Introduction to Cultural Data Science: Exam Portfolio - Assignments

Luke Ring (202009983)

12 January, 2023

Contents

Assignment 1: W35: Regular expressions and spreadsheets

1. What regular expressions do you use to extract all the dates in this blurb: http://bit.ly/regexexercise2 and to put them into the following format YYYY-MM-DD?

```
s/([0-9]{1,2})[^0-9]{1,2}([0-9]{1,2})[^0-9]{1,2}([0-9]{2}|[0-9]{4})/\3-\1-\2/g
```

2. Write a regular expression to convert the stopwordlist (list of most frequent Danish words) from Voyant in http://bit.ly/regexexercise3 into a neat stopword list for R (which comprises "words" separated by commas, such as http://bit.ly/regexexercise4). Then take the stopwordlist from R http://bit.ly/regexexercise4 and convert it into a Voyant list (words on separate line without interpunction)

1. `s/^(.*)$\n?/"\1", /g`
2. `s/"([^"]+)"[, ]*/$1\n/g`

3. In 250 words, answer the following question: "What are the basic principles for using spreadsheets for good data organisation?"

> The principles for good data organisation when using spreadsheets are to be consistant, clear, and concise. 1. The data should be stored in a single table. 2. Each column should contain one type of data. 3. Each row should contain one record. 4. Each cell should contain one value. 5. Values should be stored in the most appropriate format. (i.e. numbers should not have things like dollar signs) 6. Value types, including categories should be documented

Challenge (OPTIONAL)!Can you find all the instances of 'Dis Manibus' invocation in the EDH inscriptions in https://bit.ly/regexexercise5? Beware of the six possible canonical versions of the Dis Manibus formula (see day 1 slides)!

```
D(i{1,2}s)?\s+M(anibus)?(\s+S((acrum)|(\b)))?
```

Assignment 2:W35: Open Refine

1. Create a spreadsheet listing the names of Danish monarchs with their birth- and death-date and start and end year of reign. Make it *tidy*! They should be sortable by year of birth. Suitable source websites are here and here, but you can also use another source, provided you reference it. (Group collaboration is expected and welcome. Remember to attach this spreadsheet to Brightspace submission)

The collated CSV is available here: https://github.com/Digital-Methods-HASS/au662726_Ring_Luke/blob/main/data/king_birth_deaths.csv

2. Does OpenRefine alter the raw data during sorting and filtering?

> No, it does not. The raw data is not altered in any way. The sorting and filtering is done on a copy of the data, and the original data is not changed. This allows you to save data at different stages and reproduce the processing steps.

3. Fix the interviews dataset in OpenRefine enough to answer this question: "Which two months are reported as the most water-deprived/driest by the interviewed farmer households?"

> October, September

4. Real-Data-Challenge: What are the 10 most frequent occupations (erhverv) among unmarried men and women in 1801 Aarhus? (hint: some expert judgement interpretation is necessary, look at the HISCO classification "Historical International Standard of Classification of Occupations" on Dataverse if ambitious)

Danish isn't my native language, so it's kind of difficult but something like:

1. Soldier
2. Apprentice

3. Weaver
4. Farmer
5. Seamstress
6. Sailor
7. Day Laborer
8. … ?

Assignment 3:W35: Start with R

1. Use R to figure out how many elements in the vector below are greater than 2 and then tell me what their sum (of the larger than 2 elements) is.

```
rooms <- c(1, 2, 4, 5, 1, 3, 1, NA, 3, 1, 3, 2, 1, NA, 1, 8, 3, 1, 4, NA, 1, 3, 1, 2, 1,
7, 1, 9, 3, NA)
```

`sum(rooms[rooms > 2], na.rm = TRUE)` : the sum is 55

2. What type of data is in the 'rooms' vector?

It is a numeric vector (integers)

3. Submit the following image to Github

Inside your R Project (.Rproj), install the 'tidyverse' package and use the download.file() and read_csv() function to read the SAFI_clean.csv dataset into your R project as 'interviews' digital object (see instructions in https://datacarpentry.org/r-socialsci/setup.html and 'Starting with Data' section). Take a screenshot of your RStudio interface showing

a) the line of code you used to create the object,

b) the 'interviews' object in the Environment, and

c) the file structure of your R project in the bottom right "Files" pane.

Save the screenshot as an image and put it in your AUID_lastname_firstname repository inside our Github organisation (github.com/Digital-Methods-HASS) or equivalent. Place here the URL leading to the screenshot in your repository.

https://github.com/Digital-Methods-HASS/au662726_Ring_Luke/blob/main/figures/IDE_Screenshot_with_data.png

4. Challenge

If you managed to create your own Danish king dataset, use it. If not, you the one attached to this assignment (it might need to be cleaned up a bit). Load the dataset into R as a tibble. Calculate the mean() and median() duration of rule over time and find the three mondarchs ruling the longest. How many days did they rule (accounting for transition year?)

Mean: 18.68 Median: 14

1. Christian IV: 60 years
2. Erik VII af Pommern: 43 years
3. Christian VII: 42 years

The top three kings ruled for a total of 52959 days.

```r
# make sure danish characters load correctly
loc <- locale(encoding = "ISO-8859-15")

# read in data
king_data <- read_csv2("../data/kings.csv", locale = loc, col_types = c("c", "i",
    "i", "c"  # has the word 'unknown'
))

# filter NA values
king_data <- king_data[!is.na(king_data$Start_date), ]
king_data <- king_data[!is.na(king_data$End_date), ]

# now that the unknowns have gone, convert to numeric
king_data$Yearasruler <- as.numeric(king_data$Yearasruler)
# caluclate mean and median ruling time
mean_years_ruling <- mean(king_data$Yearasruler)
median_years_ruling <- median(king_data$Yearasruler)

mean_years_ruling
median_years_ruling

# find the top 3 kings with the most years ruling
top_3_ruling <- king_data[order(king_data$Yearasruler, decreasing = TRUE)[1:3], ]

paste(top_3_ruling$Kings, ", ", top_3_ruling$Yearasruler, " years", collapse = "; ")

top_3_ruling

# convert start date to a date type
top_3_ruling$Start_date <- as.Date(paste0(top_3_ruling$Start_date, "-01-01"), format =
↪  "%Y-%m-%d")

# convert end date to a date type
top_3_ruling$End_date <- as.Date(paste0(top_3_ruling$End_date, "-01-01"), format =
↪  "%Y-%m-%d")

# calculate the days spent ruling
days_ruling <- difftime(top_3_ruling$End_date, top_3_ruling$Start_date, units = "days")

# add the top three kings' ruling days together
```

```r
total_days_ruling <- sum(days_ruling)

total_days_ruling
```

Assignment 4:W35: Visualize data (not only) with ggplot

This exercise is based on the dataset provided by OurWorldInData project based at the Oxford University.

The long-term trend in Homicides in Western Europe

Understanding how homicide rates have changed prior to the modern era requires the help of historians and archivists. Manuel Eisner, a criminology professor at the University of Cambridge, and his colleagues published the Historical Violence Database : a compilation of data on long-term trends in homicide rates, in addition to qualitative information such as the cause of death, perpetrator and victim. This database is limited to countries with relatively complete historical records on violence and crime – mainly Western Europe and the US.

Starting in the second half of the nineteenth century, these European regions have consistent police records of those accused of murder or manslaughter and annual counts of homicide victims. To go back further in time, reaching as far back as the thirteenth century, Eisner collected estimates (from historical records of coroner reports, court trials, and the police) of homicide rates made in over ninety publications by scholars.

Homicide rates – measured as the number of homicides per 100,000 individuals – up to 1990 are sourced from Eisner's (2003) publication and the Historical Violence Database.

Are homicide rates in Europe today lower or higher than in the past? Using the provided dataset, display and describe the long-run homicide rates for the five European regions: Italy, England, Germany, Netherlands and Scandinavia.

Load the available data from ourworldindata.org

You should always interrogate the source of your data. Who compiled it, from where, what is missing, how representative the data are? Check the data/Metadata.txt to learn about the data provenance.

- OurWorldInData, multiple data sets
- Variable geographic coverage Western Europe
- Variable time span 1300 – 2016

```r
Western_Europe <- read_csv("../data/homicide-rates-across-western-europe.csv")
```

```
## Rows: 206 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Entity, Code
## dbl (2): Year, Homicide rate in Europe over long-term (per 100,000) (homicid...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Inspect the data

How clean and analysis-ready is the dataset? Do you understand what the column names represent? What is the difference between rate and homicide number?

```
head(Western_Europe)
```

Entity	Code	Year	Homicide rate in Europe over long-term (per 100,000) (homicides per 100,000 people)
England	NA	1300	23
England	NA	1550	7
England	NA	1625	6
England	NA	1675	4
England	NA	1725	2
England	NA	1775	1

Ok, the data look good except for the column `Homicide rate in Europe over long-term (per 100,000)` which is not very easy to work with.

- Use the `names()` function and assignment key to relabel this column to `homicides_per_100k`

```
names(Western_Europe)[4] <- "homicides_per_100k"
head(Western_Europe)
```

Entity	Code	Year	homicides_per_100k
England	NA	1300	23
England	NA	1550	7
England	NA	1625	6
England	NA	1675	4
England	NA	1725	2
England	NA	1775	1

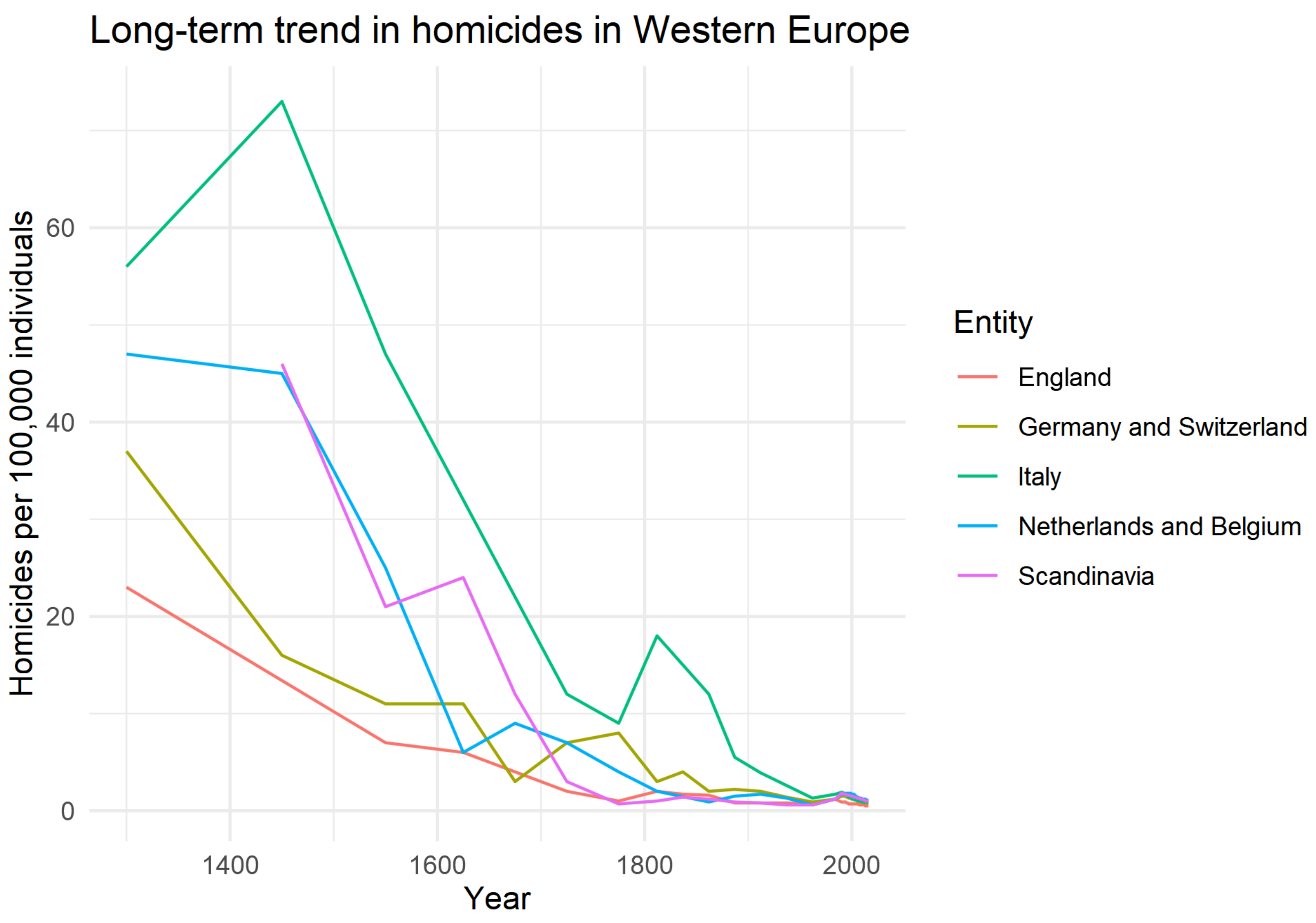
Now, that you have looked at what the data looks like and what it represents, and streamlined it, let's see what big picture it contains.

Let's see what the long-term trend is in homicides

- use `ggplot()` function and remember the`+` at the end of the line

- chose a meaningful `geom_......()` for geometry (hint: points are not great)
- load `Year` on the `x` axis and `homicides_per_100k` column in y axis
- to color individual country entries consistently, assign the country column to the argument `color`.
- provide meaningful title and axis labels
- remember to change the `eval` flag so that the code chunk renders when knitted

