DATA VISUALIZATION

LECTURE 2 INTRO TO R PROGRAMMING

MARIA MONTOYA-AGUIRRE

M1 APE @ PARIS SCHOOL OF ECONOMICS

HOMEWORK REVIEW

✅Inspect the IMDB dataset `glimpse()` `table()` and describe 5 issues we should solve in the data cleaning/tidying process that we haven't addressed in this lecture. Write down the name of at least one function that would help for this. Google if you need!

- The year variable is not numeric and has string characters. Some functions: `str_length()`, `str_sub()`, `as.numeric()`

✅ Are older movies better?

1) Fix the year variable. This can be done in multiple ways! Try with `str_length()` and `str_sub()`

2) Create a variable that indicates whether a movie is older or more recent than the average year in the sample.

3) Create a table with the average rating by decade (start from the 50s).

EXERCISE 1

✅Inspect the IMDB dataset `glimpse()` `table()` and describe 5 issues we should solve in the data cleaning/tidying process that we haven't addressed in this lecture. Write down the name of at least one function that would help for this. Google if you need!

```
imdb <- read.csv("../data/01_imdb-top250-french.csv", encoding = "UTF-8")
glimpse(imdb)
```

```
## Rows: 250
## Columns: 13
## $ X             <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, …
## $ Name          <chr> "Shoah", "Home", "Untouchable", "Le Trou", "The Man Who Planted Trees", "Ch…
## $ Rank          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, …
## $ Year          <chr> "-1985", "(I) (2009)", "-2011", "-1960", "-1987", "-1945", "-2001", "-2010"…
## $ Type          <chr> "PG", "U", "15", "A", "", "A", "15", "15", "A", "U", "U", "A", "AA", "15", …
## $ Duration      <chr> "566 min", "118 min", "112 min", "131 min", "30 min", "189 min", "122 min",…
## $ Genre         <chr> "Documentary, History, War", "Documentary", "Biography, Comedy, Drama", "Cr…
## $ Rating        <dbl> 8.7, 8.5, 8.5, 8.5, 8.5, 8.3, 8.3, 8.3, 8.2, 8.2, 8.2, 8.2, 8.1, 8.1, 8.1, …
## $ MetaScore     <int> 99, 47, 57, NA, NA, 96, 69, 80, NA, NA, NA, 86, 99, NA, NA, 100, 95, 85, 96…
## $ Desc          <chr> "Claude Lanzmann's epic documentary recounts the story of the Holocaust thr…
## $ Director_Stars <chr> "Director:\nClaude Lanzmann\n                | \n    Stars:\nSimon Srebnik…
## $ Votes         <chr> "9,984", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
## $ Gross         <chr> "$0.02M", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
```

EXAMPLE

```
## Rows: 250
## Columns: 13
## $ X             <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, …
## $ Name          <chr> "Shoah", "Home", "Untouchable", "Le Trou", "The Man Who Planted Trees", "Ch…
## $ Rank          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, …
## $ Year          <chr> "-1985", "(I) (2009)", "-2011", "-1960", "-1987", "-1945", "-2001", "-2010"…
## $ Type          <chr> "PG", "U", "15", "A", "", "A", "15", "15", "A", "U", "U", "A", "AA", "15", …
## $ Duration      <chr> "566 min", "118 min", "112 min", "131 min", "30 min", "189 min", "122 min",…
## $ Genre         <chr> "Documentary, History, War", "Documentary", "Biography, Comedy, Drama", "Cr…
## $ Rating        <dbl> 8.7, 8.5, 8.5, 8.5, 8.5, 8.3, 8.3, 8.3, 8.2, 8.2, 8.2, 8.2, 8.1, 8.1, 8.1, …
## $ MetaScore     <int> 99, 47, 57, NA, NA, 96, 69, 80, NA, NA, NA, 86, 99, NA, NA, 100, 95, 85, 96…
## $ Desc          <chr> "Claude Lanzmann's epic documentary recounts the story of the Holocaust thr…
## $ Director_Stars <chr> "Director:\nClaude Lanzmann\n                | \n    Stars:\nSimon Srebnik…
## $ Votes         <chr> "9,984", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
## $ Gross         <chr> "$0.02M", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
```

- The `Year` variable is not numeric and has strings that should not be there "-" "()"

  `str_length()`, `str_sub()`, `as.numeric()`

1

```
## Rows: 250
## Columns: 13
## $ X             <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, …
## $ Name          <chr> "Shoah", "Home", "Untouchable", "Le Trou", "The Man Who Planted Trees", "Ch…
## $ Rank          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, …
## $ Year          <chr> "-1985", "(I) (2009)", "-2011", "-1960", "-1987", "-1945", "-2001", "-2010"…
## $ Type          <chr> "PG", "U", "15", "A", "", "A", "15", "15", "A", "U", "U", "A", "AA", "15", …
## $ Duration      <chr> "566 min", "118 min", "112 min", "131 min", "30 min", "189 min", "122 min",…
## $ Genre         <chr> "Documentary, History, War", "Documentary", "Biography, Comedy, Drama", "Cr…
## $ Rating        <dbl> 8.7, 8.5, 8.5, 8.5, 8.5, 8.3, 8.3, 8.3, 8.2, 8.2, 8.2, 8.2, 8.1, 8.1, 8.1, …
## $ MetaScore     <int> 99, 47, 57, NA, NA, 96, 69, 80, NA, NA, NA, 86, 99, NA, NA, 100, 95, 85, 96…
## $ Desc          <chr> "Claude Lanzmann's epic documentary recounts the story of the Holocaust thr…
## $ Director_Stars <chr> "Director:\nClaude Lanzmann\n                | \n    Stars:\nSimon Srebnik…
## $ Votes         <chr> "9,984", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
## $ Gross         <chr> "$0.02M", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
```

- There are missing values in the `Type` variable that are not coded as missing but are instead only blank `""`. `if_else()`, `NA_character_`

2

```
## Rows: 250
## Columns: 13
## $ X             <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, …
## $ Name          <chr> "Shoah", "Home", "Untouchable", "Le Trou", "The Man Who Planted Trees", "Ch…
## $ Rank          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, …
## $ Year          <chr> "-1985", "(I) (2009)", "-2011", "-1960", "-1987", "-1945", "-2001", "-2010"…
## $ Type          <chr> "PG", "U", "15", "A", "", "A", "15", "15", "A", "U", "U", "A", "AA", "15", …
## $ Duration      <chr> "566 min", "118 min", "112 min", "131 min", "30 min", "189 min", "122 min",…
## $ Genre         <chr> "Documentary, History, War", "Documentary", "Biography, Comedy, Drama", "Cr…
## $ Rating        <dbl> 8.7, 8.5, 8.5, 8.5, 8.5, 8.3, 8.3, 8.3, 8.2, 8.2, 8.2, 8.2, 8.1, 8.1, 8.1, …
## $ MetaScore     <int> 99, 47, 57, NA, NA, 96, 69, 80, NA, NA, NA, 86, 99, NA, NA, 100, 95, 85, 96…
## $ Desc          <chr> "Claude Lanzmann's epic documentary recounts the story of the Holocaust thr…
## $ Director_Stars <chr> "Director:\nClaude Lanzmann\n                | \n    Stars:\nSimon Srebnik…
## $ Votes         <chr> "9,984", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
## $ Gross         <chr> "$0.02M", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
```

- The `Genre` variable is not useful as it is. It should be categorical variable or a dummy. `str_detect()`, `case_when()`

3 & 4

```
## Rows: 250
## Columns: 13
## $ X              <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, …
## $ Name           <chr> "Shoah", "Home", "Untouchable", "Le Trou", "The Man Who Planted Trees", "Ch…
## $ Rank           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, …
## $ Year           <chr> "-1985", "(I) (2009)", "-2011", "-1960", "-1987", "-1945", "-2001", "-2010"…
## $ Type           <chr> "PG", "U", "15", "A", "", "A", "15", "15", "A", "U", "U", "A", "AA", "15", …
## $ Duration       <chr> "566 min", "118 min", "112 min", "131 min", "30 min", "189 min", "122 min",…
## $ Genre          <chr> "Documentary, History, War", "Documentary", "Biography, Comedy, Drama", "Cr…
## $ Rating         <dbl> 8.7, 8.5, 8.5, 8.5, 8.5, 8.3, 8.3, 8.3, 8.2, 8.2, 8.2, 8.2, 8.1, 8.1, 8.1, …
## $ MetaScore      <int> 99, 47, 57, NA, NA, 96, 69, 80, NA, NA, NA, 86, 99, NA, NA, 100, 95, 85, 96…
## $ Desc           <chr> "Claude Lanzmann's epic documentary recounts the story of the Holocaust thr…
## $ Director_Stars <chr> "Director:\nClaude Lanzmann\n                | \n    Stars:\nSimon Srebnik…
## $ Votes          <chr> "9,984", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
## $ Gross          <chr> "$0.02M", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
## [1] "Director:\nClaude Lanzmann\n                | \n    Stars:\nSimon Srebnik, \nMichael Podchlebnik,
\nMotke Zaïdl, \nHanna Zaïdl"
```

- The variable `Director_Stars` stores two variables in one column, it should be split

`separate_wider_delim()`

- The variable Director_Stars should not have the variable name in the content `gsub()`, `str_remove()`

5

```
## Rows: 250
## Columns: 13
## $ X             <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, …
## $ Name          <chr> "Shoah", "Home", "Untouchable", "Le Trou", "The Man Who…
## $ Rank          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, …
## $ Year          <chr> "-1985", "(I) (2009)", "-2011", "-1960", "-1987", "-194…
## $ Type          <chr> "PG", "U", "15", "A", "", "A", "15", "15", "A", "U", "U…
## $ Duration      <chr> "566 min", "118 min", "112 min", "131 min", "30 min", "…
## $ Genre         <chr> "Documentary, History, War", "Documentary", "Biography,…
## $ Rating        <dbl> 8.7, 8.5, 8.5, 8.5, 8.5, 8.3, 8.3, 8.3, 8.2, 8.2, 8.2, …
## $ MetaScore     <int> 99, 47, 57, NA, NA, 96, 69, 80, NA, NA, NA, 86, 99, NA,…
## $ Desc          <chr> "Claude Lanzmann's epic documentary recounts the story …
## $ Director_Stars <chr> "Director:\nClaude Lanzmann\n                | \n    S…
## $ Votes         <chr> "9,984", "", "", "", "", "", "", "", "", "", "", "", ""…
## $ Gross         <chr> "$0.02M", "", "", "", "", "", "", "", "", "", "", "", "…
```

- All the values in the `Gross` and `Votes` variables are missing

`select()`

EXERCISE 2

1) Fix the year variable. This can be done in multiple ways! Try with `str_length()` and `str_sub()`

```
imdb$Year[1:50]
```

```
##  [1] "-1985"      "(I) (2009)" "-2011"      "-1960"      "-1987"
##  [6] "-1945"      "-2001"      "-2010"      "-1962"      "-1956"
## [11] "-1902"      "-1969"      "-1969"      "-1995"      "-1959"
## [16] "-1994"      "-2019"      "-1953"      "-1966"      "(I) (2014)"
## [21] "-1928"      "-1967"      "-1937"      "-1955"      "-2007"
## [26] "-2007"      "-1955"      "-1952"      "-1987"      "-1962"
## [31] "-1939"      "-1986"      "-1970"      "-1958"      "-1986"
## [36] "-1973"      "-1983"      "-1966"      "-1950"      "-1953"
## [41] "-1963"      "-1966"      "-1973"      "(I) (2011)" "-2009"
## [46] "-2012"      "-1993"      "-1960"      "-2004"      "-1972"
```

- There are two formats: `-YYYY` and `(I) (YYYY)`. We need to keep only the `YYYY` part to convert it to numeric.
- What do `str_length()` and `str_sub()` do?

1) Fix the year variable. This can be done in multiple ways! Try with `str_length()` and `str_sub()`

2) Create a variable that indicates whether a movie is older or more recent than the average year in the sample.

```
imdb <-
  imdb %>%
  # Fix year variable
  mutate(year = if_else(str_length(Year) == 5,   # If format is "-YYYY",
                        str_sub(Year, 2, 5),     #  extract positions 2-5
                        str_sub(Year, 6, 9)),    #  otherwise it's "(I) (YYYY)"
                                                 #  so, keep positions 6-9
         year = as.numeric(year)) %>%            # Convert to numeric




#
```

1) Fix the year variable. This can be done in multiple ways! Try with `str_length()` and `str_sub()`

2) Create a variable that indicates whether a movie is older or more recent than the average year in the sample.

```
imdb <-
  imdb %>%
  # Fix year variable
  mutate(year = if_else(str_length(Year) == 5,   # If format is "-YYYY",
                        str_sub(Year, 2, 5),     #  extract positions 2-5
                        str_sub(Year, 6, 9)),    #  otherwise it's "(I) (YYYY)"
                                                 #  so, keep positions 6-9
         year = as.numeric(year)) %>%            # Convert to numeric

  # Create an "older than average" indicator
  mutate(avg_year = mean(year),                  # Calculate average year
         older_than_avg = year < avg_year) %>%   # Create indicator
  select(Name, year, avg_year, older_than_avg, Rating )

head(imdb)
```

OTHER WAYS TO FIX YEAR FROM YOUR ANSWERS

Using **regular expressions**, a concise language for describing patterns strings. See a cheatsheet on how to use them with package {stringr} here.

```
imdb %>%
  mutate(Year = ...)

str_extract(Year,"[:digit:]{4}") # Extract exactly 4 digits
str_replace_all(Year, "[()I-]", "") # Replace the characters ()I- for nothing
- gsub("[^0-9.]+", "", Year)  # Replace anything that is not 0-9 (one or more)
```

3) Create a table with the average rating by decade (start from the 50s).

```
imdb %>%
  # Create decade indicator
  mutate(decade = case_when(year %in% c(1950:1959) ~ "50s",
                            year %in% c(1960:1969) ~ "60s",
                            year %in% c(1970:1979) ~ "70s",
                            year %in% c(1980:1989) ~ "80s",
                            year %in% c(1990:1999) ~ "90s",
                            year %in% c(2000:2009) ~ "2000s",
                            year %in% c(2010:2019) ~ "2010s",
                            year %in% c(2019:2029) ~ "2020s",
                            .default = "40s or older")) %>%




#
```

3) Create a table with the average rating by decade (start from the 50s).

```
  imdb %>%
    # Create decade indicator
    mutate(decade = case_when(year %in% c(1950:1959) ~ "50s",
                              year %in% c(1960:1969) ~ "60s",
                              year %in% c(1970:1979) ~ "70s",
                              year %in% c(1980:1989) ~ "80s",
                              year %in% c(1990:1999) ~ "90s",
                              year %in% c(2000:2009) ~ "2000s",
                              year %in% c(2010:2019) ~ "2010s",
                              year %in% c(2019:2029) ~ "2020s",
                              .default = "40s or older")) %>%

    # Create a table with average rating by decade
    group_by(decade) %>%
    summarise(avg_rating = mean(Rating))
#
```

3) Create a table with the average rating by decade (start from the 50s).

```
imdb %>%
  # Create decade indicator
  mutate(decade = case_when(year %in% c(1950:1959) ~ "50s",
                            year %in% c(1960:1969) ~ "60s",
                            year %in% c(1970:1979) ~ "70s",
                            year %in% c(1980:1989) ~ "80s",
                            year %in% c(1990:1999) ~ "90s",
                            year %in% c(2000:2009) ~ "2000s",
                            year %in% c(2010:2019) ~ "2010s",
                            year %in% c(2019:2029) ~ "2020s",
                            .default = "40s or older")) %>%

  # Create a table with average rating by decade
  group_by(decade) %>%
  summarise(avg_rating = mean(Rating)) %>%
  arrange(-avg_rating)
```

OTHER WAYS TO CREATE THE DECADE FROM YOUR ANSWERS

```
imdb %>%
  mutate(decade = ...)

paste(str_sub(Year, start = 3, end = 3),"0s", sep="") # Get the third string, and add "0s"

as.numeric(substr(Year, start = 1, stop=3)) * 10 # Get first three digits of year and multiply by 10

floor(Year/10) * 10 # Divide the year by 10 and round downwards

Year %/% 10 * 10 # Integer division, get the quotient
```

WARM-UP USING {dplyr}

- Import `02_taylor-swift-spotify.csv` (originally obtained here) and `View()` the data
- Inspect the structure of the data using `glimpse()`
- Use `summarise()` to compute for each album the average danceability and the the number of songs included (1 row per song)
- Create a subset of the data called `maxpop`containing the variables `album`, `release_date`, `danceability` and `popularity` for the 10 most popular songs. Use function `arrange()` and `row_number()`

10:00

WARM-UP

- Import `02_taylor-swift-spotify.csv` (originally obtained here) and `View()` the data
- Inspect the structure of the data using `glimpse()`

```
ts <- read.csv("../data/02_taylor-swift-spotify.csv")

glimpse(ts)
```

```
## Rows: 487
## Columns: 21
## $ X                 <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, …
## $ name              <chr> "Mine (Taylor's Version)", "Sparks Fly (Taylor's Ve…
## $ album             <chr> "Speak Now", "Speak Now", "Speak Now", "Speak Now",…
## $ release_date      <chr> "2023-07-07", "2023-07-07", "2023-07-07", "2023-07-…
## $ track_number      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, …
## $ id                <chr> "7G0gBu6nLdhFDPRLc0HdDG", "3MytWN8L7shNYzGl4tAKRp",…
## $ uri               <chr> "spotify:track:7G0gBu6nLdhFDPRLc0HdDG", "spotify:tr…
## $ acousticness      <dbl> 0.004440, 0.025100, 0.006210, 0.248000, 0.023600, 0…
....
```

WARM-UP

- Use `summarise()` to compute for each album the average danceability and the the number of songs included (1 row per song)

```
ts %>%
group_by(album) %>%
summarise(avg_danceability = mean(danceability),
        songs = n())
```

WARM-UP

- Create a subset of the data called `maxpop` containing the variables `album`, `release_date`, `danceability` and `popularity` for the 10 most popular songs. Use function `arrange()` and `row_number()`

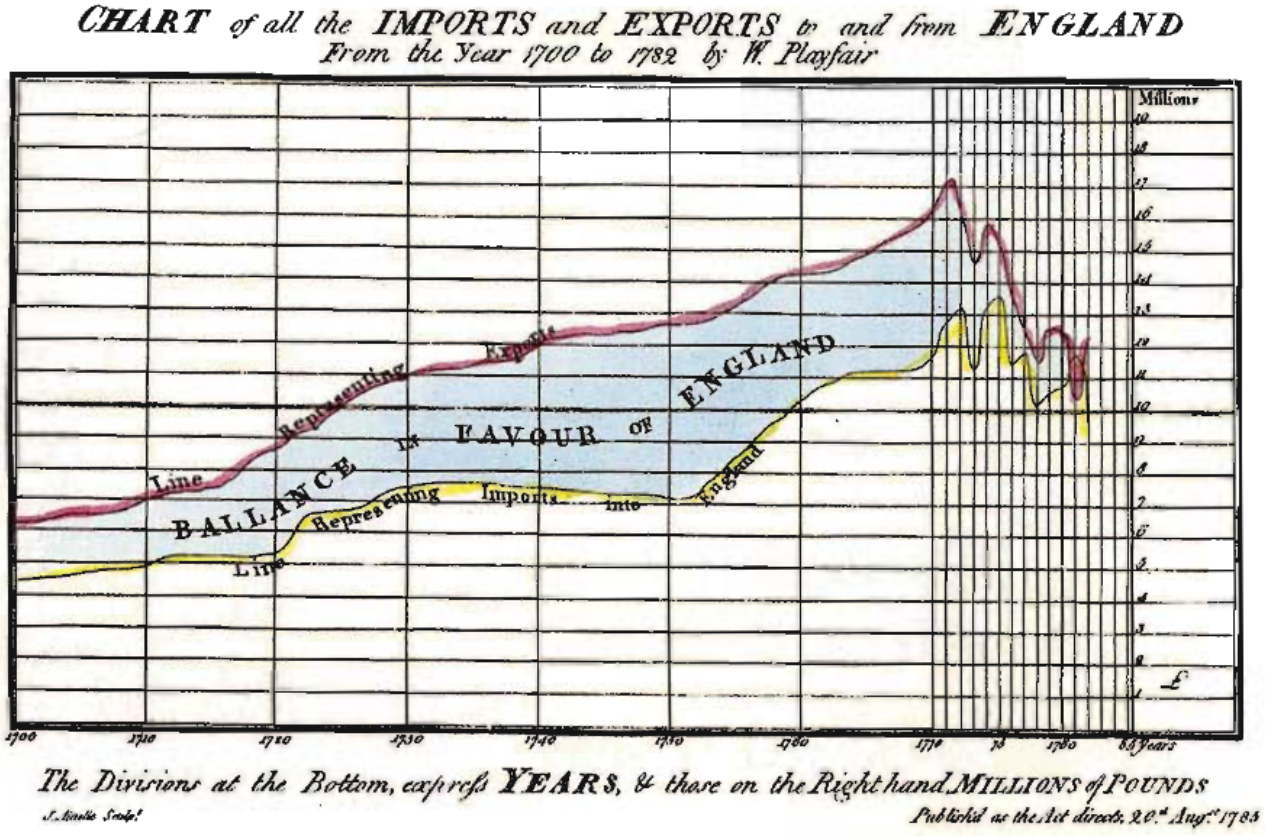
```
maxpop <-
  ts %>%
  arrange(-popularity) %>%          # Order by popularity (descending)
  select(album, release_date, danceability, popularity) %>%
  filter(row_number() <= 10)        # Keep first 10 rows

maxpop
```

AGENDA

- AN INTRO TO DATA VISUALIZATION
- THE ggplot() FUNCTION
    1. MAIN STRUCTURE
    2. AESTHETICS
    3. ADDING DIMENSIONS
    4. MAPPING vs. SETTING ATTRIBUTES
    5. SAVING
- BROWSING THE TOOLBOX
    1. LABELS
    2. ANNOTATIONS
    3. ADDING LAYERS
    4. GROUPING DATA
    5. STATISTICAL SUMMARIES
- WHICH GRAPH SHOULD I USE?
- GRAPHICAL EXCELLENCE AND INTEGRITY

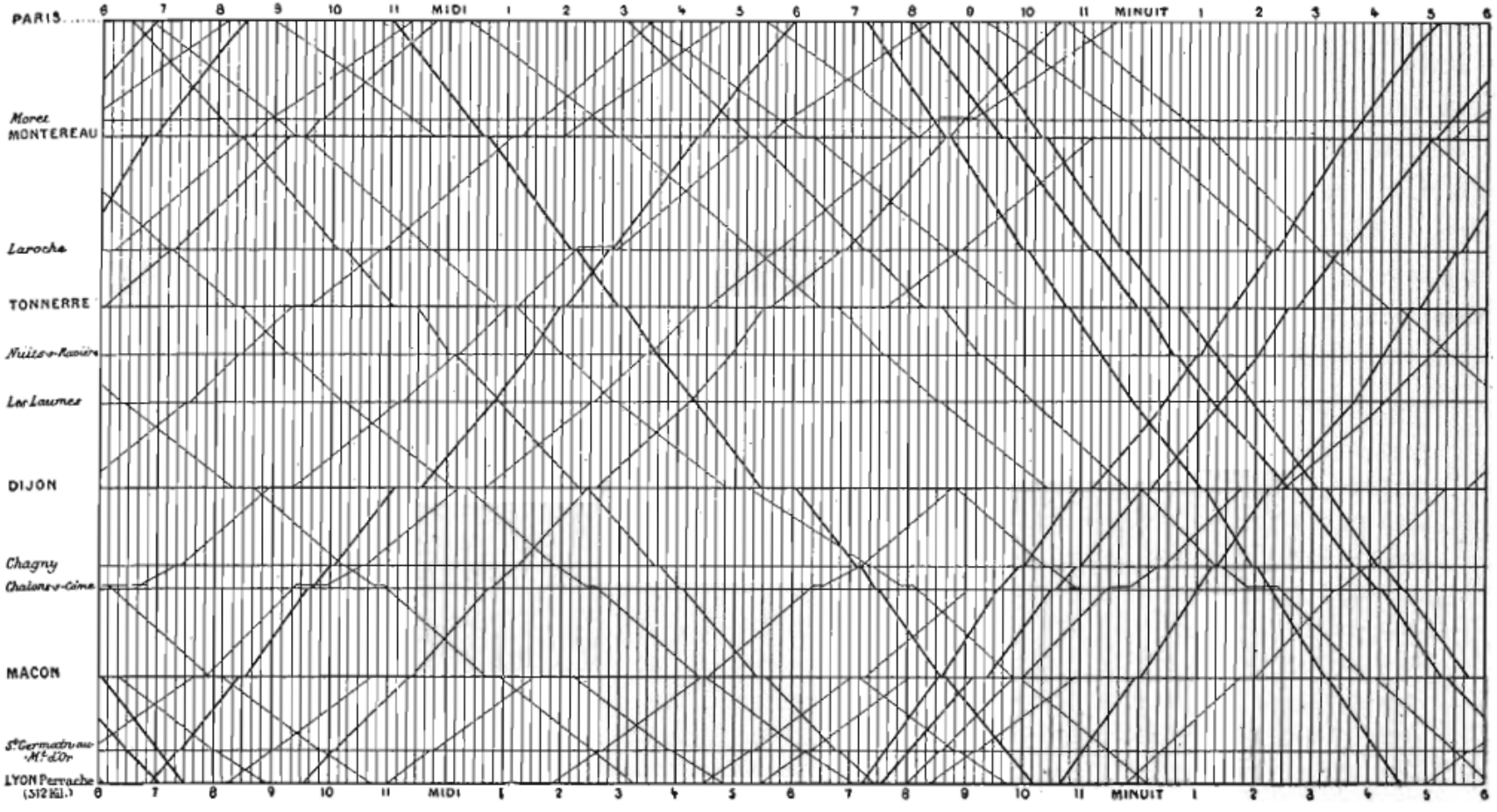
DATA VISUALIZATION

First known graphics applied to economic data were produced by William Playfair (1759-1823), a Scottish political economist in his book *The Commercial and Political Atlas* [1]



