

ScPoEconometrics

Tidying, Visualising and Summarising Data

Florian Oswald, Gustave Kenedi and Pierre Villedieu
SciencesPo Paris
2021-01-19

Quick "Quiz" on Last Week's Material

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

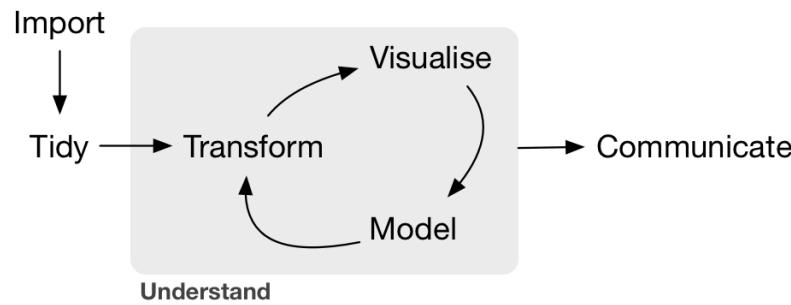
OR

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



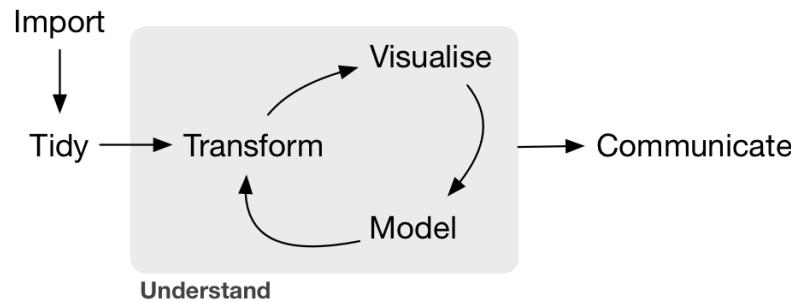
Working With Data

- Econometrics is about data.



Working With Data

- Econometrics is about data.



- According to a 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 *must* read through Hadley Wickham's chapter. It's clear and concise.



dplyr Overview

- You *must* read through **Hadley Wickham's chapter**. It's clear and 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`.)



dplyr Overview

- You *must* read through **Hadley Wickham's chapter**. It's clear and 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* work as follows:

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


dplyr Overview

- You *must* read through **Hadley Wickham's chapter**. It's clear and 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* 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 `%>%`:

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


Main dplyr Verbs

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



Main dplyr Verbs

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



Main dplyr Verbs

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



Main dplyr 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



Main dplyr 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()`: Collapse data to a single summary



Main dplyr 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()`: 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 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(tidyverse)
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]) # 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 Research/... A        1295
```

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

What variables does this dataset contain?



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 %>%
  filter(grade == "A" & samplesize > 2000 & rawpoll_trump > 45)

## # A tibble: 1 x 15
##   state startdate enddate   pollster grade samplesize population rawpoll_clinton rawpoll_johnson rawpoll_mcmullin adjpoll_clinton adjpoll_johnson adjpoll_mcmullin rawpoll_trump adjpoll_trump
##   <fct> <date>    <date>    <fct>    <fct>      <int> <chr>           <dbl>          <dbl>          <dbl>          <dbl>          <dbl>          <dbl>
## 1 Indi... 2016-04-26 2016-04-28 Marist ... A         2149 rv                 41
## # ... with 6 more variables: rawpoll_clinton <dbl>, rawpoll_johnson <dbl>, rawpoll_mcmullin <dbl>, adjpoll_clinton <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

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)

```
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"                 "pollster"
## [5] "grade"                   "samplesize"               "population"              "rawpoll_clinton"
## [9] "rawpoll_trump"           "rawpoll_johnson"          "rawpoll_mcmullin"        "adjpoll_clinton"
## [13] "adjpoll_trump"           "adjpoll_johnson"          "adjpoll_mcmullin"        "trump_clinton_tot"
## [17] "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)

```
polls_us_election_2016 %>%  
  select(state,startdate,enddate,pollster,rawpoll_clinton,rawpoll_trump) %>%  
  names()  
  
## [1] "state"          "startdate"       "enddate"        "pollster"       "rawpoll_clinton"  
## [6] "rawpoll_trump"
```

```
select()
```



dplyr 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

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

Select

```
## # 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
```

Summarise

Apply function by group

```
group_by()
```



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

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
```

But also with other `R` commands:

```
polls_us_election_2016$samplesize %>% mean(na.rm = T)
## [1] 1148.216
```



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
```

But also with other `R` 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  
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

10 : 00

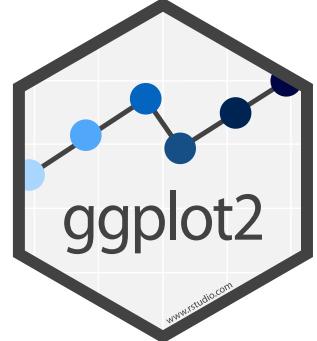
Load the data as explained on slide 8.

1. How many polls have 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?
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) had a sample size greater than 1,000?
4. 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?
5. 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 a dataset with these commands:

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



The gapminder dataset: Overview

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

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

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

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

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

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

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



Task 2

05 : 00

1. What variables does this dataset contain?
2. How are the data stored?
3. Create a new variable called `gdppercap` corresponding to `gdp` divided by `population`.
4. Compute the average population per continent per year, `mean_pop`, removing missing values and assign the output to a new object `gapminder_new`.



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

Map data to visual elements or parameters

- year
- population
- country

Aesthetics

+ aes()



gg is for Grammar of Graphics

Data

Map data to visual elements or parameters

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

Aesthetics

+ aes()