```
plt_hr <- ggplot(data = Western_Europe) + geom_line(aes(x = Year, y = homicides_per_100k,
    color = Entity)) + labs(title = "Long-term trend in homicides in Western Europe",
    x = "Year", y = "Homicides per 100,000 individuals") + theme_minimal()
plt_hr
```



Alright, the homicide rates should all be descending over time. What a comfort. But the viz is not super clear. Let's check the rates for individual countries.

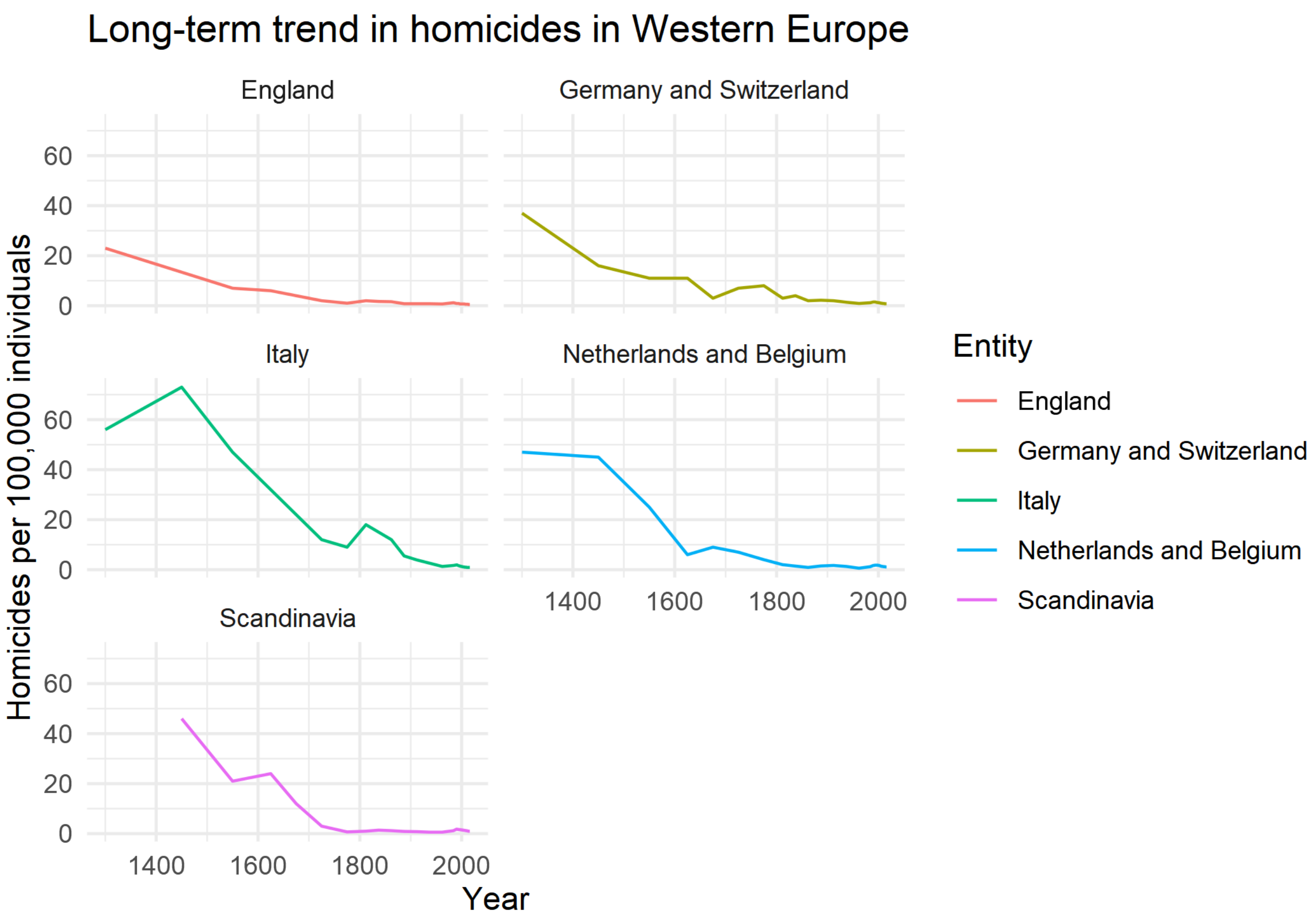
Uncouple the homicides of individual countries for easier view

You can visualize each country's trend separately by adding an extra argument to the ggplot, the `facet_wrap()` and feeding it the country column. If in doubt, check your ggplot tutorial and your country column name for exact usage.

- reuse the ggplot from the chunk above

- insert `facet_wrap()` after the specification of geometry to split countries in separate charts
- change the facet "layout" to two columns and three rows so that the trends are easier to see in horizontal layout.

```
plt_hr + facet_wrap(~Entity, ncol = 2, nrow = 3)
```



Compare the trends in homicide with the pattern of reign duration among Danish rulers through time.

- Load your Danish king dataset. Hopefully it is tidy and your years and duration of reign are all numeric.
- You need to have a consistent way of plotting the rulers' reign on the x axis, so I recommend you create a midyear column by calculating the middle of each monarch's rule (Hint: `midyear` = `endyear - (endyear-startyear)/2`)
- Start a ggplot plotting midyear on x axis and duration on y axis
- Try `geom_smooth()` for geometry
- Provide meaningful labels and a title
- How would you characterize the trend compared to the homicides above?

```r
kings_data <- read_delim("../data/king_birth_deaths.csv", delim = ";")
```

```
## Rows: 55 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (1): king
## dbl (4): rule_start, rule_end, birth, death
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

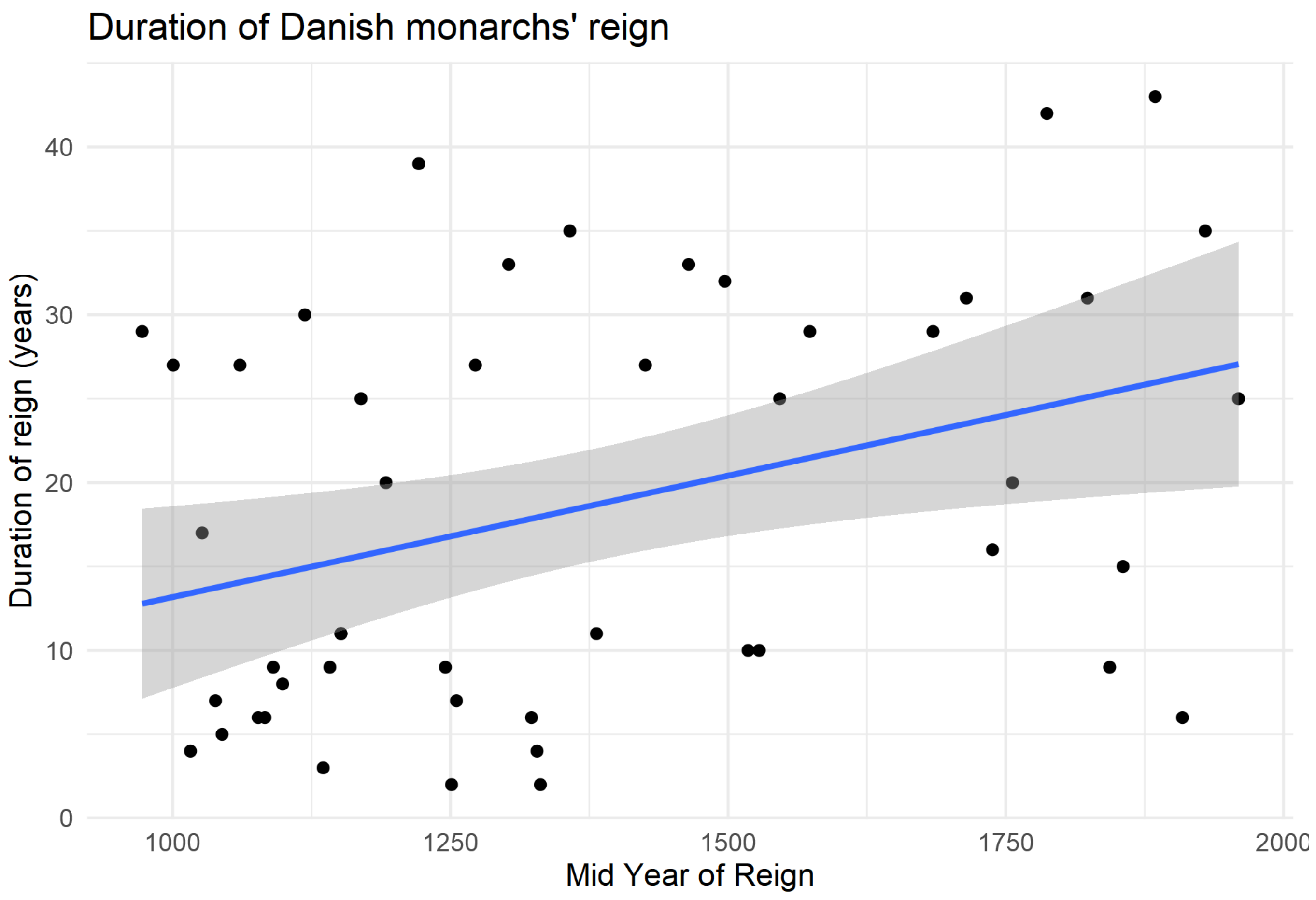
```r
kings_data <- kings_data %>%
    mutate(midyear = rule_end - (rule_end - rule_start)/2, rule_duration = rule_end -
        rule_start)


ggplot(data = kings_data) + geom_point(aes(x = midyear, y = rule_duration)) +
↪   geom_smooth(aes(x = midyear,
    y = rule_duration), method = "lm") + labs(title = "Duration of Danish monarchs'
     ↪  reign",
    x = "Mid Year of Reign", y = "Duration of reign (years)") + theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 7 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 7 rows containing missing values (`geom_point()`).
```

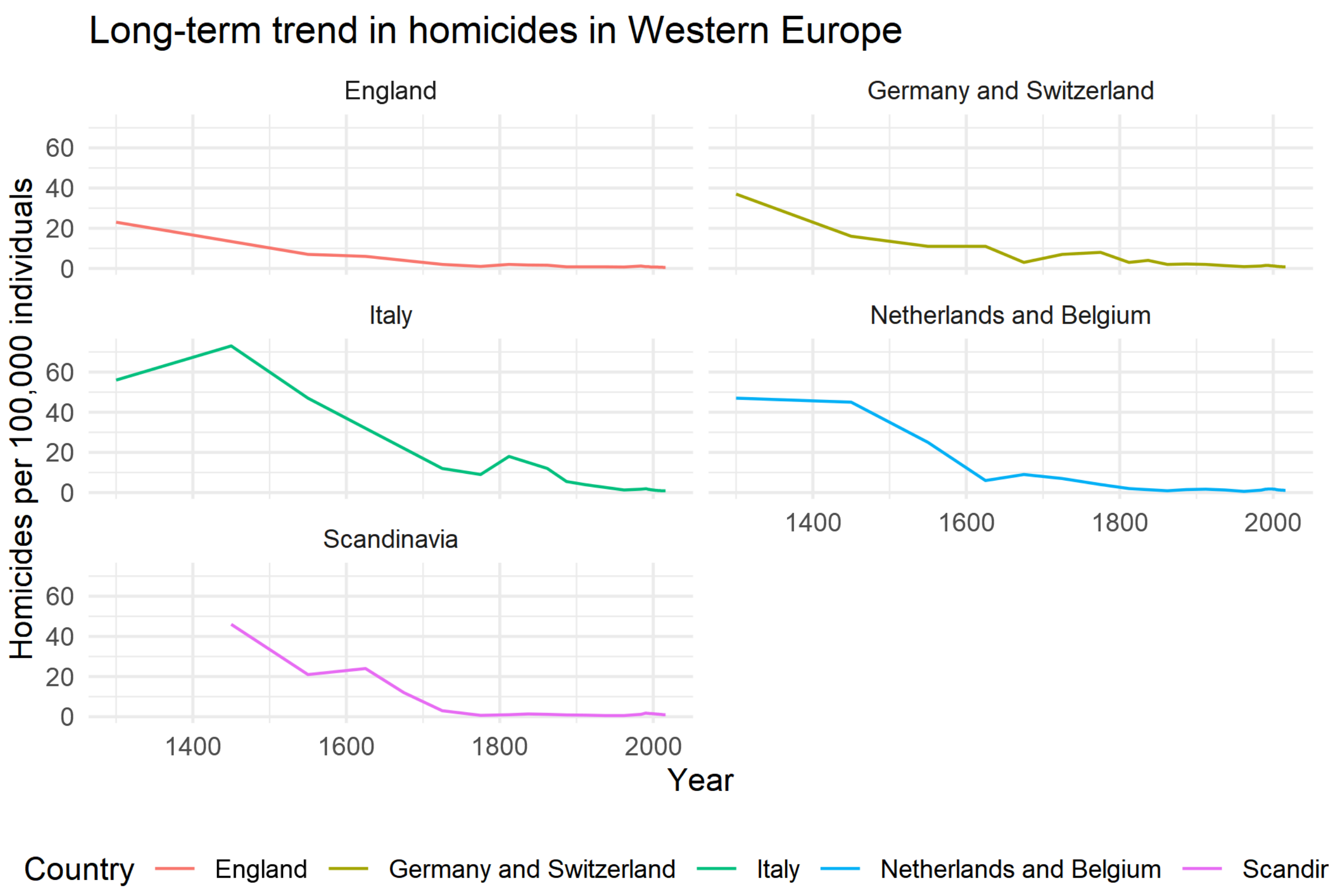


- Overall it seems that there is an increasing trend in the duration of Danish monarchs' reign. This is the opposite of the trend in homicides.

Final Tasks

1) Plot: In the facetted plot above, move the legend from the current position on the side to below the facets, and label it "Country" instead of "Entity".

```
plt_hr + facet_wrap(~Entity, ncol = 2, nrow = 3) + theme(legend.position = "bottom") +
    labs(color = "Country")
```



2) Rmarkdown:

- edit the author of the document, and convert 'Final Tasks' into heading #2 (like the other headings)
- add a `floating table of contents` to your Rmarkdown document,
- provide informative `chunk-names` and edit flags in your R chunks, and
- automatically generate a `timestamp` to show when the document was last updated. (Hint: check the Rmarkdown episode in our Data Carpentry tutorial)

3) Question: In <250 words articulate your answer on the basis of the data visualisations to the following question: are we more civilized today?

That's not an easy question to answer, in general we can say that homocide rates in the 7 countries (5 regions) investigated seem to have declined, which is a positive thing, but that doesn't take into account so many factors, and these data are only showing trends in a few select Western European countries, so we cannot say anything about if this is a global trend. There are also many aspects that I would consider relevant to what most people think of as "civilized", including things like general attitudes to other humans and animal, access to healthcare, assistance for underprivileged individuals and education systems. Of course, I think most people, myself included, will agree that a reduction in homocides is a good thing.

Assignment 5:W35: Managing Files on Steroids with Shell

Your supervisor has shared a folder of photos on Sciencedata.dk with you (password is 2020CDS, folder is 500Mb and contains 189 images) and needs your help with a couple diagnostics:

1) Identify the names and format of the 3 biggest files. Can you come up with a command to generate a numerically ordered list of 3 biggest files? (hint: consider using wc to gauge image size)

```
# This shell script does the following
# 1. Finds the N biggest image files in the specified folder and prints them
# 2. Finds images  that are 0 bytes in size, prints the total number and lists the
↪  filenames
# 3. optionally replaces the 0 byte files with a file from the specified folder matching
↪  the filename

# Set the folder to search
$folder = ".\images\HW"
# Set the folder to search for replacement files
$replacementFolder = ".\images\goodphotos"
# do replacement?
$replace = $true

# Set the number of files to find
$number = 3

# Find the N biggest files in the folder
$biggest = Get-ChildItem $folder -Recurse -File | Sort-Object -Property Length
↪  -Descending | Select-Object -First $number

# Print the N biggest files
Write-Host "The $number biggest files in $folder are:"
$biggest | Format-Table -AutoSize

# Find the files that are 0 bytes in size
$zero = Get-ChildItem $folder -Recurse -File | Where-Object {$_.Length -eq 0}

# Print the number of files that are 0 bytes in size
Write-Host "There are $($zero.Count) files that are 0 bytes in size in $folder"

# Print the filenames of the files that are 0 bytes in size
Write-Host "The files that are 0 bytes in size are:"
$zero | Format-Table -AutoSize

# Replace the 0 byte files with a file from the specified folder matching the filename
if ($replace) {
    foreach ($file in $zero) {
        $replacementFile = Get-ChildItem $replacementFolder -Recurse -File |
        ↪  Where-Object {$_.Name -eq $file.Name}
        if ($replacementFile) {
            Write-Host "Replacing $file with $replacementFile"
            Copy-Item $replacementFile.FullName $file.FullName -Force
        }
    }
}
```

The three largest files are:

1. size: 14761472 bytes, name: 9240_Overview_S.RW2, format: RW2 (RAW file format)
2. size: 14733312 bytes, name: 9247_Overview_SW.RW2, format: RW2 (RAW file format)
3. size: 14713856 bytes, name: 9237_Overview_W.RW2, format: RW2 (RAW file format)

2) Some of the image files are empty, a sign of corruption. Can you find the empty photo files (0 kb size) , count them, and generate a list of their filenames to make their later replacement easier?