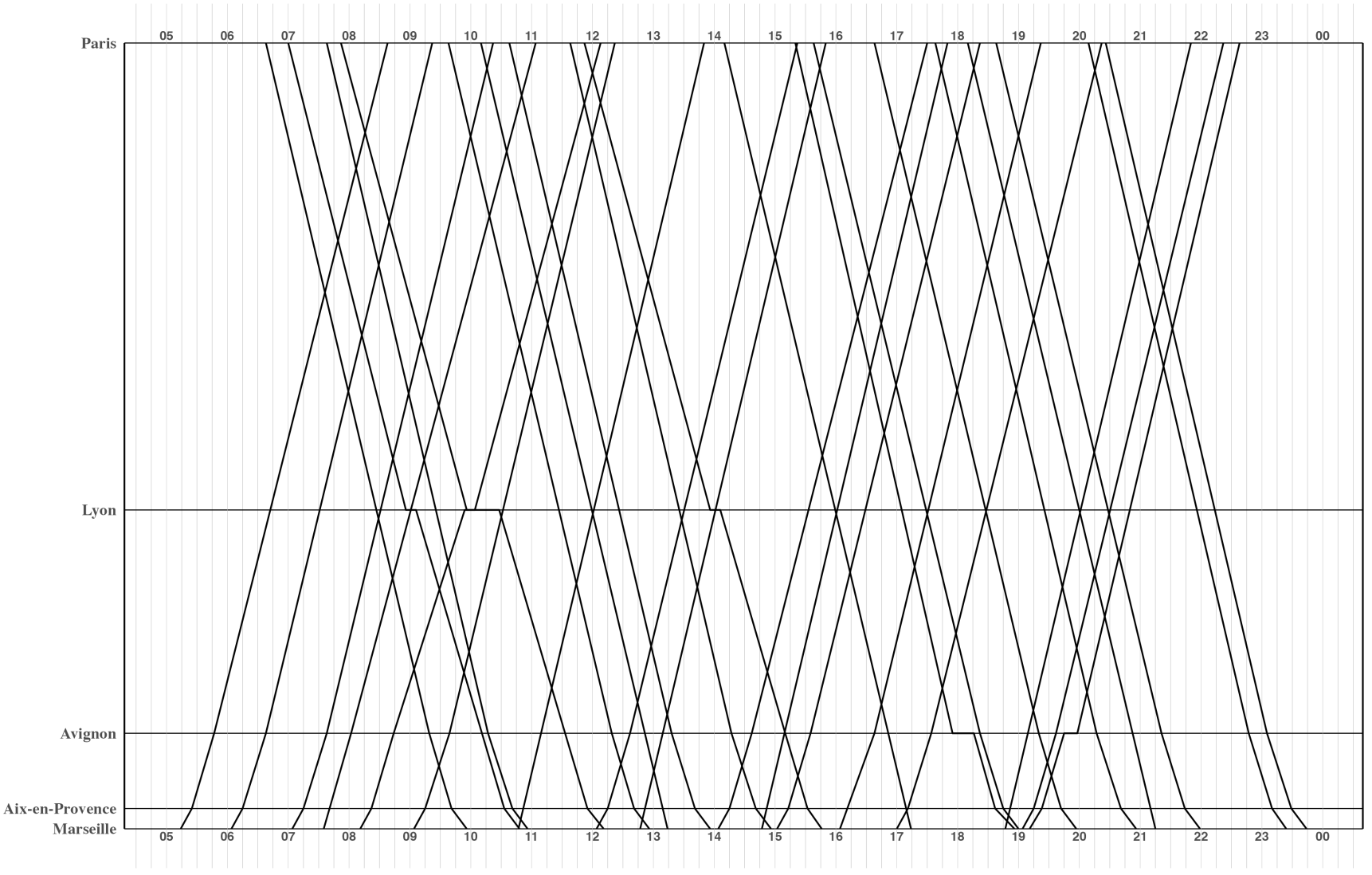
[1]Tilling, 1975 as cited in Tufte, 2007.

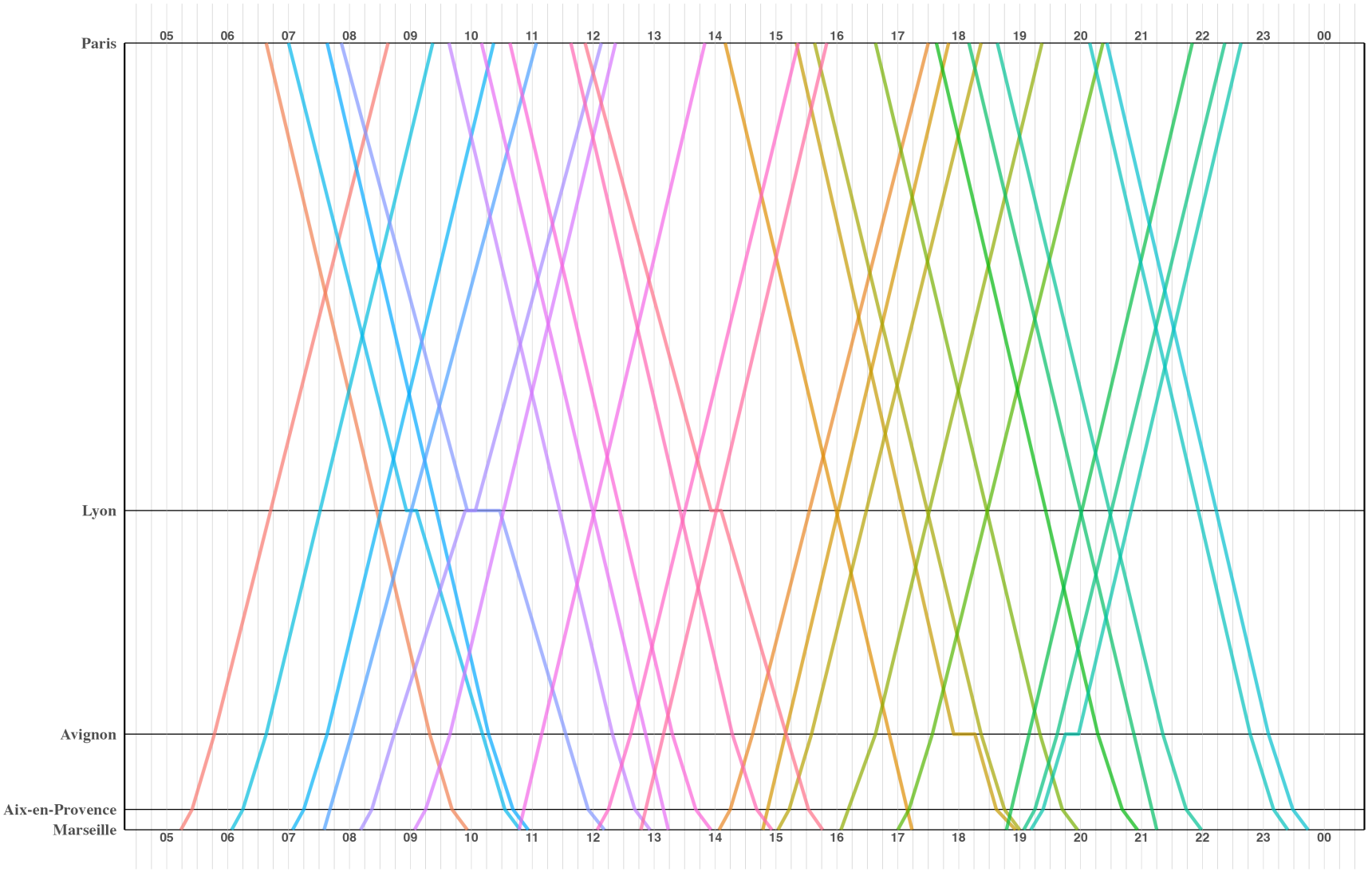
> Information, that is imperfectly acquired, is generally imperfectly retained; and a man (*person*) who has carefully investigated **a printed table**, finds, when done, that (s)he has **only a very faint and partial idea** of what (s)he has read; and that like a figure imprinted on sand, is **soon totally erased and defaced**. [...]
>
> On **inspecting any one of these Charts** attentively, a **sufficiently distinct impression** will be made, to **remain unimpaired for a considerable time**, and the idea which does remain will be **simple and complete** [...]
>
> --- William Playfair in The Commercial and Political Atlas

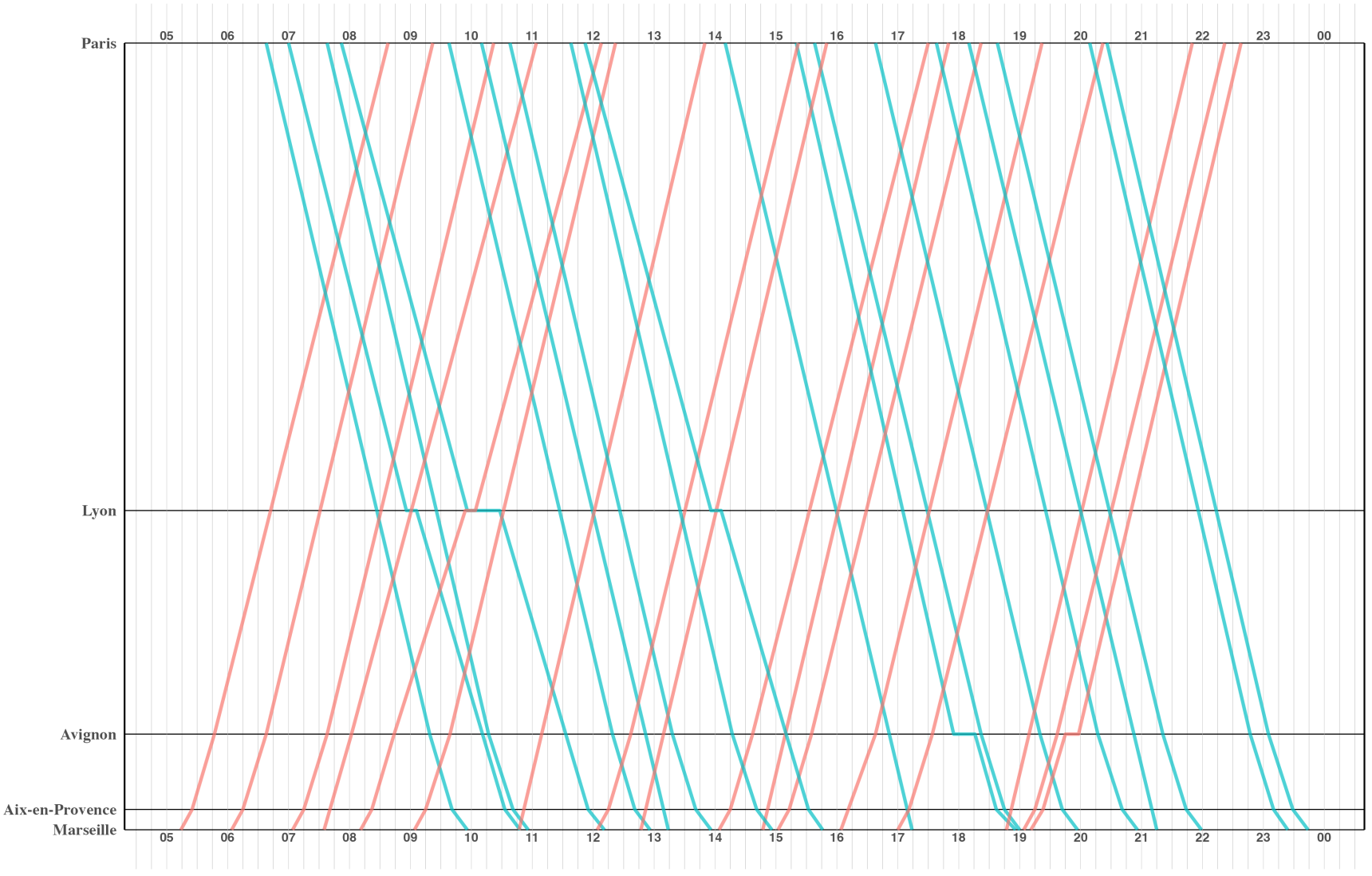


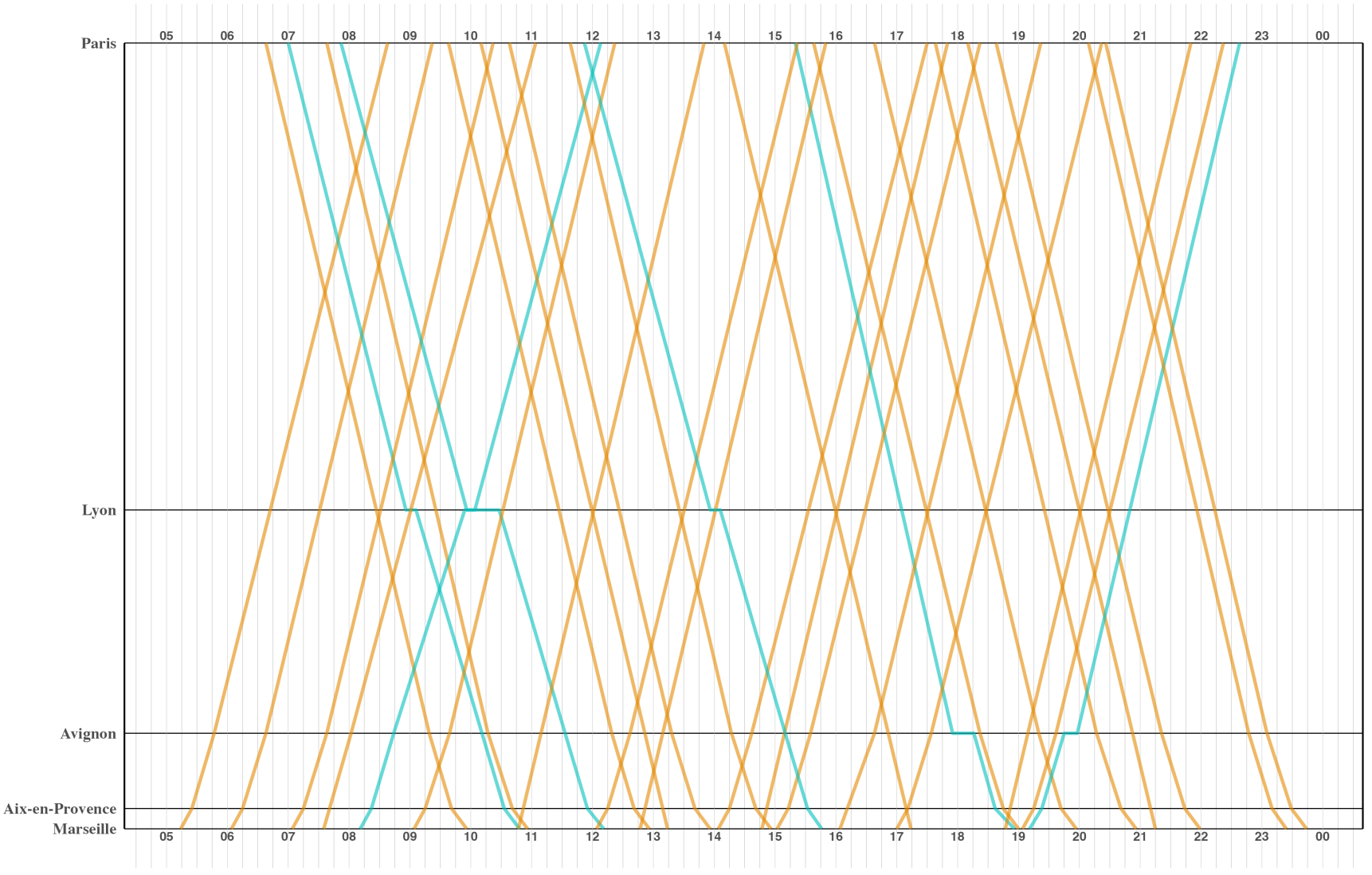
E. J. Marey, *La méthode graphique* (Paris, 1885), p. 20. The method is attributed to the French engineer, Ibry.

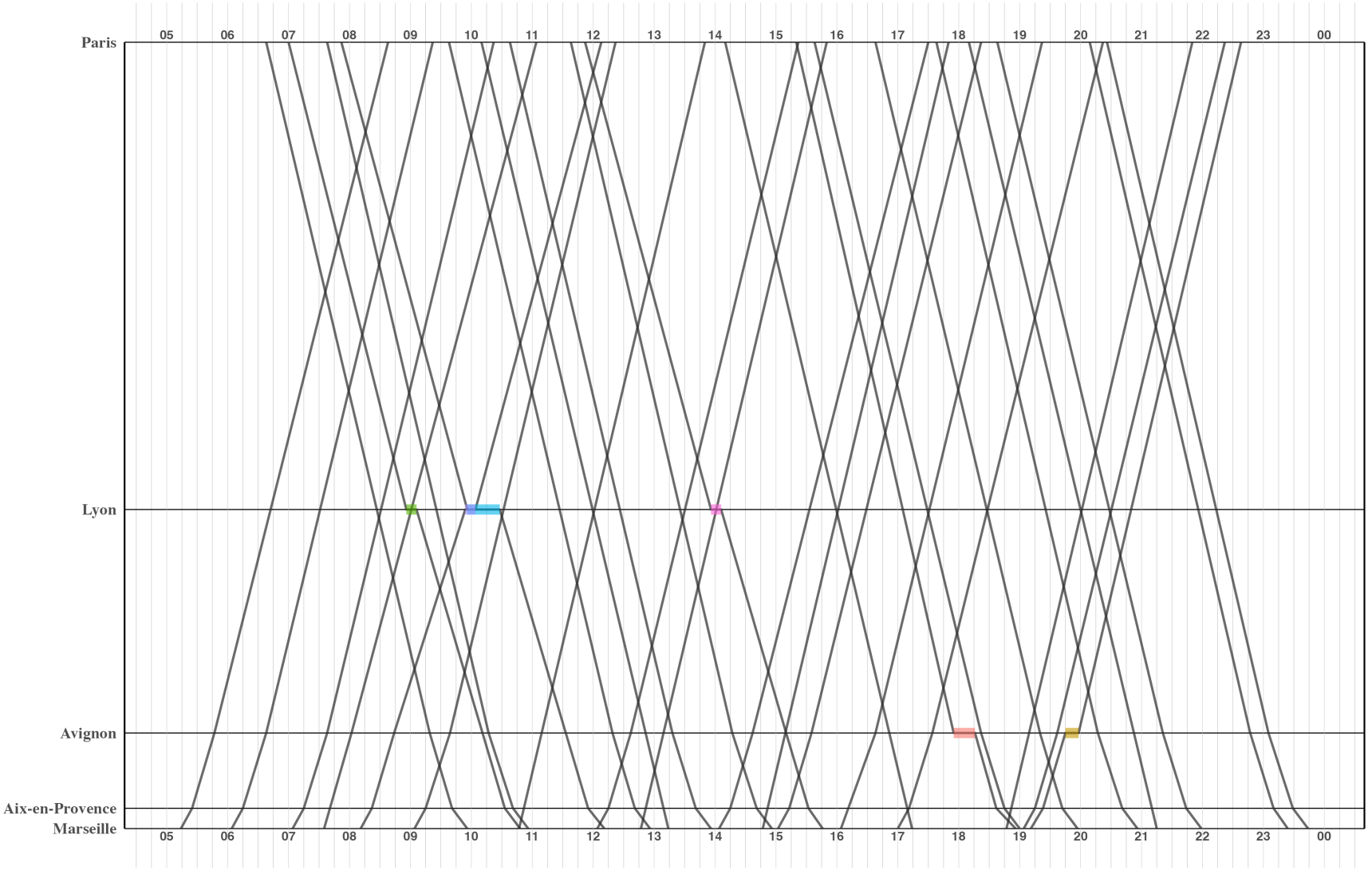
Now, using R with the {ggplot2} package and the current Paris - Marseille schedule


Paris
Lyon
Avignon
Aix-en-Provence
Marseille
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
00


Paris
Lyon
Avignon
Aix-en-Provence
Marseille
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
00


Paris
Lyon
Avignon
Aix-en-Provence
Marseille
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
00


Paris
Lyon
Avignon
Aix-en-Provence
Marseille
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
00


Paris
Lyon
Avignon
Aix-en-Provence
Marseille
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
00


ggplot2:
Build a data
MASTERPIECE
themes
geoms
aes
fill
color
ggplot
gganimate
plotly
HORST '18

PREPARING THE DATA FOR TODAY

We are going to use data from the World Inequality Database.

```
wid <- read.csv("../data/02_wid.csv")
glimpse(wid)
```

```
## Rows: 1,610
## Columns: 6
## $ country   <chr> "Algeria", "Algeria", "Algeria", "Algeria", "Algeria", "Alge…
## $ continent <chr> "Africa", "Africa", "Africa", "Africa", "Africa", "Africa", …
## $ year      <int> 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, …
## $ fshare    <dbl> 0.0992, 0.1120, 0.1201, 0.1206, 0.1160, 0.1221, 0.1232, 0.12…
## $ top1      <dbl> 0.1003, 0.0991, 0.0991, 0.0991, 0.0991, 0.0991, 0.0991, 0.09…
## $ inc_head  <dbl> 12610.627, 12619.984, 12634.026, 12531.988, 12546.430, 12533…
```

What is the observation level? Country-year
Variables:

- `fshare` Female labor income share
- `top1` Top 1% income share
- `inc_head` Income per capita (adults)

ggplot() BASICS

Based on *The Grammar of graphics* (Wilkinson, 2005).
What is a statistical graphic? A mapping from data to aesthetic attributes (color, shape, size) of geometric objects (points, lines, bars).

`ggplot()` creates a canvas to draw on.

```
#install.packages("ggplot2")
library(ggplot2)

ggplot()
```

ggplot() BASICS

Based on *The Grammar of graphics* (Wilkinson, 2005).
What is a statistical graphic? A mapping from data to aesthetic attributes (color, shape, size) of geometric objects (points, lines, bars).

`ggplot()` creates a canvas to draw on. Then we add:

🔢🔤 **Data:** specifies the dataframe with the **values** to plot

```
#install.packages("ggplot2")
library(ggplot2)

ggplot(data = wid,          # Data
```

ggplot() BASICS

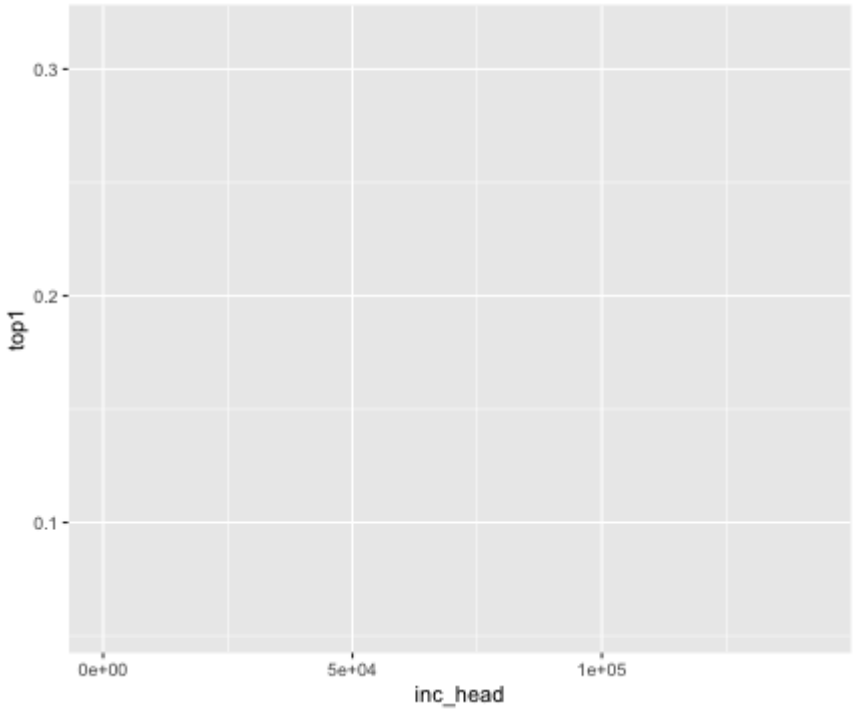
Based on *The Grammar of graphics* (Wilkinson, 2005).
What is a statistical graphic? A mapping from data to aesthetic attributes (color, shape, size) of geometric objects (points, lines, bars).

`ggplot()` creates a canvas to draw on. Then we add:

🔢🔡 **Data:** specifies the dataframe with the **values** to plot

📐🖌️**Aesthetic mappings:** relate variables in the data to visual characteristics of the plot.

```
#install.packages("ggplot2")
library(ggplot2)

ggplot(data = wid,           # Data
       aes(x = inc_head,     # Aesthetics
           y = top1)) +
```



ggplot() BASICS

Based on *The Grammar of graphics* (Wilkinson, 2005).
What is a statistical graphic? A mapping from data to aesthetic attributes (color, shape, size) of geometric objects (points, lines, bars).