gg is for Grammar of Graphics

Data

Map data to visual elements or parameters

Aesthetics

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

```
+ aes()
```



gg is for Grammar of Graphics

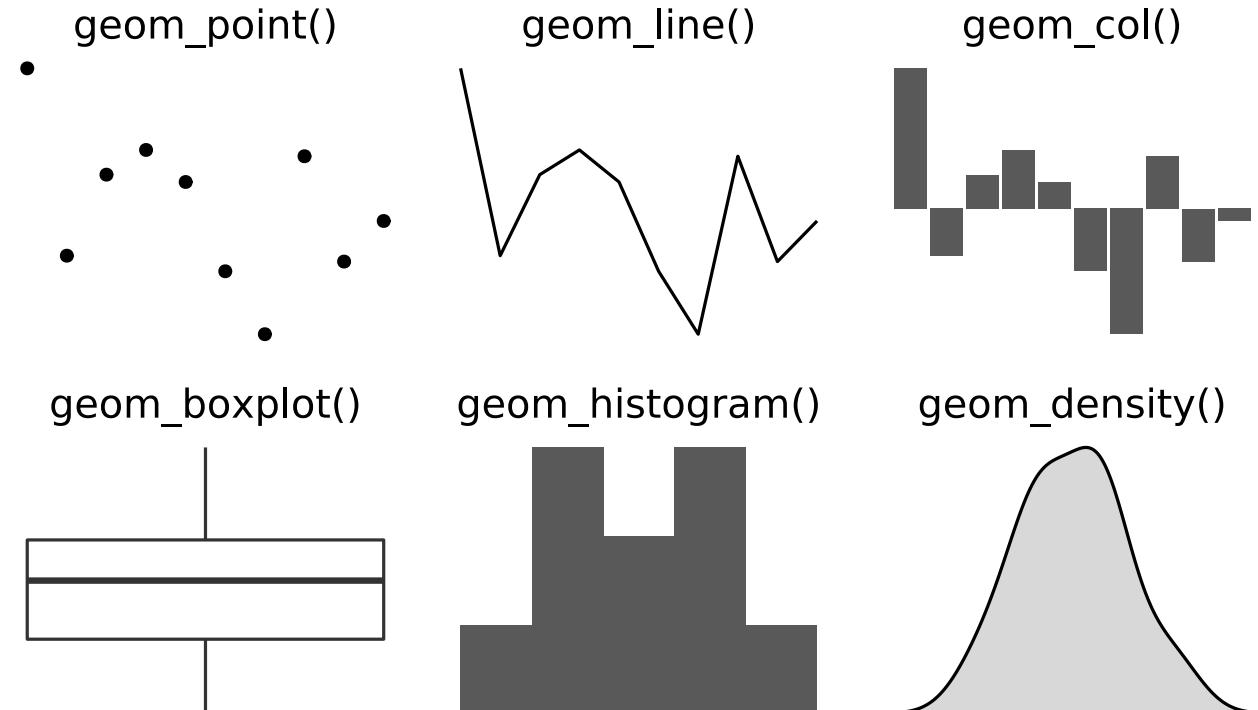
Data

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) +
```

|

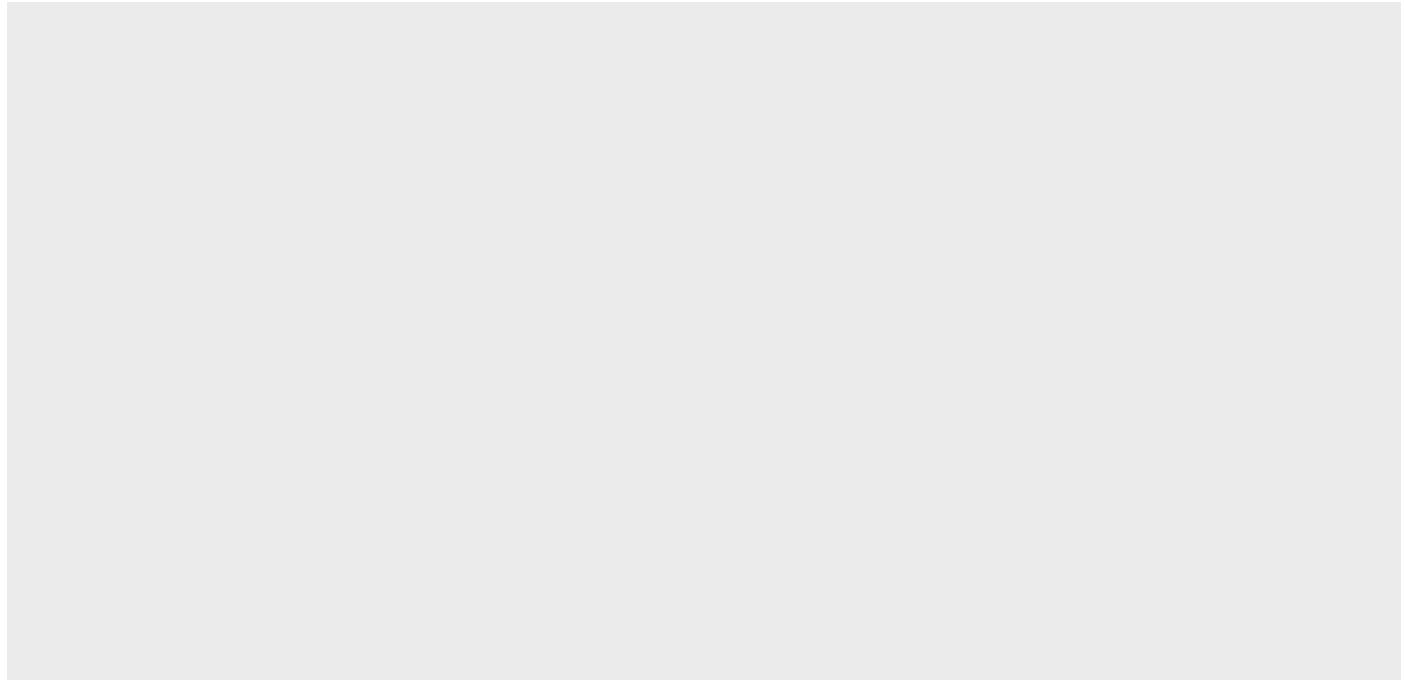
Geoms