73 files are 0 bytes in size.

```
There are 73 files that are 0 bytes in size in .\images\HW
The files that are 0 bytes in size are:

    Directory: C:\Users\xxxx\Documents\github_projects\au662726_Ring_Luke\images\HW

Mode        LastWriteTime Length Name
----        ------------- ------ ----
-a--- 18/01/2017    11.01      0 9260_Detail_Drain.JPG
-a--- 18/01/2017    11.01      0 9260_Detail_Stratigraphy.JPG
-a--- 18/01/2017    11.01      0 9260_Overview_E.JPG
-a--- 18/01/2017    11.01      0 9260_Overview_S.JPG
-a--- 18/01/2017    11.01      0 9261_Overview_E.JPG
-a--- 18/01/2017    11.01      0 9262_Overview_S.JPG
-a--- 18/01/2017    11.01      0 9263_Overview_W.JPG
-a--- 18/01/2017    11.01      0 9264_Overview_W.JPG
-a--- 18/01/2017    11.01      0 9265_Overview_W.JPG
-a--- 18/01/2017    11.01      0 9266_Overview_E.JPG
-a--- 18/01/2017    11.01      0 9267_Overview_W.JPG
-a--- 18/01/2017    11.01      0 9268_Overview_E.JPG
-a--- 18/01/2017    11.01      0 9269_Overview_S.JPG
-a--- 18/01/2017    11.01      0 9270_Overview_S.JPG
-a--- 18/01/2017    11.01      0 9271_Overview_W.JPG
-a--- 18/01/2017    11.01      0 9272_Overvieew_SE.JPG
-a--- 18/01/2017    11.01      0 9272_Overview_with_Dich_E.JPG
-a--- 18/01/2017    11.01      0 9273_Overview_S.JPG
-a--- 18/01/2017    11.01      0 9274_Overview_W.JPG
-a--- 18/01/2017    11.01      0 9275_Overview_SW.JPG
```

```
-a--- 18/01/2017    11.01       0 9276_Overview_SW.JPG
-a--- 18/01/2017    11.01       0 9277_Overview_NW.JPG
-a--- 18/01/2017    11.01       0 9278_Overview_W.JPG
-a--- 18/01/2017    11.01       0 9279_Overview_SW.JPG
-a--- 18/01/2017    11.01       0 9280_Overview_SW.JPG
-a--- 18/01/2017    11.02       0 9281_Overvierview_W.JPG
-a--- 18/01/2017    11.02       0 9282_Overview_W.JPG
-a--- 18/01/2017    11.48       0 9283_Overview_W.JPG
-a--- 18/01/2017    11.50       0 9284_Overview_S.JPG
-a--- 18/01/2017    11.48       0 9285_Overview_W.JPG
-a--- 18/01/2017    11.47       0 9286_Overview_SW.JPG
-a--- 18/01/2017    11.46       0 9287_Overview_S.JPG
-a--- 18/01/2017    11.46       0 9288_Overview_S.JPG
-a--- 18/01/2017    11.46       0 9289_Overview_S.JPG
-a--- 18/01/2017    11.46       0 9290_Overview_S.JPG
-a--- 18/01/2017    11.45       0 9291_Overview_W.JPG
-a--- 18/01/2017    11.45       0 9292_Overview_SE.JPG
-a--- 18/01/2017    11.45       0 9293_Overview_NW.JPG
-a--- 18/01/2017    11.02       0 9301_Overview_S.JPG
-a--- 18/01/2017    11.02       0 9302_overview_N.JPG
-a--- 18/01/2017    11.02       0 9302_overview_W.JPG
-a--- 18/01/2017    11.02       0 9303_overview_W.JPG
-a--- 18/01/2017    11.02       0 9304_overview_pool.JPG
-a--- 18/01/2017    11.02       0 9304_overview_W.JPG
-a--- 18/01/2017    11.02       0 9305_overview_RT.JPG
-a--- 18/01/2017    11.02       0 9305_overview_S.JPG
-a--- 18/01/2017    11.02       0 9306_overview_S.JPG
-a--- 18/01/2017    11.02       0 9306_ovrview_N.JPG
-a--- 18/01/2017    11.02       0 9307_overview_S.JPG
-a--- 18/01/2017    11.02       0 9308_overview_E.JPG
-a--- 18/01/2017    11.02       0 9309_overview_W.JPG
-a--- 18/01/2017    11.02       0 9310_overview_S.JPG
-a--- 18/01/2017    11.02       0 9311_overview_RT_W.JPG
-a--- 18/01/2017    11.02       0 9311_overview_W.JPG
-a--- 18/01/2017    11.02       0 9312_overview_W.JPG
-a--- 18/01/2017    11.02       0 9313_overview_W.JPG
-a--- 18/01/2017    11.03       0 9314_overview_S.JPG
-a--- 18/01/2017    11.03       0 9315_overview_S.JPG
-a--- 18/01/2017    11.03       0 9316_overview_N.JPG
-a--- 18/01/2017    11.03       0 9317_overview_E.JPG
-a--- 18/01/2017    11.03       0 9318_overview_W.JPG
-a--- 18/01/2017    11.03       0 9319_overview_W.JPG
-a--- 18/01/2017    11.03       0 9320_overview_W.JPG
-a--- 18/01/2017    11.03       0 9321_overview_W.JPG
-a--- 18/01/2017    11.03       0 9321_RT(detail)_S.JPG
-a--- 18/01/2017    11.03       0 9322_overview1_S.JPG
-a--- 18/01/2017    11.03       0 9322_overview2_S.JPG
-a--- 18/01/2017    11.03       0 9322_RT(detail).JPG
```

```
-a--- 18/01/2017    11.03      0 9323_overview_S.JPG
-a--- 18/01/2017    11.03      0 9324_overview_W.JPG
-a--- 18/01/2017    11.03      0 9325_overview_S.JPG
-a--- 18/01/2017    11.03      0 9326_overview_W.JPG
-a--- 18/01/2017    11.03      0 9327_overview_E.JPG
```

3) Optional/Advanced: Imagine you have a directory goodphotos/ (same password as above) with original non-zero-length files sitting at the same level as the current directory. How would you write a loop to replace the zero length files?

From part 1):

```
# Replace the 0 byte files with a file from the specified folder matching the filename
if ($replace) {
    foreach ($file in $zero) {
        $replacementFile = Get-ChildItem $replacementFolder -Recurse -File |
        ↪  Where-Object {$_.Name -eq $file.Name}
        if ($replacementFile) {
            Write-Host "Replacing $file with $replacementFile"
            Copy-Item $replacementFile.FullName $file.FullName -Force
        }
    }
}
```

Assignment 6:W43: Practicing functions with Gapminder

Use the gapminder dataset from Week 43 to produce solutions to the three tasks below.

Task 1: Defensive GDP Function

> Define a defensive function that calculates the Gross Domestic Product of a nation from the data available in the gapminder dataset. You can use the population and GDPpercapita columns for it. Using that function, calculate the GDP of Denmark in the following years: 1967, 1977, 1987, 1997, 2007, and 2017. First we define the function:

```
# simple function to test if a value is interger-like
is_wholenumber <- function(x) {
    if (is.numeric(x) & x%%1 == 0) {
        return(TRUE)
    } else {
        return(FALSE)
    }
}
# The main GDP calculation function
gapminder_gdp <- function(for_country, for_year) {
```

```r
    # first check that country is a character
    if (!is.character(for_country)) {
        stop("country must be a character")
    }
    # then check that year is an integer
    if (!is_wholenumber(for_year)) {
        stop("year must be an integer")
    }
    # ensure dplyr is loaded
    if (!require(dplyr)) {
        stop(paste("dplyr is required to run this function.", "Please install it with",
            "install.packages('dplyr')"))
    }
    # make sure the gampnider dataset exists
    if (!require(gapminder)) {
        stop(paste("Gapminder package not available,", "you can install by running",
            "install.packages(\"gapminder\")"))
    }
    # make the country and gapminder countries lowercase to avoid case
    # sensitivity issues
    country_orig <- for_country
    for_country <- tolower(for_country)
    levels(gapminder$country) <- tolower(levels(gapminder$country))
    # check that the country is in the gapminder dataset
    if (!for_country %in% gapminder$country) {
        stop(paste("Country", country_orig, "not found in gapminder dataset,", "please
         ↪  check the spelling of the country"))
    }
    # check that the year is in the gapminder dataset
    if (!for_year %in% gapminder[gapminder$country == for_country, ]$year) {
        # we can report available range of years for the country to help the
        # user
        years <- gapminder %>%
            filter(country == for_country) %>%
            pull(year)
        stop(paste("Year", for_year, "not found in gapminder dataset for",
         ↪  paste0(country_orig,
            ".")), "\nPlease ensure the year is between", min(years), "and",
             ↪  paste0(max(years),
            ".")))
    }
    # calculate the GDP
    gdp <- gapminder %>%
        filter(country == for_country, year == for_year) %>%
        mutate(gdp = pop * gdpPercap) %>%
        pull(gdp)
    # return the GDP
    return(gdp)
```

```
}
```

Now we can test the function output for the years requested:

```r
# test the function
test_country <- "Denmark"
test_years <- c(1967, 1977, 1987, 1997, 2007, 2017)
# loop through the years and print the results
for (test_year in test_years) {
    # get the GDP for the country and year
    tryCatch({
        gdp <- gapminder_gdp(test_country, test_year)
        # print the results
        print(paste("The GDP of", test_country, "in", test_year, "was",
         ↪  (scales::label_dollar(prefix = "US$"))(gdp)))
    }, error = function(e) {
        # print the error message
        print(e)
    })
}
```

```
## [1] "The GDP of Denmark in 1967 was US$77,116,977,700"
## [1] "The GDP of Denmark in 1977 was US$103,920,280,028"
## [1] "The GDP of Denmark in 1987 was US$128,771,236,166"
## [1] "The GDP of Denmark in 1997 was US$157,476,118,456"
## [1] "The GDP of Denmark in 2007 was US$192,906,627,081"
## <simpleError in gapminder_gdp(test_country, test_year): Year 2017 not found in gapminder datas
## Please ensure the year is between 1952 and 2007.>
```

We can see that the output shows the GDP for all the years except for 2017, which is not in the dataset. In this case an error is thrown which stops execution. This means you can't accidentally get strange results from the function, or if you try to do something that isn't possible, it will not just silently continue, but will let you know immediately, allowing you to fix the issue.

In the test examples above, we have a `try / catch` block which allows us to continue our script / Rmd knitting even if there is an error, this is fine for testing but you would not want to do that with real data unless you are absolutely sure that you don't care about data that couldn't be found.

Task 2: Life Expectancy for countries starting with "B"

> Write a script that loops over each country in the gapminder dataset, tests whether the country starts with a 'B' , and prints out whether the life expectancy is smaller than 50, between 50 and 70, or greater than 70. (Hint: remember the grepl function, and review the Control Flow tutorial)

```r
# loop through the countries
for (cntry in unique(gapminder$country)) {
    # test if the country starts with a B
    if (!grepl("^B", cntry)) {
        # if not, skip to the next country
        next
    }
    # get the life expectancy
    life_expectancy <- gapminder %>%
        filter(country == cntry) %>%
        pull(lifeExp)
    # calculate the mean life expectancy for the years found
    mean_life_expectancy <- mean(life_expectancy, na.rm = TRUE)
    # get the range of years
    years <- gapminder %>%
        filter(country == cntry) %>%
        pull(year)
    # print the results
    print(paste("The mean life expectancy for", cntry, "from", min(years), "to",
        max(years), "was: "))
    # test the life expectancy
    if (mean_life_expectancy < 50) {
        print(paste("less than 50 years"))
    } else if (mean_life_expectancy >= 50 & mean_life_expectancy <= 70) {
        print(paste("between 50 and 70 years"))
    } else {
        print(paste("greater than 70 years"))
    }
}
```

```
## [1] "The mean life expectancy for Bahrain from 1952 to 2007 was: "
## [1] "between 50 and 70 years"
## [1] "The mean life expectancy for Bangladesh from 1952 to 2007 was: "
## [1] "less than 50 years"
## [1] "The mean life expectancy for Belgium from 1952 to 2007 was: "
## [1] "greater than 70 years"
## [1] "The mean life expectancy for Benin from 1952 to 2007 was: "
## [1] "less than 50 years"
## [1] "The mean life expectancy for Bolivia from 1952 to 2007 was: "
## [1] "between 50 and 70 years"
## [1] "The mean life expectancy for Bosnia and Herzegovina from 1952 to 2007 was: "
## [1] "between 50 and 70 years"
## [1] "The mean life expectancy for Botswana from 1952 to 2007 was: "
## [1] "between 50 and 70 years"
## [1] "The mean life expectancy for Brazil from 1952 to 2007 was: "
## [1] "between 50 and 70 years"
## [1] "The mean life expectancy for Bulgaria from 1952 to 2007 was: "
## [1] "between 50 and 70 years"
## [1] "The mean life expectancy for Burkina Faso from 1952 to 2007 was: "
## [1] "less than 50 years"
## [1] "The mean life expectancy for Burundi from 1952 to 2007 was: "
## [1] "less than 50 years"
```

Bonus: the same, but using `tidyverse`

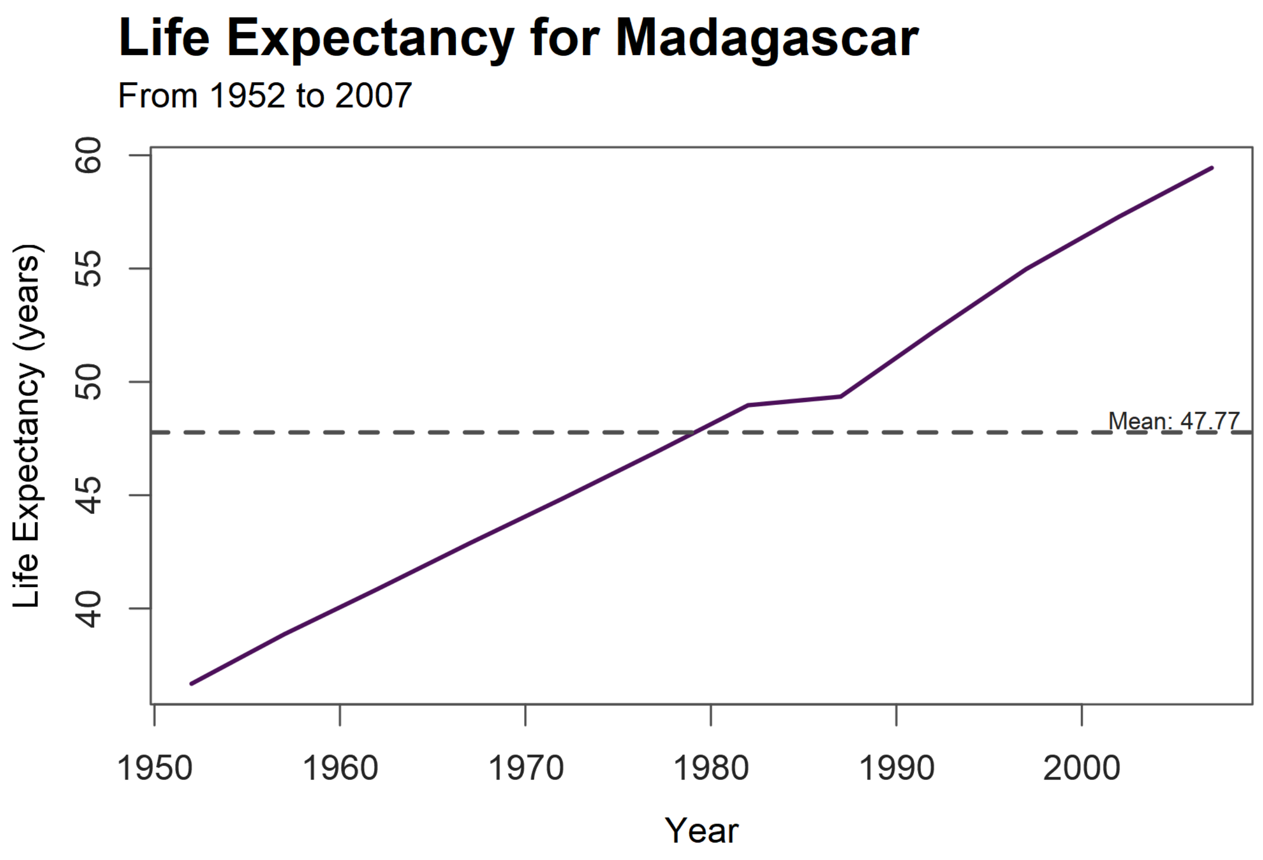
```
# find countries starting with B, and get the mean life expectancy
gapminder %>%
    filter(grepl("^B", country)) %>%
    group_by(country) %>%
    summarise(mean_life_expectancy = mean(lifeExp, na.rm = TRUE), years =
     ↪  paste0(min(year),
        "-", max(year))) %>%
    mutate(life_expectancy = case_when(mean_life_expectancy < 50 ~ "less than 50 years",
        mean_life_expectancy >= 50 & mean_life_expectancy <= 70 ~ "between 50 and 70
↪   years",
        TRUE ~ "greater than 70 years")) %>%
    select(country, years, life_expectancy) %>%
    print(n = Inf)
```

