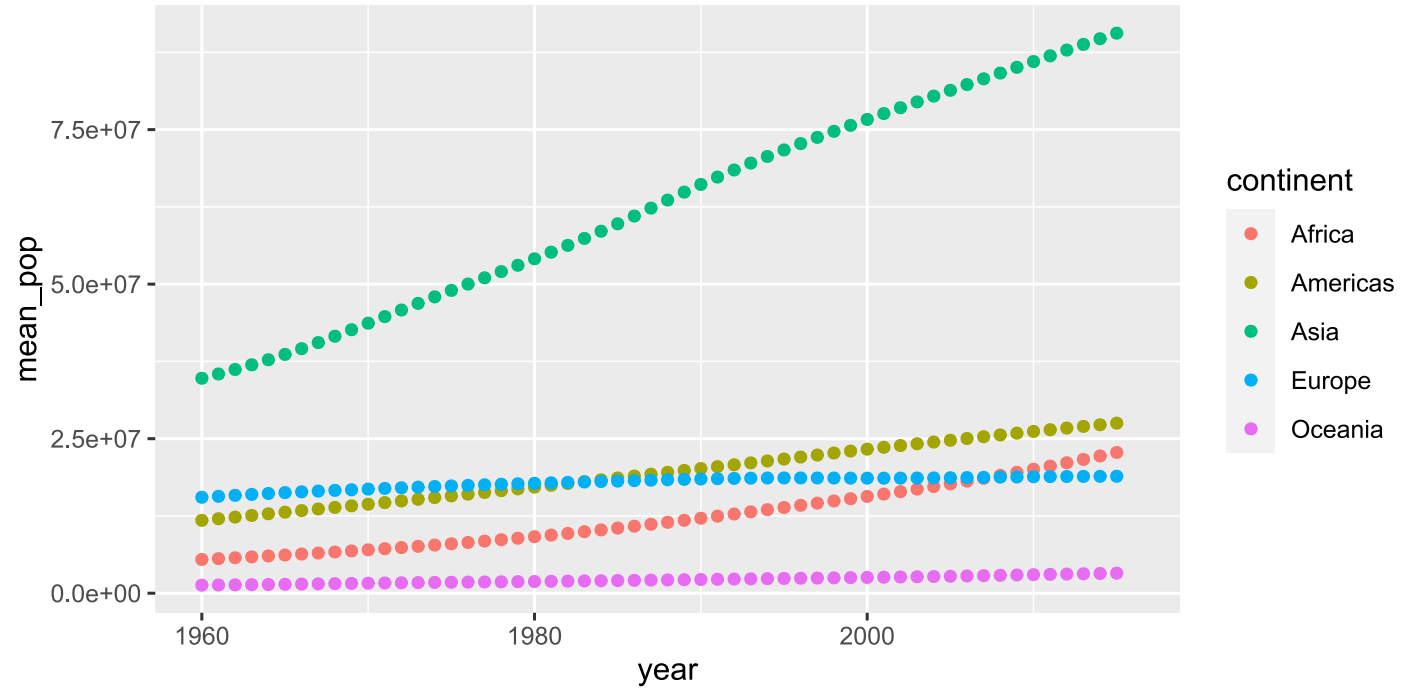
(Y)Our first plot!

```
gapminder_mean %>%  
  ggplot() +  
  aes(x = year,  
       y = mean_pop) +  
  geom_point()
```



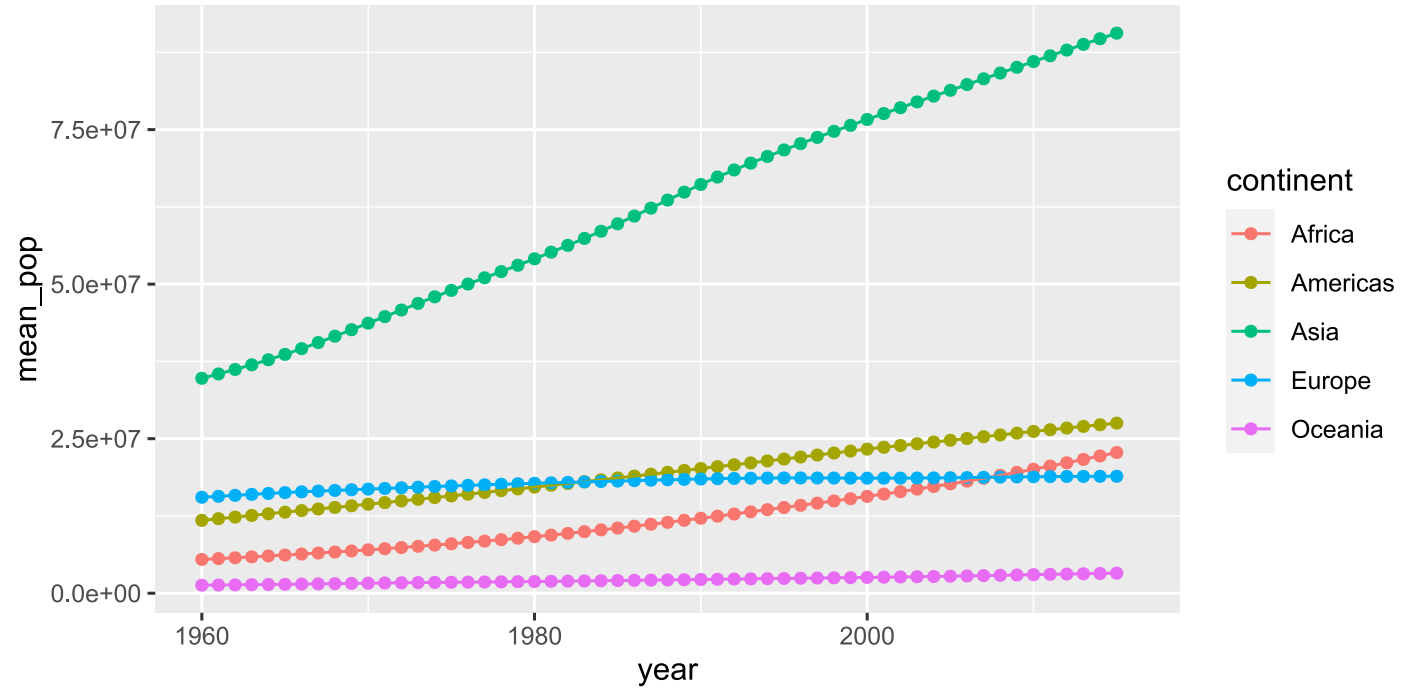
(Y)Our first plot!

```
gapminder_mean %>%  
  ggplot() +  
  aes(x = year,  
       y = mean_pop,  
       color = continent) +  
  geom_point()
```