```
+ geom_()
```



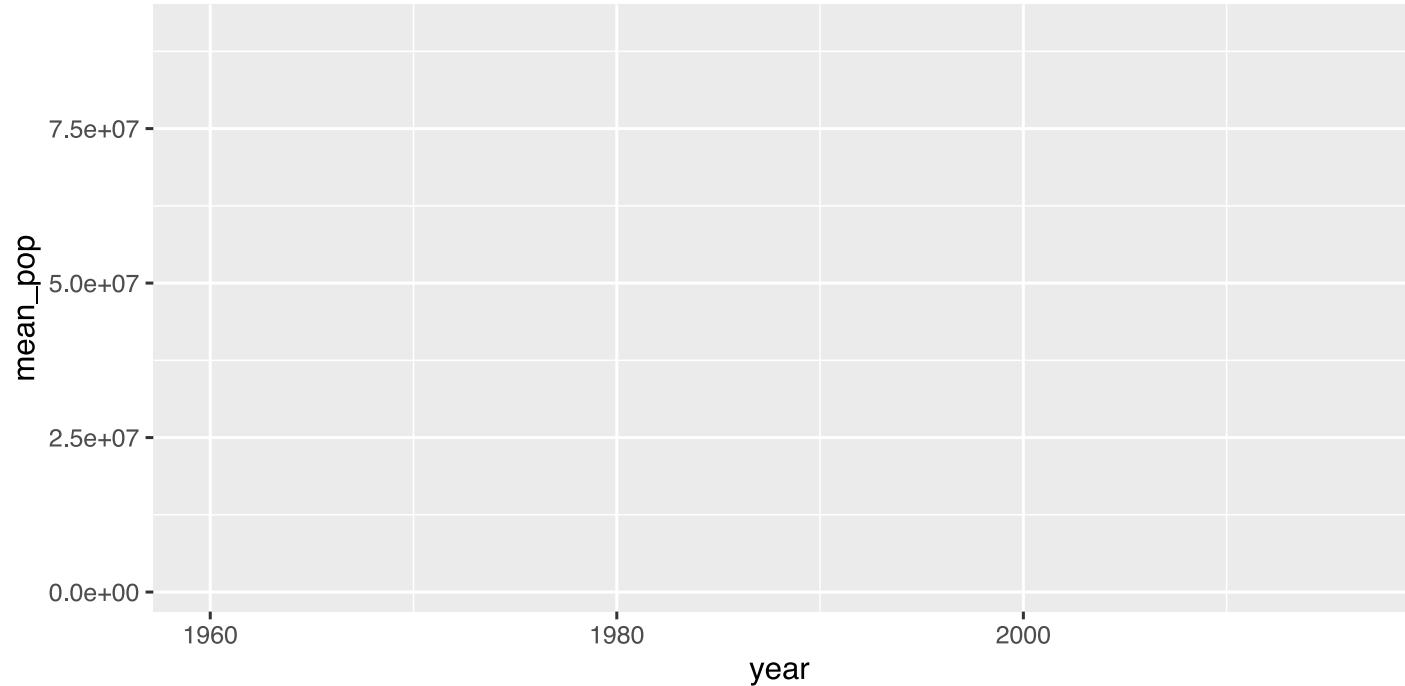
(Y)Our first plot!

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



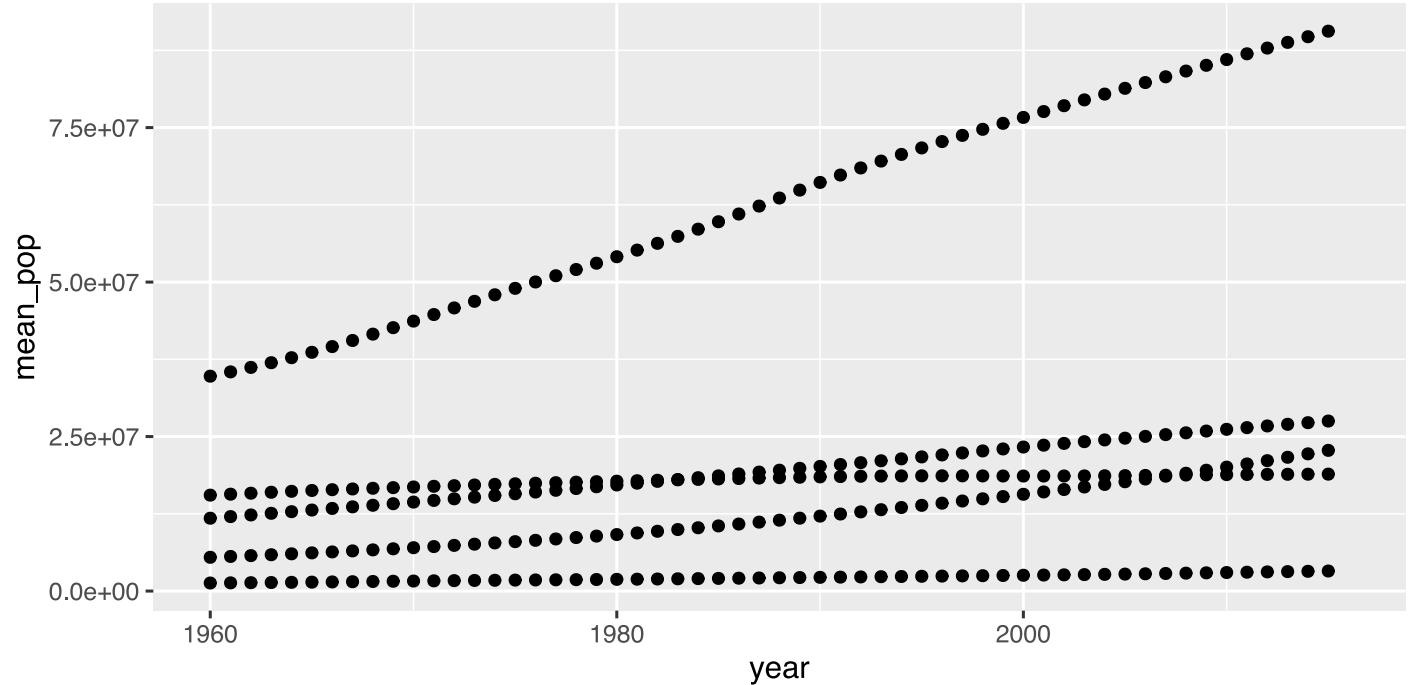
(Y)Our first plot!

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