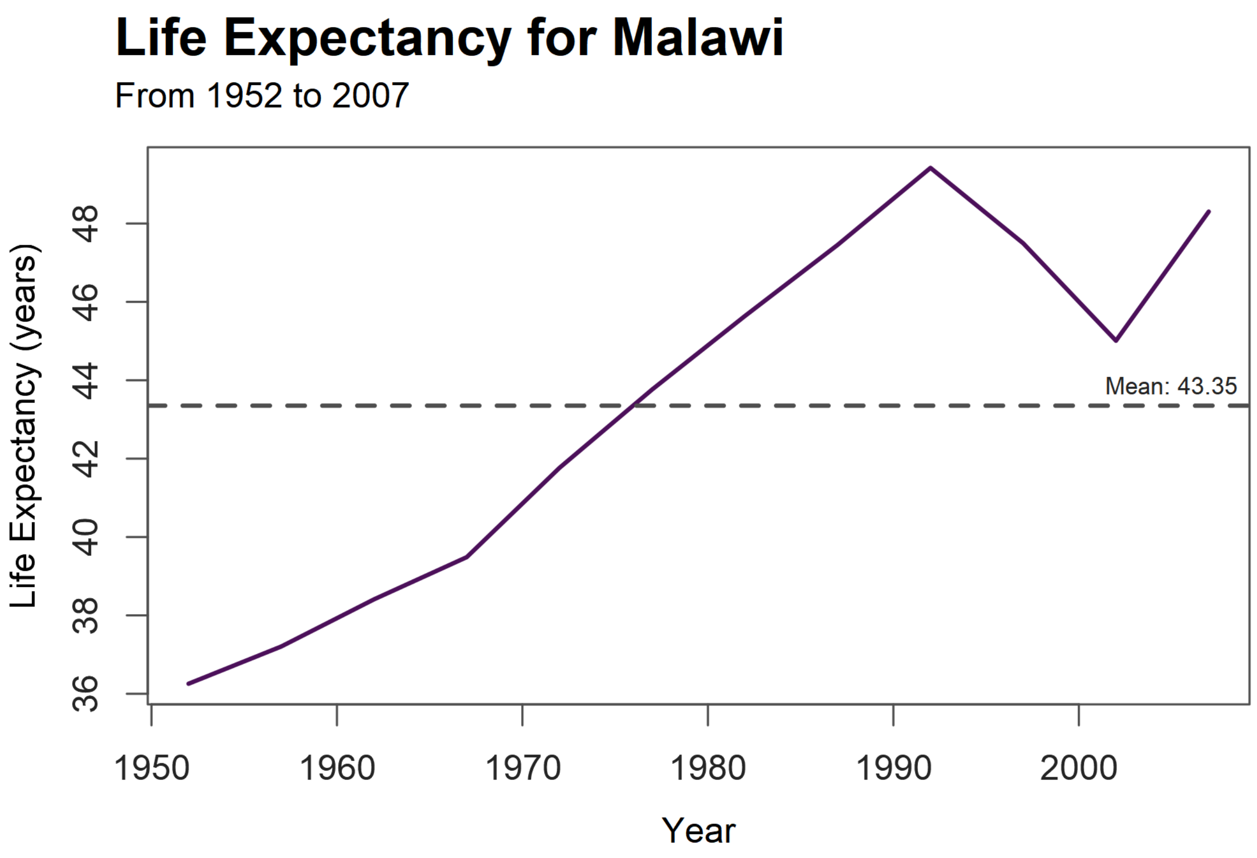
```
## # A tibble: 11 x 3
##    country                years     life_expectancy
##    <fct>                  <chr>     <chr>
##  1 Bahrain                1952-2007 between 50 and 70 years
##  2 Bangladesh             1952-2007 less than 50 years
##  3 Belgium                1952-2007 greater than 70 years
##  4 Benin                  1952-2007 less than 50 years
##  5 Bolivia                1952-2007 between 50 and 70 years
##  6 Bosnia and Herzegovina 1952-2007 between 50 and 70 years
##  7 Botswana               1952-2007 between 50 and 70 years
##  8 Brazil                 1952-2007 between 50 and 70 years
##  9 Bulgaria               1952-2007 between 50 and 70 years
## 10 Burkina Faso           1952-2007 less than 50 years
## 11 Burundi                1952-2007 less than 50 years
```

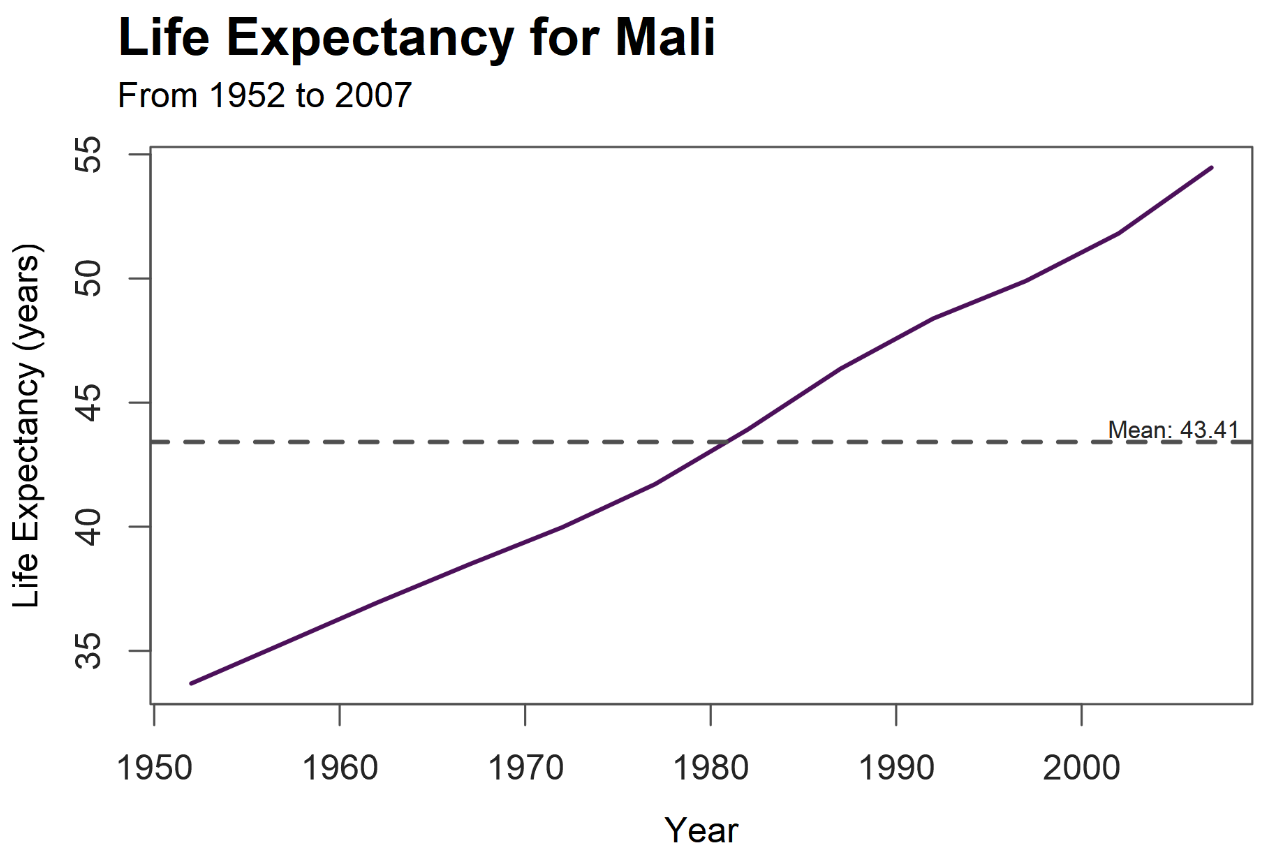
Task 3: Plot `life expectancy/time` for countries starting with "M"

> Challenge/Optional: Write a script that loops over each country in the gapminder dataset, tests whether the country starts with a 'M' and graphs life expectancy against time (using plot() function) as a line graph if the mean life expectancy is under 50 years.

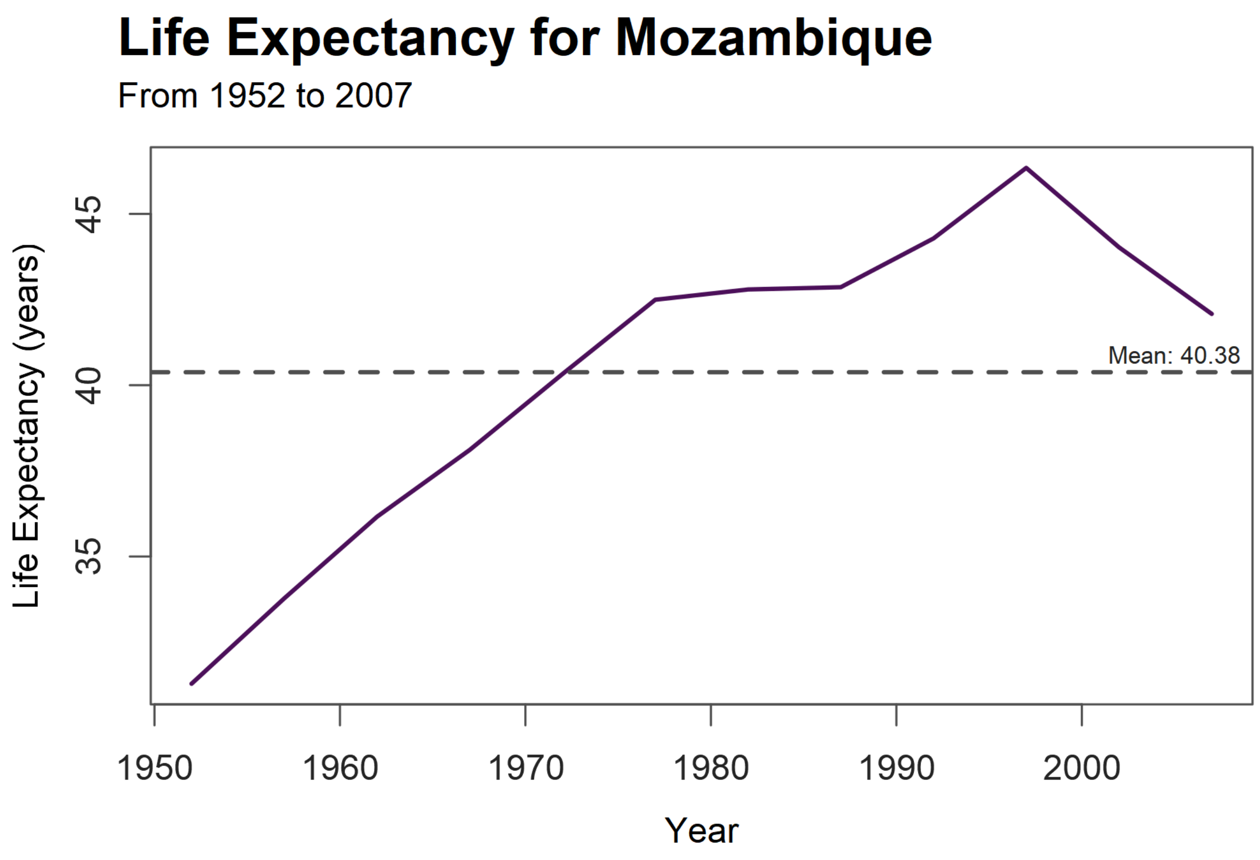
```
# loop through the countries
for (cntry in unique(gapminder$country)) {
    # test if the country starts with a M
    if (!grepl("^M", cntry)) {
        # if not, skip to the next country
        next
    }
    # get the life expectancy
    life_expectancy <- gapminder %>%
        filter(country == cntry) %>%
        pull(lifeExp)
    # calculate the mean life expectancy for the years found
    mean_life_expectancy <- mean(life_expectancy, na.rm = TRUE)
    # test the life expectancy
    if (mean_life_expectancy < 50) {
        # get the range of years
        years <- gapminder %>%
            filter(country == cntry) %>%
            pull(year)
        # plot the life expectancy and make it look nice
        par(bg = "#ffffff")
        plot(years, life_expectancy, type = "l", col = "#4c105a", lwd = 2, cex = 1.2,
            xlab = "", ylab = "", axes = FALSE, ann = FALSE, oma = c(2, 3, 5, 2))
        # add title
        mtext(side = 3, line = 2, at = min(years) - 4, adj = 0, cex = 1.5, col =
         ↪  "#000000",
            paste("Life Expectancy for", cntry), font = 2)
        # add subtitle
        mtext(side = 3, line = 0.75, at = min(years) - 4, adj = 0, cex = 1, col =
         ↪  "#000000",
            paste("From", min(years), "to", max(years)))
        # add the axis labels
        title(xlab = "Year", ylab = "Life Expectancy (years)", col.lab = "#000000",
            mgp = c(2.5, 0.5, 0))
        # add mean line
        abline(h = mean_life_expectancy, col = "#4f4f4f", lwd = 2, lty = 2)
        # add mean life expectancy line label
        text(x = max(years) - 2, y = mean_life_expectancy + 0.5, labels = paste("Mean:",
            round(mean_life_expectancy, 2)), cex = 0.7, col = "#242424")
        box(col = "#4f4f4f")
        axis(1, col = "#4f4f4f", col.axis = "#242424")
        axis(2, col = "#4f4f4f", col.axis = "#242424")
    }
}
```


Life Expectancy for Madagascar
From 1952 to 2007
Life Expectancy (years)
60
55
50
45
40
Mean: 47.77
1950
1960
1970
1980
1990
2000
Year


Life Expectancy for Malawi
From 1952 to 2007
Life Expectancy (years)
36
38
40
42
44
46
48
Mean: 43.35
1950
1960
1970
1980
1990
2000
Year



Life Expectancy for Mali
From 1952 to 2007
Life Expectancy (years)
55
50
45
40
35
Mean: 43.41
1950
1960
1970
1980
1990
2000
Year