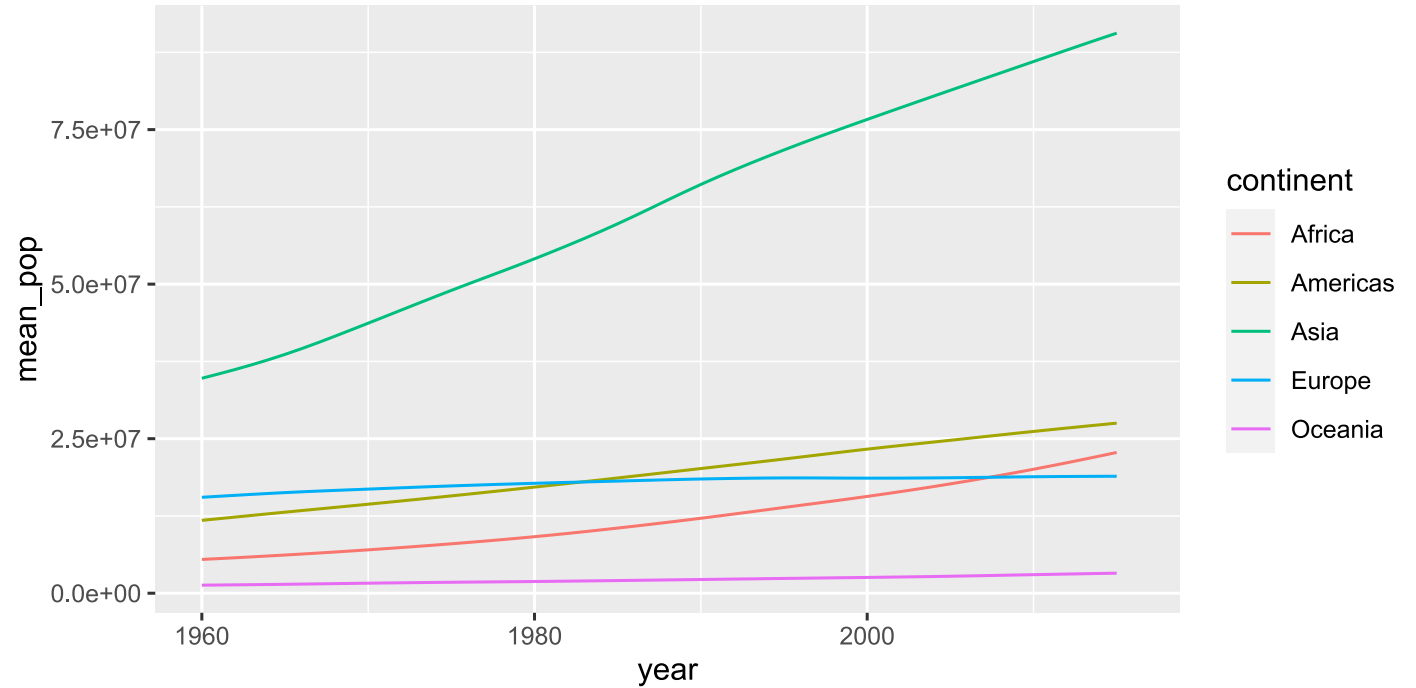
(Y)Our first plot!

```
gapminder_mean %>%  
  ggplot() +  
    aes(x = year,  
        y = mean_pop,  
        color = continent) +  
    geom_point() +  
    geom_line()
```



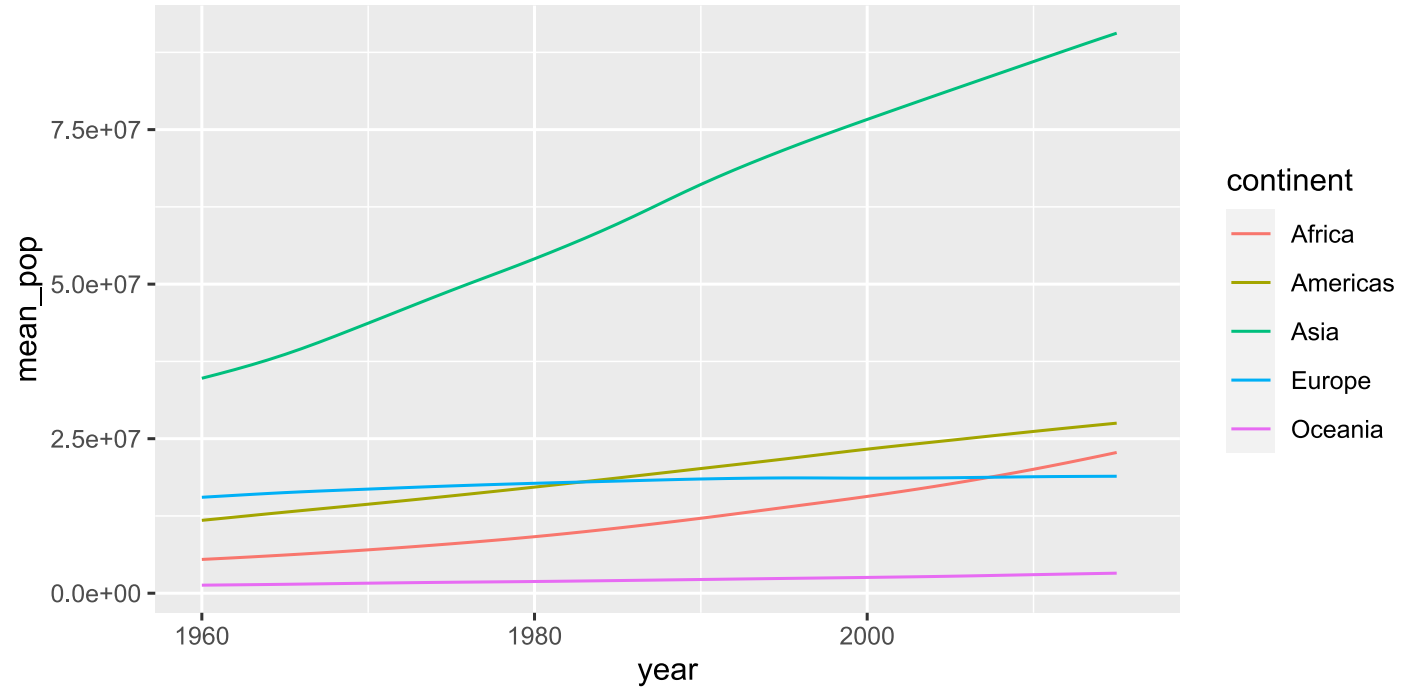
(Y)Our first plot!

```
gapminder_mean %>%  
  ggplot() +  
  aes(x = year,  
       y = mean_pop,  
       color = continent) +  
  # geom_point() +  
  geom_line()
```