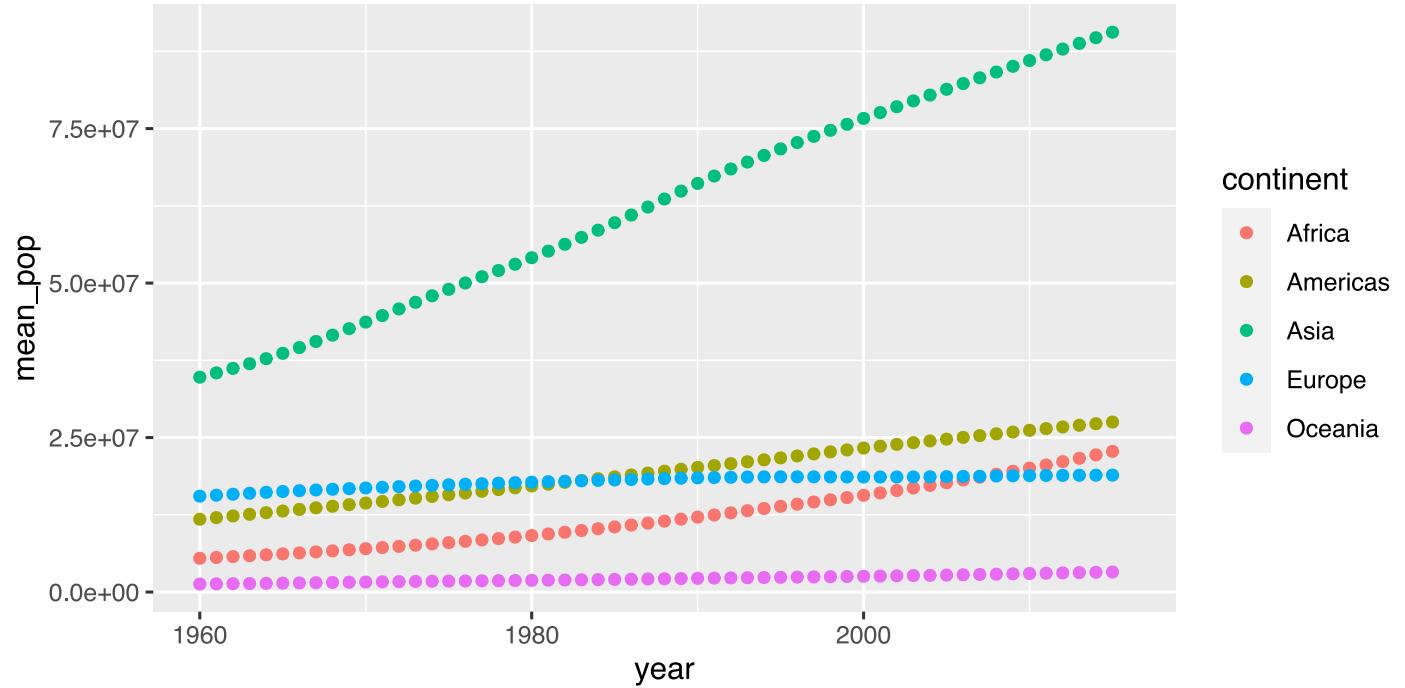
(Y)Our first plot!

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



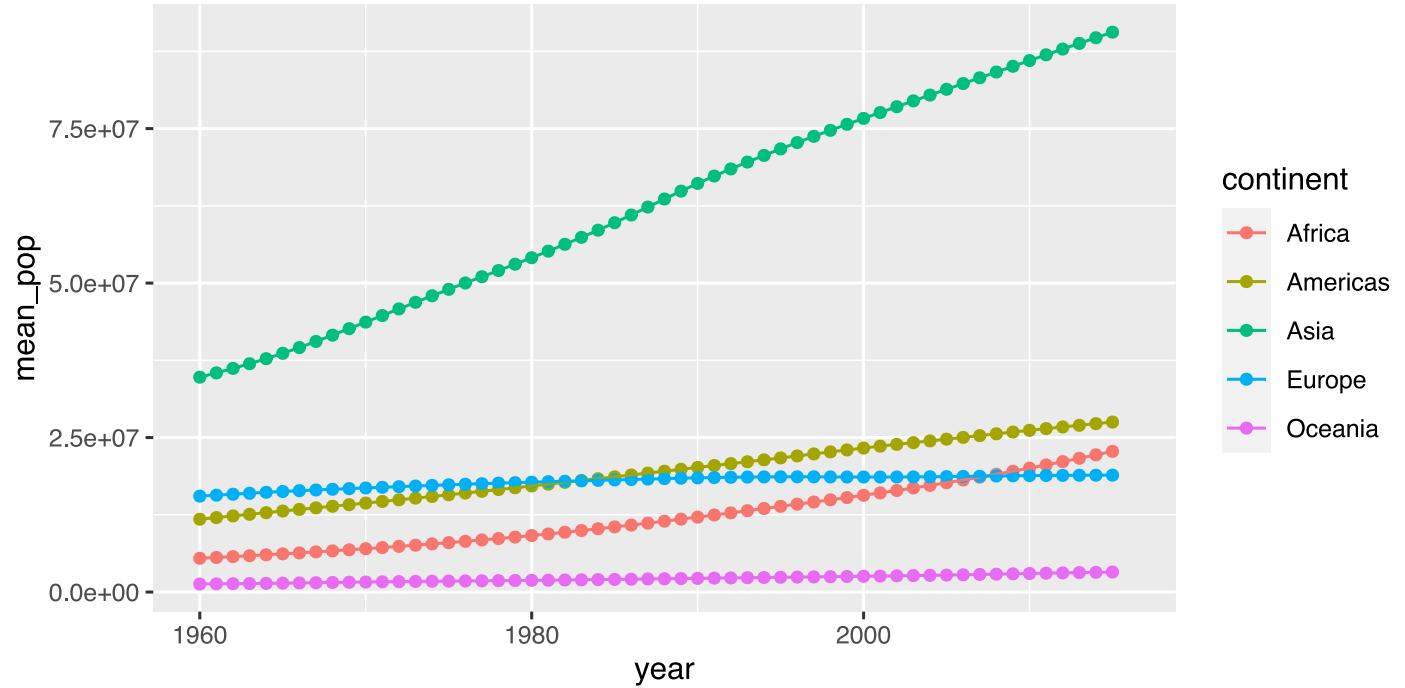
(Y)Our first plot!

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



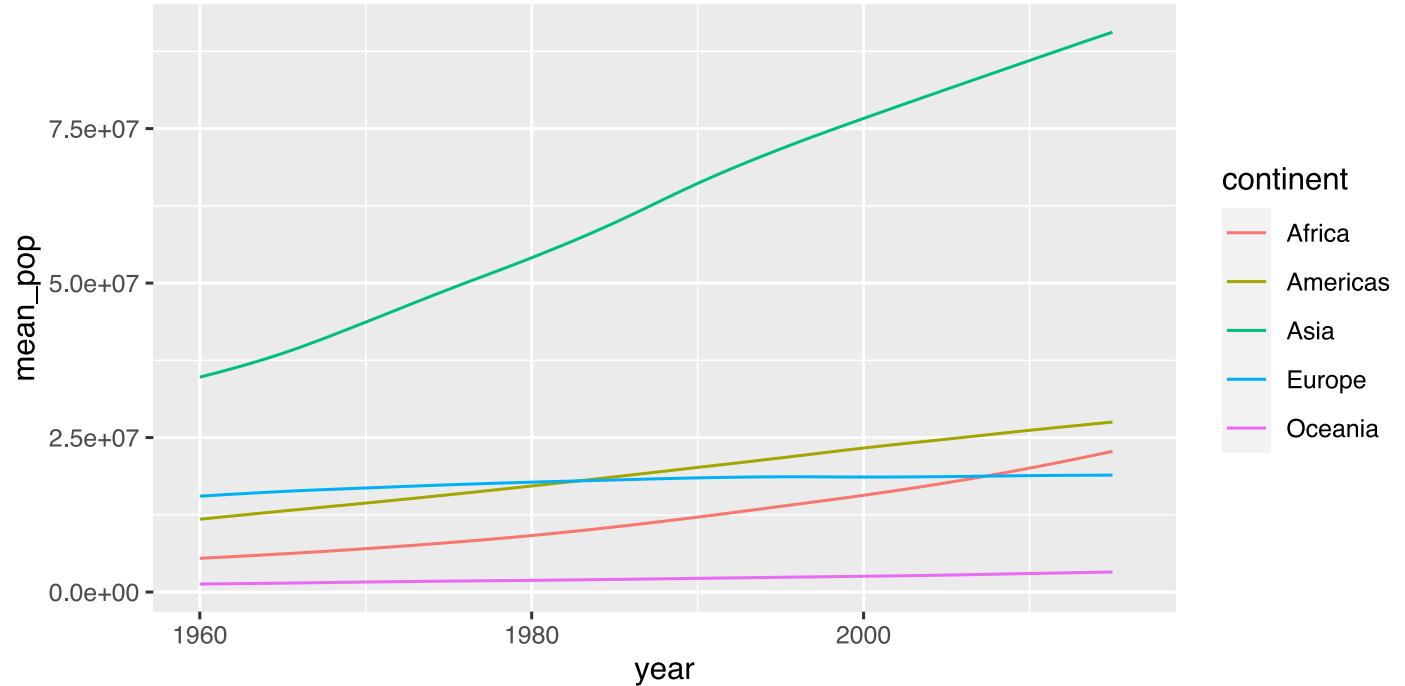
(Y)Our first plot!