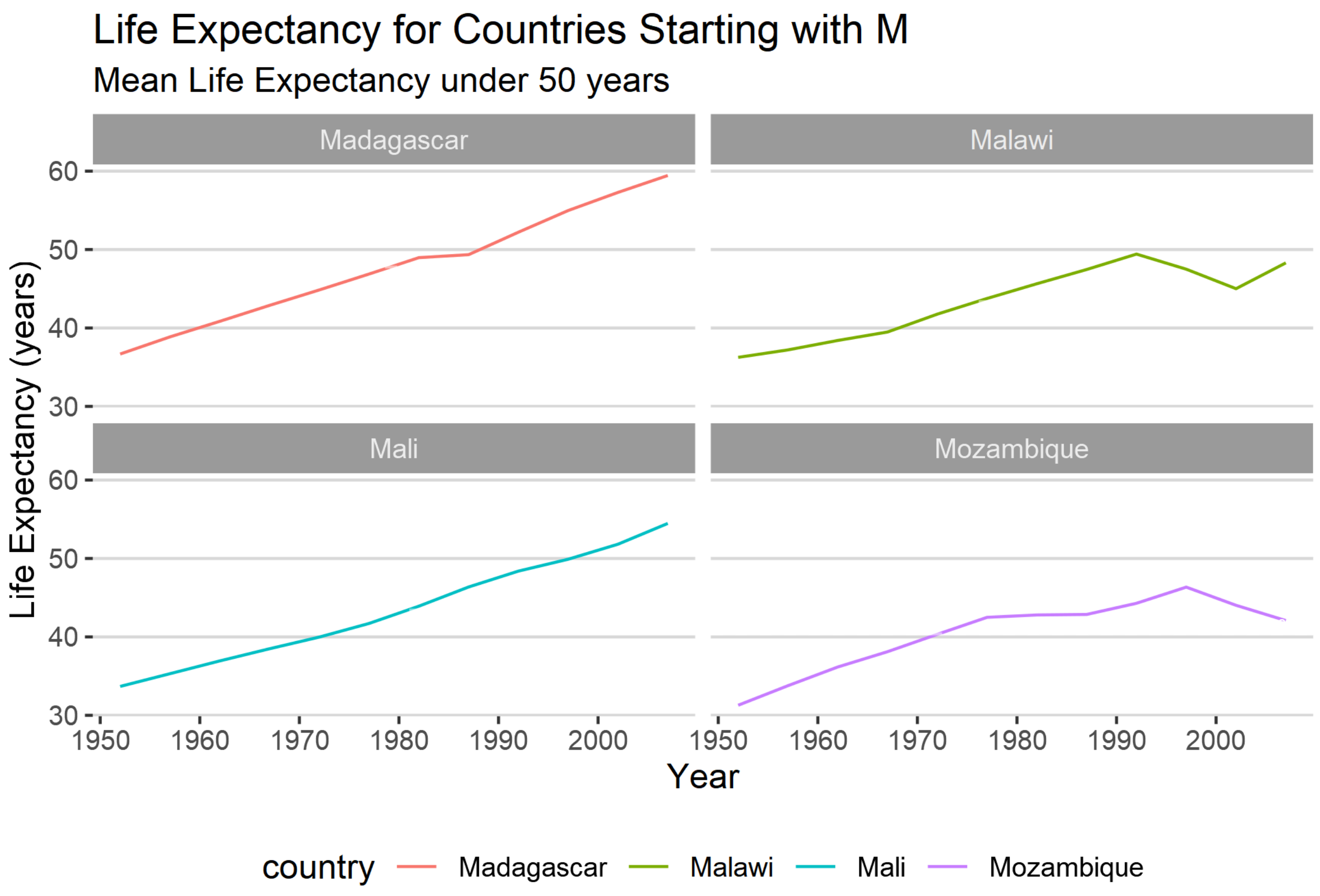


Bonus: Similar, but using `tidyverse` and `ggplot2`

```
# find countries starting with M, and mean life expectancy under 50
m_countries <- gapminder %>%
    filter(grepl("^M", country)) %>%
    group_by(country) %>%
    summarise(mean_life_expectancy = mean(lifeExp, na.rm = TRUE), years =
     ↪  paste0(min(year),
        "-", max(year)), year_min = min(year), year_max = max(year)) %>%
    filter(mean_life_expectancy < 50) %>%
    select(country, years, mean_life_expectancy, year_min, year_max)
# print the countries found and their mean life expectancy
print(m_countries, n = Inf)
```

```
## # A tibble: 4 x 5
##   country    years     mean_life_expectancy year_min year_max
##   <fct>      <chr>                    <dbl>    <int>    <int>
## 1 Madagascar 1952-2007                 47.8     1952     2007
## 2 Malawi     1952-2007                 43.4     1952     2007
## 3 Mali       1952-2007                      43.4      1952      2007
## 4 Mozambique 1952-2007                      40.4      1952      2007
```

```
# plot the life expectancy (using facet wrap)
gapminder %>%
    filter(country %in% m_countries$country) %>%
    ggplot(aes(x = year, y = lifeExp, color = country)) + geom_line() + geom_hline(data
    ↪  = m_countries,
    aes(yintercept = mean_life_expectancy), linetype = "dashed", color = "white",
    alpha = 0.5) + geom_text(data = m_countries, aes(x = year_max, y =
    ↪  mean_life_expectancy +
    1, label = paste("Mean:", round(mean_life_expectancy, 2))), nudge_x = -2.5, color =
    ↪  "white",
    size = 2.25) + theme_hc(style = "default") + theme(strip.background =
    ↪  element_rect(colour = "#00000000",
    fill = "#00000065"), strip.text = element_text(color = "#ececec")) +
    ↪  facet_wrap(~country,
    ncol = 2) + labs(title = "Life Expectancy for Countries Starting with M", subtitle =
    ↪  "Mean Life Expectancy under 50 years",
    x = "Year", y = "Life Expectancy (years)")
```



Assignment 7:W44: Practice Web Scraping

> Use the rvest library to scrape data of your interest (football statistics in Wikipedia?, gender representatives in different governments? global population by country in https://www.worldometers.info/world-population/population-by-country/)

I will look at the population changes since 1950 for the top 10 most populous countries in the world.

First, install a handful of classic R packages and load their libraries:

- `rvest` for web-scraping
- `tidyverse` for data-wrangling, transformation and plots
- `stringr` for string manipulation
- `janitor` for clean headers that your OCD will love you for

```r
library(rvest)
library(tidyverse)
library(stringr)
library(janitor)
library(ggthemes)
```

Scrape the data

First we need to find the top 10 most populous countries.

```r
url <- "https://www.worldometers.info/world-population/population-by-country/"
# scrape the website
parsed_html <- read_html(url)
```

```r
population_table <- parsed_html %>%
    html_elements("table") %>%
    html_table()
# get the actual data from the table
population_table <- population_table[[1]]
# peek at the data
head(population_table)
```

#	Country (or dependency)	Population (2020)	Yearly Change	Net Change	Density (P/Km²)	Land Area (Km²)	Migrants (net)	Fert. Rate	Med. Age	Urban Pop %	World Share
1	China	1,439,323,776	0.39 %	5,540,090	153	9,388,211	-348,399	1.7	38	61 %	18.47 %
2	India	1,380,004,385	0.99 %	13,586,631	464	2,973,190	-532,687	2.2	28	35 %	17.70 %

#	Country (or dependency)	Population (2020)	Yearly Change	Net Change	Density (P/Km²)	Land Area (Km²)	Migrants (net)	Fert. Rate	Med. Age	Urban Pop %	World Share
3	United States	331,002,651	0.59 %	1,937,734	36	9,147,420	954,806	1.8	38	83 %	4.25 %
4	Indonesia	273,523,615	1.07 %	2,898,047	151	1,811,570	-98,955	2.3	30	56 %	3.51 %
5	Pakistan	220,892,340	2.00 %	4,327,022	287	770,880	-233,379	3.6	23	35 %	2.83 %
6	Brazil	212,559,417	0.72 %	1,509,890	25	8,358,140	21,200	1.7	33	88 %	2.73 %

```r
# now limit to the top 10 countries
population_table <- population_table[1:10, ]
# give the headers nice names
population_table <- population_table %>%
    clean_names() %>%
    rename(country = country_or_dependency, population = population_2020)
# rename 'number' column to 'year' and set to 2020
population_table <- population_table %>%
    rename(year = number) %>%
    mutate(year = 2020)
# peek at the cleaned
head(population_table)
```

year	country	population	yearly_change	net_change	density_p_km2	land_area_km2	migrants_net	fert_rate	med_age	urban_pop_percent	world_share
2020	China	1,439,323,776	0.39 %	5,540,090	153	9,388,211	-348,399	1.7	38	61 %	18.47 %
2020	India	1,380,004,385	0.99 %	13,586,631	464	2,973,190	-532,687	2.2	28	35 %	17.70 %
2020	United States	331,002,651	0.59 %	1,937,734	36	9,147,420	954,806	1.8	38	83 %	4.25 %
2020	Indonesia	273,523,615	1.07 %	2,898,047	151	1,811,570	-98,955	2.3	30	56 %	3.51 %
2020	Pakistan	220,892,340	2.00 %	4,327,022	287	770,880	-233,379	3.6	23	35 %	2.83 %
2020	Brazil	212,559,417	0.72 %	1,509,890	25	8,358,140	21,200	1.7	33	88 %	2.73 %

Now we need to get the data for each of the top 10 countries.

```r
add_country_col <- function(x) {
    # extract data
    x <- x %>%
        magrittr::extract2(1) %>%
        clean_names()
    # now get current country name
    country <- colnames(x)[13] %>%
        str_remove("_global_rank")
    # add country column
    x <- x %>%
        select(1:11) %>%
        mutate(country = country)
}
# get the country names
country_names <- population_table$country
# for the links United States is US
country_names <- str_replace(country_names, "United States", "US")
# get the country links
country_links <- country_names %>%
    str_replace_all(" ", "-") %>%
    str_replace_all("\\.", "") %>%
    str_to_lower() %>%
    paste0("https://www.worldometers.info/world-population/", ., "-population/")
population_data_css_selector <- "table.table-list:first-of-type"
# get the population data for each country, add the country name as a column
# and combine into a single data frame, matching columns
population_data <- country_links %>%
    map(read_html) %>%
    map(html_nodes, population_data_css_selector) %>%
    map(html_table, convert = FALSE) %>%
    map_df(add_country_col)
# peek at the data
head(population_data)
```

year	population	yearly_percent_change	yearly_change	migrants_net	median_age	fertility_rate	density_p_km2	urban_pop_percent	urban_population	countrys_share_of_world_	country
2020	1,439,323,776	0.39 %	5,540,090	-348,399	38.4	1.69	153	60.8 %	875,075,919	18.47 %	china
2019	1,433,783,686	0.43 %	6,135,900	-348,399	37.0	1.65	153	59.7 %	856,409,297	18.59 %	china
2018	1,427,647,786	0.47 %	6,625,995	-348,399	37.0	1.65	152	58.6 %	837,022,095	18.71 %	china
2017	1,421,021,791	0.49 %	6,972,440	-348,399	37.0	1.65	151	57.5 %	816,957,613	18.83 %	china
2016	1,414,049,351	0.51 %	7,201,481	-348,399	37.0	1.65	151	56.3 %	796,289,491	18.94 %	china
2015	1,406,847,870	0.57 %	7,607,451	-310,442	36.7	1.64	150	55.1 %	775,352,917	19.06 %	china

```
replace_pct <- function(x) {
    x <- str_replace_all(x, "%", "")
    x <- str_replace_all(x, " ", "")
    x <- as.numeric(x)
}
replace_thousands <- function(x) {
    x <- str_replace_all(x, ",", "")
    x <- as.numeric(x)
}
# now we can convert the data to numeric
population_data <- population_data %>%
    mutate_at(c("yearly_percent_change", "urban_pop_percent",
     ↪  "countrys_share_of_world_pop"),
        replace_pct)
population_data <- population_data %>%
    mutate_at(c("population", "yearly_change", "migrants_net", "median_age",
     ↪  "fertility_rate",
        "density_p_km2", "urban_population"), replace_thousands)
population_data$year <- as.numeric(population_data$year)
# replace u_s to united states
population_data$country <- str_replace(population_data$country, "u_s", "united states")
# make country names start with a capital letter
population_data$country <- str_to_title(population_data$country)
# peek at the data
head(population_data)
```

year	population	yearly_percent_change	yearly_change	migrants_net	median_age	fertility_rate	density_p_km2	urban_pop_percent	urban_population	countrys_share_of_world_pop	country	world_
2020	1.4e+09	0.39	5540090	-348399	38	1.7	153	61	8.8e+08	18	China	
2019	1.4e+09	0.43	6135900	-348399	37	1.6	153	60	8.6e+08	19	China	
2018	1.4e+09	0.47	6625995	-348399	37	1.6	152	59	8.4e+08	19	China	
2017	1.4e+09	0.49	6972440	-348399	37	1.6	151	58	8.2e+08	19	China	
2016	1.4e+09	0.51	7201481	-348399	37	1.6	151	56	8.0e+08	19	China	
2015	1.4e+09	0.55	7607451	-310442	37	1.6	150	55	7.8e+08	19	China	

Write result to file

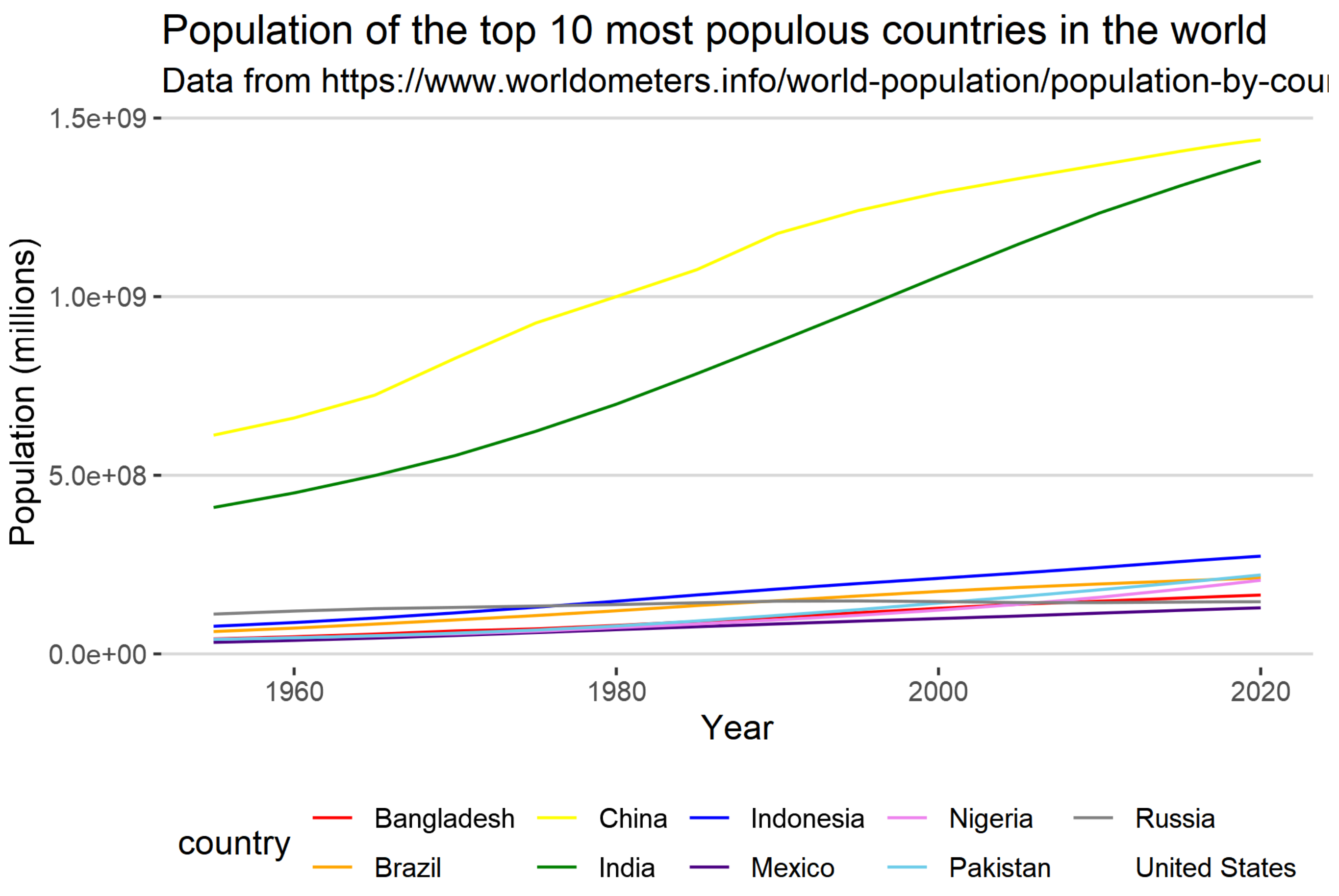
Now we can write the data to a csv file.

```
write_csv(population_data, "data/top10_country_historical_population_data.csv")
```