(Y)Our first plot!

```
g = gapminder_mean %>%  
  ggplot() +  
  aes(x = year,  
       y = mean_pop,  
       color = continent) +  
  # geom_point() +  
  geom_line()  
g  
# graphs can be saved as  
# objects!
```



gg is for Grammar of Graphics

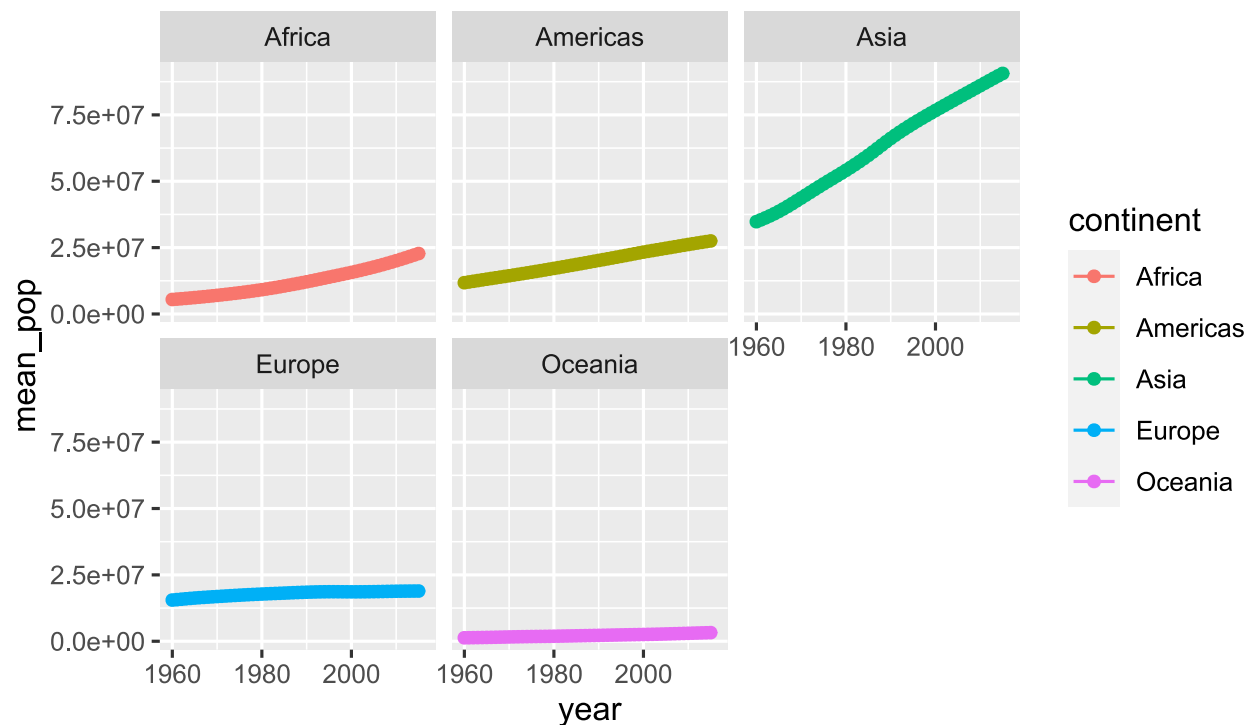
Data

Aesthetics

Geoms

Facet

```
g + facet_wrap(~ continent)
```



```
+ facet_wrap()  
+ facet_grid()
```



gg is for Grammar of Graphics

Data

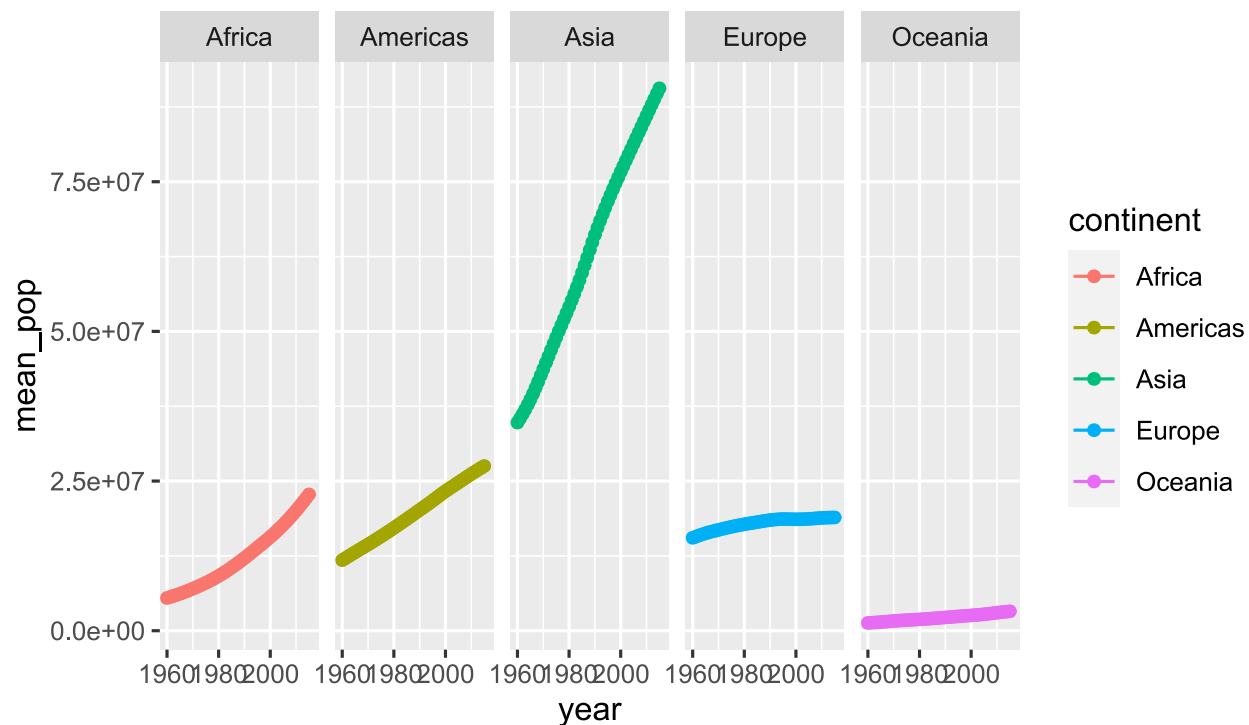
Aesthetics

Geoms

Facet

```
+ facet_wrap()  
+ facet_grid()
```

```
g + facet_grid(~ continent)
```