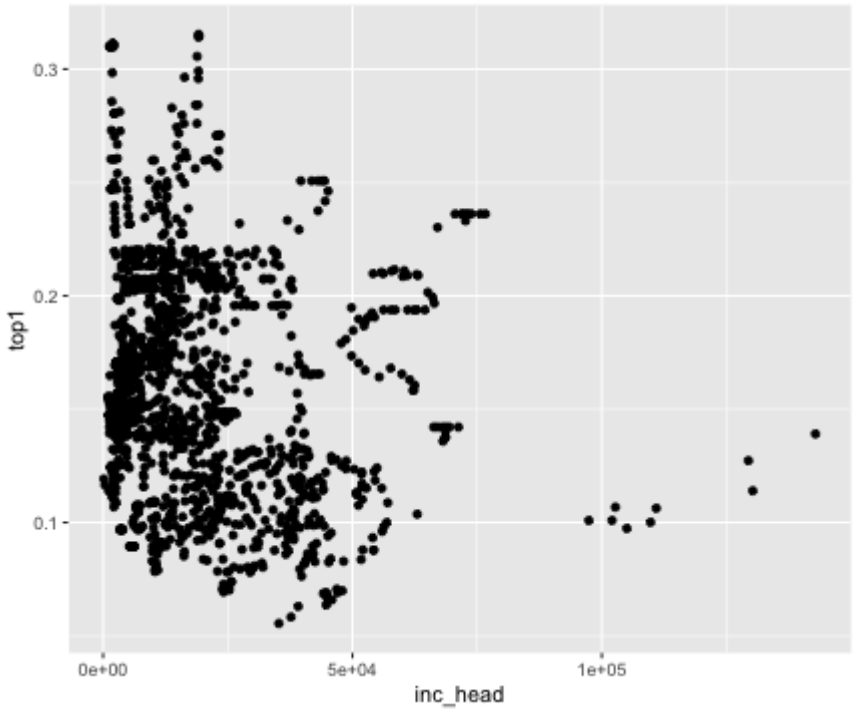
`ggplot()` creates a canvas to draw on. Then we add:

🔢🔡 **Data:** specifies the dataframe with the **values** to plot

📐🖌️**Aesthetic mappings:** relate variables in the data to visual characteristics of the plot.

📈📊 **Geometries:** describe how to render each observation. Can be layered to have multiple representations of the data.

```
#install.packages("ggplot2")
library(ggplot2)

ggplot(data = wid,           # Data
       aes(x = inc_head,     # Aesthetics
           y = top1)) +
  geom_point()               # Geometry
```



ggplot() BASICS

Based on *The Grammar of graphics* (Wilkinson, 2005).
What is a statistical graphic? A mapping from data to aesthetic attributes (color, shape, size) of geometric objects (points, lines, bars).

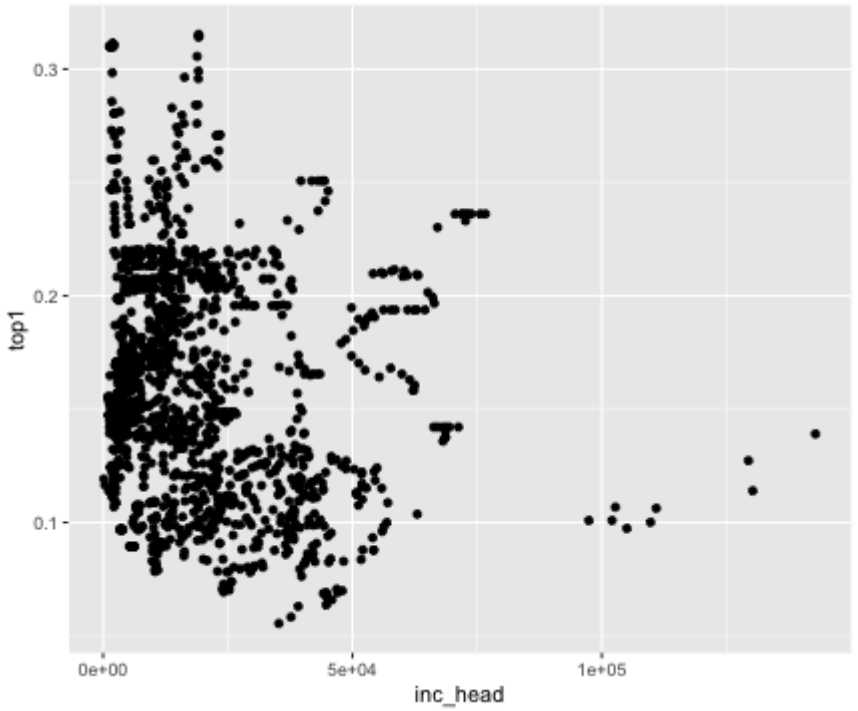
`ggplot()` creates a canvas to draw on. Then we add:

🔢🔡 **Data:** specifies the dataframe with the **values** to plot

📐🖌️**Aesthetic mappings:** relate variables in the data to visual characteristics of the plot.

📈📊 **Geometries:** describe how to render each observation. Can be layered to have multiple representations of the data.

```
#install.packages("ggplot2")
library(ggplot2)

ggplot(data = wid,           # Data
       aes(x = inc_head,     # Aesthetics
           y = top1)) +
  geom_point()               # Geometry
```



MAIN STRUCTURE

- Data and mapping should be specified in parentheses
- The first two arguments in `aes()` are almost always x,y
- Geometry and other elements should be added with `+`

```
ggplot(wid,
       aes(inc_head, top1)) +
  geom_point()
```

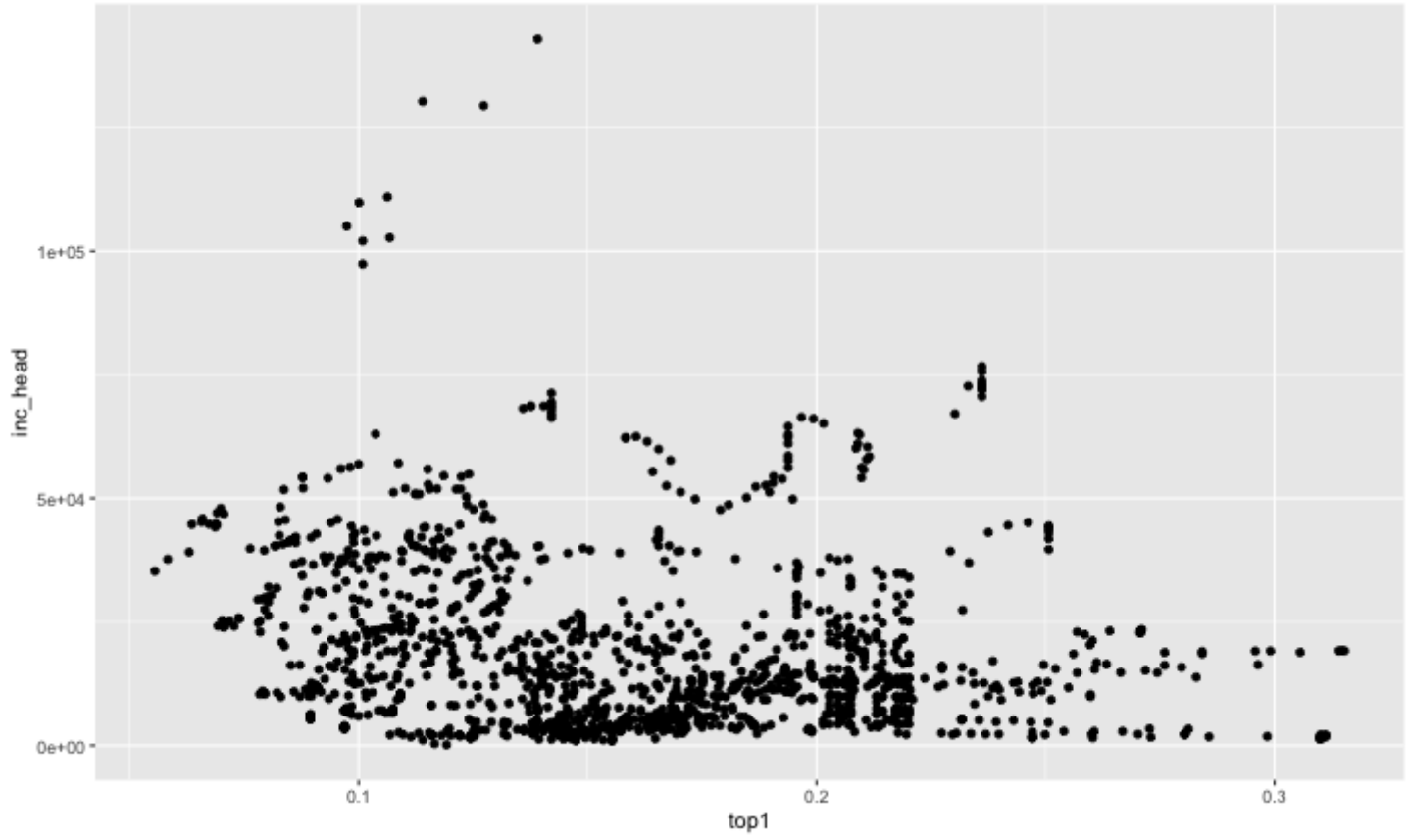
Remember the pipe?

- We can also apply `ggplot()` to our data with the pipe `%>%` OR `|>`

```
wid %>%                     # Data
ggplot(aes(x = inc_head,    # Aesthetics
           y = top1)) +
  geom_point()              # Geometry
```

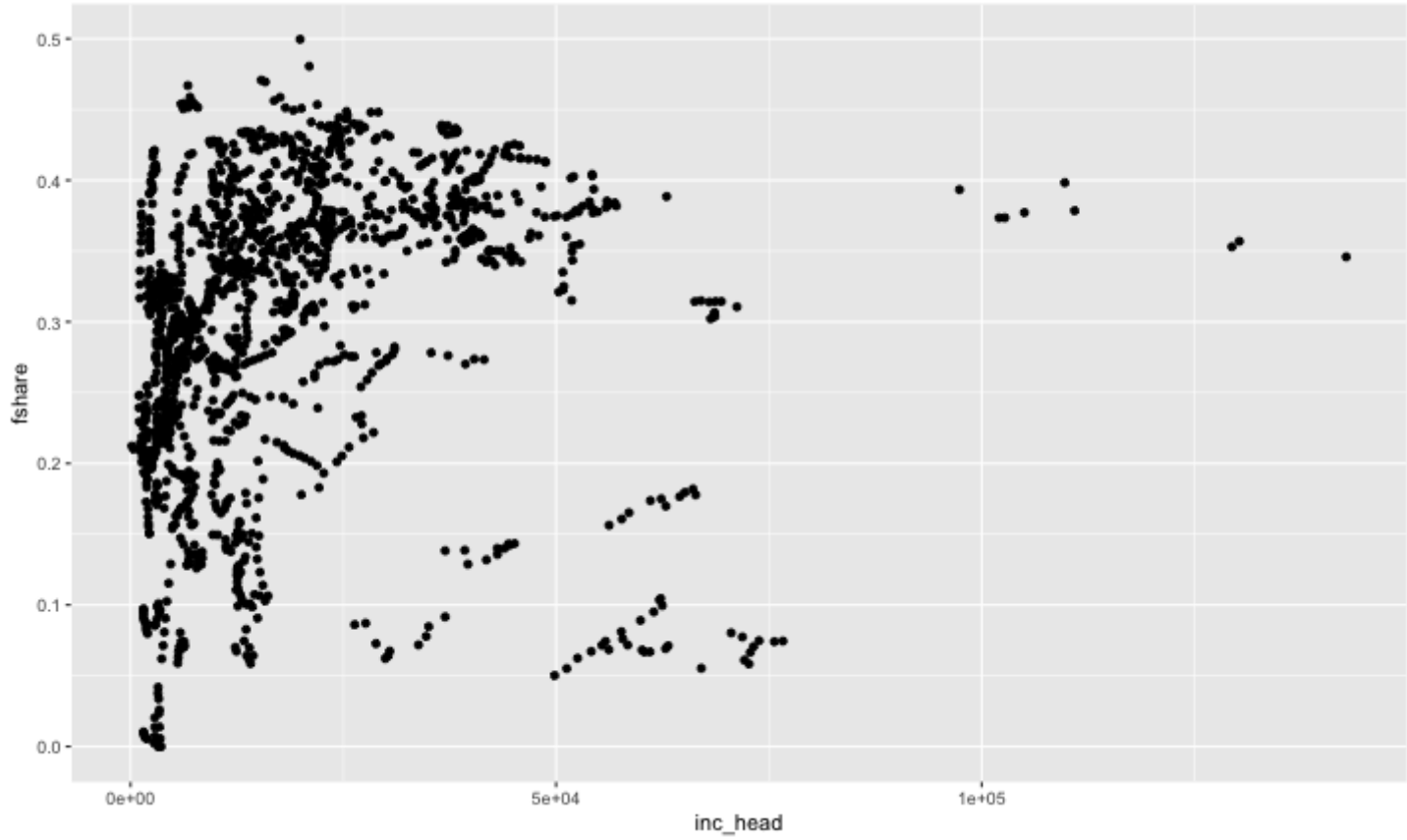
Can you guess the plots?

```
wid %>%
  ggplot(aes(top1, inc_head)) +
  geom_point()
```



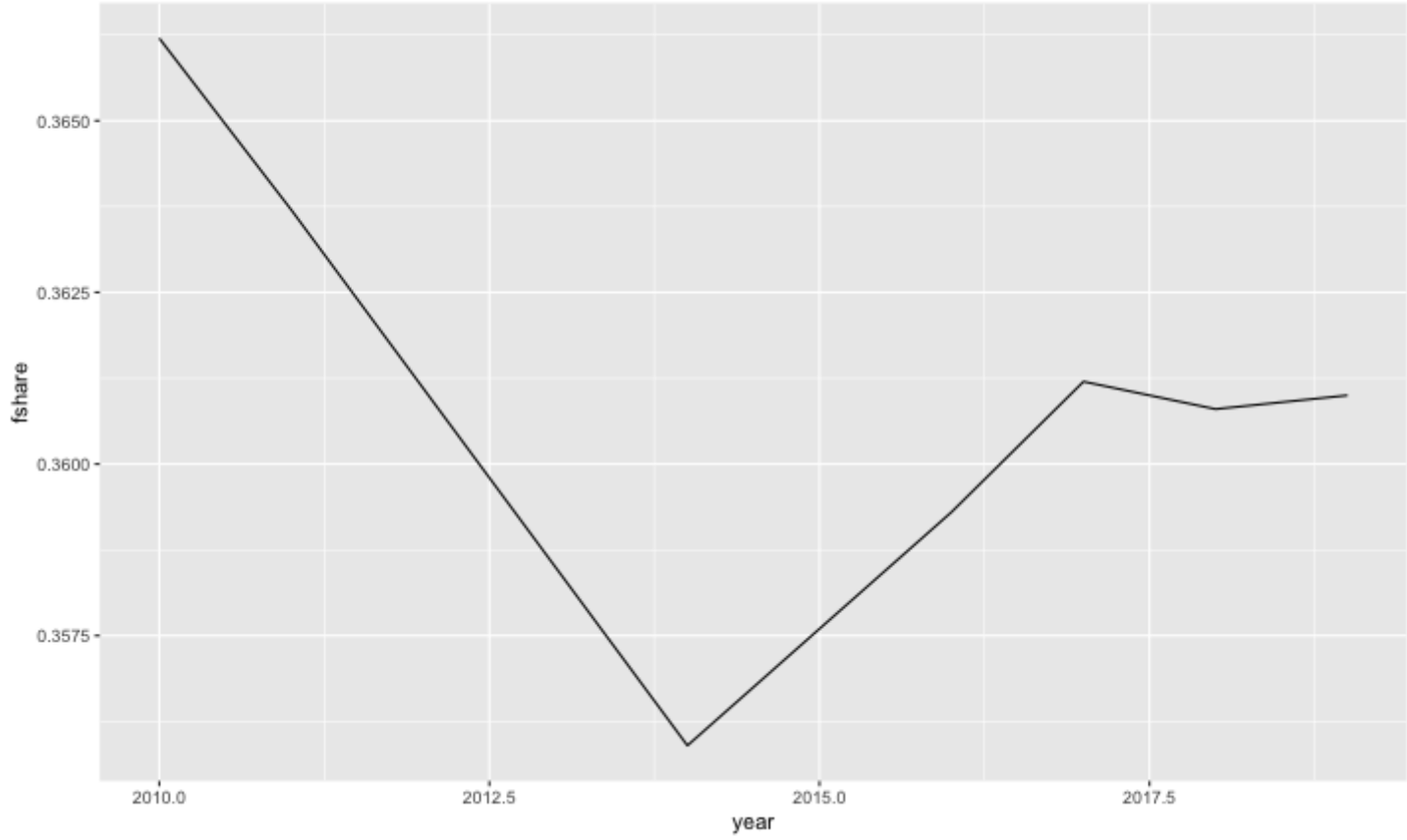
Can you guess the plots?

```
wid %>%
  ggplot(aes(inc_head, fshare)) +
  geom_point()
```



Can you guess the plots?

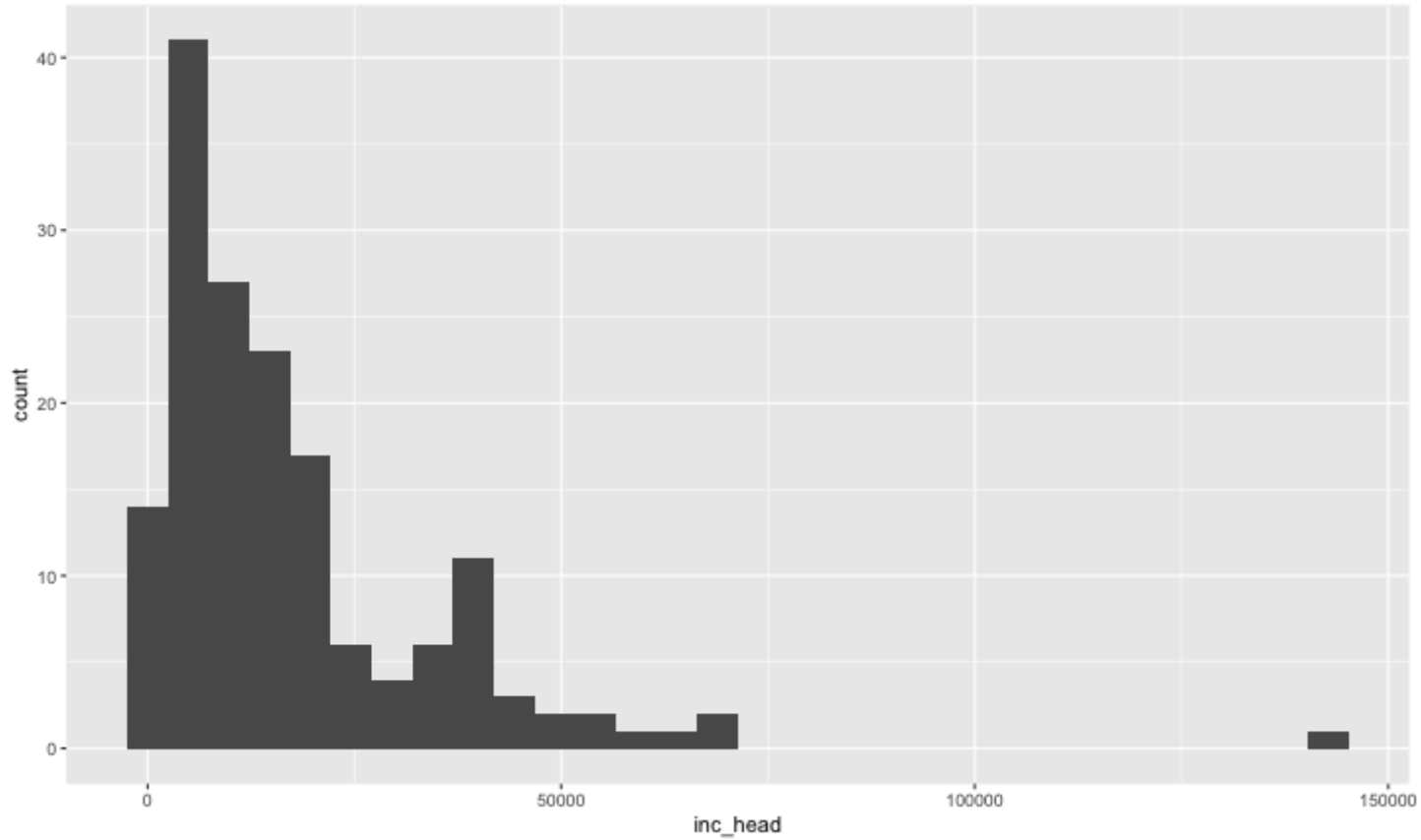
```
wid %>%
  filter(country == "Italy") %>%
  ggplot(aes(year, fshare)) +
  geom_line()
```



Can you guess the plots?

```
wid %>%
  filter(year == "2010") %>%
  ggplot(aes(inc_head)) +
  geom_histogram()
```

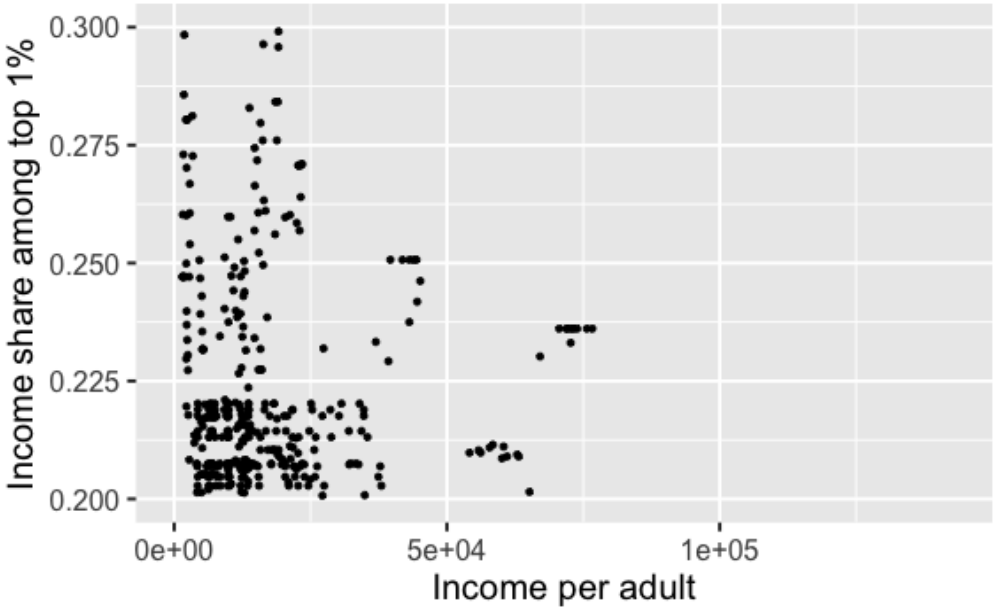
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



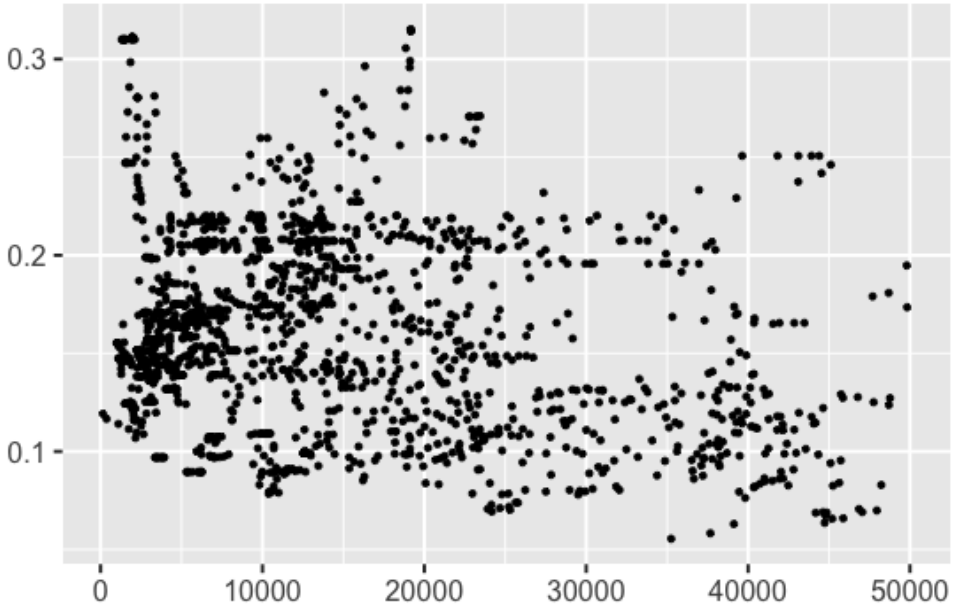
AESTHETICS:

AXES The quickest functions for the most common modifications:

```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point() +
  xlab("Income per adult") +
  ylab("Income share among top 1%") +
  xlim(0.2, 0.3)
```




```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point() +
  xlab(NULL) +
  ylab("") +
  ylim(NA, 50000)
```