```
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()
```



gg is for Grammar of Graphics

Data

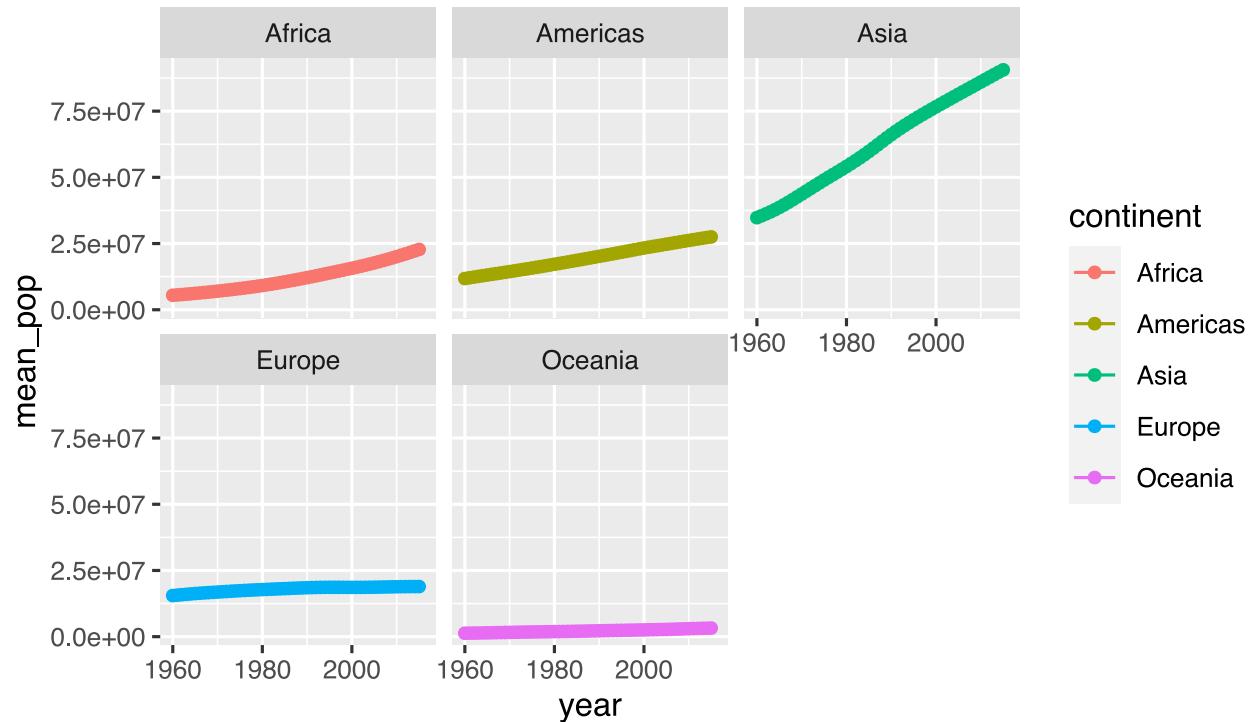
Aesthetics

Geoms

Facet

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

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



gg is for Grammar of Graphics

Data

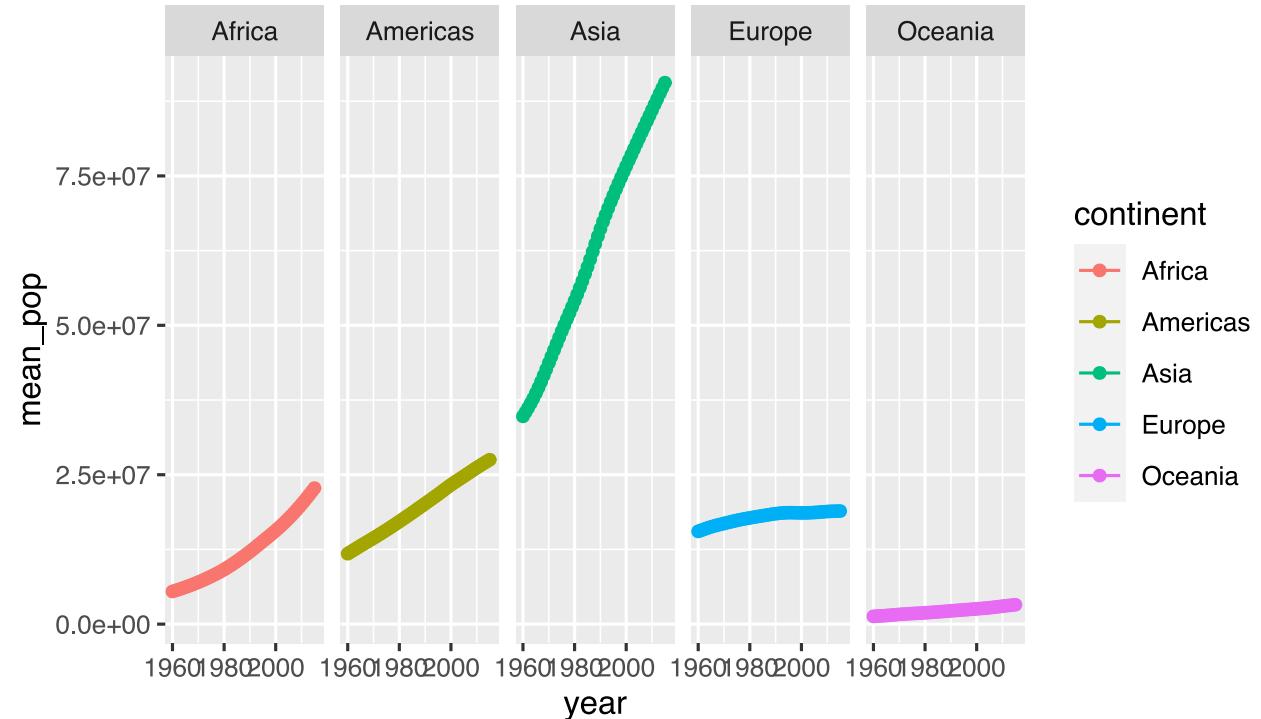
Aesthetics

Geoms

Facet

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

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



gg is for Grammar of Graphics

Data

Aesthetics

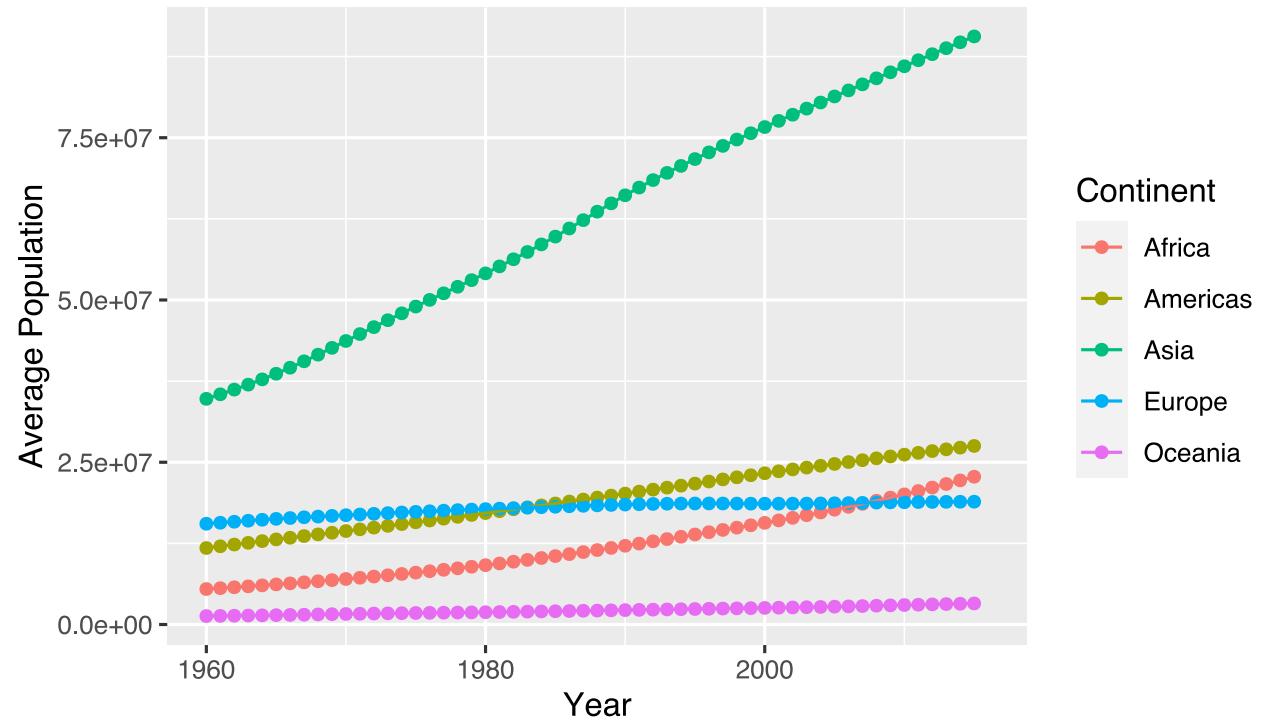
Geoms

Facet

Labels

+ labs()

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



gg is for Grammar of Graphics

Data

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

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

Aesthetics

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

Geoms

- I want to change my discrete x-axis
`scale_x_discrete()`

Facet

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

`scale_size_continuous()`

Labels

- I want to rescale y-axis as log

`scale_y_log10()`

Scales

- I want to use a different color palette

`scale_fill_discrete()`

`scale_color_manual()`

+ `scale_*_*()`



gg is for Grammar of Graphics

Data

Aesthetics

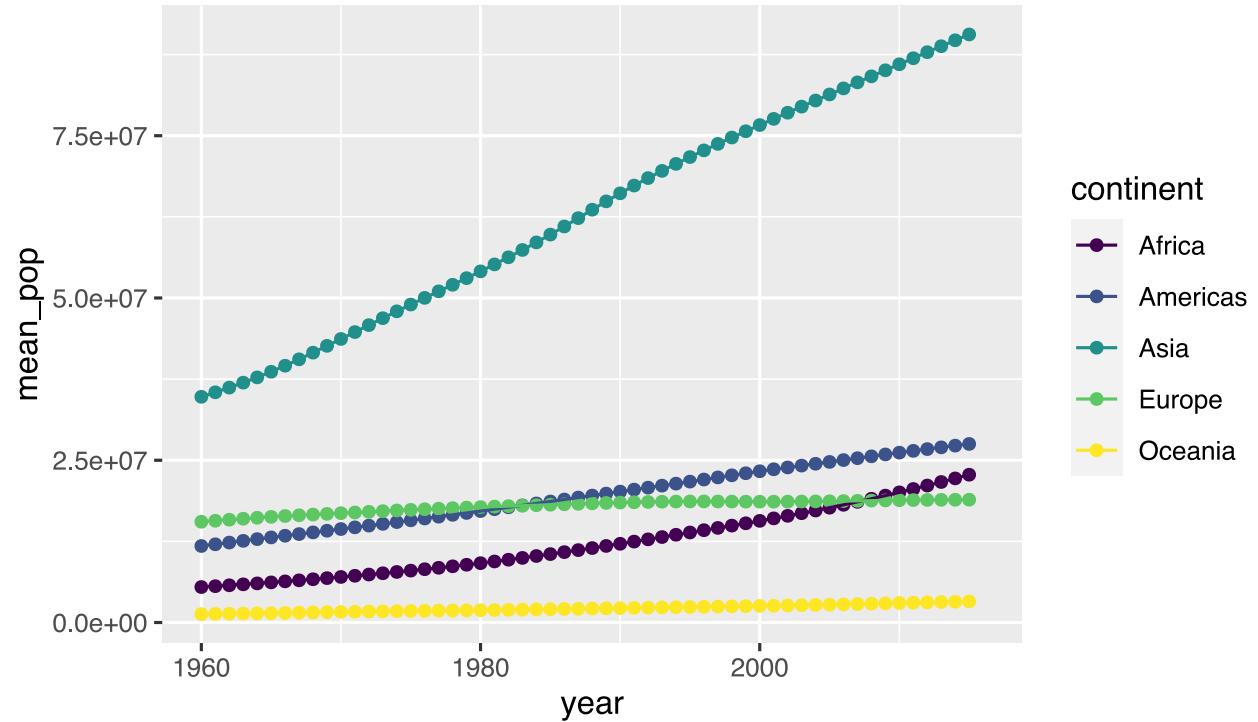
Geoms

Facet

Labels

Scales