gg is for Grammar of Graphics

Data

Aesthetics

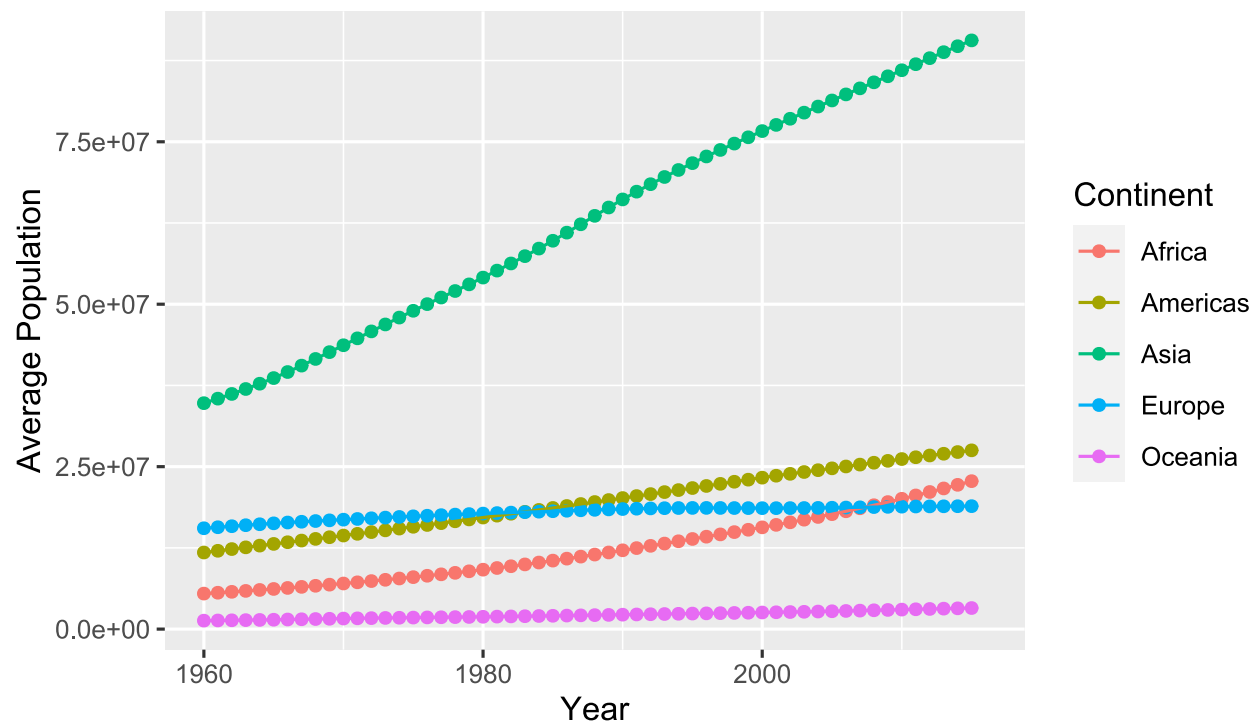
Geoms

Facet

Labels

```
+ labs()
```

```
g + labs(x = "Year", y = "Average Population", color = "Continent")
```



gg is for Grammar of Graphics

Data

`scale + _ + <aes> + _ + <type> + ()`

Aesthetics

What parameter do you want to adjust? → `<aes>`

What type is the parameter? → `<type>`

Geoms

- I want to change my discrete x-axis

`scale_x_discrete()`

- I want to change range of point sizes from continuous variable

`scale_size_continuous()`

- I want to rescale y-axis as log10

`scale_y_log10()`

- I want to use a different color palette

`scale_fill_discrete()`

`scale_color_manual()`

Scales

+ `scale_*_*`()