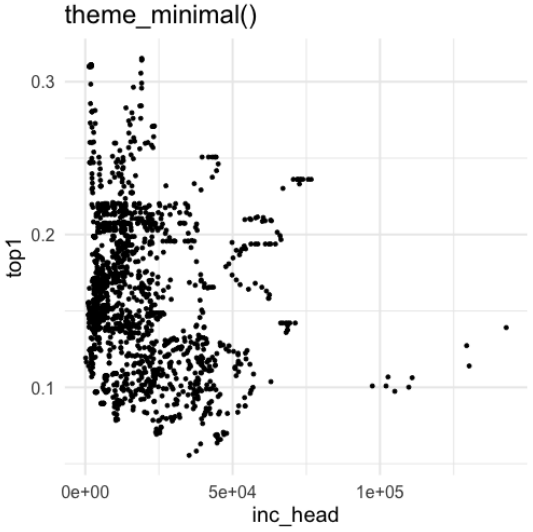


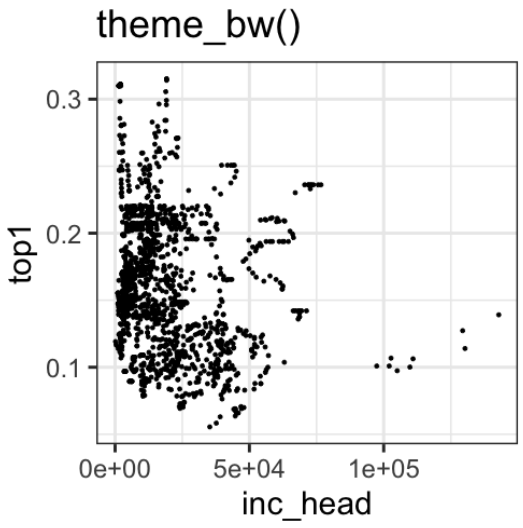
AESTHETICS

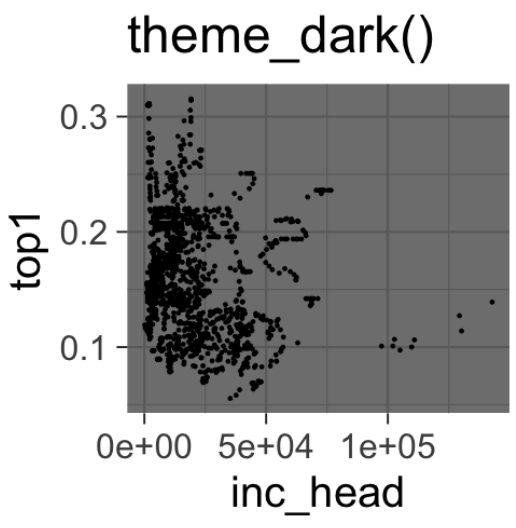
THEME

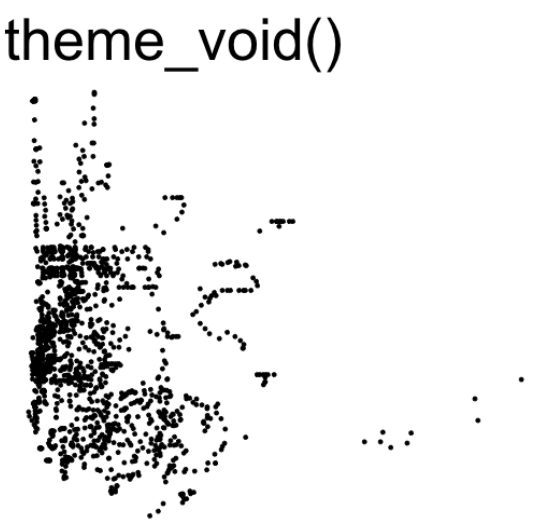
- The theme system allows you to have control over the appearance of all the **non-data** elements of the plot
- You can use any of the **default R themes**. Also, you can specify the **font size** using `base_size`

```
... +
  theme_gray(base_size = 10) #<< # The default theme
```







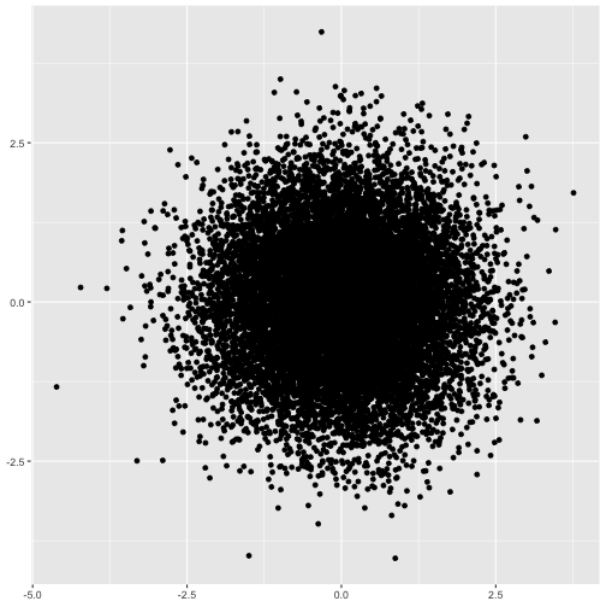


- You can customize the graph completely using the theme() function
- See `?theme` for the endless possibilities and Chapter 8 of Wickham (2016)

AESTHETICS GEOM ATTRIBUTES

We can set the appearance of our observations by specifying their shape, color, fill, size/width, and transparency (*alpha*). **Aesthetic attributes can make or break a plot:**

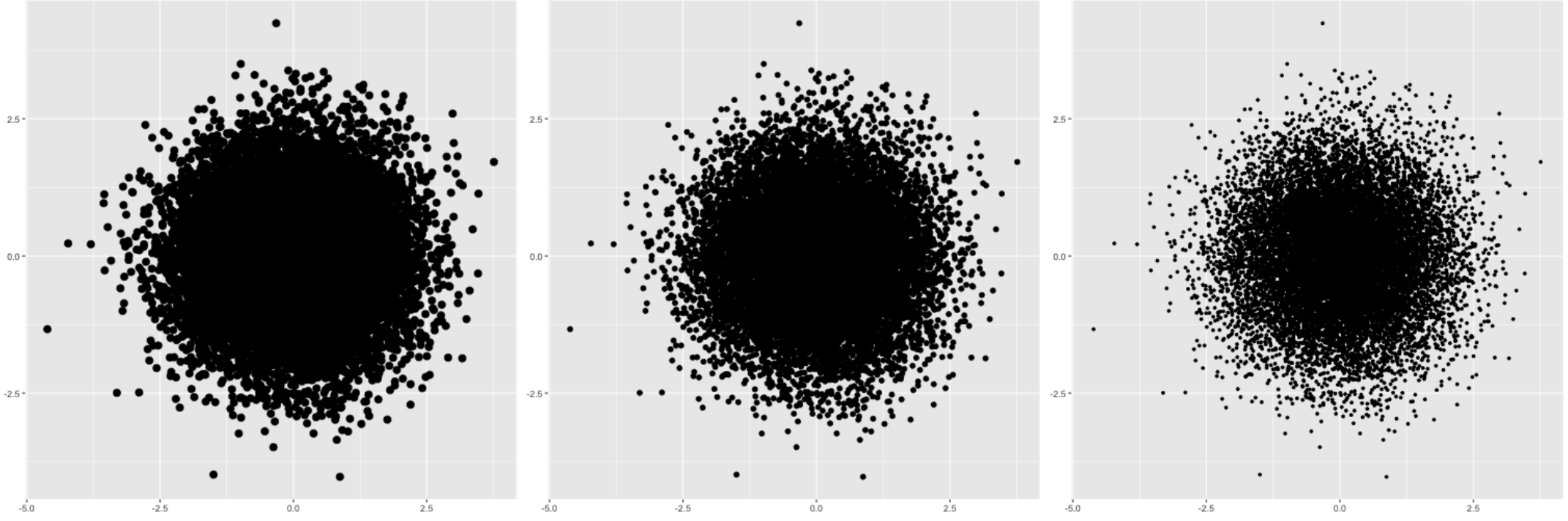
```
df <- data.frame(x = rnorm(15000), y = rnorm(15000))
norm <-
  df %>%
  ggplot(aes(x,y))

norm + geom_point()
```



How to deal with overplotting?

Changing the size

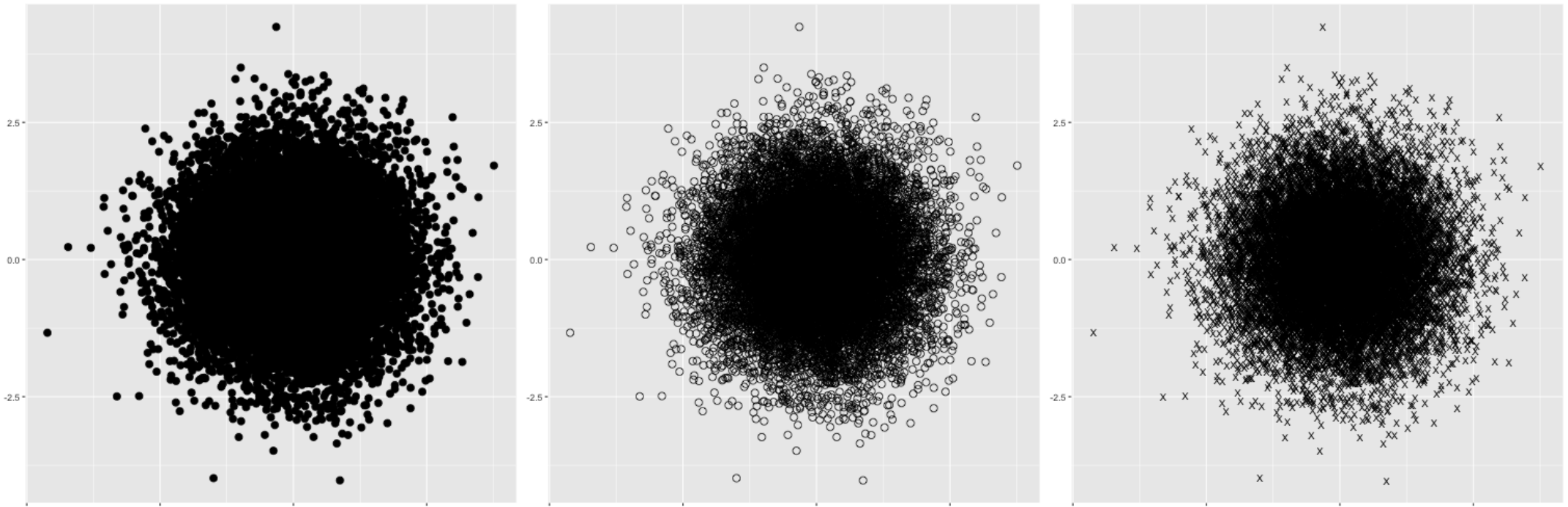
```
par(mar = c(3, 3, .1, .1))

norm + geom_point(size = 3)
norm + geom_point(size = 2)
norm + geom_point(size = 1)
```



How to deal with overplotting?

Changing the shape

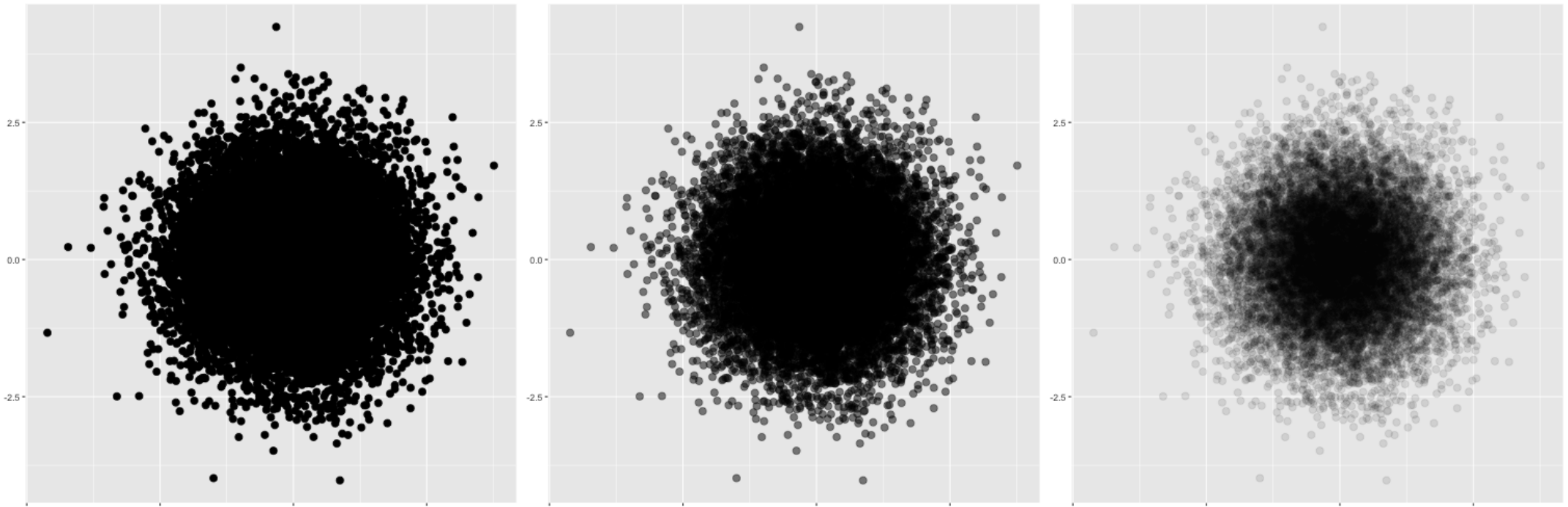
```
par(mar = c(3, 3, .1, .1))

norm + geom_point(size =3)
norm + geom_point(size = 3,
                  shape = 1)
norm + geom_point(size = 3,
                  shape = "X")
```



How to deal with overplotting?

Changing the transparency

```
par(mar = c(3, 3, .1, .1))

norm + geom_point(size = 3)
norm + geom_point(size = 3,
                  alpha = 1 / 2)
norm + geom_point(size = 3,
                  alpha = 0.1)
```

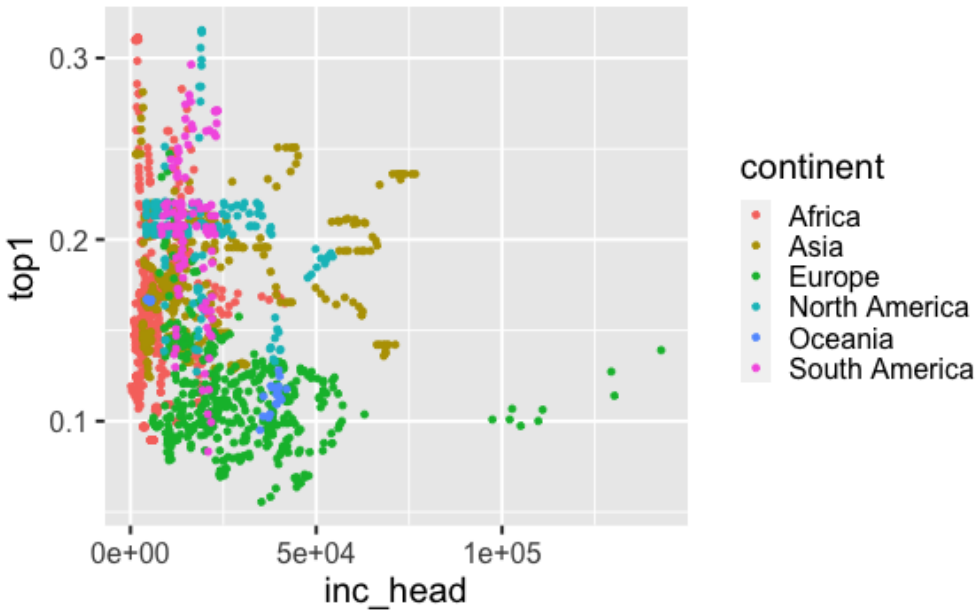


ADDING DIMENSIONS

So far, we've added at most two variables to our plots.

We can add more variables using aesthetics like the **color, shape, and size** of the geometries.

```
wid %>%
  ggplot(aes(inc_head, top1,
             color = continent)) +
  geom_point()
```

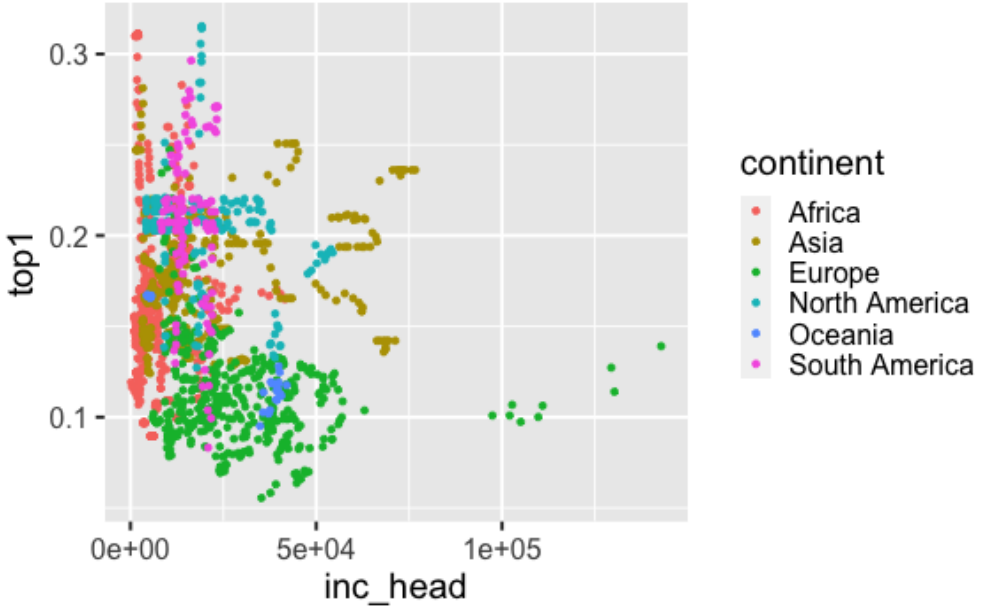


ADDING DIMENSIONS

So far, we've added at most two variables to our plots.

We can add more variables using aesthetics like the **color, shape, and size** of the geometries. We are **mapping** variable values into visual attributes.

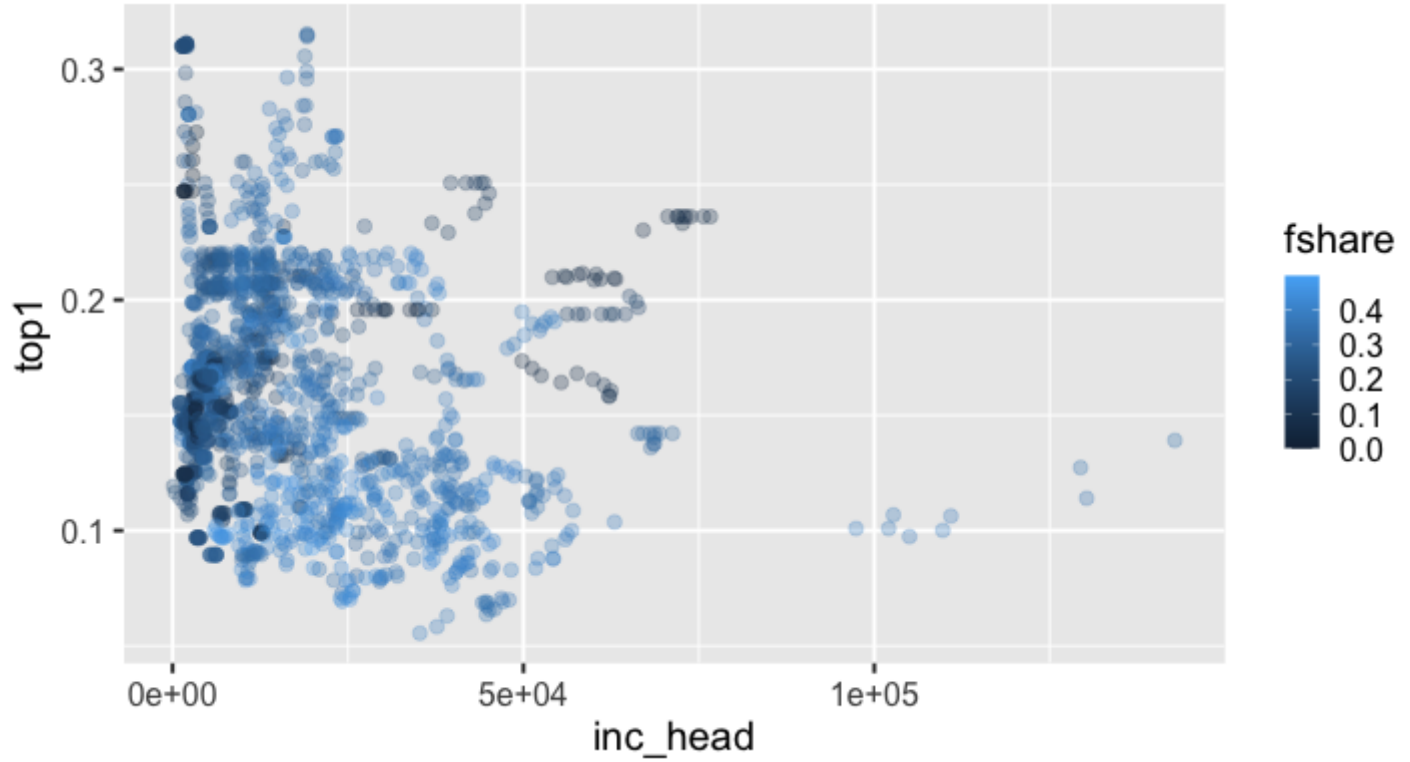
```
wid %>%
  ggplot(aes(inc_head, top1,
             color = continent)) +
  geom_point()
```



- ggplot2 takes care of the details converting data ("Europe", "Africa") to aesthetics (Green, Red) with an automatic **scale**
- When our mapping variable is continuous, a gradient is used instead
- The scale can be overridden and customized. See Chapter 6 of ggplot2 book for more details.
  - `scale_color_discrete()`
  - `scale_color_manual(values = c("red", "blue","#000000"))`

ADDING DIMENSIONS

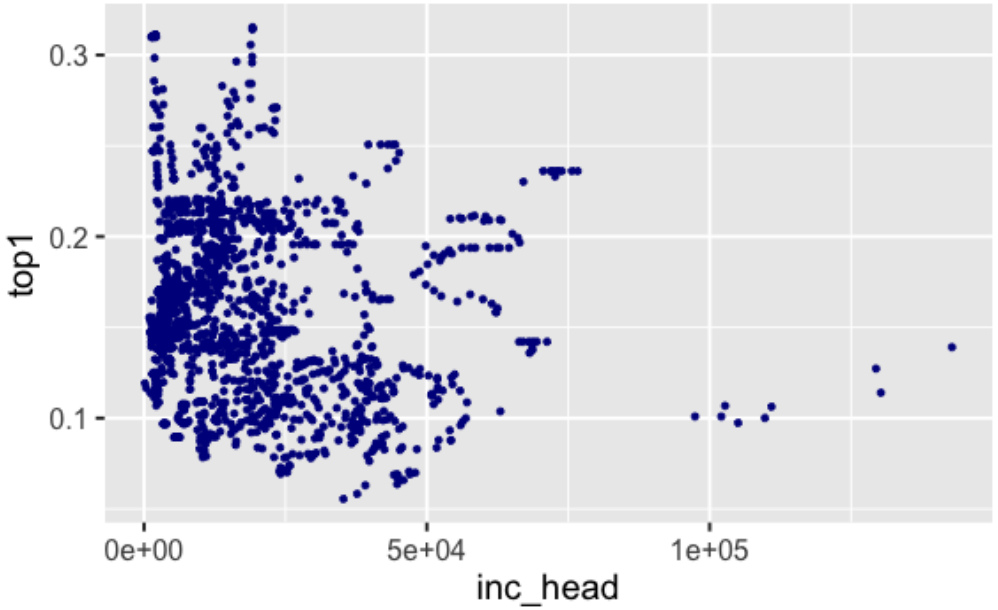
```
wid %>%
  ggplot(aes(inc_head, top1,
             color = fshare)) +
  geom_point(alpha = 0.3)
```



Setting vs. mapping visual attributes

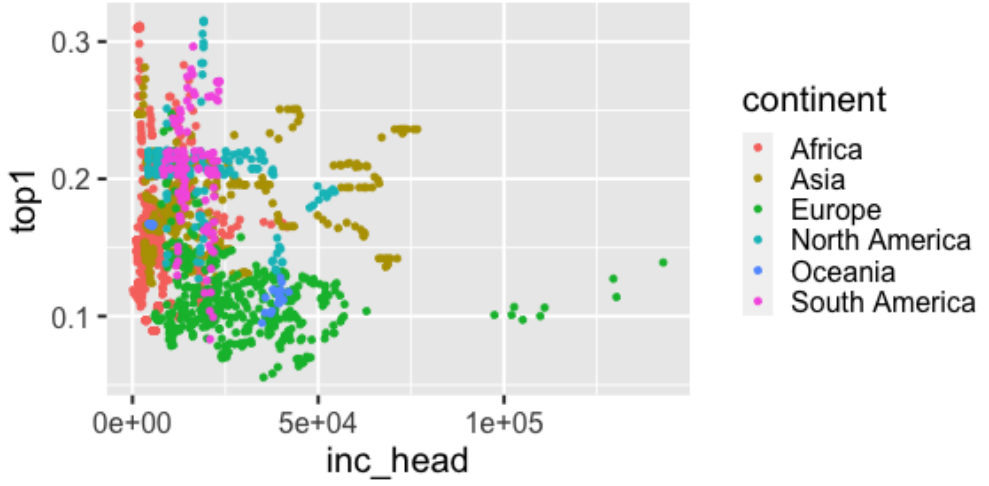
Setting: Defining a **fixed** value

```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point(color = "darkblue")
```