```
g + scale_color_viridis_d()
```



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



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

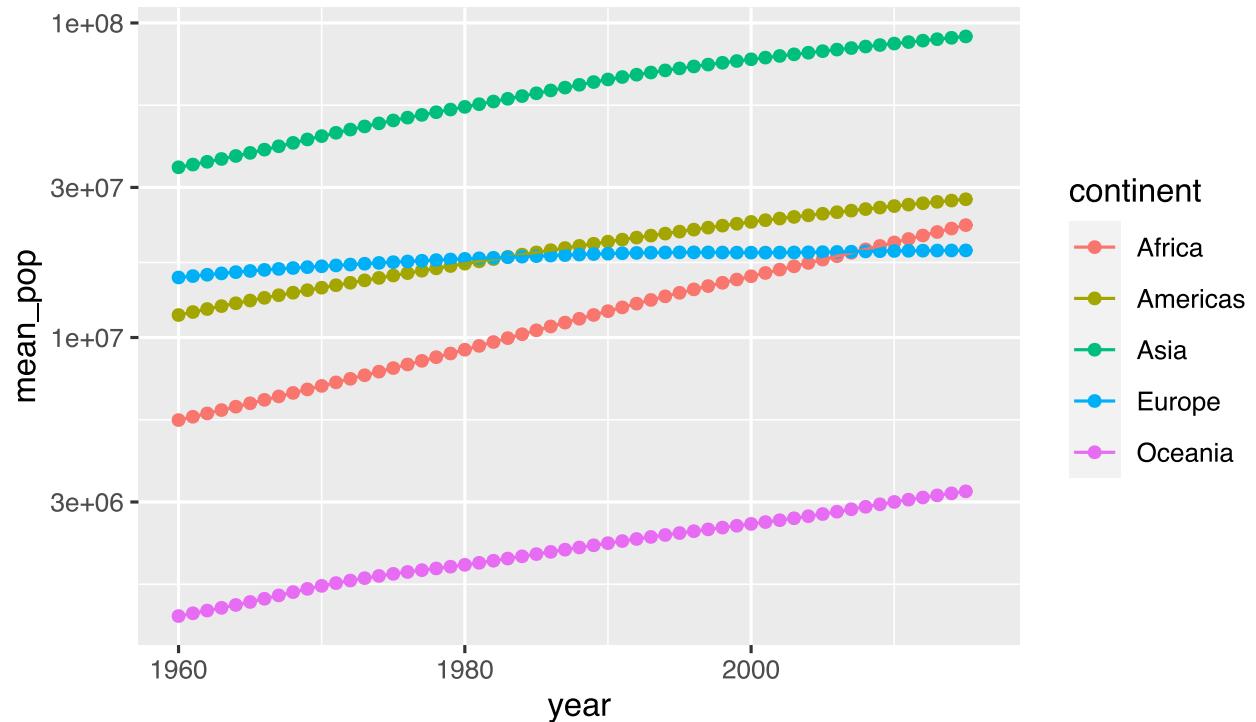
Facet

Labels

Scales

+ scale_**()

g + scale_y_log10()



continent

- Africa
- Americas
- Asia
- Europe
- Oceania



gg is for Grammar of Graphics

Data

Aesthetics

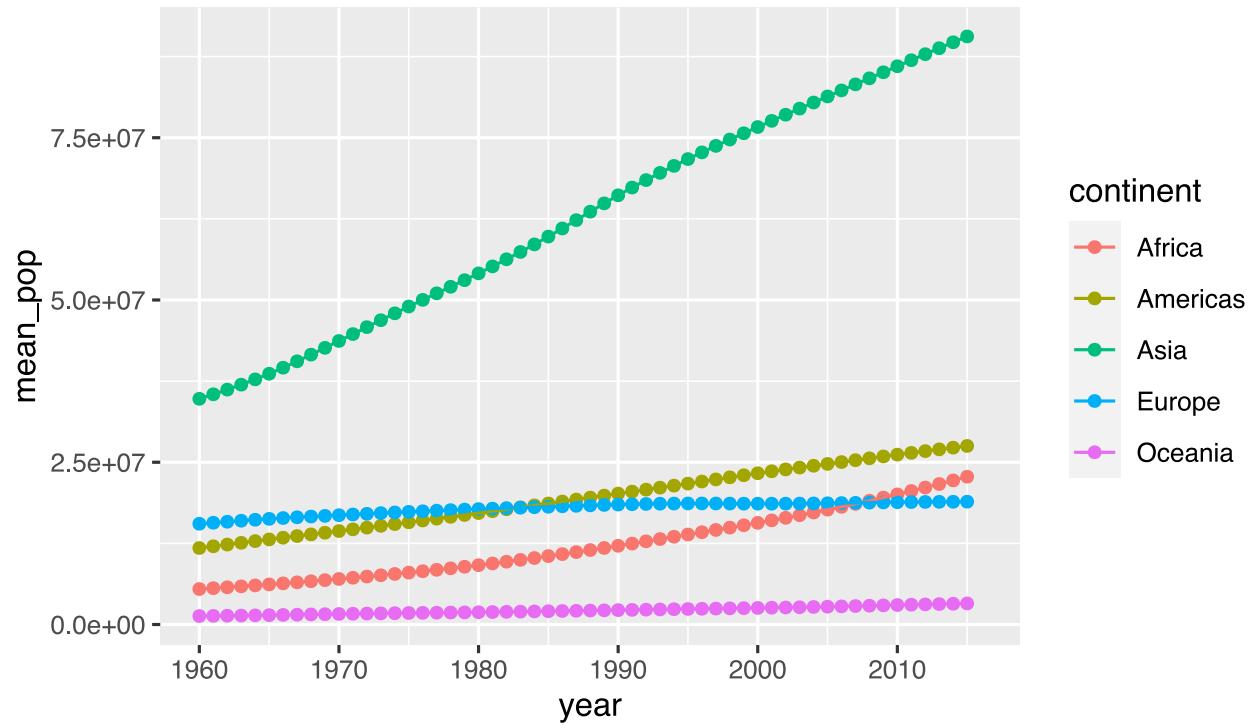
Geoms

Facet

Labels

Scales

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