gg is for Grammar of Graphics

Data

Aesthetics

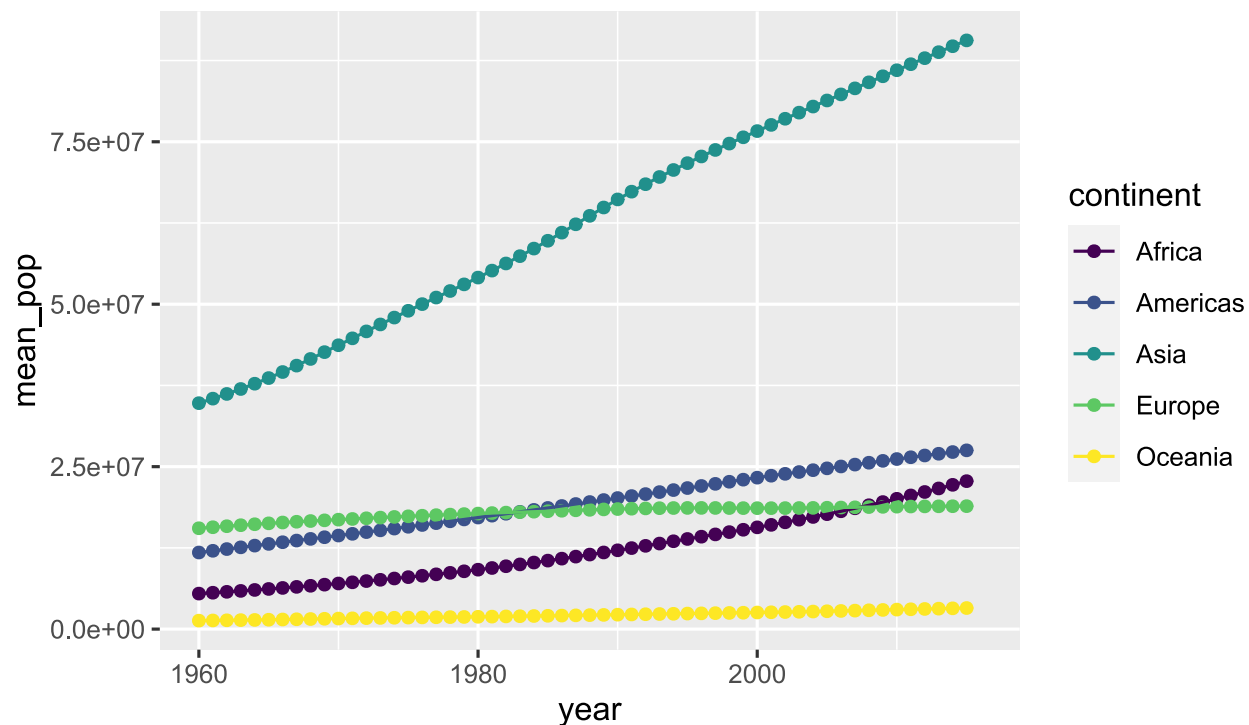
Geoms

Facet

Labels

Scales

```
g + scale_color_viridis_d()
```



```
+ scale_*_*( )
```



gg is for Grammar of Graphics

Data

Aesthetics

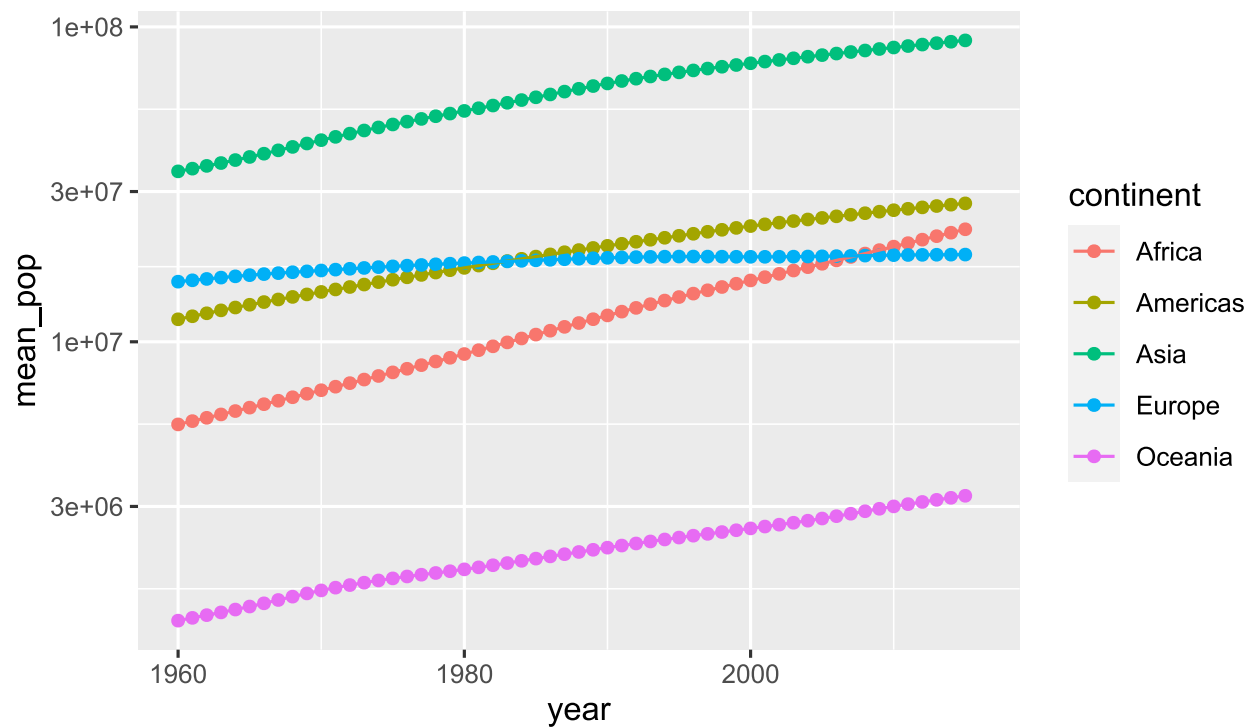
Geoms

Facet

Labels

Scales

```
g + scale_y_log10()
```



```
+ scale_*_*()
```



gg is for Grammar of Graphics

```
g + scale_x_continuous(breaks = seq(1950, 2020, 10))
```

Data

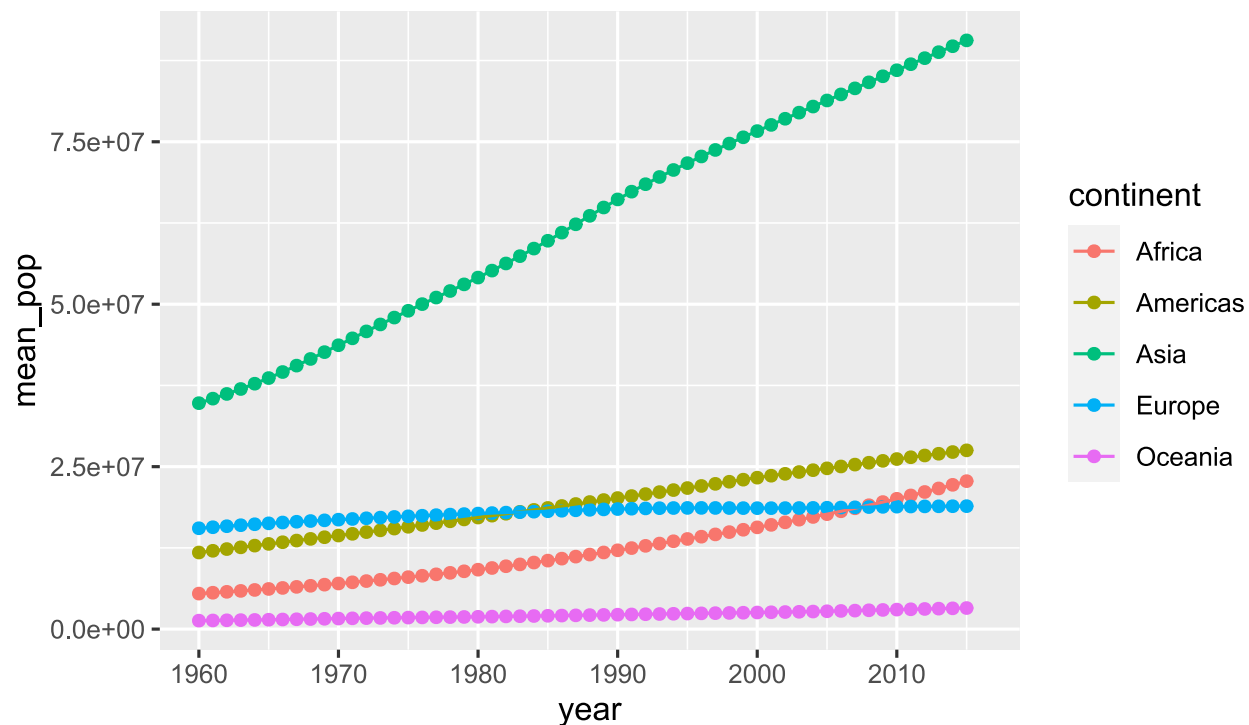
Aesthetics

Geoms

Facet

Labels

Scales



```
+ scale_*_*( )
```

Delving Deeper into ggplot

- Each graph is different and `ggplot2` provides a zillion options to customize your graph to perfection.