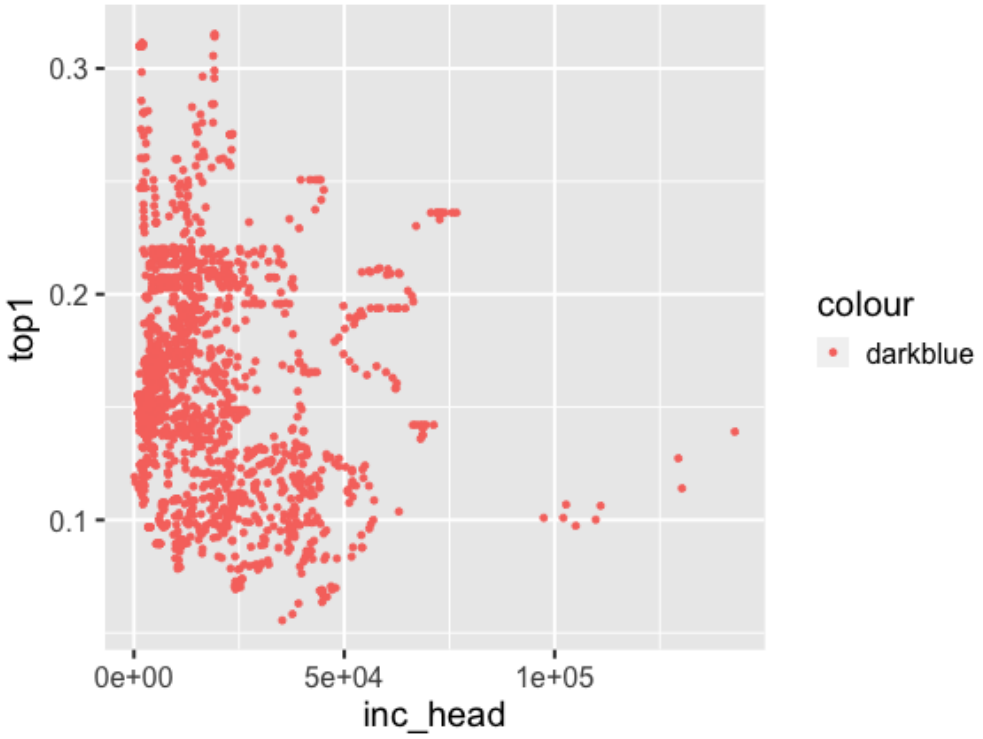
Mapping: Relating an aesthetic to a **variable**

```
wid %>%
  ggplot(aes(inc_head, top1,
             color = continent)) +
  geom_point()

# Equivalent to this:
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point(aes(color = continent))
```



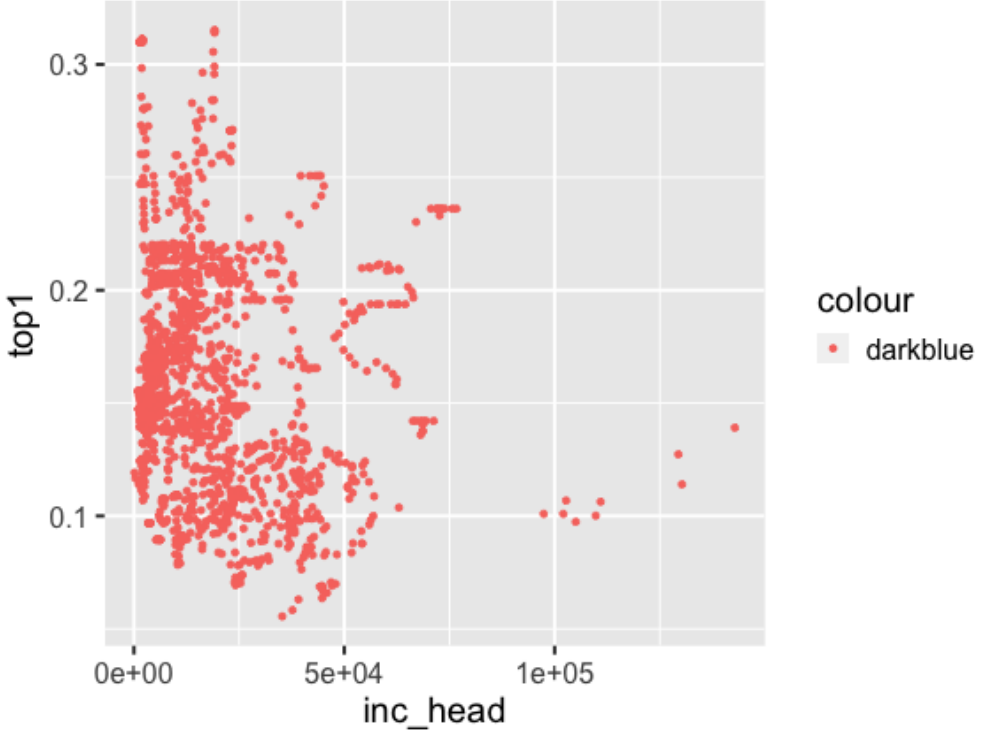
What about this?

```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point(aes(color = "darkblue"))
```



What about this?

```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point(aes(color = "darkblue"))
```



- This is mapping the value "dark blue" to a color.
- It temporarily creates a new variable containing only the value "darkblue" and then scales it with a color scale.

```
wid %>%
  mutate(colour = "darkblue") %>%
  ggplot(aes(inc_head, top1,
             color = colour)) +
  geom_point()
```

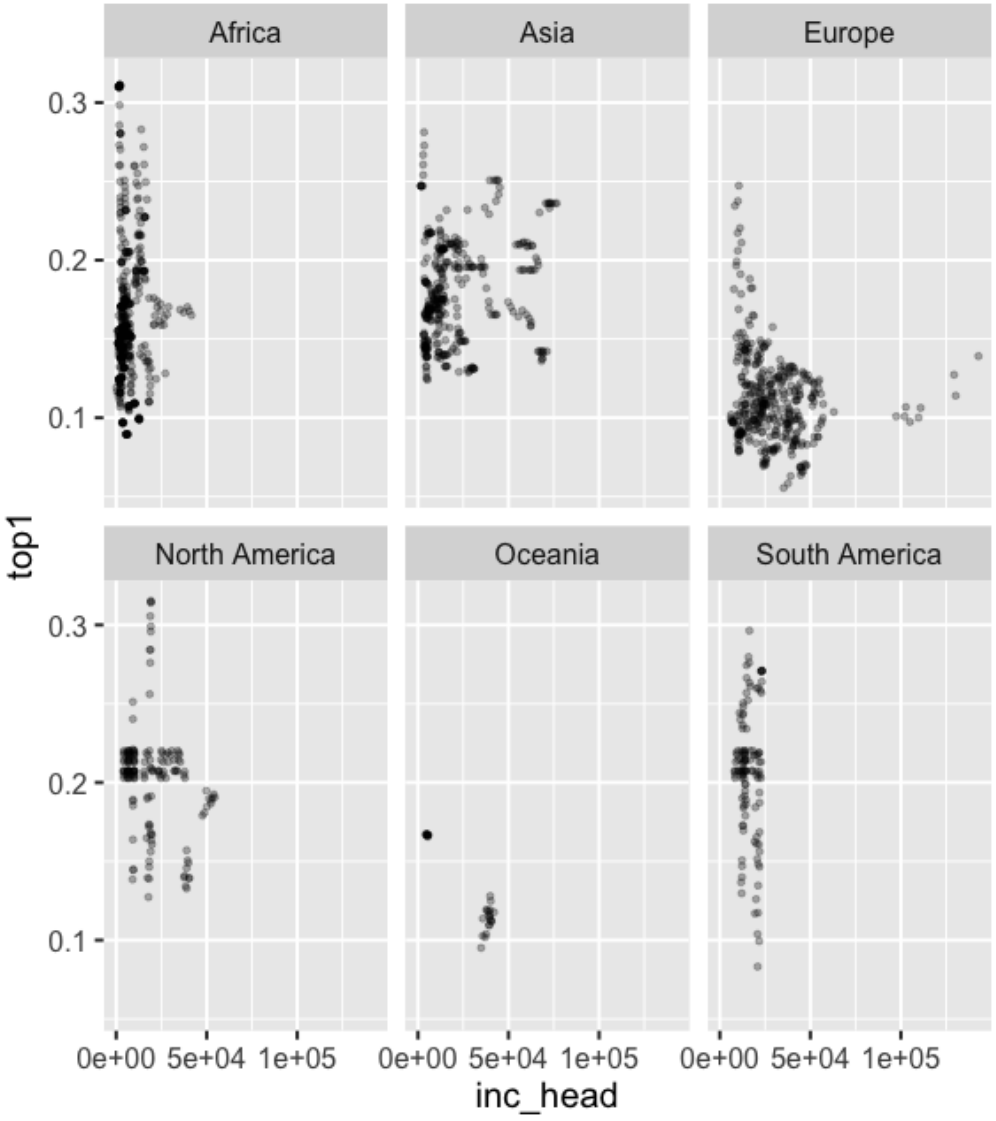
- We'll learn how it can be useful in a few slides

FACETTING

- Another technique for displaying additional categorical variables on a plot is facetting.

```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point(alpha = 0.3) +
  facet_wrap(~continent)
```

- It creates tables of plots by splitting the data into subsets and displaying different graphs for each subset

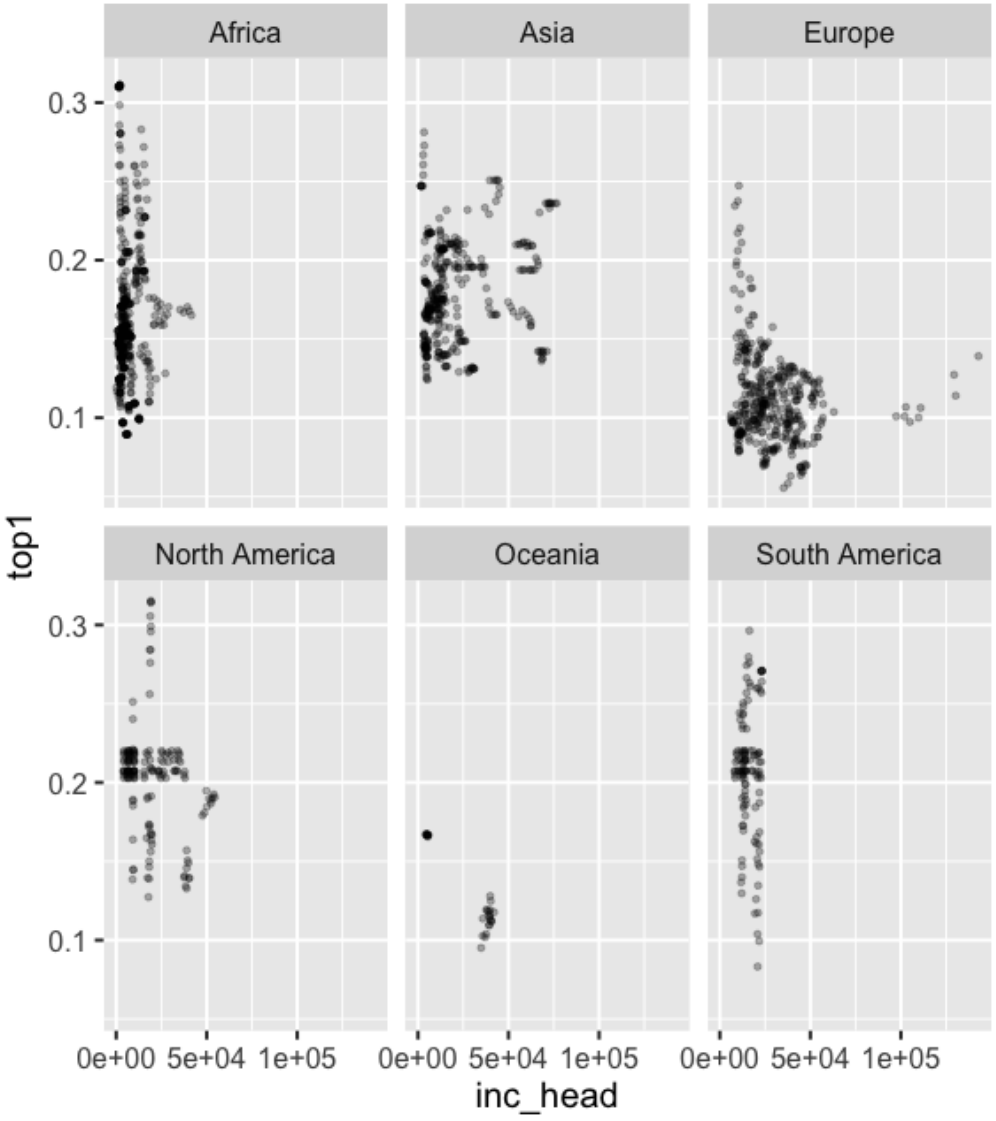


FACETTING

- Another technique for displaying additional categorical variables on a plot is facetting.

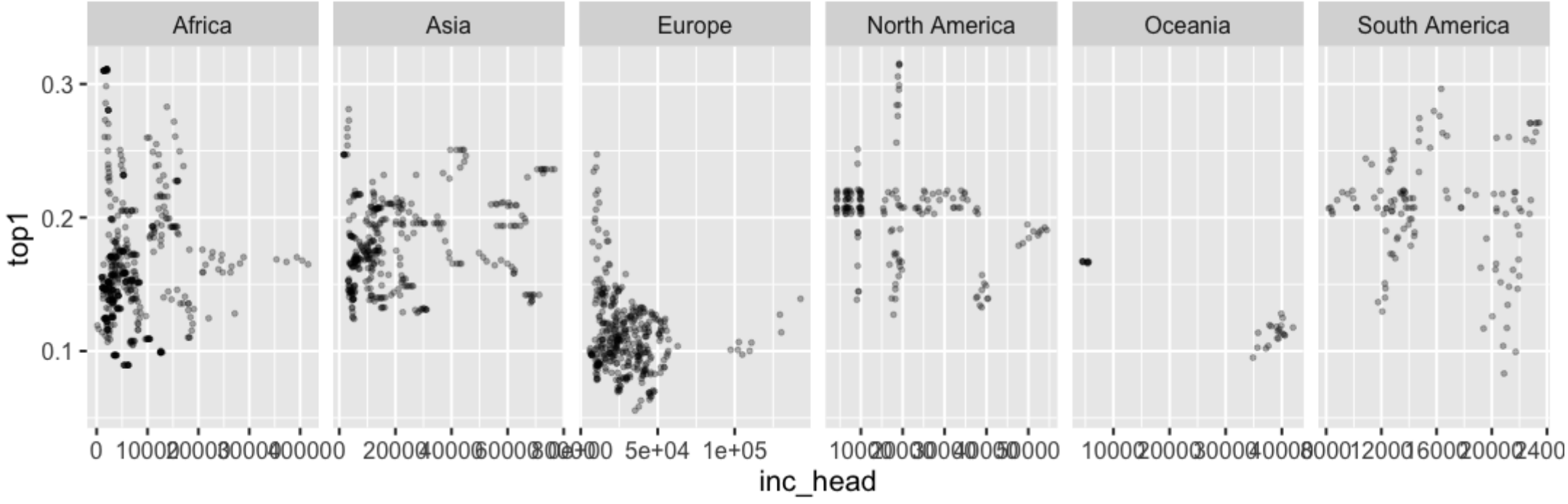
```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point(alpha = 0.3) +
  facet_wrap(~continent)
```

- It creates tables of plots by splitting the data into subsets and displaying different graphs for each subset
- Choose the facet arrangement using `nrow` and `ncol` to indicate the number of rows and columns
- Specify the type of scale you want:
  - **free**: adjusted separately to each facet
  - **fixed**: common to all



FACETTING

```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point(alpha = 0.3) +
  facet_wrap(~continent,
             ncol = 6,
             scales = "free_x")
```



SAVING

- `ggsave()` will save what is in the plot panel

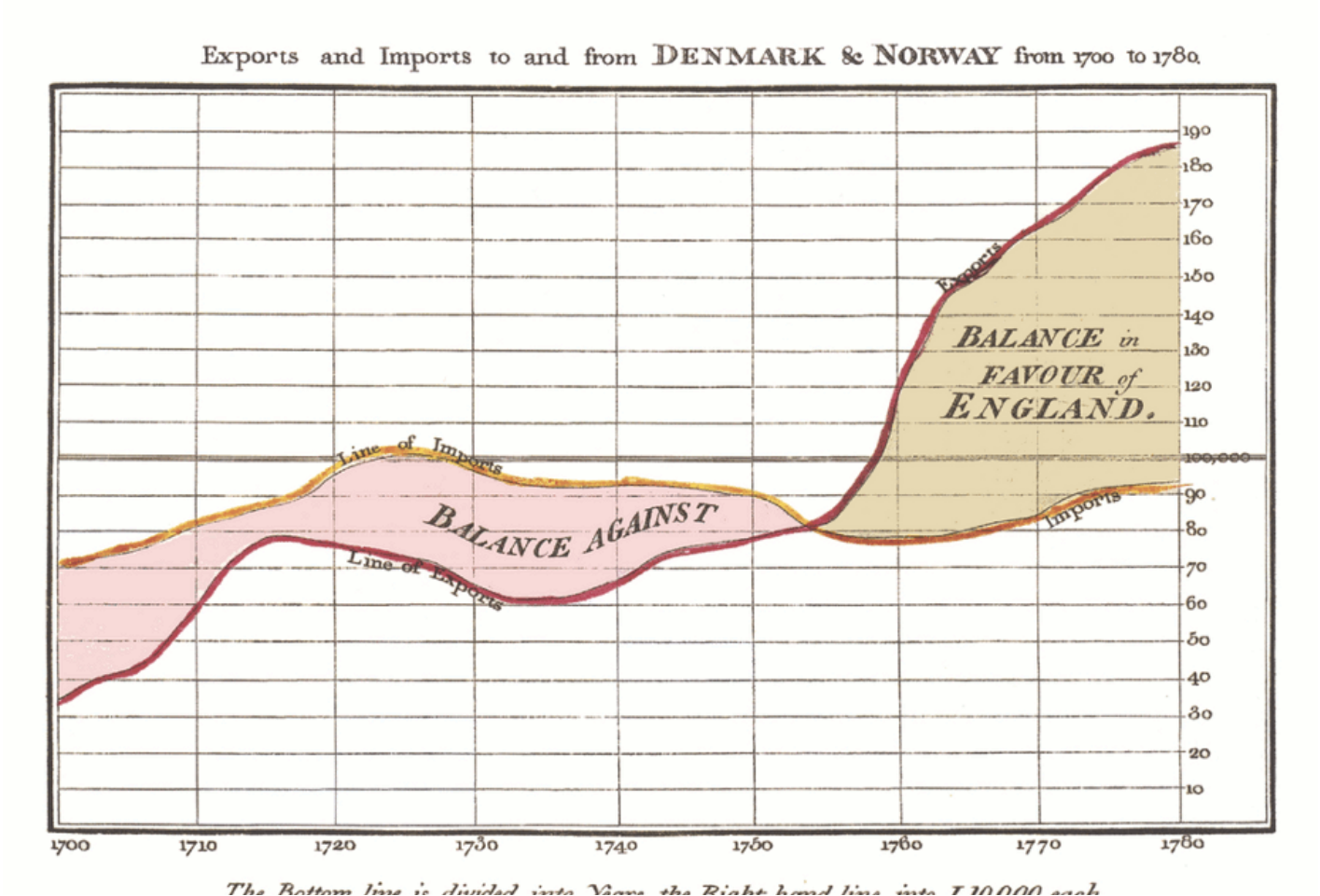
```
ggsave("../myplot.png")   # Saves what is in the Plots tab

ggsave("../myplot.png",    # Where
       plot = last_plot(), # What
       width = 16,         # Dimensions
       height = 9,
       unit = "cm")        # Units
```

AGENDA

- AN INTRO TO DATA VISUALIZATION
- THE ggplot() FUNCTION
    1. MAIN STRUCTURE
    2. AESTHETICS
    3. ADDING DIMENSIONS
    4. MAPPING vs. SETTING ATTRIBUTES
    5. SAVING
- BROWSING THE TOOLBOX
    1. ADDING LAYERS
    2. LABELS
    3. ANNOTATIONS
    4. GROUPING DATA
    5. STATISTICAL SUMMARIES
- WHICH GRAPH SHOULD I USE?
- GRAPHICAL EXCELLENCE AND INTEGRITY

BROWSING THE TOOLBOX

There is a lot. Really **A LOT** you can do with ggplot.
Let's illustrate these tools creating a `ggplot()` modern version of one of Playfair's trade balance graphs:
- Download and import `02_playfair-balance.csv`

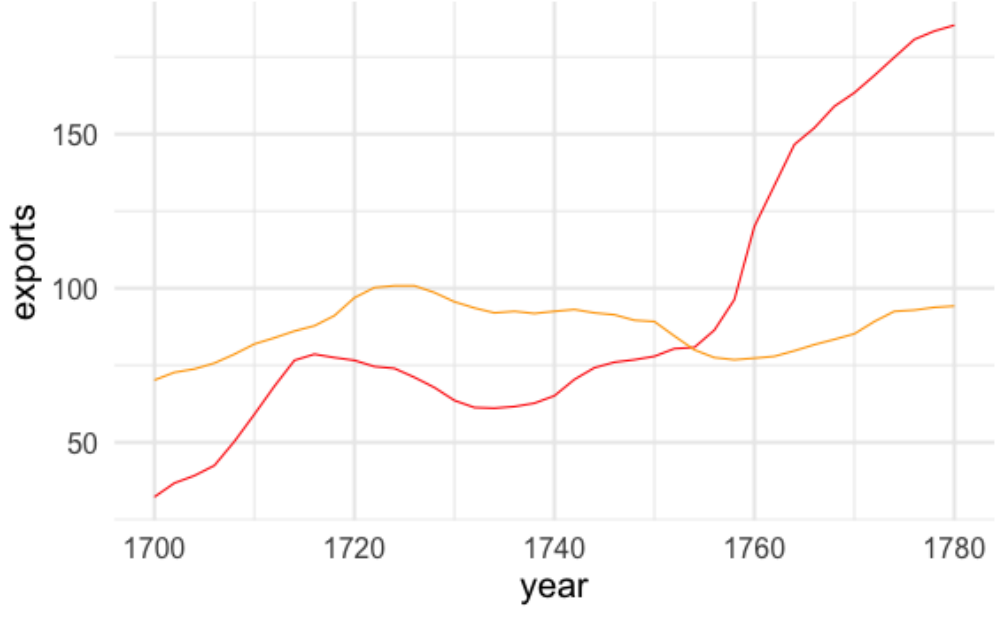


ADDING MORE LAYERS

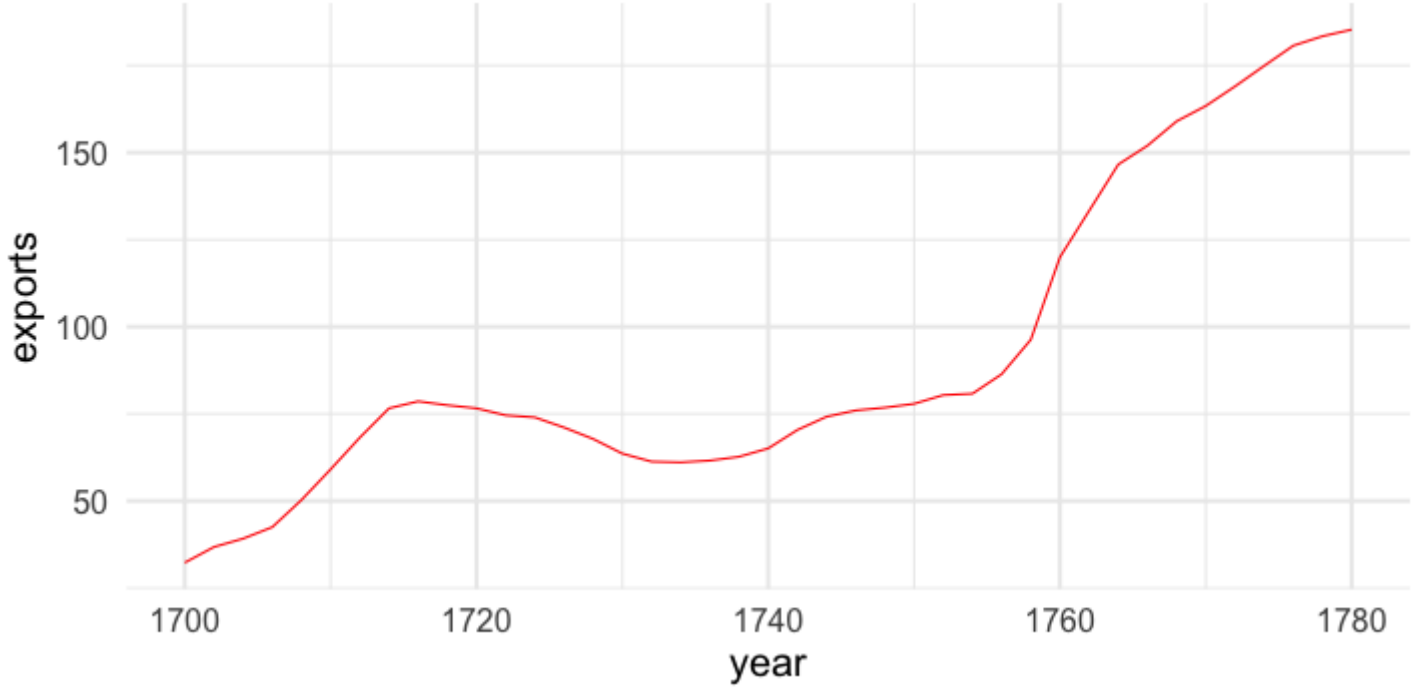
We can keep on adding geometries and geometries to build our dataset.