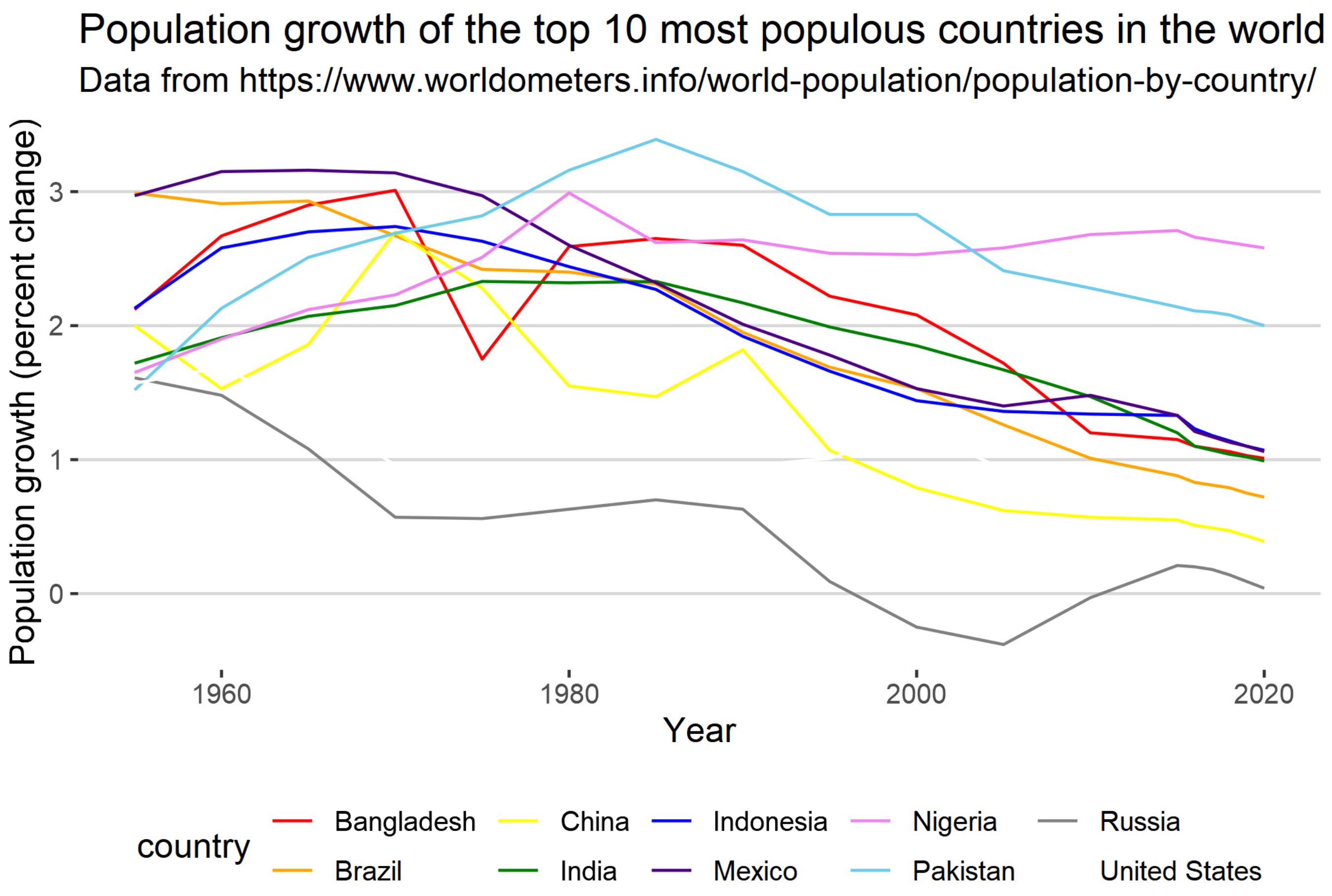
Plot the data

Now we can plot the data.

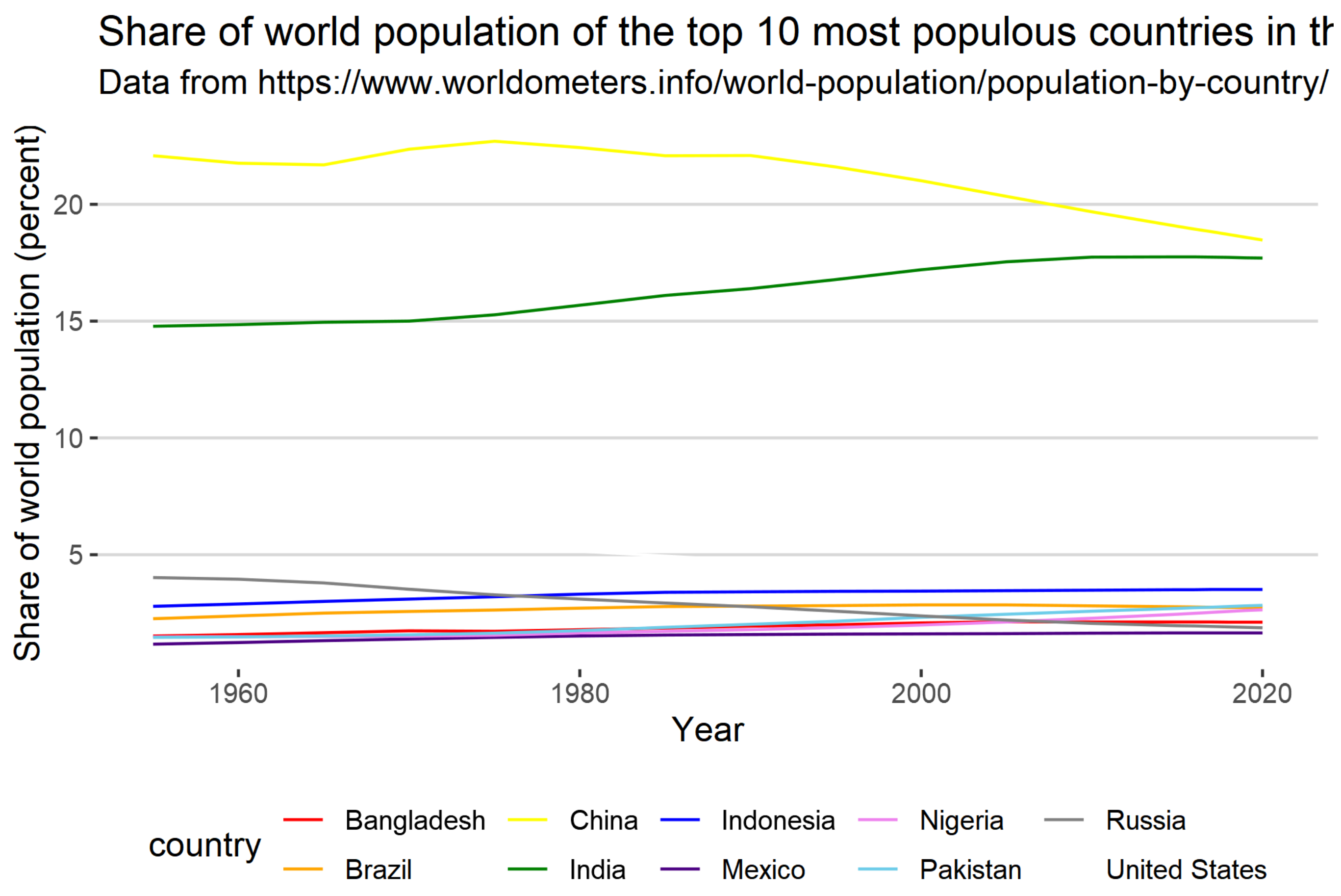
```
# define colors
country_colors <- c("#FF0000", "#FFA500", "#FFFF00", "#008000", "#0000FF", "#4B0082",
    "#EE82EE", "#6ccce9", "#808080", "#FFFFFF")
# overall population
population_data %>%
    ggplot(aes(x = year, y = population, color = country)) + geom_line() +
    ↪ scale_color_manual(values = country_colors) +
    labs(title = "Population of the top 10 most populous countries in the world",
        subtitle = "Data from
        ↪ https://www.worldometers.info/world-population/population-by-country/",
        x = "Year", y = "Population (millions)") + theme_hc(style = "default") +
    theme(legend.position = "bottom")
```



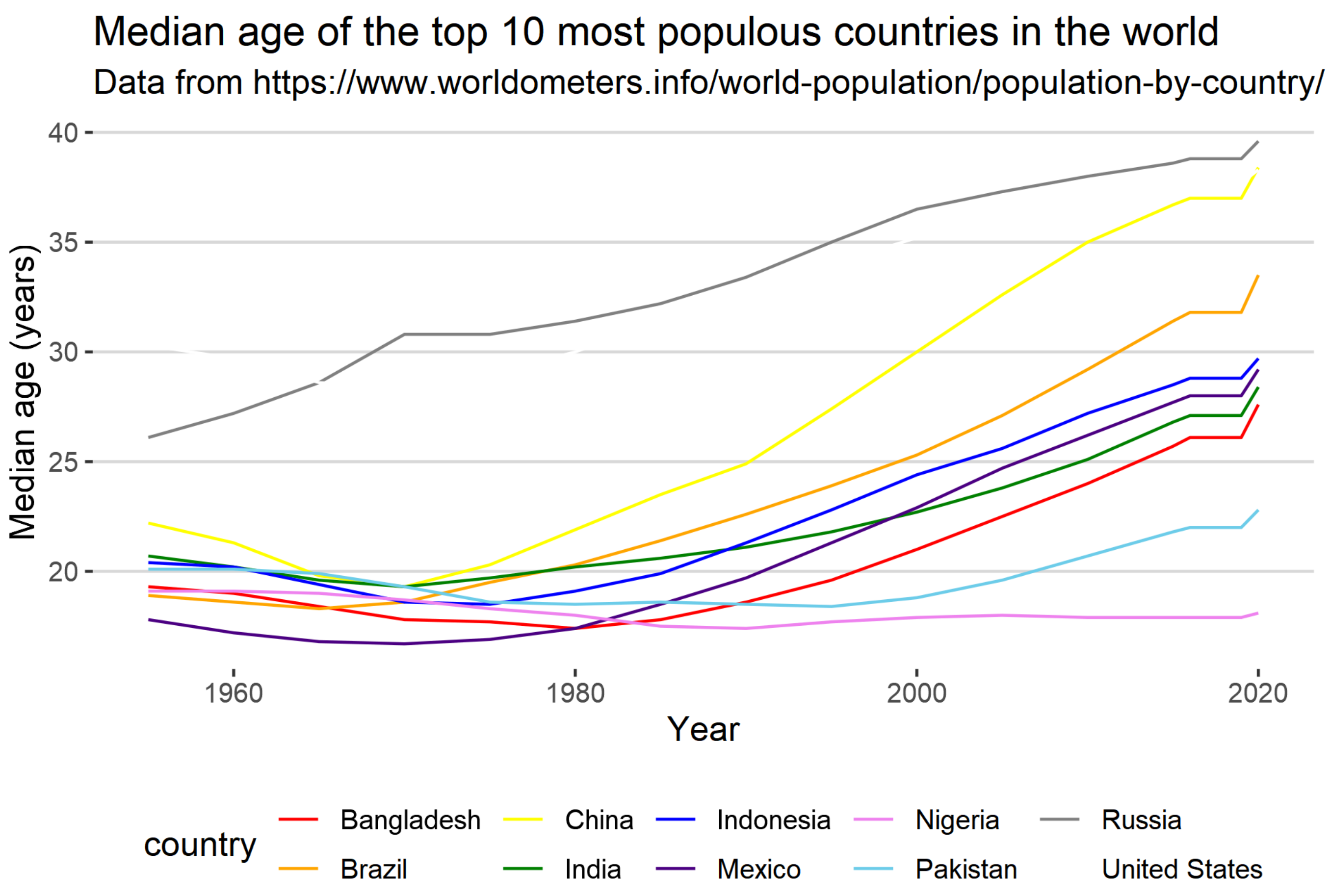
```
# population growth
population_data %>%
    ggplot(aes(x = year, y = yearly_percent_change, color = country)) + geom_line() +
    scale_color_manual(values = country_colors) + labs(title = "Population growth of the
     ↪  top 10 most populous countries in the world",
    subtitle = "Data from
     ↪  https://www.worldometers.info/world-population/population-by-country/",
    x = "Year", y = "Population growth (percent change)") + theme_hc(style = "default") +
    theme(legend.position = "bottom")
```