Delving Deeper into ggplot

- Each graph is different and `ggplot2` provides a zillion options to customize your graph to perfection.
- Excellent cheatsheet on [project website](#).



Delving Deeper into ggplot

- Each graph is different and `ggplot2` provides a zillion options to customize your graph to perfection.
- Excellent cheatsheet on [project website](#).
- [Garrick Aden-Buie](#)'s wonderful [Gentle Guide to the Grammar of Graphics with `ggplot2`](#) from which the previous slides were taken from.



Types of Plots

Histograms: counts how many observations fall within a certain bin.



Types of Plots

Histograms: counts how many observations fall within a certain bin.

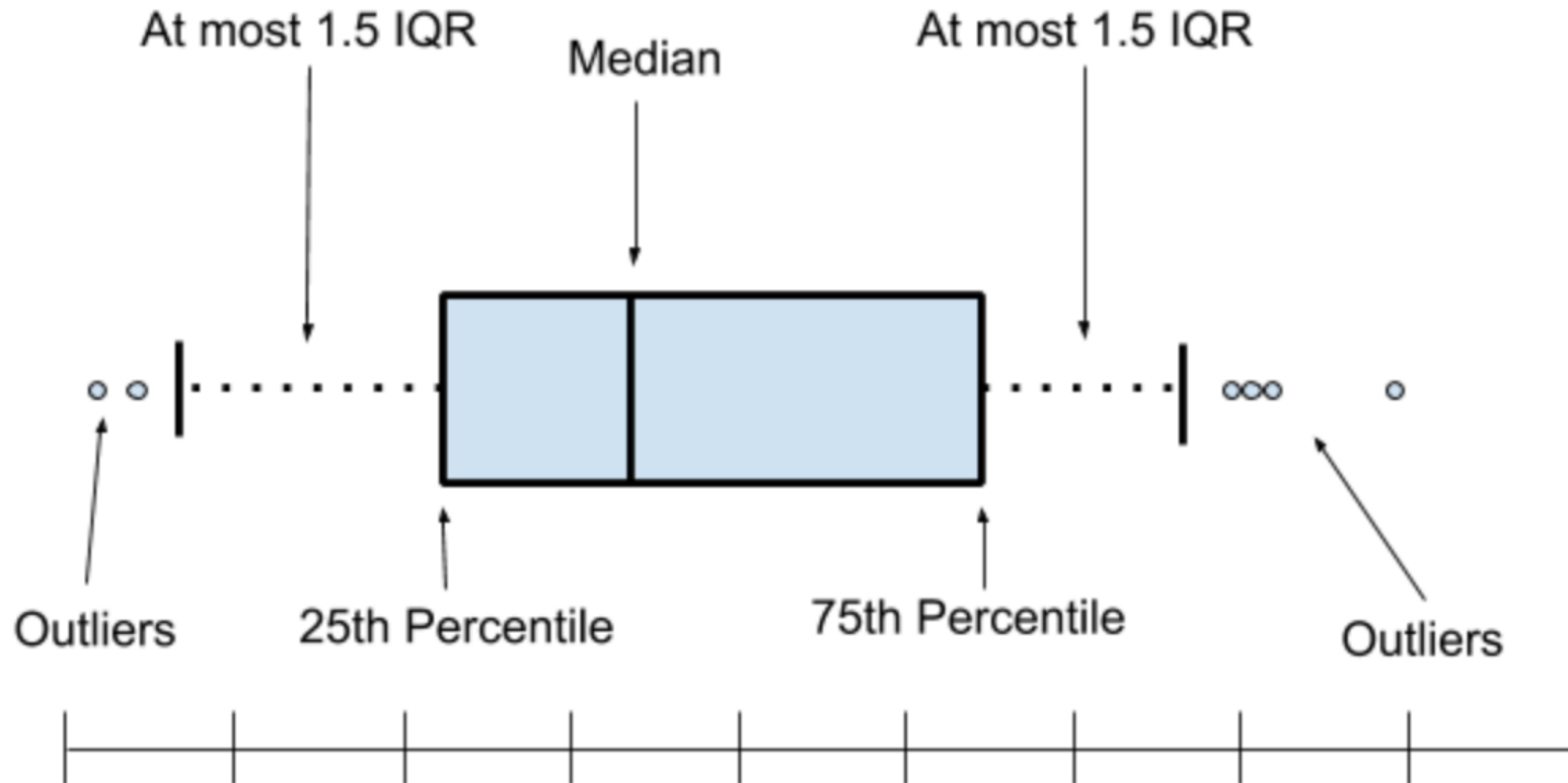
Boxplots: displays the distribution of a variable.



Types of Plots

Histograms: counts how many observations fall within a certain bin.

Boxplots: displays the distribution of a variable.



Types of Plots

Histograms: counts how many observations fall within a certain bin.

Boxplots: displays the distribution of a variable.

Scatter plots: shows the association between two variables.



Task 3: Visualising data

10:00

Using the `gapminder` data, create the following plots using `ggplot2`.

1. A histogram of life expectancy in 2015. (Hint: do you need to specify a `y` in `aes()` for a histogram?) Once you've created the histogram, within the appropriate `geom_*` set: `binwidth` to 5, `boundary` to 45, `colour` to "white" and `fill` to "#d90502". What does each of these options do?
Optional: Using the previous graph, facet it by continent such that each continent's plot is a new row. (Hint: check the help for `facet_grid`.)
2. A boxplot of average life expectancy per year by continent. Within the appropriate `geom_*` set: `colour` to "black" and `fill` to "#d90502". (Hint: you need to group by both `continent` and `year`.)
3. A scatter plot of fertility rate (y-axis) with respect to infant mortality (x-axis) in 2015. Once you've created the scatter plot, within the appropriate `geom_*` set: `size` to 3, `alpha` to 0.5, `colour` to "#d90502". Add labels (`labs`) to the plot so that it is cleaner.



Summarising

Summarising Data

- In general, we can learn from the data by visualising it and/or computing summary statistics



Summarising Data

- In general, we can learn from the data by visualising it and/or computing summary statistics
- Let's now turn to summary statistics!