- Every layer we add must have some data associated with it
- The data on each layer doesn't need to be the same. We can specify the mappings/aesthetics of each geom (data, x, y) separately.
  - When a new geom is added it inherits the aesthetics inside `ggplot(aes())` unless specified otherwise
  - It doesn't hurt to be explicit about which data you are using when you are handling multiple layers

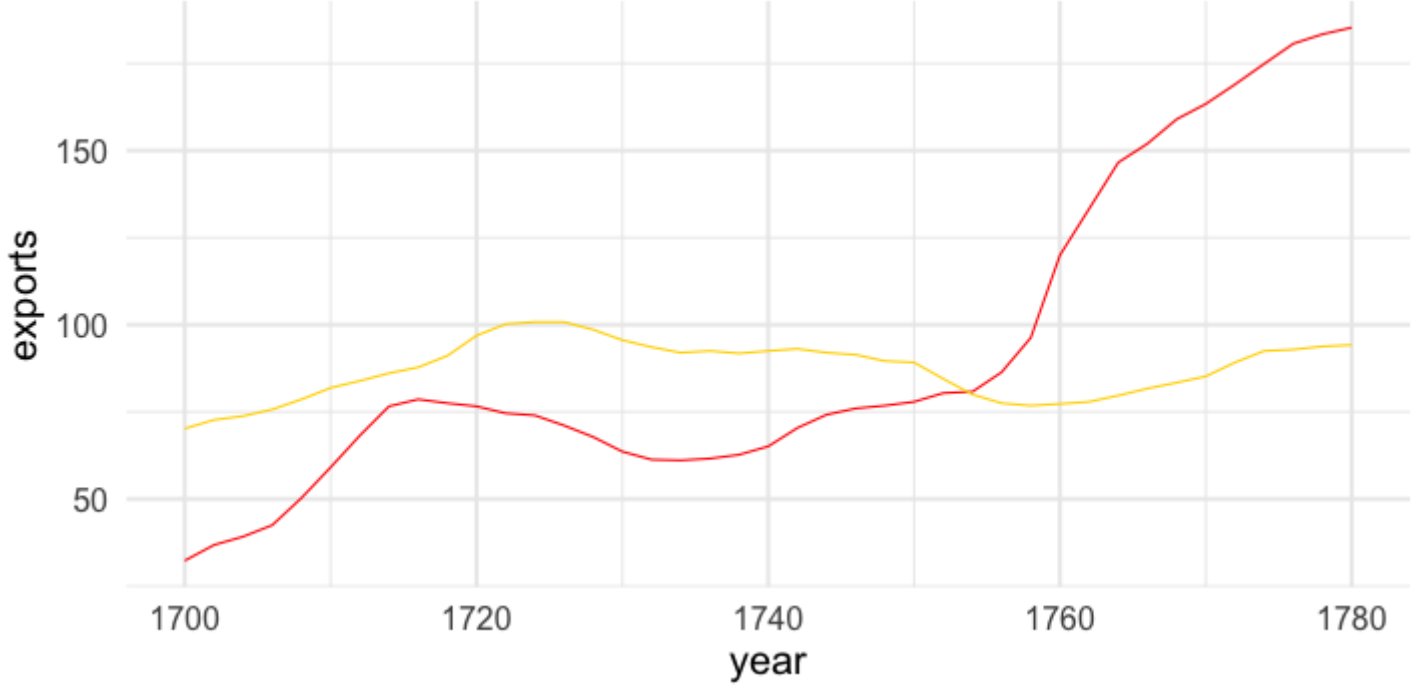
```
balance <- read.csv("../data/raw/02_playfair-balance.csv")

balance %>%
  ggplot(aes(x = year, y = exports)) +
  geom_line(aes(y = exports),      # First layer
            color = "red") +
  geom_line(aes(y = imports),      # Second layer
            color = "orange") +
  theme_minimal(base_size = 20)
```



```r
balance %>%
  ggplot(aes(x = year, y = exports)) +
  geom_line(aes(y = exports),      # First layer
            color = "red") +
  theme_minimal(base_size = 20)
```

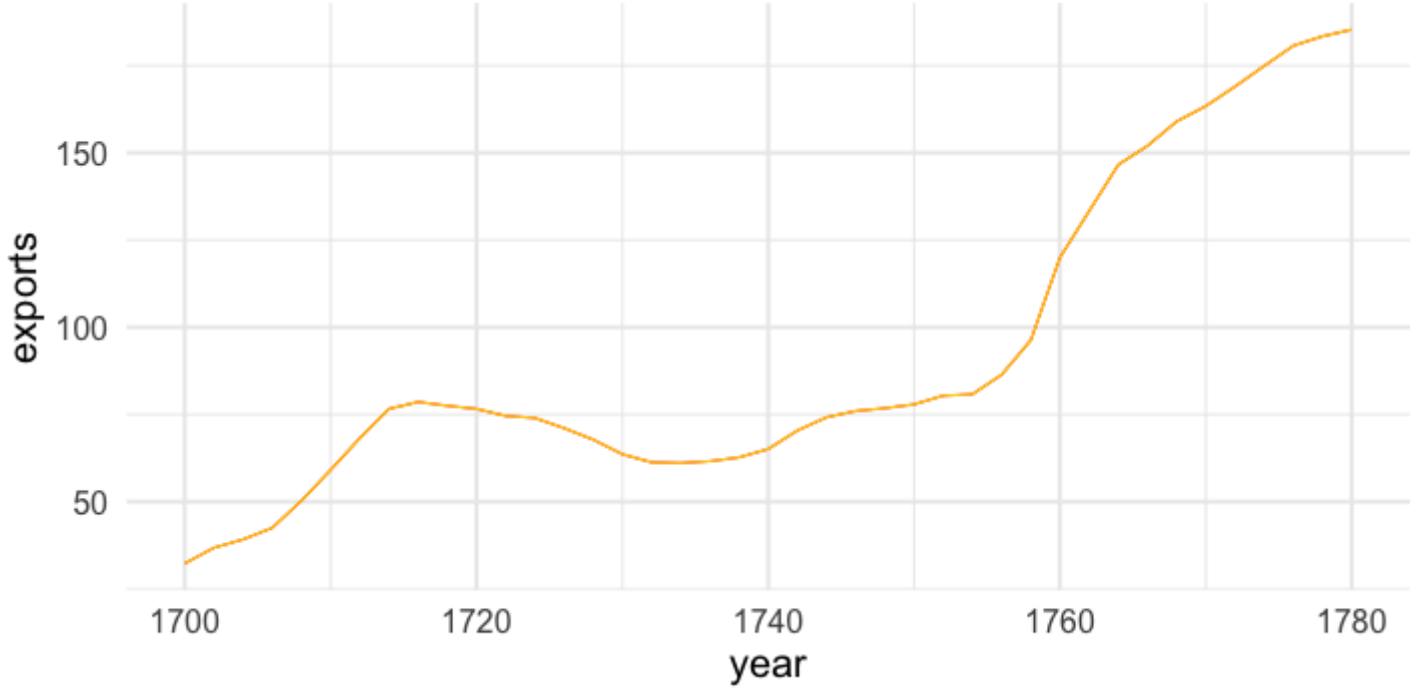


```
balance %>%
  ggplot(aes(x = year, y = exports)) +
  geom_line(aes(y = exports),      # First layer
            color = "red") +
  geom_line(aes(y = imports),      # Second layer
            color = "gold") +
  theme_minimal(base_size = 20)
```



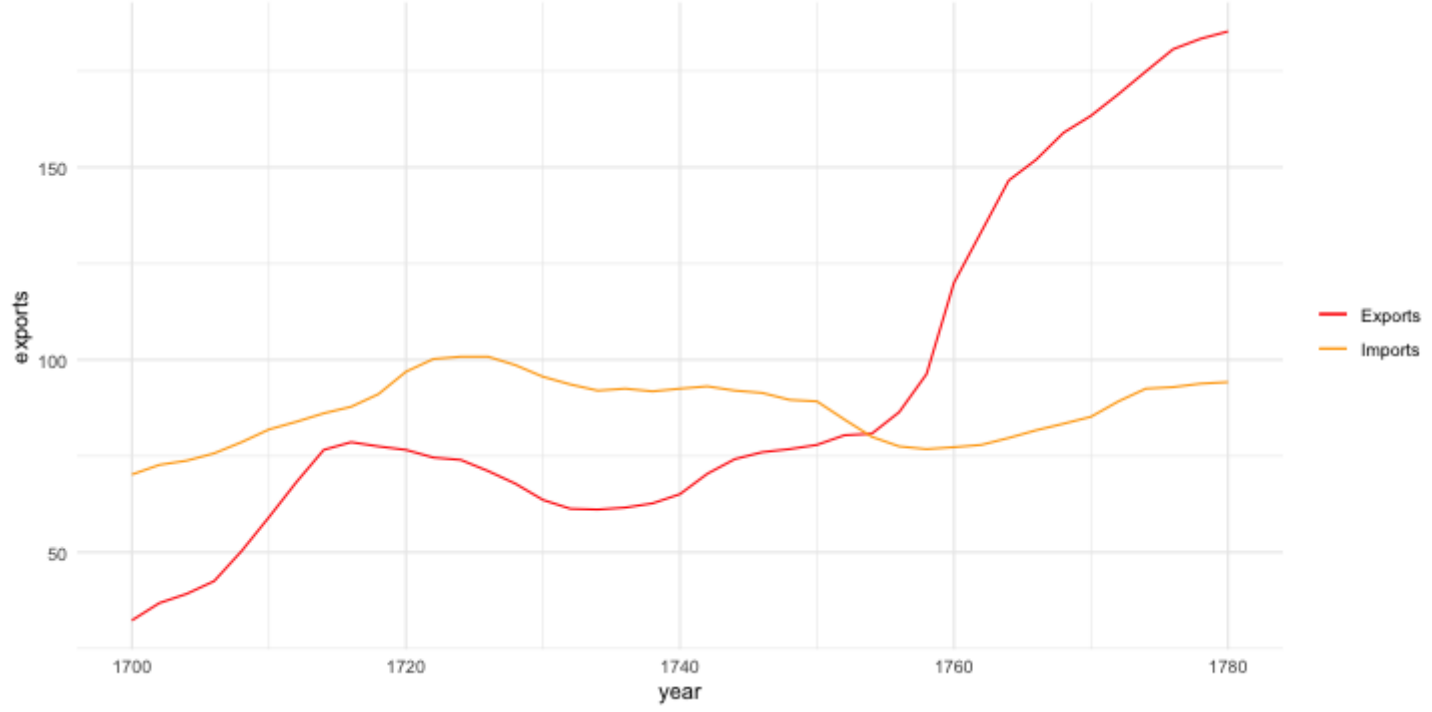
What happens if we don't specify new mappings for each geom?

```
balance %>%
  ggplot(aes(x = year, y = exports)) +
  geom_line(color = "red") +
  geom_line(color = "gold") +
  theme_minimal(base_size = 20)
```



LABELING EACH LAYER

```
balance %>%
  ggplot(aes(x = year, y = exports)) +
  geom_line(aes(y = exports,
                color = "Exports")) +
  geom_line(aes(y = imports,
                color = "Imports")) +
  theme_minimal() +
scale_color_manual(values = c("red", "orange"),
                   name = NULL)
```



Let's save this plot to add more things to it later:

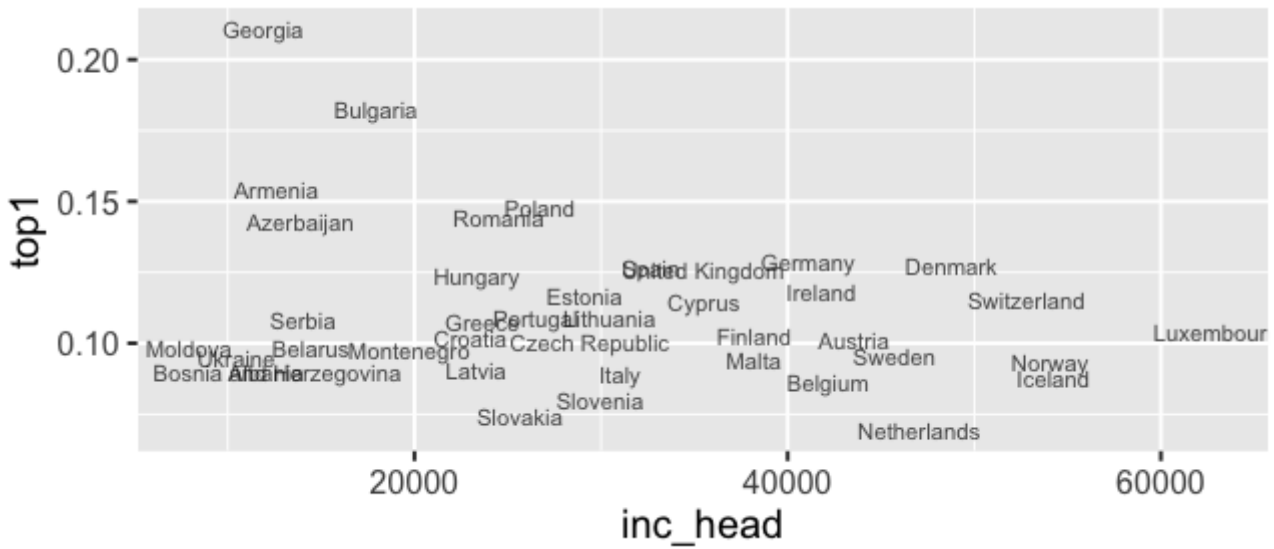
```
balance_plot <-
  balance %>%
    ggplot(aes(x = year, y = exports)) +
    geom_line(aes(y = exports,
                  color = "Exports")) +
    geom_line(aes(y = imports,
                  color = "Imports")) +
    theme_minimal() +
  scale_color_manual(values = c("red", "orange"),
                     name = NULL)
```

LABELS

- Adding text to a plot. Your obvious best friend: `geom_text()`. Same as `geom_point()` but with text labels instead of points. Duh 🙄

For example, with our WID data:

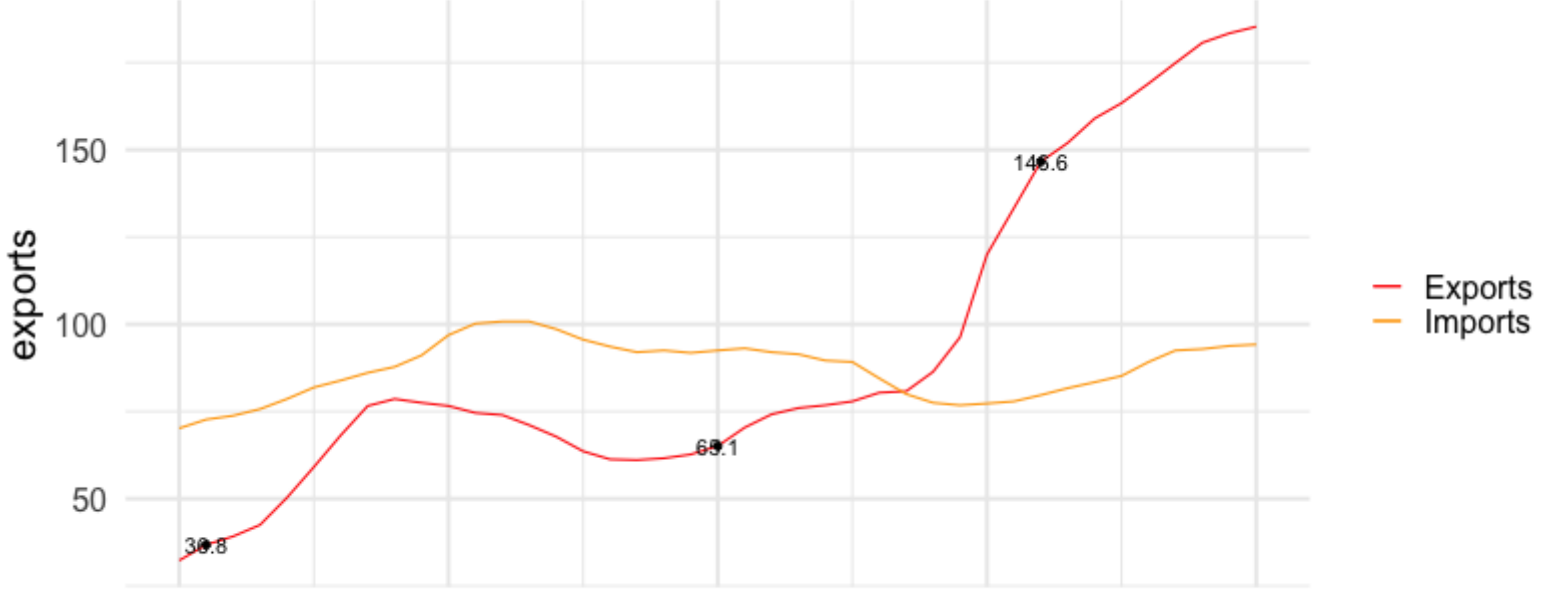
```
wid %>%
  filter(year == 2019 & continent == "Europe") %>%
  ggplot(aes(x = inc_head, y = top1)) +
  geom_text(aes(label = country),
            alpha = 0.7)
```



LABELS

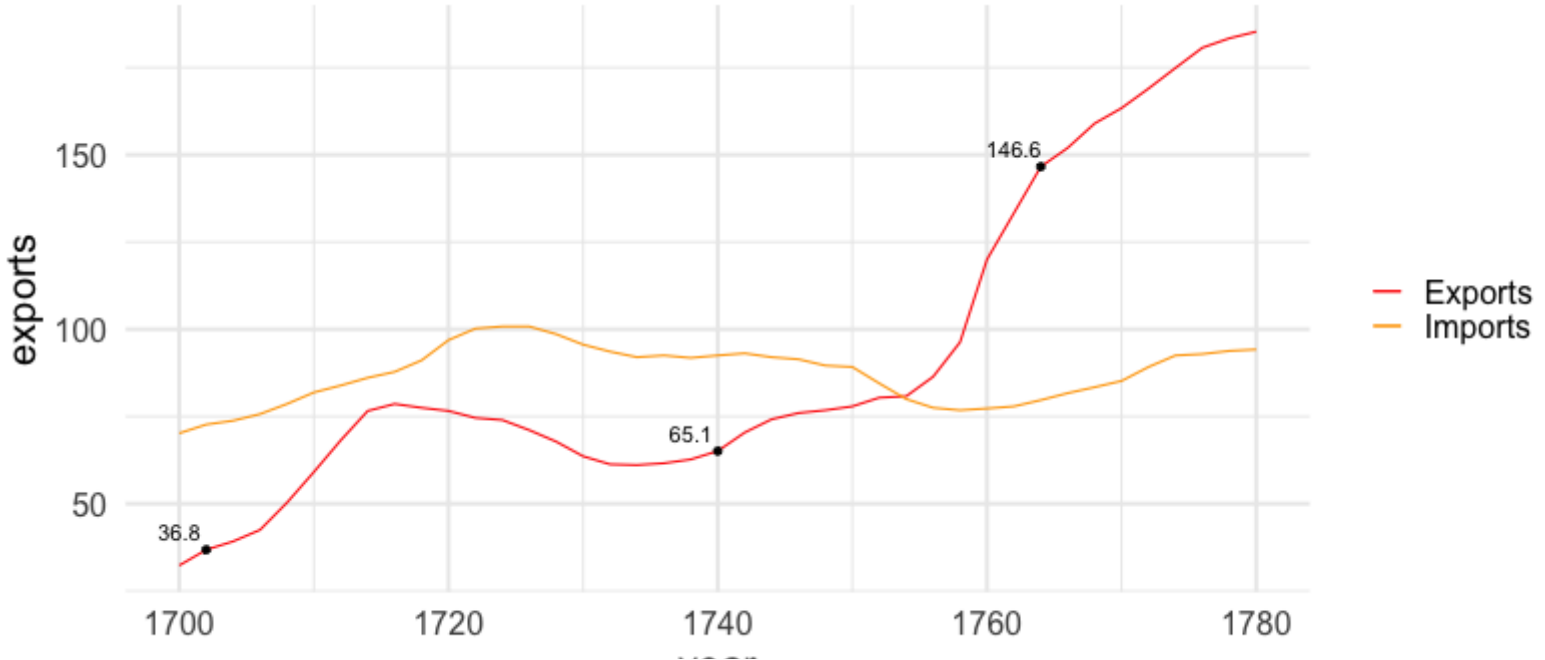
Now, using it with our trade balance plot to highlight three points:

```
bal_labels <- # Keep only the data for three years
  balance %>%
  filter(year %in% c(1702, 1740, 1764))

balance_plot +
  geom_point(data = bal_labels) +
  geom_text(aes(x = year, y = exports,
                label = exports),
            data = bal_labels)
```



Fixing things:

```
bal_labels <-
  balance %>%
  filter(year %in% c(1702, 1740, 1764)) # Keep only these

balance_plot +
  geom_point(data = bal_labels) +
  geom_text(aes(x = year, y = exports,
                label = exports),
            data = bal_labels,
            nudge_y = 4,
            nudge_x = -2)
```

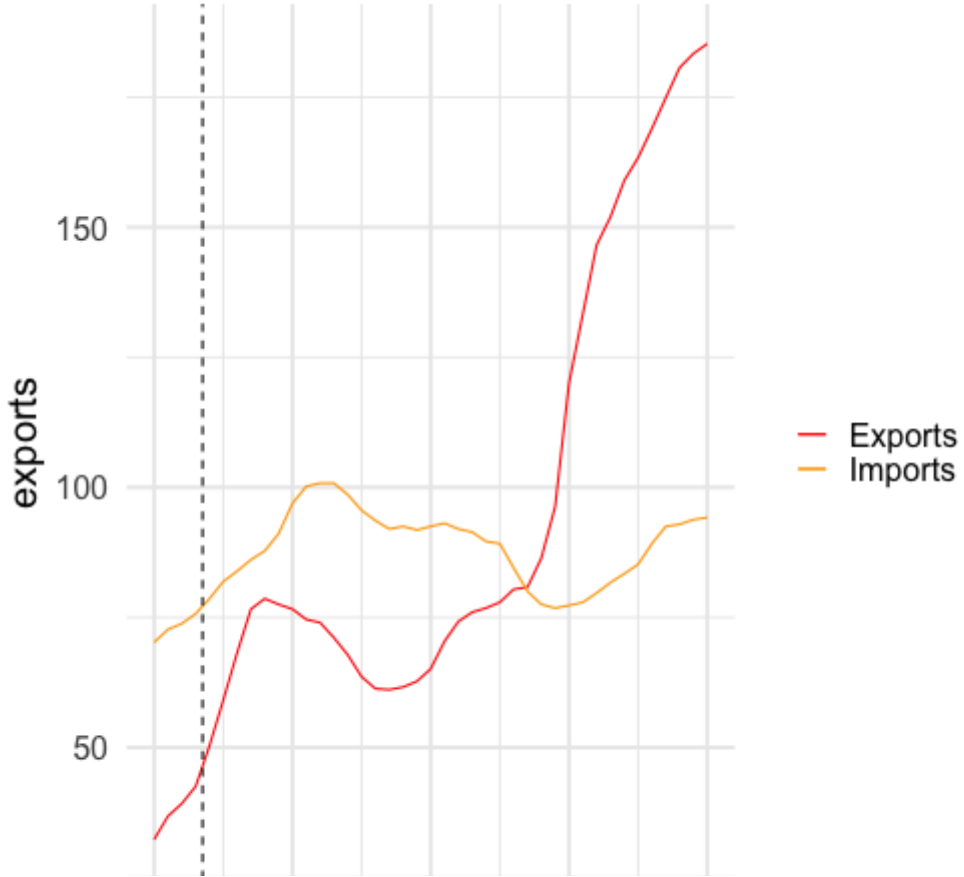


ANNOTATIONS

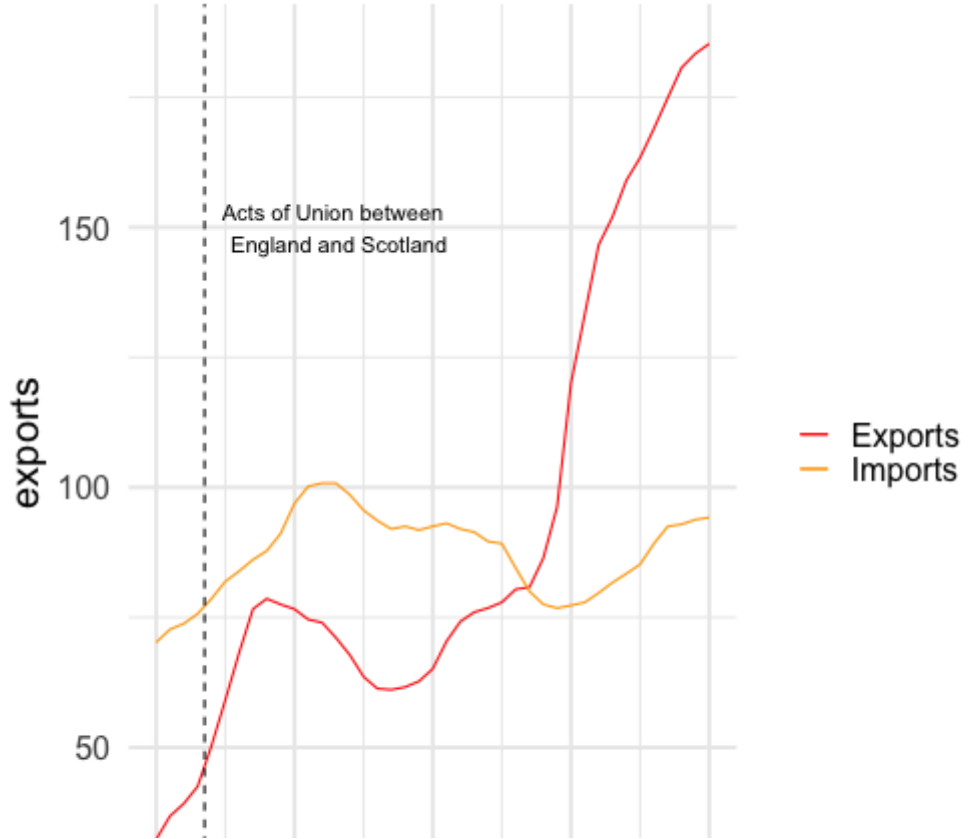
- Annotations add metadata to your plot to **highlight** certain features. You can use:
  - `geom_text()` to add text descriptions or label points (ex. outliers).
  - `geom_rect()` to highlight rectangular regions of the plot
  - `geom_line()`, `geom_path()`, `geom_segment()` to add lines
  - `geom_vline()`, `geom_hline()` to add rulers (lines that span the whole graph)