```
+ 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.

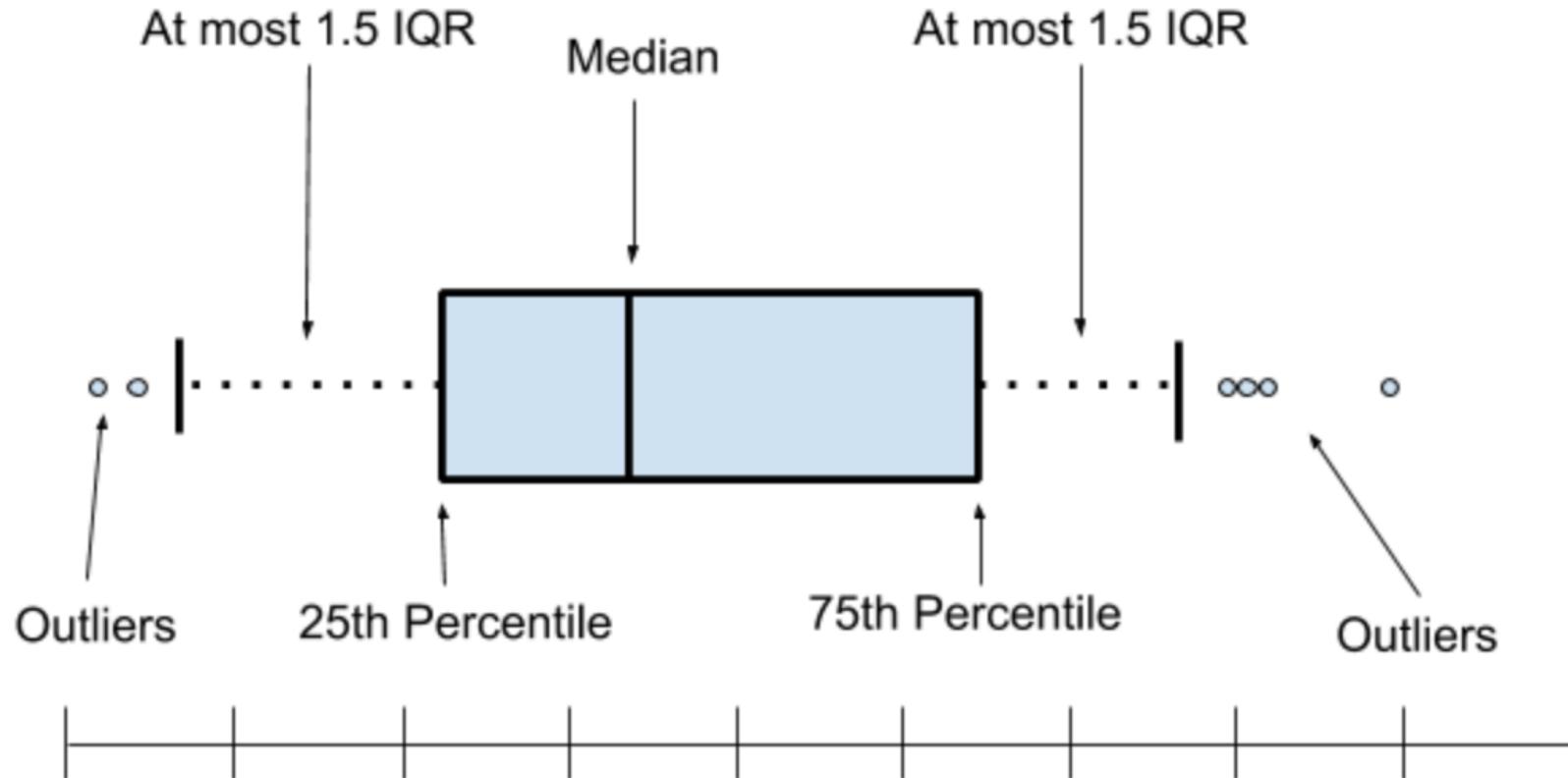
Boxplots: displays the distribution of a variable.



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

10 : 00

Using the `gapminder` data, create the following plots using `ggplot2`. Don't forget to label the axes.

1. A histogram of life expectancy in 2015. Within the appropriate `geom_*` set: `binwidth` to 5, `boundary` to 45, `colour` to "white" and `fill` to "#d90502".
2. 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`.*)
3. A boxplot of average life expectancy per year by continent (removing missing values). Within the appropriate `geom_*` set: `colour` to "black" and `fill` to "#d90502". (*Hint: you need to group by both `continent` and `year`.*)
4. A scatter plot of fertility rate (y-axis) with respect to infant mortality (x-axis) in 2015. Within the appropriate `geom_*` set: `size` to 3, `alpha` to 0.5, `colour` to "#d90502".



Summarising

Summarising Data

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



Summarising 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**.



Summarising 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 **summarise** the data for us to learn from it.



Summarising 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 **summarise** the data for us to learn from it.
- In general, we can compute summary statistics and/or visualise the data with plots.



Summarising 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 **summarise** the data for us to learn from it.
- In general, we can compute summary statistics and/or visualise the data with plots.
- Let's now turn to summary statistics!



Summarising 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 **summarise** the data for us to learn from it.
- In general, we can compute summary statistics and/or visualise the data with plots.
- 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

$\Pr(X \leq m) \geq 0.5$ 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.

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

Consider two normal distributions with equal mean at 0:



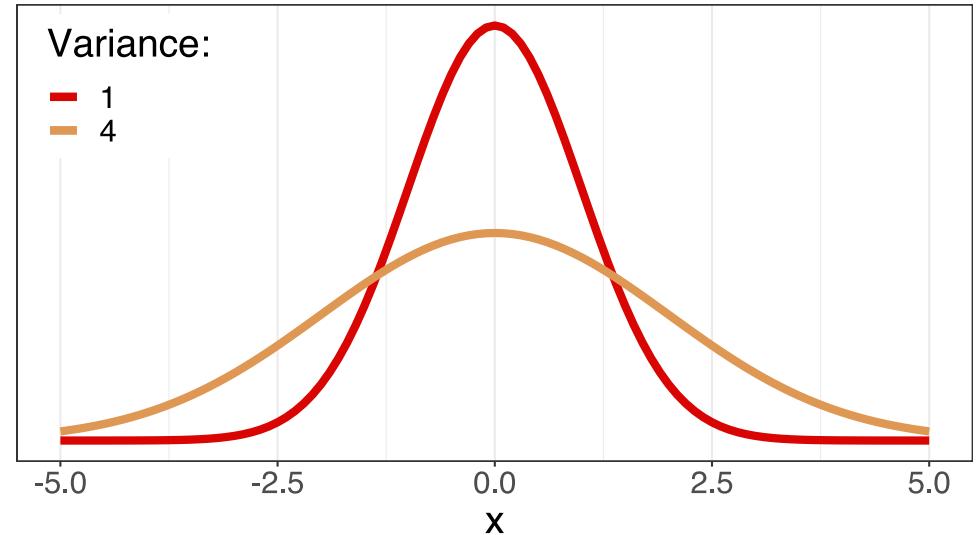
Spread

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.

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

Consider two **normal distributions** with equal mean at 0:



Compute with:

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



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 = (gapminder_2015$fertility > 2.1)) # dummy variable for fertility rate above replacement
table(gapminder_new$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 = (gapminder_2015$fertility > 2.1)) # dummy variable for fertility rate above replacement
table(gapminder_new$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:

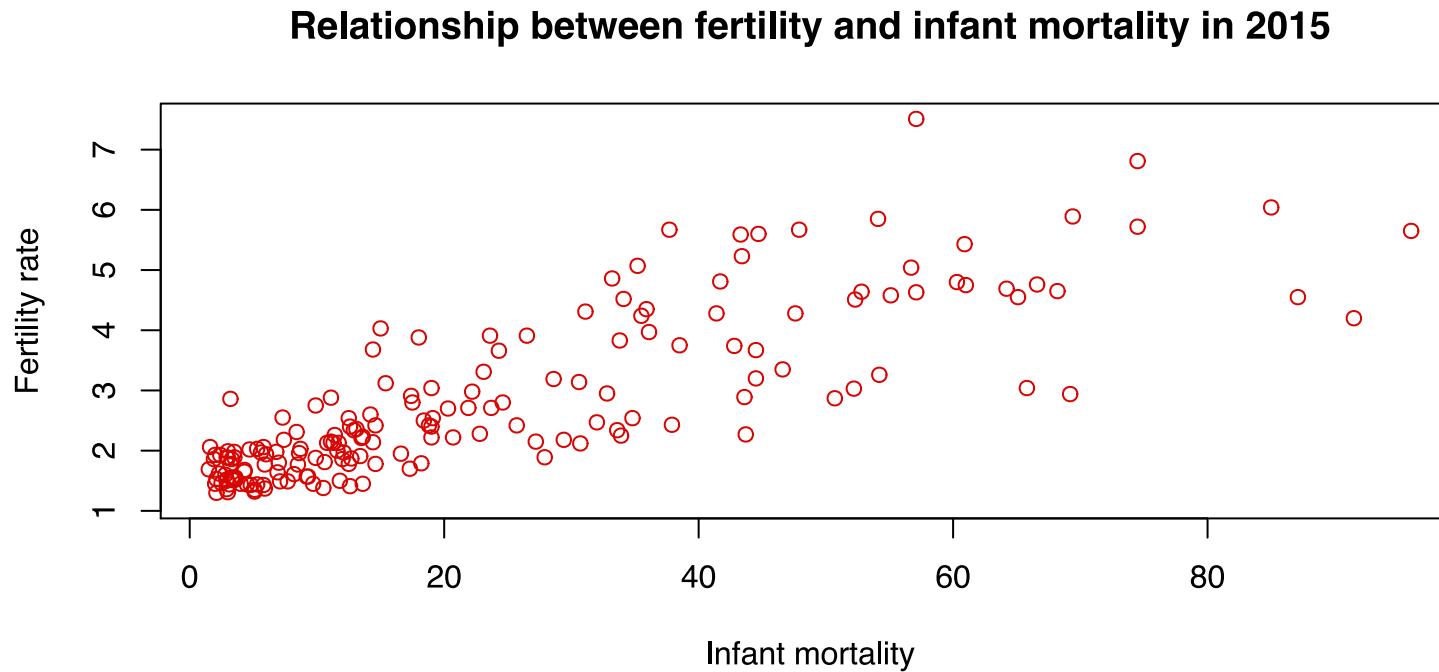
```
gapminder_new %>%
  count(continent, fertility_above_2)

## #  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 `NAs` 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.

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



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.

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

- 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
```