Summarising Data

- In general, we can learn from the data by visualising it and/or computing summary statistics
- Let's now turn to summary statistics!
- In particular, let's look at two features: *central tendency* and *spread*.



Central Tendency

`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
```



Central Tendency

`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
```

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

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

The median is robust against *outliers*.

```
median(x)
```

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



Spread

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

The *variance* is such a measure.

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

Consider two normal distributions with equal mean at 0:



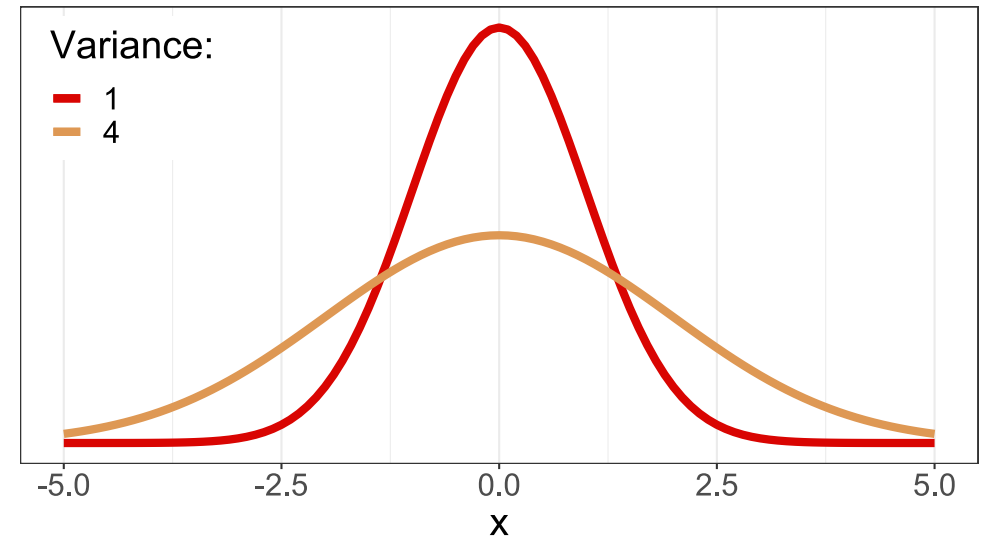
Spread

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

The *variance* is such a measure.

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

Consider two normal distributions with equal mean at 0:



Compute with:

```
var(x)
```



Tabulating Data

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

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

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



Tabulating Data

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

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

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

The same can be achieved using the `count` function (from `dplyr`)

```
gapminder %>% count(continent)
```

```
## continent n  
## 1 Africa 2907  
## 2 Americas 2052  
## 3 Asia 2679  
## 4 Europe 2223  
## 5 Oceania 684
```



Tabulating Data

Given two variables, `table` produces a contingency table:

```
gapminder_new <- gapminder %>%  
  filter(year == 2015) %>%  
  mutate(fertility_above_2 = (fertility > 2.1)) # dummy variable for fertility rate above replacement rate
```

```
table(gapminder_new$fertility_above_2)
```

```
##  
## FALSE TRUE  
##      80  104
```



Tabulating Data

Given two variables, `table` produces a contingency table:

```
gapminder_new <- gapminder %>%  
  filter(year == 2015) %>%  
  mutate(fertility_above_2 = (fertility > 2.1)) # dummy variable for fertility rate above replacement rate
```

```
table(gapminder_new$fertility_above_2)
```

```
##  
## FALSE  TRUE  
##    80   104
```

```
table(gapminder_new$fertility_above_2, gapminder_new$continent)
```

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



Tabulating Data

Given two variables, `table` produces a contingency table:

```
gapminder_new <- gapminder %>%  
  filter(year == 2015) %>%  
  mutate(fertility_above_2 = (fertility > 2.1)) # dummy variable for fertility rate above replacement rate
```

```
table(gapminder_new$fertility_above_2)
```

```
##  
## FALSE  TRUE  
##    80   104
```

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

- ⚠ To obtain `tables` or `crosstables` with `NAs`, use the `useNA = "always"` or `useNA = "ifany"`



Tabulating Data

Again the `count` function can get you there as well:

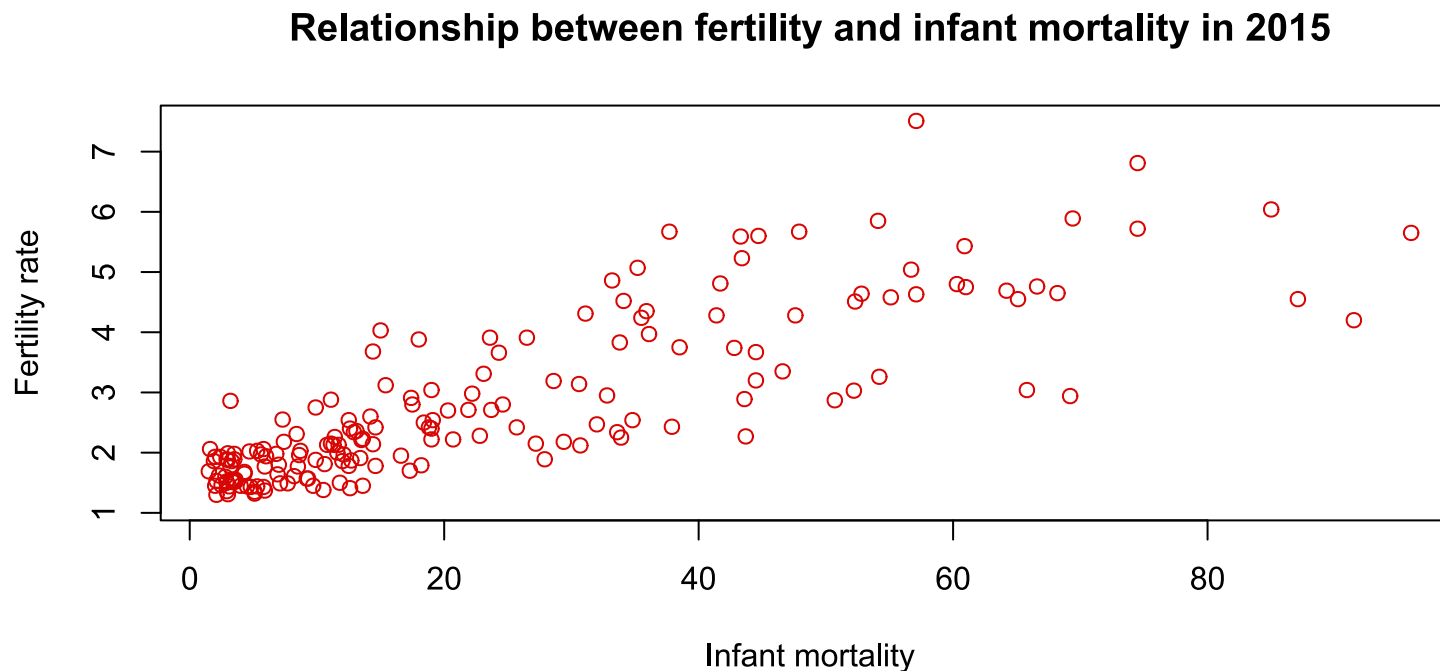
```
gapminder_new %>%  
  count(continent, fertility_above_2)
```

```
##   continent fertility_above_2   n  
## 1    Africa              FALSE   2  
## 2    Africa              TRUE  49  
## 3  Americas              FALSE  15  
## 4  Americas              TRUE  20  
## 5  Americas              NA    1  
## 6    Asia                FALSE  20  
## 7    Asia                TRUE  27  
## 8   Europe              FALSE  39  
## 9  Oceania              FALSE   4  
## 10 Oceania              TRUE   8
```