ANNOTATIONS

```
balance_plot +
  geom_vline(xintercept = 1707,
             linetype = "dashed")
```

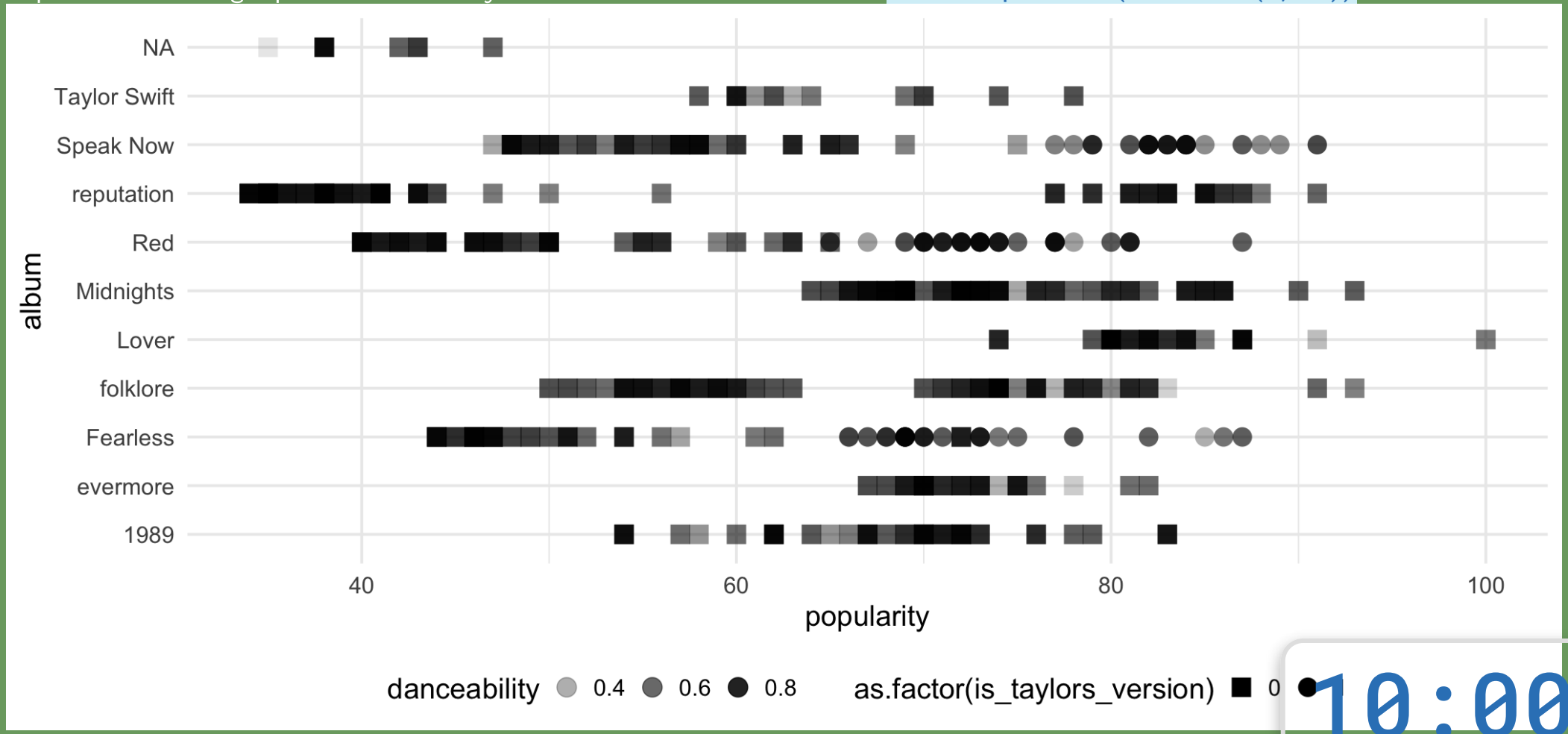



```
balance_plot +
  geom_vline(xintercept = 1707,
             linetype = "dashed") +
  annotate("text", x = 1716, y = 150,
           label = "Acts of Union between \n England and
                    Scotland")
```



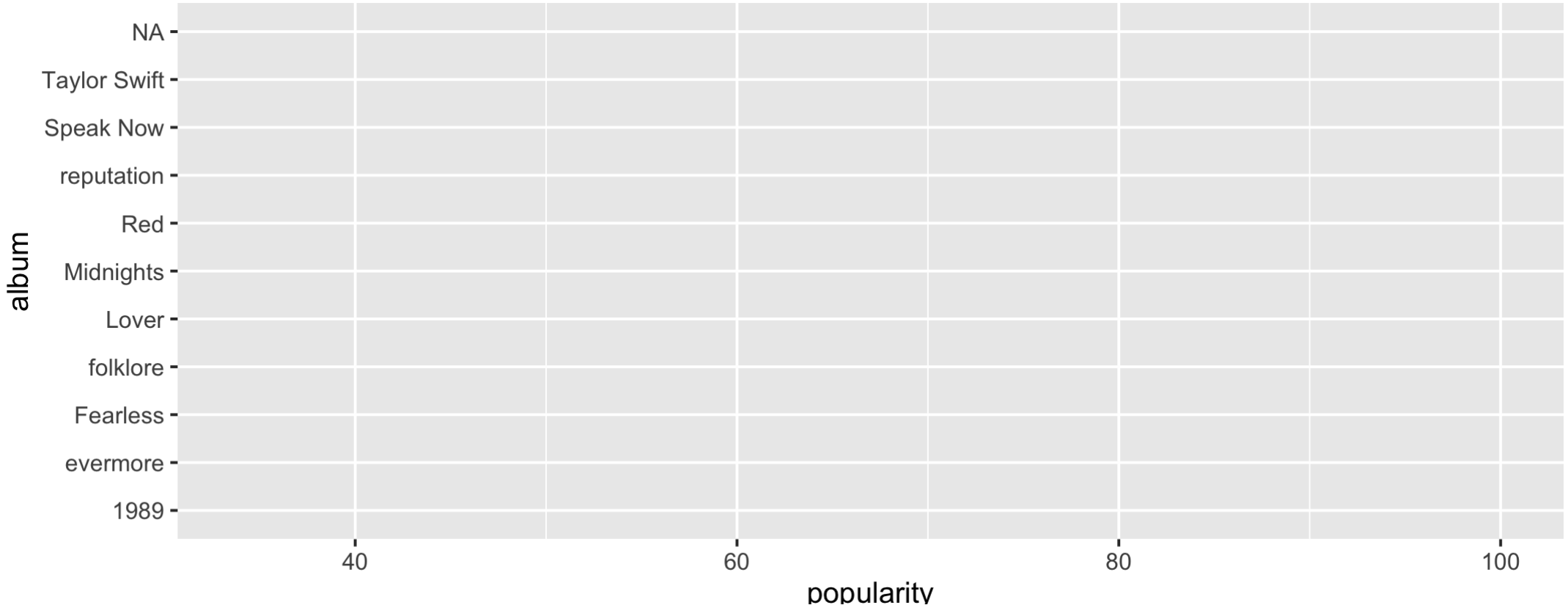
EXERCISE

- Reproduce this graph with the Taylor Swift data. Hint: add `scale_shape_manual(values = c(1, 16))` at the end.

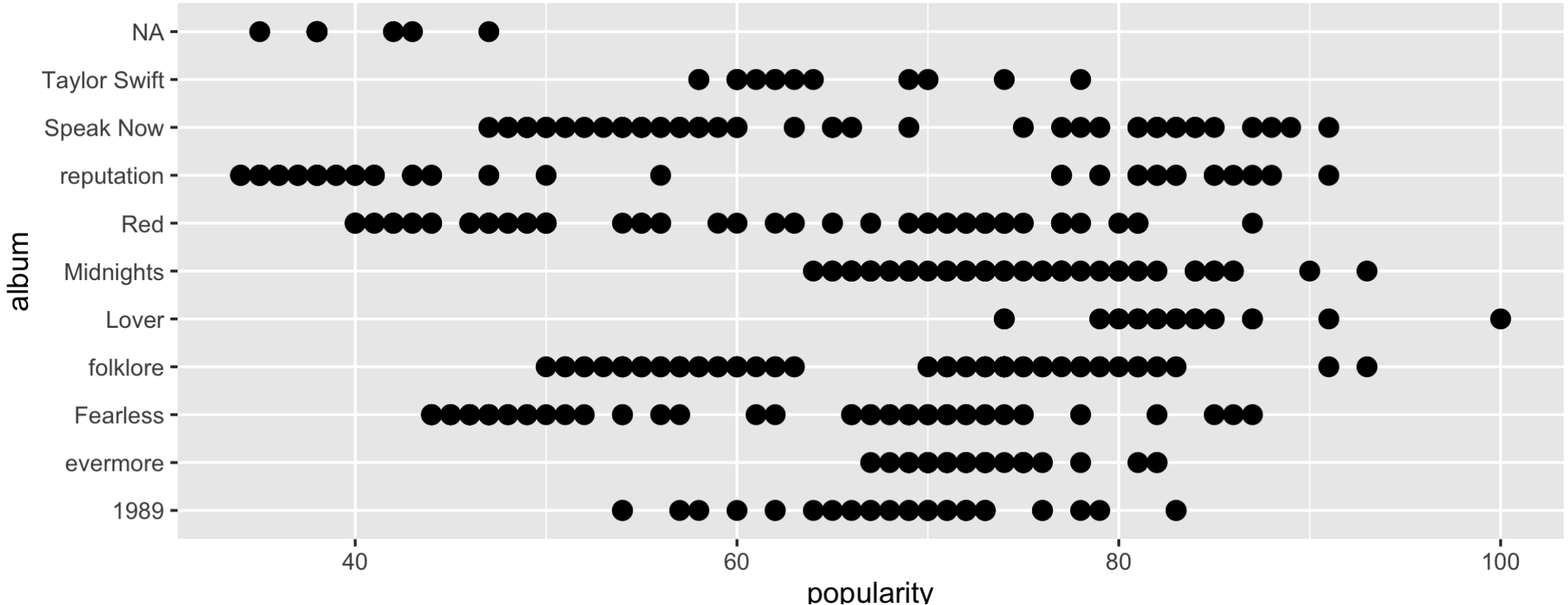


10:00

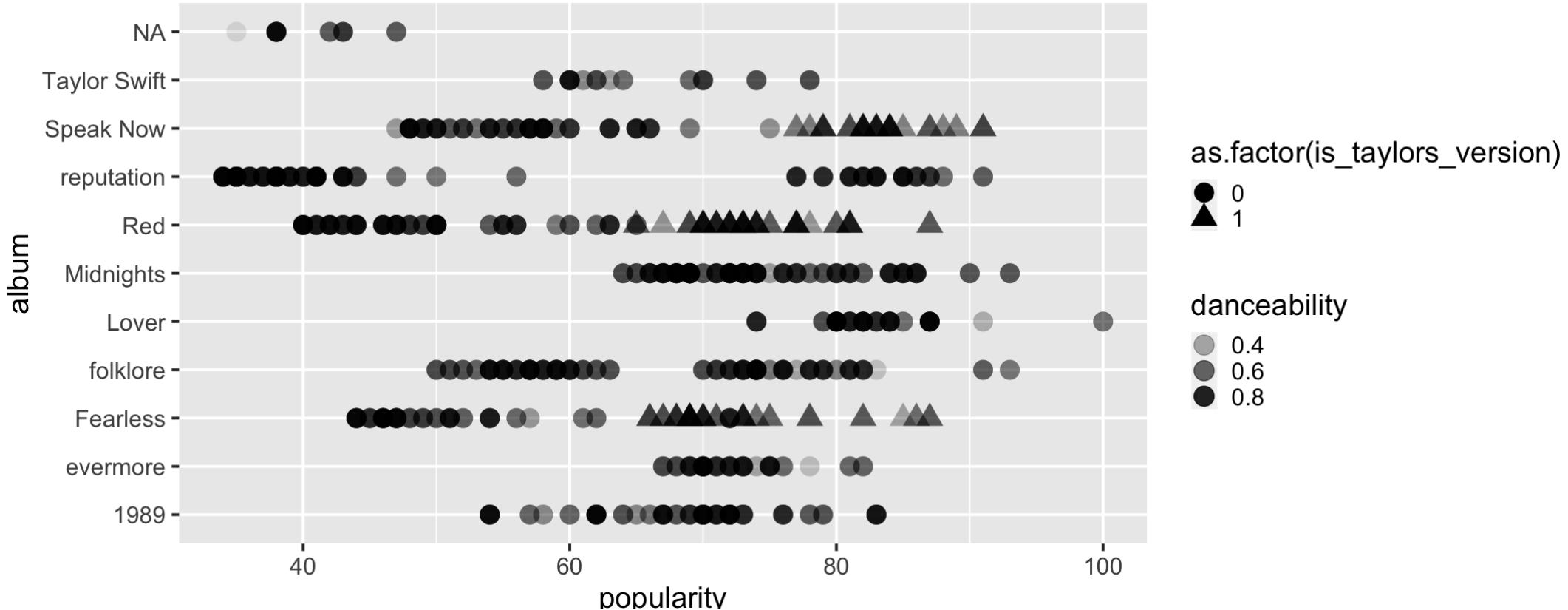
```
ts %>%
  ggplot(aes(x = popularity, y = album))
                                                                    #
                                                                    #
                                                                    #
                                                                    #
                                                                    #
                                                                    #
                                                                    #
```



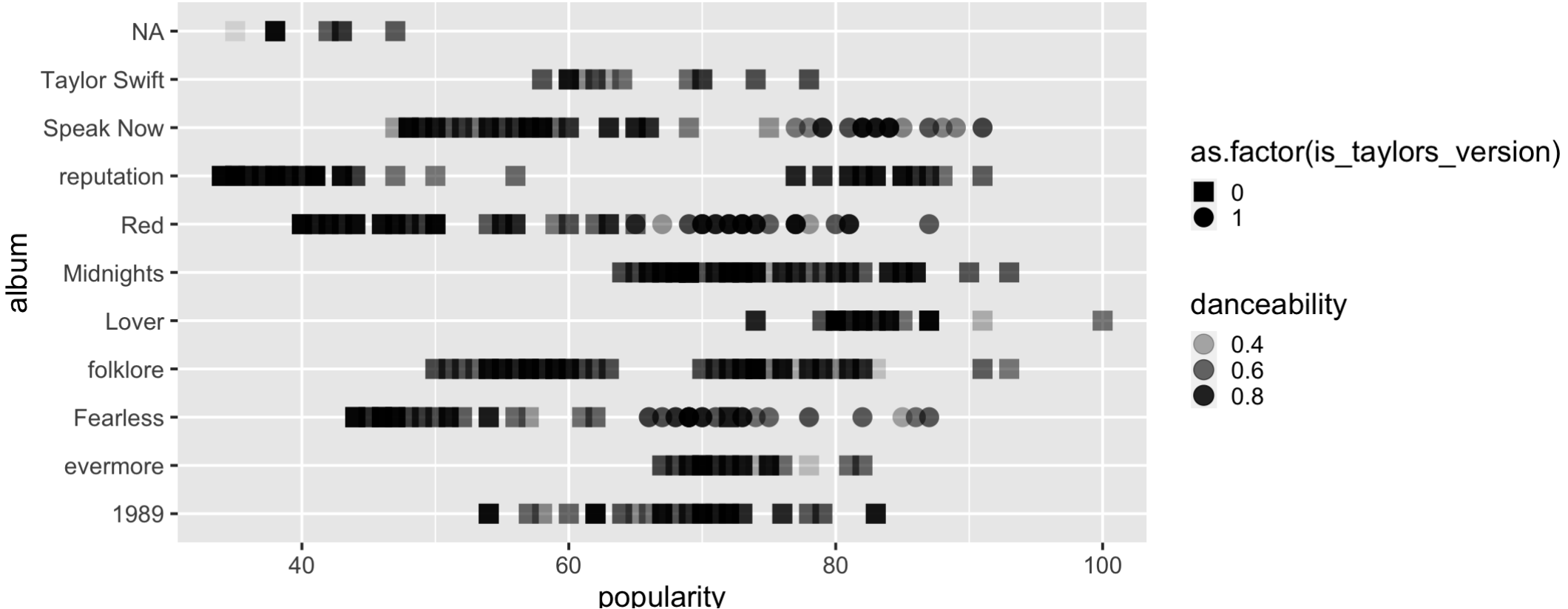
```
ts %>%
  ggplot(aes(x = popularity, y = album)) +
  geom_point(size = 6)
```



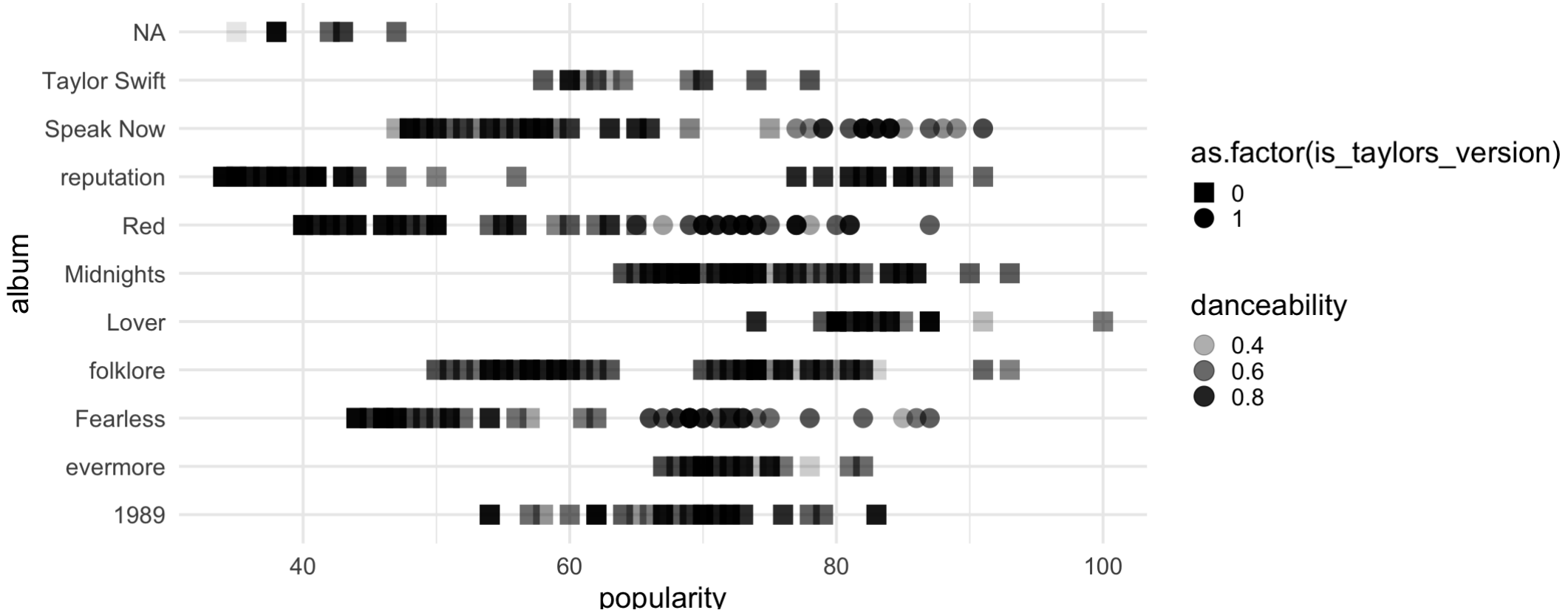
```
ts %>%
  ggplot(aes(x = popularity, y = album,
             shape = as.factor(is_taylors_version),
             alpha = danceability)) +
  geom_point(size = 6)
                                                                    #
                                                                    #
                                                                    #
                                                                    #
```



```
ts %>%
  ggplot(aes(x = popularity, y = album,
             shape = as.factor(is_taylors_version),
             alpha = danceability)) +
  geom_point(size = 6) +
  scale_shape_manual(values = c(15, 16))
                                                                    #
                                                                    #
                                                                    #
```

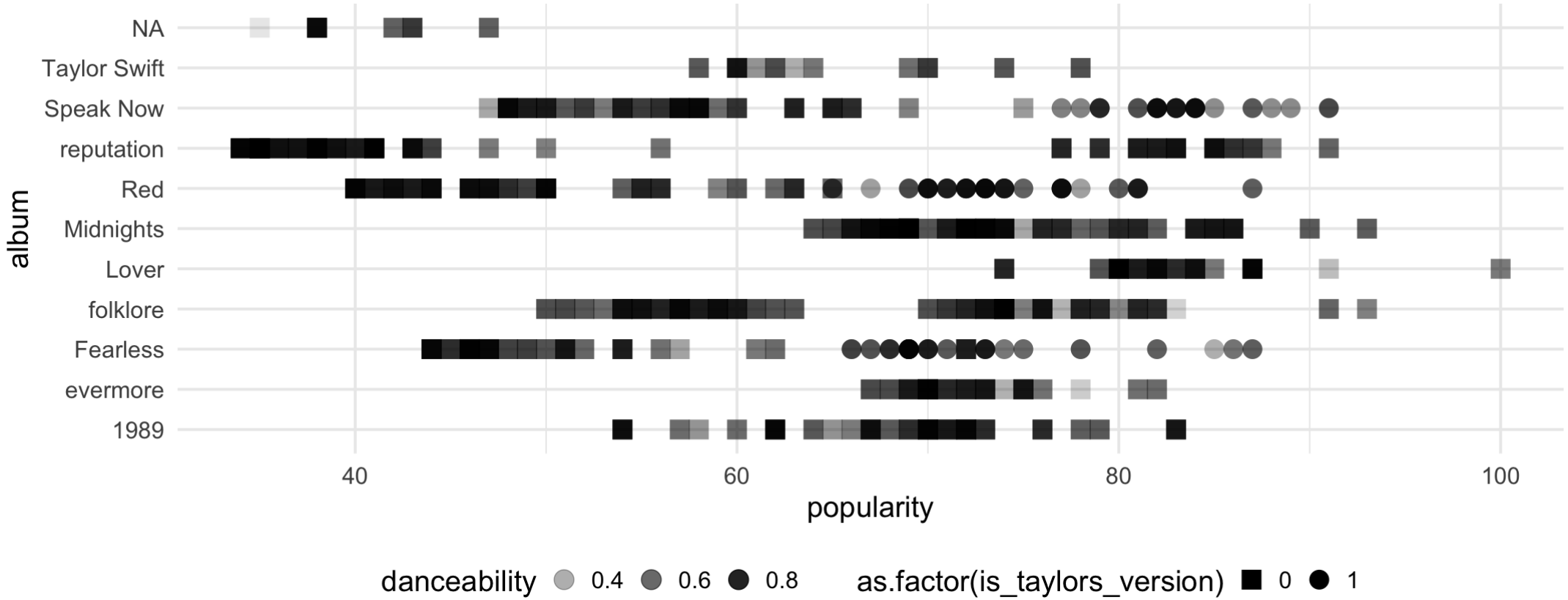


```
ts %>%
  ggplot(aes(x = popularity, y = album,
             shape = as.factor(is_taylors_version),
             alpha = danceability)) +
  geom_point(size = 6) +
  scale_shape_manual(values = c(15, 16)) +
  theme_minimal(base_size = 20)
                                                                  #
                                                                  #
```



```
ts %>%
  ggplot(aes(x = popularity, y = album,
             shape = as.factor(is_taylors_version),
             alpha = danceability)) +
  geom_point(size = 6) +
  scale_shape_manual(values = c(15, 16)) +
  theme_minimal(base_size = 20) +
  theme(legend.position = "bottom")
```

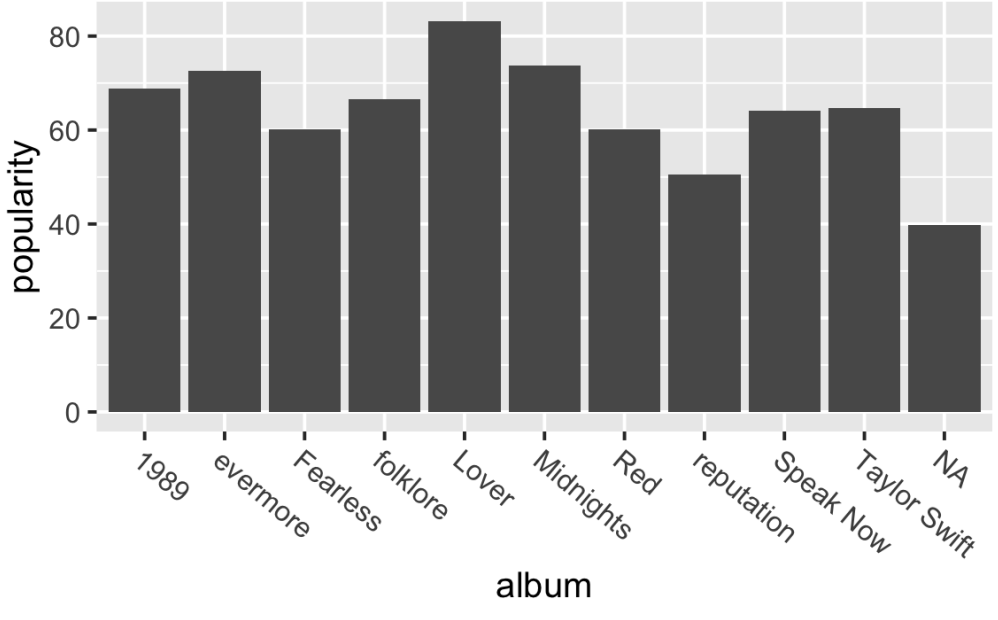
#



Statistical summaries

So far, we've only seen **individual geoms**, where there is a graphical object drawn for each observation (e.g. `geom_point()` draws one point per row)

A **collective geom** displays multiple observations with one geometric object. They are often used to display grouped summary statistics (e.g. the average popularity by album) or variable distributions through boxplots or histograms/densities.