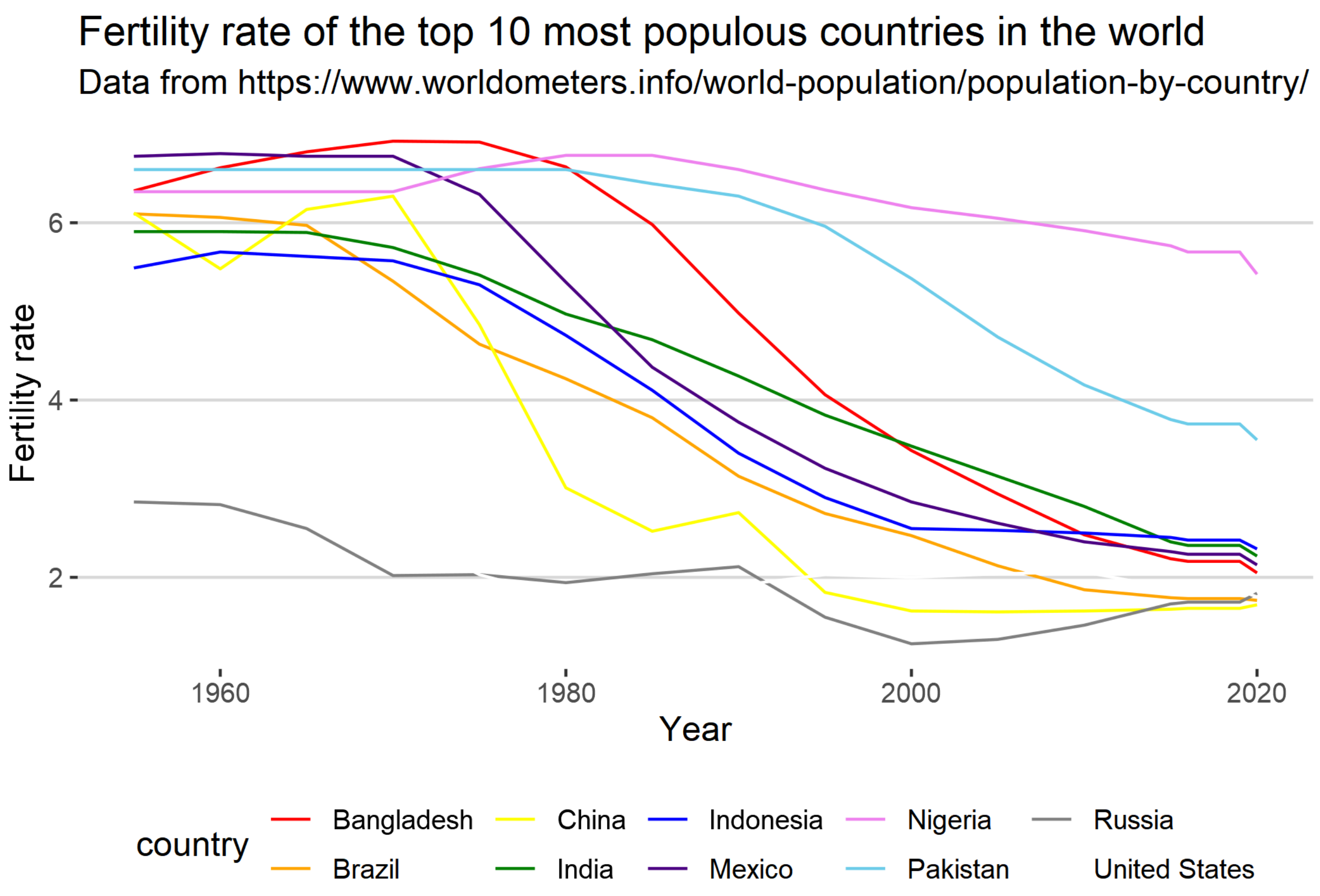
```
# by share of world population
population_data %>%
    ggplot(aes(x = year, y = countrys_share_of_world_pop, color = country)) +
     ↪  geom_line() +
    scale_color_manual(values = country_colors) + labs(title = "Share of world
     ↪  population of the top 10 most populous countries in the world",
    subtitle = "Data from
     ↪  https://www.worldometers.info/world-population/population-by-country/",
    x = "Year", y = "Share of world population (percent)") + theme_hc(style = "default")
     ↪  +
    theme(legend.position = "bottom")
```



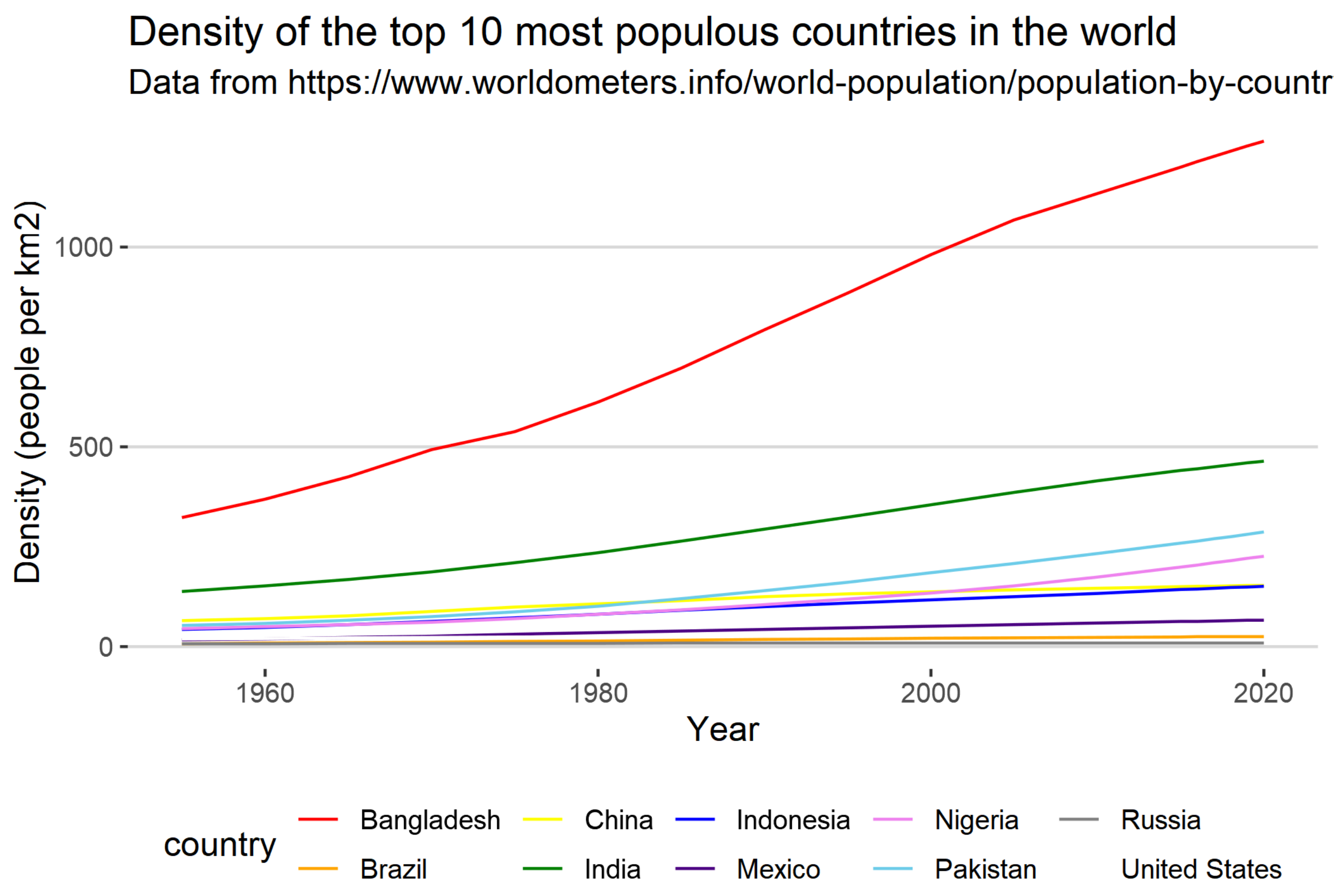
```
# by median age
population_data %>%
    ggplot(aes(x = year, y = median_age, color = country)) + geom_line() +
    ↪ scale_color_manual(values = country_colors) +
    labs(title = "Median age of the top 10 most populous countries in the world",
        subtitle = "Data from
        ↪ https://www.worldometers.info/world-population/population-by-country/",
        x = "Year", y = "Median age (years)") + theme_hc(style = "default") +
        ↪ theme(legend.position = "bottom")
```



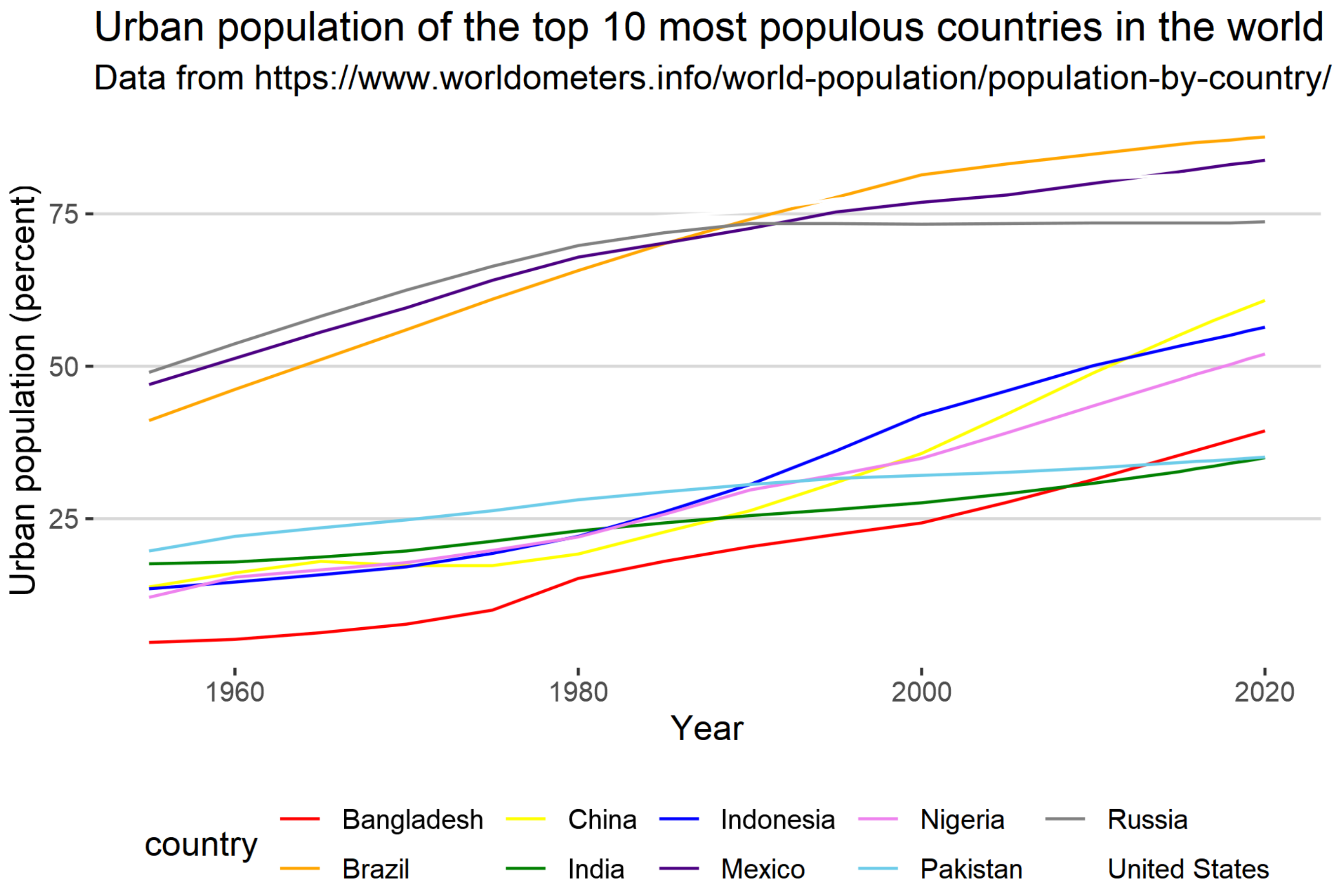
```
# by fertility rate
population_data %>%
    ggplot(aes(x = year, y = fertility_rate, color = country)) + geom_line() +
    ↪  scale_color_manual(values = country_colors) +
    labs(title = "Fertility rate of the top 10 most populous countries in the world",
        subtitle = "Data from
        ↪  https://www.worldometers.info/world-population/population-by-country/",
        x = "Year", y = "Fertility rate") + theme_hc(style = "default") +
        ↪  theme(legend.position = "bottom")
```



```
# by density
population_data %>%
    ggplot(aes(x = year, y = density_p_km2, color = country)) + geom_line() +
    ↪  scale_color_manual(values = country_colors) +
    labs(title = "Density of the top 10 most populous countries in the world", subtitle
    ↪  = "Data from
    ↪  https://www.worldometers.info/world-population/population-by-country/",
       x = "Year", y = "Density (people per km2)") + theme_hc(style = "default") +
    theme(legend.position = "bottom")
```



```r
# by urban population
population_data %>%
    ggplot(aes(x = year, y = urban_pop_percent, color = country)) + geom_line() +
    scale_color_manual(values = country_colors) + labs(title = "Urban population of the
    ↪  top 10 most populous countries in the world",
    subtitle = "Data from
    ↪  https://www.worldometers.info/world-population/population-by-country/",
    x = "Year", y = "Urban population (percent)") + theme_hc(style = "default") +
    theme(legend.position = "bottom")
```



Assignment 8:W45: Sentiment Analysis or Text Mining

Note: for more text analysis, you can fork & work through Casey O'Hara and Jessica Couture's eco-data-sci workshop (available here https://github.com/oharac/text_workshop)

Your task

Taking this script as a point of departure, apply sentiment analysis on the Game of Thrones. You will find a pdf in the data folder. What are the most common meaningful words and what emotions do you expect will dominate this volume? Are there any terms that are similarly ambiguous to the 'confidence' above?

Credits:

This tutorial is inspired by Allison Horst's Advanced Statistics and Data Analysis.

My Work (LR)

Load Game of Thrones PDF

```
page_range <- 6:731
# load the pdf
got_text <- pdf_text(here("assignment_8", "data", "got.pdf"))
# let's get all the main text
got_text <- got_text[page_range]
```

Clean up text

```
# split the pages into lines, move to columns and remove whitespace around
got_df <- data.frame(got_text, page_no = page_range) %>%
    mutate(text_full = str_split(got_text, pattern = "\\n")) %>%
    select(-got_text) %>%
    unnest(text_full) %>%
    mutate(text_full = str_trim(text_full))
# remove empty lines
got_df <- got_df %>%
    filter(nchar(text_full) > 0)
# replace U+2019 with apostrophe
got_df <- got_df %>%
    mutate(text_full = str_replace_all(text_full, "’", "'"))
```

Tokenize text

```
# tokenize the text
got_tokens <- got_df %>%
    unnest_tokens(word, text_full)
# check tokens
got_wc <- got_tokens %>%
    count(word) %>%
    arrange(-n)
head(got_wc)
```

word	n
the	17727
and	8913
to	6566
a	6394

word	n
of	5993
he	5163

```
# remove stopwords
got_tokens <- got_tokens %>%
    anti_join(stop_words)
```

```
## Joining, by = "word"
```

```
# check tokens
got_wc <- got_tokens %>%
    count(word) %>%
    arrange(-n)
head(got_wc)
```

word	n
lord	1297
ser	976
jon	781
ned	743
tyrion	590
eyes	566

```
# no need to remove numeric, count doesn't change got_tokens <- got_tokens %>%
# filter(is.na(as.numeric(word)))
```

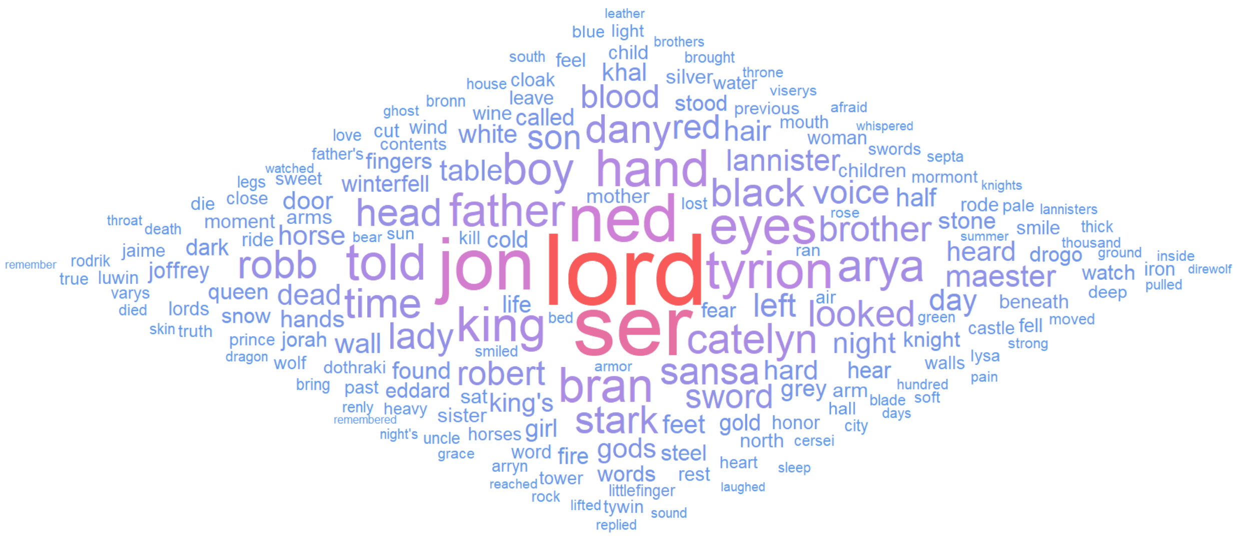
Visualise word counts

```
# Find the number of unique words
length(unique(got_tokens$word))  # 10,958
```

```
## [1] 10981
```

```
# Visualise the top 200 words
got_top200 <- got_tokens %>%
    count(word) %>%
    arrange(-n) %>%
    head(200)
# Create a word cloud
```