Note that `count` will display `NA`s only if there are some.



How are x and y related? Covariance and Correlation



Two main statistics to characterise the relationship between x and y :

1. Covariance
2. Correlation



Covariance

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

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



Covariance

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

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

- The `cov` function computes the covariance:

```
cov(gapminder_new$fertility, gapminder_new$infant_mortality, use = "complete.obs")
```

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



Covariance

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

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

- The `cov` function computes the covariance:

```
cov(gapminder_new$fertility, gapminder_new$infant_mortality, use = "complete.obs")
```

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

- Difficult to interpret because sensitive to the variables' dispersions from the mean



Correlation

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

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



Correlation

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

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

- The `cor` function computes the correlation:

```
cor(gapminder_new$fertility, gapminder_new$infant_mortality, use = "complete.obs")
```

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



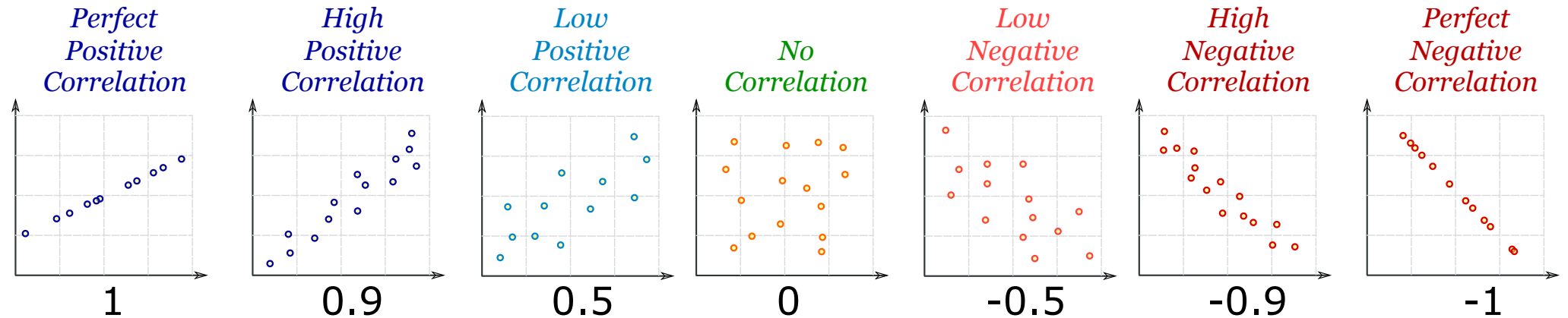
Correlation

- Correlation is always between -1 and 1!



Correlation

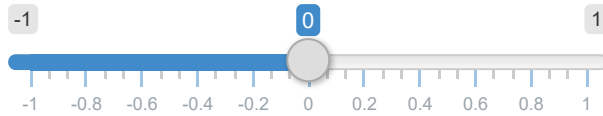
- Correlation is always between -1 and 1!



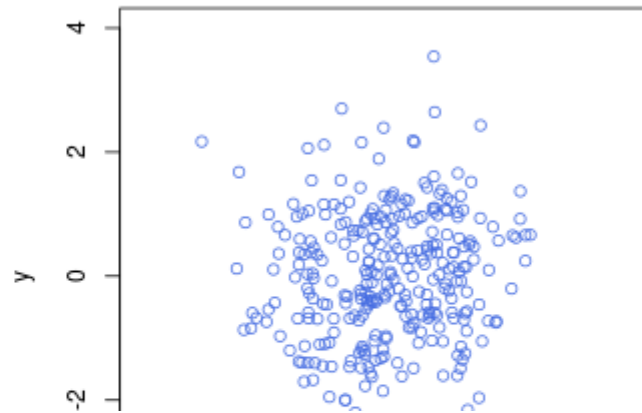
Source: *mathisfun*

Correlation

Correlation r



Correlation: 0



Task 4: Summarising data

10:00

1. Compute the mean of GDP in 2011 and assign to object `mean`. You should exclude missing values. (Hint: read the help for `mean` to remove `NAs`).
2. Compute the median of GDP in 2011 and assign to object `median`. Again, you should exclude missing values. Is it greater or smaller than the average?
3. Create a density plot of GDP in 2011 using `geom_density`. A density plot is a way of representing the distribution of a numeric variable. Add the following code to your plot to show the median and mean as vertical lines. What do you observe?

```
geom_vline(xintercept = as.numeric(mean), colour = "red") +  
geom_vline(xintercept = as.numeric(median), colour = "orange")
```
4. Compute the correlation between fertility and infant mortality in 2015. To drop `NAs` in either variable set the argument `use` to "pairwise.complete.obs" in your `cor()` function. Is this correlation consistent with the graph you produced in Task 3?

In your free time, you can do this tutorial:



```
library(ScPoApps) # this may take a while to install  
runTutorial('chapter2')
```

On the way to causality

- ☑ **How to manage data?** Read it, tidy it, visualise it...
- ✗ How to summarise relationships between variables?
- ✗ What is causality?
- ✗ What if we don't observe an entire population?
- ✗ Are our findings just due to randomness?
- ✗ How to find exogeneity in practice?



SEE YOU NEXT WEEK!

	Slides
	Book
	@ScPoEcon
	@ScPoEcon