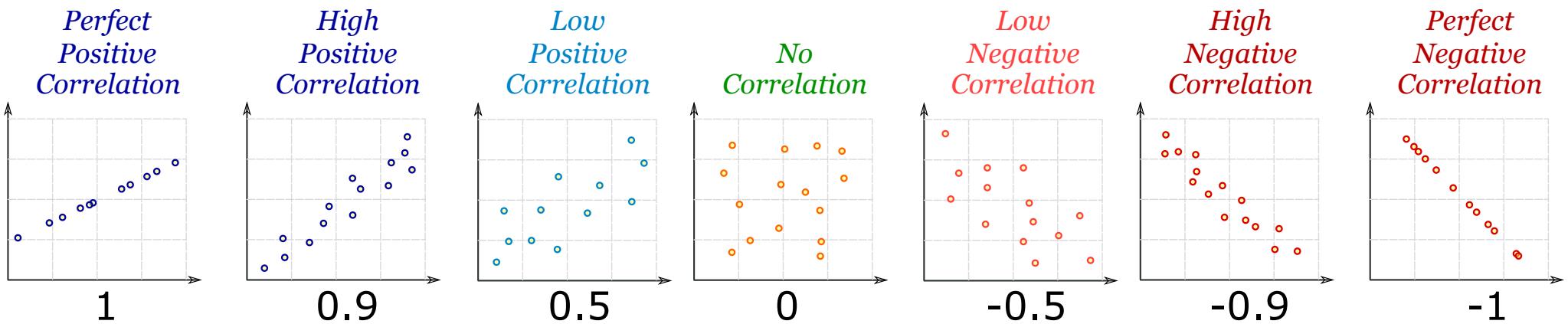
Correlation

- Correlation is always between -1 and 1!



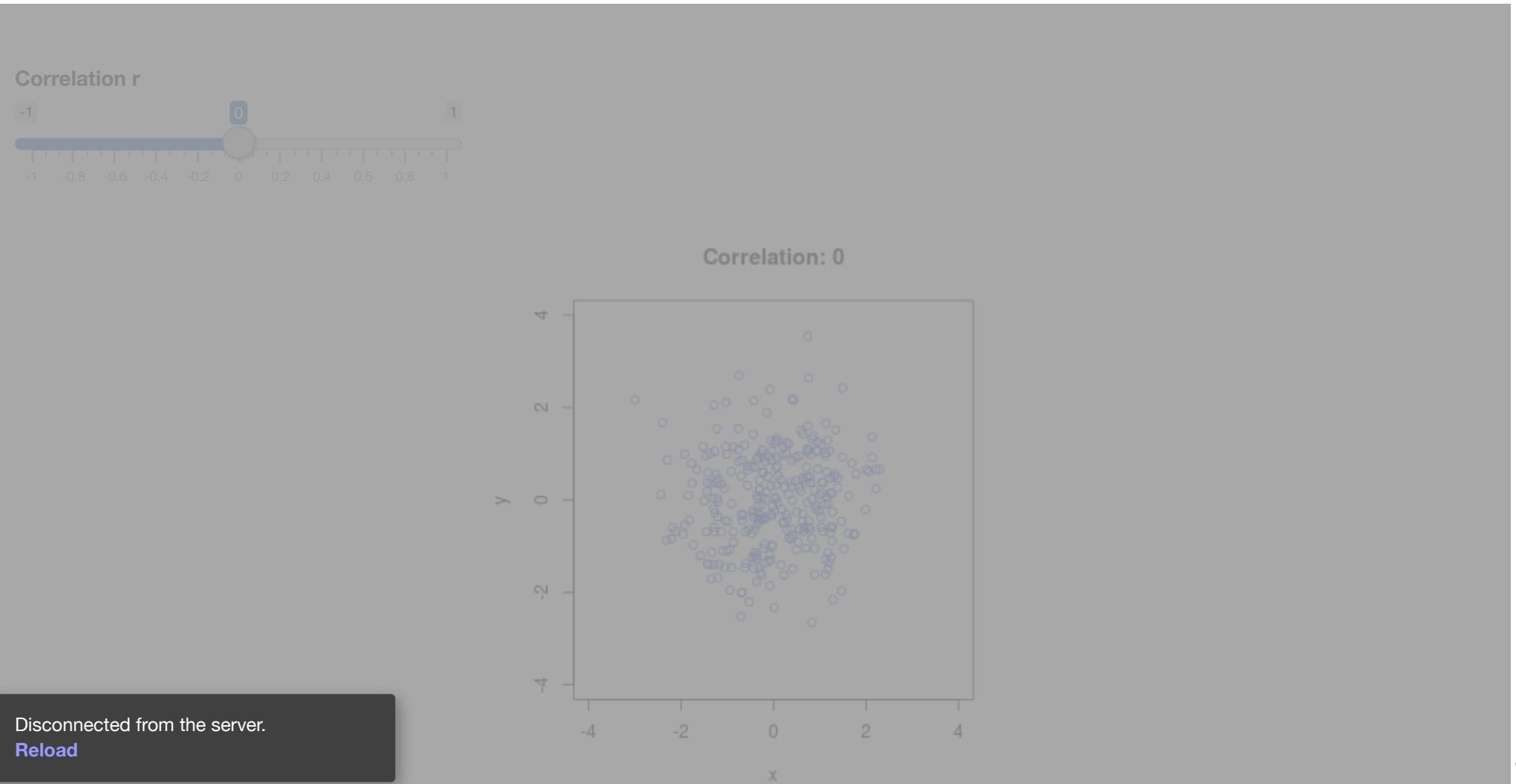
Correlation

- Correlation is always between -1 and 1!



Source: [mathisfun](#)

Correlation



Task 4

10 : 00

1. Compute the mean of population in 1960 and assign to object `mean`. Read the help for `mean` to remove `NAs`.
2. Compute the median of population in 1960 and assign to object `median`. Is it greater or smaller than the average?
3. Create of density plot using `geom_density` of population in 1960. A density plot is a way of representing the distribution of a numeric variable. Add a vertical line containing the value of `mean` and another one containing the value of `median`. Use `geom_vline` to do so and use `as.numeric` around `mean` and `median`. What do you observe?
4. Compute the correlation between fertility rate 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)  
runTutorial('chapter2')
```



SEE YOU NEXT WEEK!

 florian.oswald@sciencespo.fr

 Slides

 Book

 @ScPoEcon

 @ScPoEcon