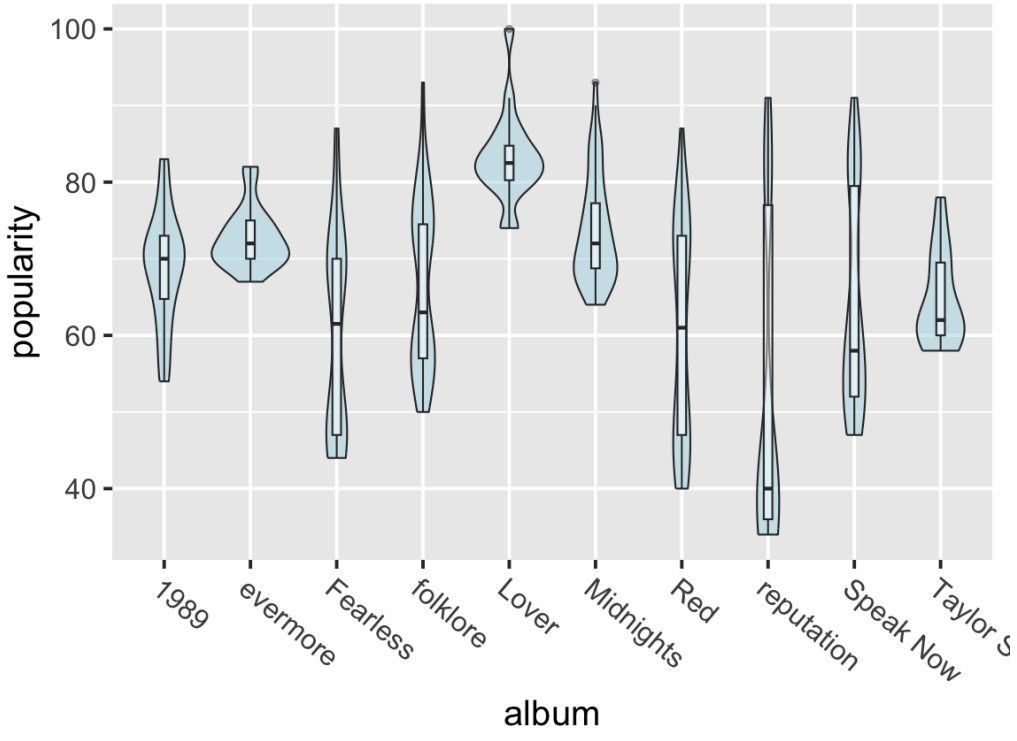



```
ts %>%
  # Create summary statistic
  group_by(album) %>%
  summarise(popularity =
              mean(popularity)) %>%
  # Plot
  ggplot(aes(x = album,
             y = popularity)) +
  geom_bar(stat = "identity") +
  # Rotate axis labels
  theme(axis.text.x =
          element_text(angle = -40,
                       vjust = 1,
                       hjust = 0))
```

Statistical summaries

So far, we've only seen **individual geoms**, where there is a graphical object drawn for each observation (e.g. `geom_point()` draws one point per row)

A **collective geom** displays multiple observations with one geometric object. They are often used to display grouped summary statistics (e.g. the average popularity by album) or variable distributions through boxplots or histograms/densities.




```
ts %>%
  filter(!is.na(album)) %>%
  ggplot(aes(x = album,
             y = popularity)) +
  geom_violin(fill = "lightblue",
              alpha = 0.5) +
  geom_boxplot(width = .1,
               alpha = 0.4)
```

HOW TO CHOOSE THE RIGHT PLOT?

- **Do you want to see all the observations or summarize the data?**
- **Are your x and y categorical or continuous?** See the recommendations in the ggplot2 cheatsheet or this decision tree by Yan Holtz & Conor Healy

Look into visualization galleries:

- The R graph ghallery for examples of use cases and inspiration on aesthetics
- 50 ggplot2 visualizations with R code
- Examples from economics research from DIME Analytics Econ Visual Library in R and Stata

BEST RESOURCE for everything related to `ggplot()`

- ggplot2: Elegant Graphics for Data Analysis by Haldey Wickham.

GRAPHICAL EXCELLENCE

4 qualities of great visualizations

1. HONEST

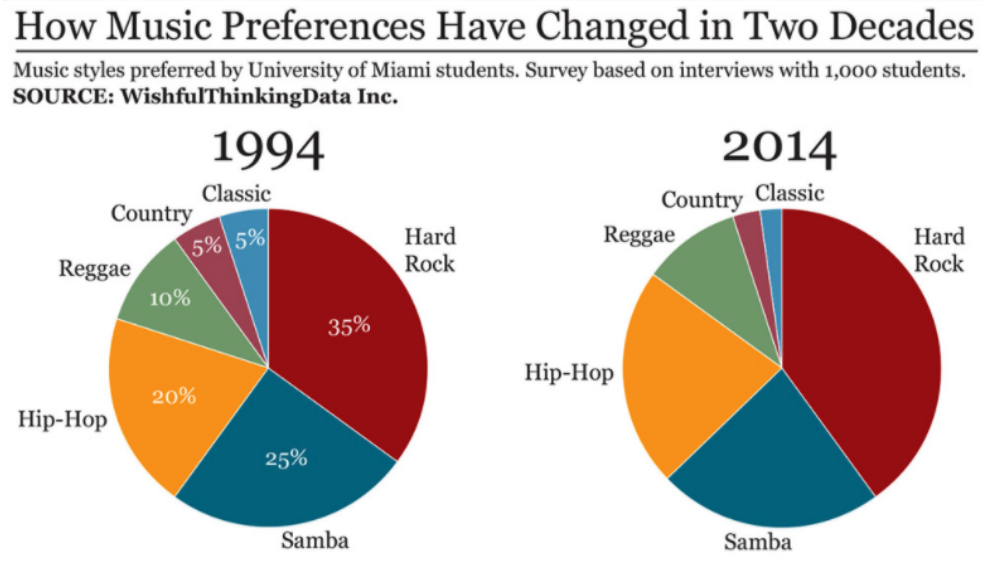
Being careless with scales and axes is dangerous

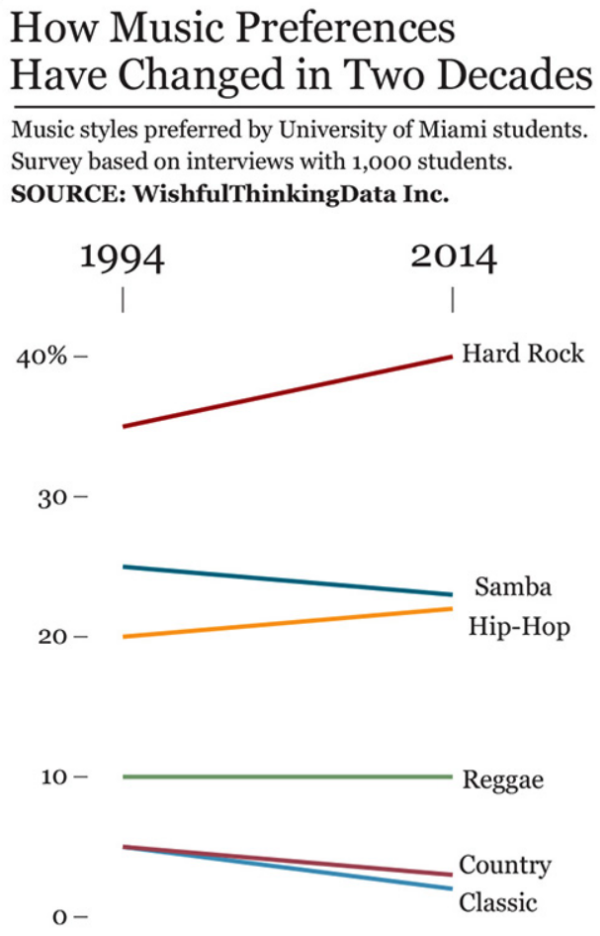
- Zooming or unzooming the graphs can be very misleading. Start your axis at 0 (most of the time).
- Double axes can say whatever you want them to say. Plot things on the same scale.

4 qualities of great visualizations

2. FUNCTIONAL

- Your graph should convey information right but also **help the audience interpret the data correctly** -
- It should be understood without additional context (label axes and add proper legends)

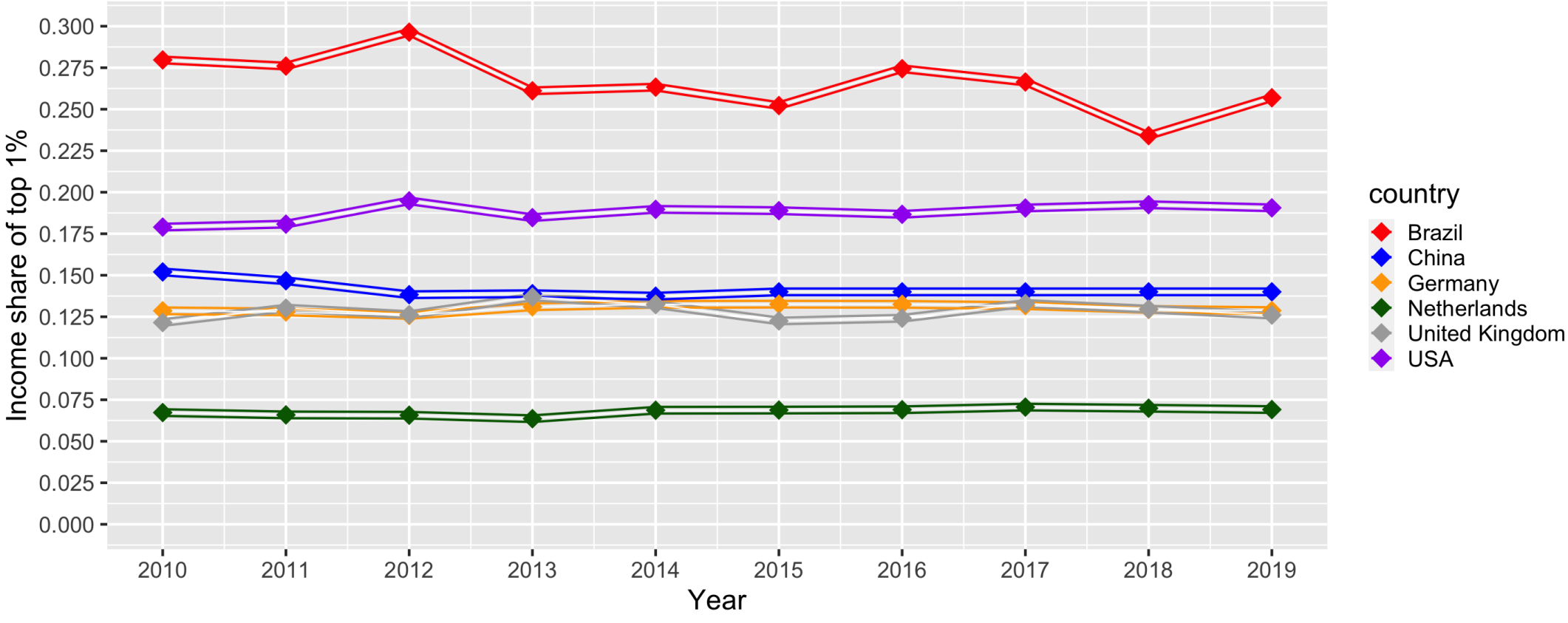




4 qualities of great visualizations

3. BEAUTIFUL

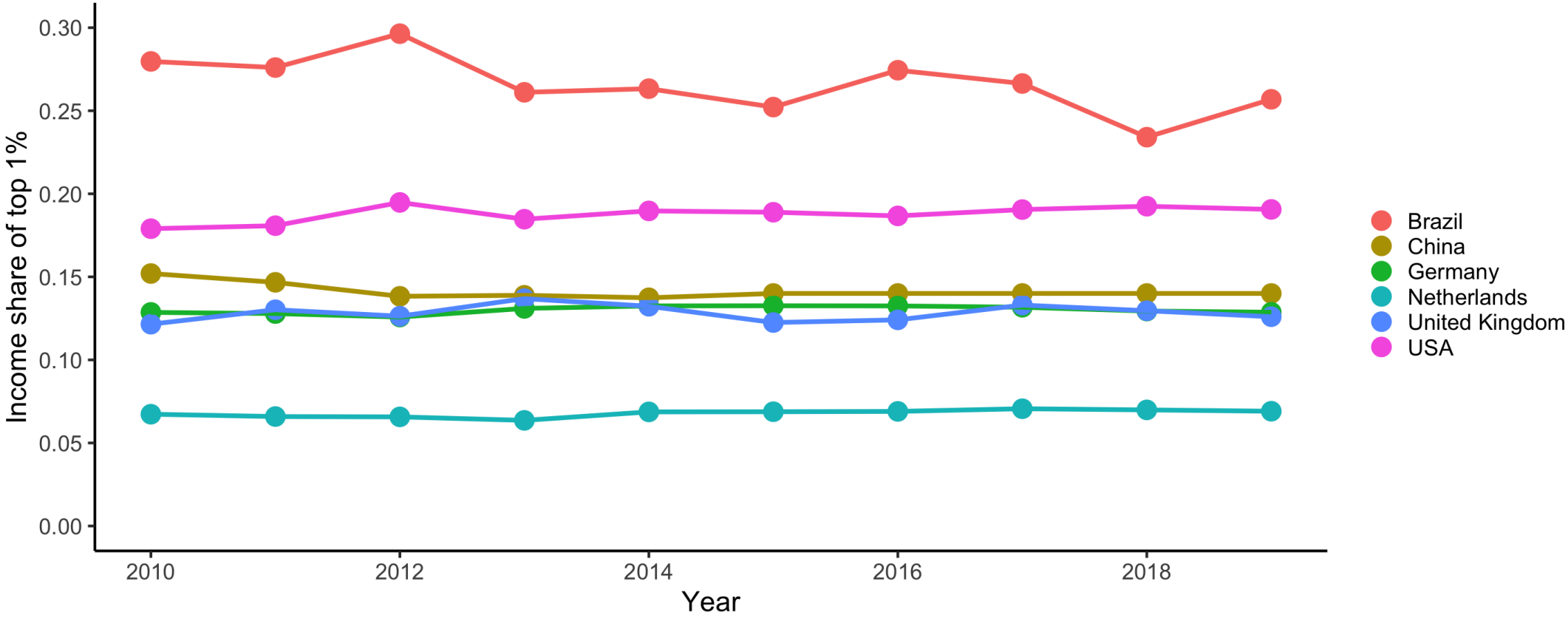
- Your graph should be attractive and ✨**aesthetically pleasing**✨. Declutter!



4 qualities of great visualizations

3. BEAUTIFUL

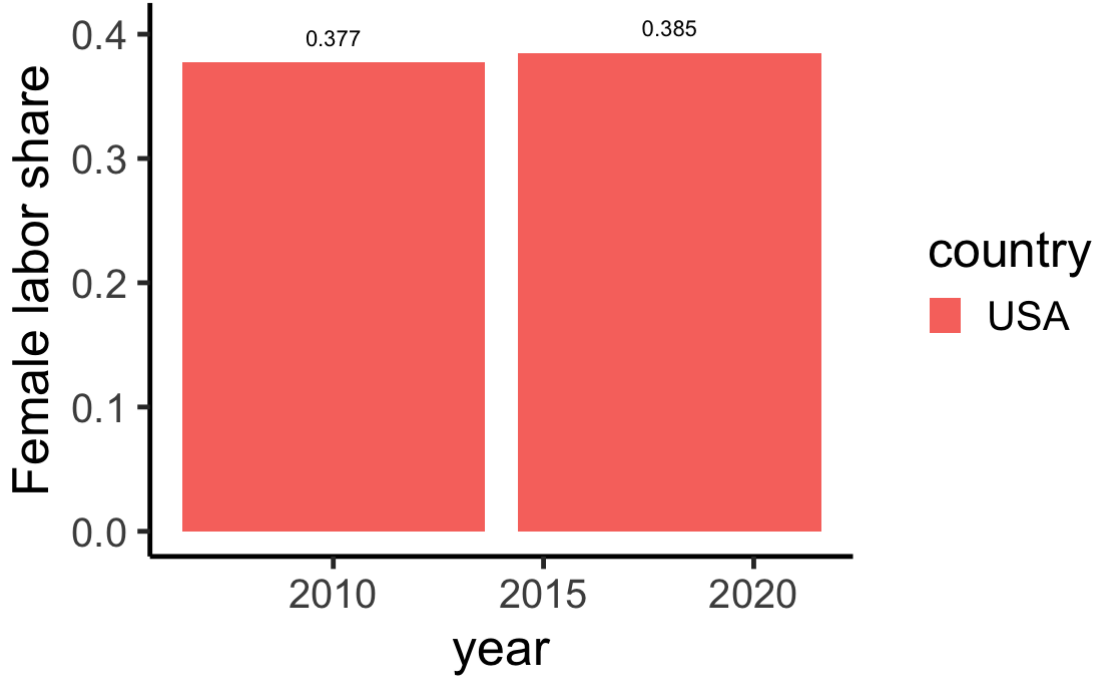
- Your graph should be attractive and ✨**aesthetically pleasing**✨. Declutter!



4 qualities of great visualizations

4. INSIGHTFUL

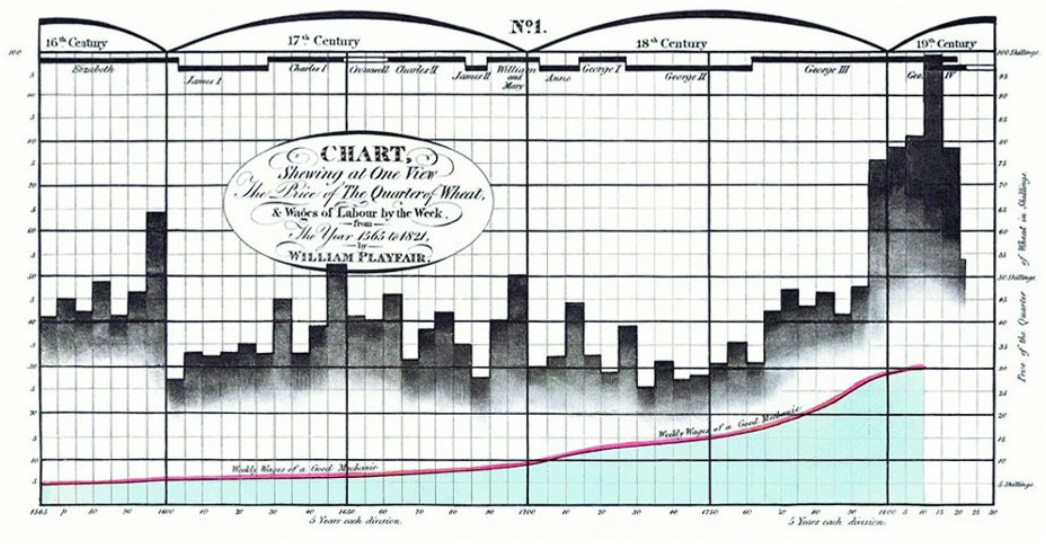
- A graphic should reveal evidence that we would have a hard time seeing otherwise. The purpose of visualization is insight, not pictures.
- From the creators of "*meetings that could have been an e-mail*", we have: "*graphs that could have been a simple table or sentence*"



HOMEWORK

Submit using this link

- Use the `02_playfair-wages-wheat,csv` dataset and and replicate this graph as best as you can using `ggplot()`



  - 🆗 OK: Plotting wheat prices, wages and the timeline of English rulers in the right geoms and colors
  - 👍🏾 Great: Getting the axes (you might need `dup_axis()`) and the overall appearance of the geoms as similar as you can, as well as including the annotation in the middle of the graph ("Chart showing...")
  - 🤩 Amazing: Adding the label over the wages series ("Weekly wages of a good mechanic"), customizing the appearance of the grid and including the labels of the English rulers

SLIDES THAT DIDN'T MAKE THE CUT

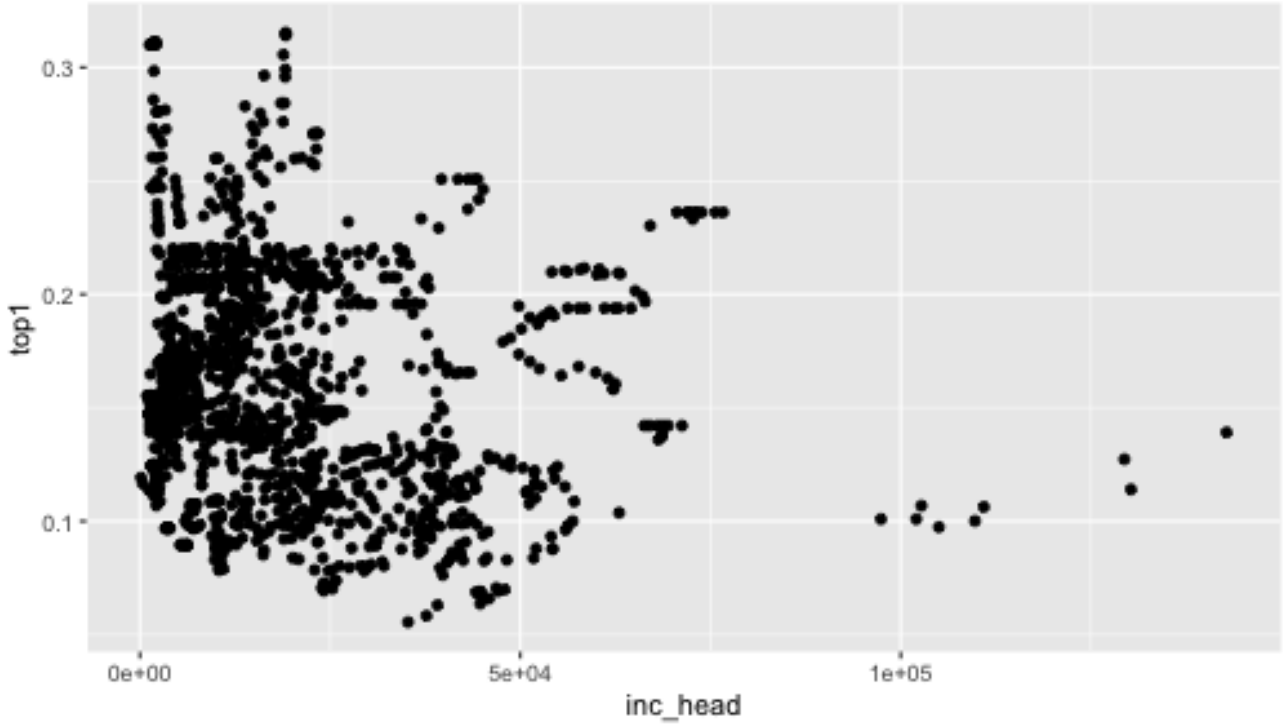
- Axes can be modified with **scale functions**. The following parameters can be specified:
  - `name` the label of the axis
  - `limits` where the axis starts and ends
  - `breaks` where to put ticks and values on the axis

```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point() +

#
```

```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point() +
  scale_x_continuous() +
  scale_y_continuous()
```

Axes

```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point()
                                                    #
```



Axes

```
wid %>%
  ggplot(aes(inc_head, top1)) +
  geom_point() +
  scale_x_continuous(name = "Income per adult",
                     limits = c(0, 150000)) +
  scale_y_continuous(name = "Share of income among top 1%")
```