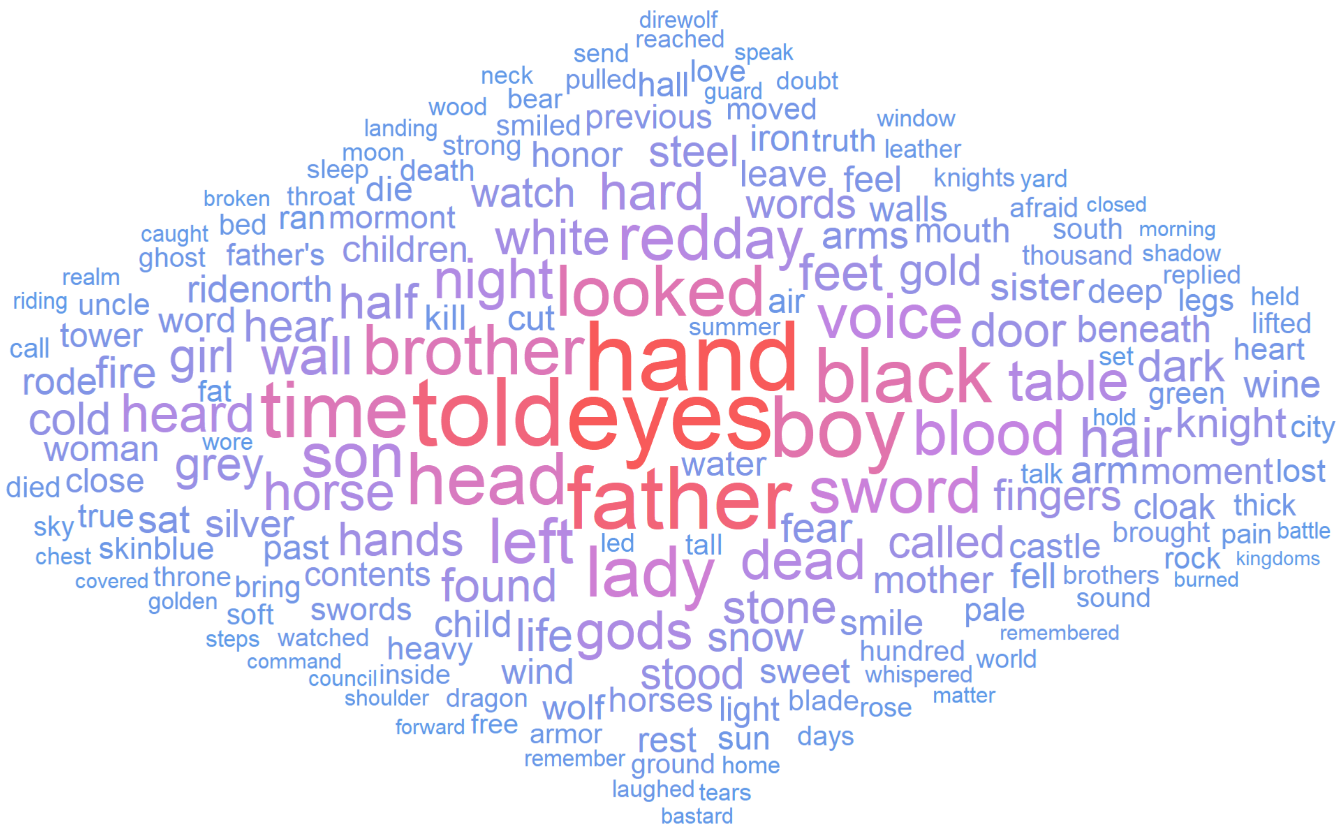
```
ggplot(data = got_top200, aes(label = word, size = n)) +
↪  geom_text_wordcloud_area(aes(color = n),
    shape = "diamond") + scale_size_area(max_size = 12) + scale_color_gradientn(colors =
     ↪  c("#5d99e8",
    "#c683e1", "#f85a5a")) + theme_hc(style = "default")
```



Actually, there are some weird things in there, while it's interesting to see who is mentioned the most, we should get rid of some of the most common titles and names.

```
# remove titles and names, and some weird puncuated stopwords
titles_names_got <- c("arryn", "arya", "bran", "brienne", "bronn", "catelyn", "cersei",
    "daenerys", "dany", "dothraki", "drogo", "eddard", "grace", "house", "jaime",
    "joffrey", "jon", "jorah", "jory", "khal", "king", "king's", "lannister",
     ↪  "lannisters",
    "littlefinger", "lord", "lords", "luwin", "lysa", "maester", "margaery",
     ↪  "melisandre",
    "ned", "night's", "petyr", "prince", "pycelle", "queen", "renly", "riverrun",
    "robb", "robert", "rodrik", "sam", "samwell", "sansa", "septa", "ser", "stannis",
    "stark", "theon", "tyrion", "tywin", "varys", "viserys", "winterfell")
got_tokens <- got_tokens %>%
    filter(!word %in% titles_names_got)
```

```
# get the new top 200 words
got_top200 <- got_tokens %>%
    count(word) %>%
    arrange(-n) %>%
    head(200)
# Create a word cloud
ggplot(data = got_top200, aes(label = word, size = n)) +
↪  geom_text_wordcloud_area(aes(color = n),
    shape = "diamond") + scale_size_area(max_size = 12) + scale_color_gradientn(colors =
    ↪  c("#5d99e8",
    "#c683e1", "#f85a5a")) + theme_hc(style = "default")
```



That looks better, I'm sure there are some we missed but, it's ok :)

Sentiment analysis

Now we will do sentiment analysis using NRC

```
# load the NRC lexicon
nrc <- get_sentiments("nrc")
```

```
# get the sentiment of the words
got_sentiment <- got_tokens %>%
    inner_join(nrc, by = c(word = "word"))
# check excluded got_tokens
got_exclude <- got_tokens %>%
    anti_join(nrc)
```

```
## Joining, by = "word"
```

```
# Count to find the most excluded:
got_exclude_n <- got_exclude %>%
    count(word, sort = TRUE)
head(got_exclude_n)
```

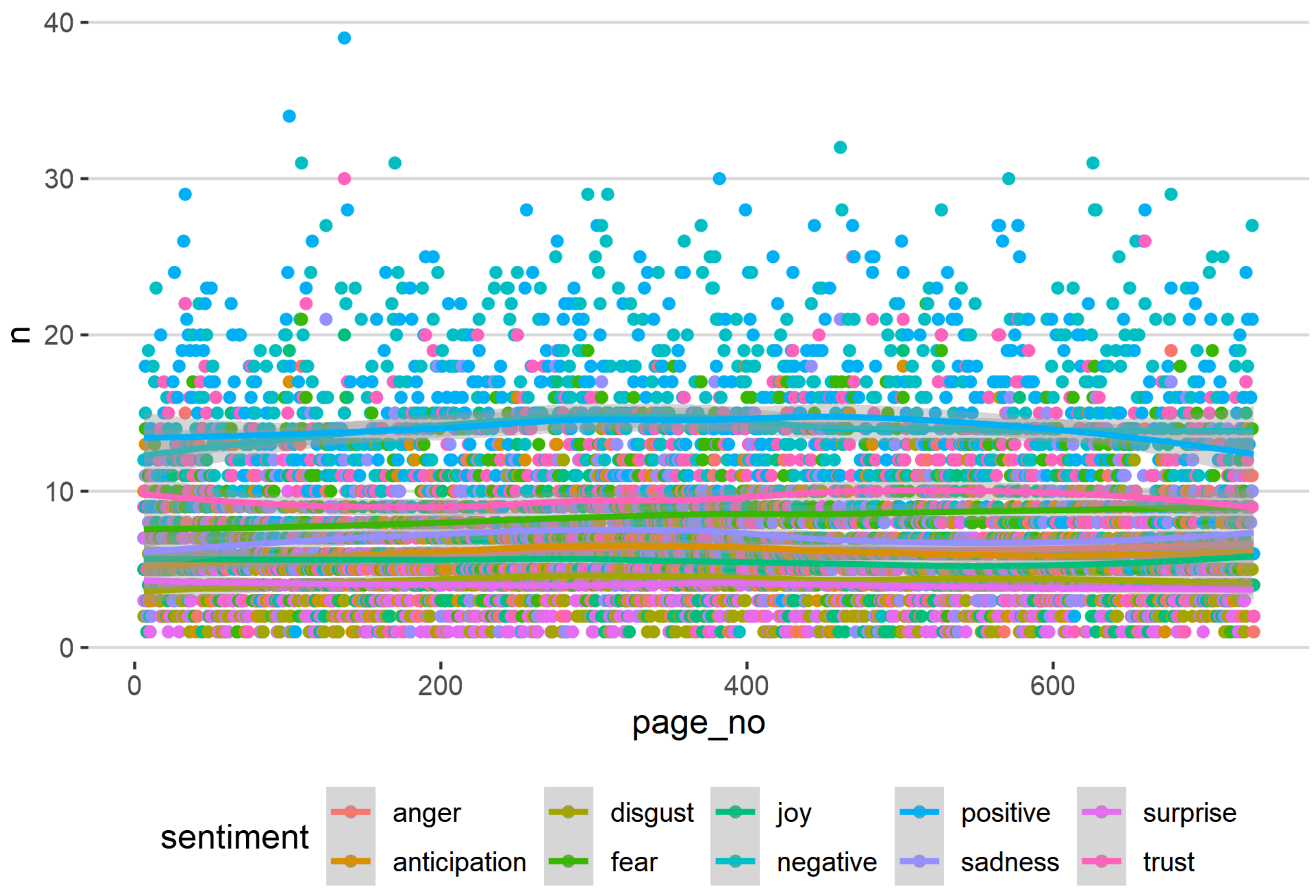
word	n
eyes	566
hand	566
told	504
boy	416
black	412
looked	405

```
# check the sentiment
got_sentiment %>%
    count(sentiment) %>%
    arrange(-n)
```

sentiment	n
positive	9995
negative	9957
trust	6701
fear	5873
sadness	4937
anger	4242
anticipation	4215
joy	3829
disgust	2885
surprise	2639

```
# get the sentiment of the text, by page and unnest the sentiment
got_sentiment_page <- got_sentiment %>%
    group_by(page_no) %>%
    summarise(sentiment = paste(sentiment, collapse = " ")) %>%
    unnest_tokens(sentiment, sentiment)
# plot the sentiment, over the course of the book for each sentiment
got_sentiment_page %>%
    count(page_no, sentiment) %>%
    ggplot(aes(x = page_no, y = n, color = sentiment)) + geom_point() + geom_smooth() +
    theme_hc(style = "default")
```

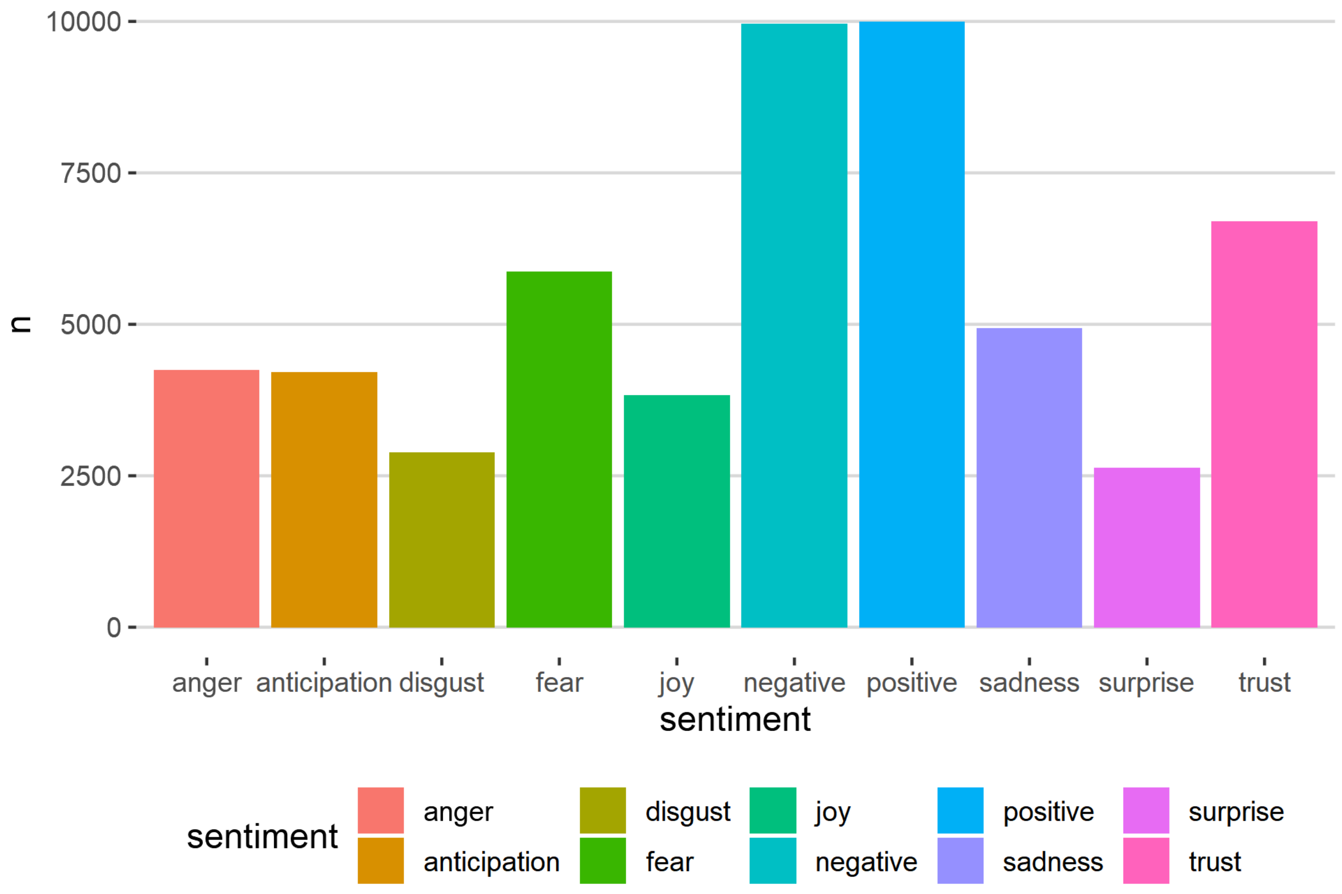
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



interesting, it seems that the sentiment across the book is fairly selectable

Finally, let's do an plot of the overall sentiments

```
got_sentiment %>%
    count(sentiment) %>%
    ggplot(aes(x = sentiment, y = n, fill = sentiment)) + geom_col() + theme_hc(style =
    ↪  "default")
```



Seems like there's overall an almost equal number of positive and negative words.

The highest categories are trust, fear and then sadness.