Connectivity between online behavior and crypto evaluations.

# Introduction

# Portfolios:

## **Portfolio 1**

**DESCRIPTION**

Upload your answers/solutions to the problems below. Beware of making the submission legible and understandable to another reader:

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 ?

**Answer:**

To find all dates: (\d+).(\d+)..(\d+)

To substitute: $3 $1 $2

1. 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)

**Answer:**

First (")|(,) replace all “ and , with nothing.

Take the text and find all white spaces \s and replace with \n

To get back.

\W replace “, “

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

**Answer:**

Have a data documentation sheet that describes each variable, its variable type, what does the variable represent i.e., what does it measure. Be concise and have a standardized way to do your data documentation so it is easily readable. Tidy format i.e., each row represents an observation, and each column corresponds to a distinct variable. Encode your variables correctly, binary, and categorical variables as factors, integers as discrete and floats as continuous etc. Remember to document whether your data is in a wide or long format. You can change these formats by functions such as pivot\_longer or pivot\_wider. Code your missing values in the same manner i.e., don’t change style. Check if dates are coded uniformly or whether they are in different formats

## **Portfolio 2:**

Portfolio 2

Sigurd Sørensen

2022-08-31

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.0 ✔ purrr 0.3.5   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.2.1 ✔ stringr 1.4.1   
## ✔ readr 2.1.3 ✔ forcats 0.5.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

## 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) Does OpenRefine alter the raw data during sorting and filtering? - You never change the raw data but it saves a log of each new itteration and all its changes. (But I like to work in R xD xD xD )

kings <- c("Margrethe 2.  
Siden 1972  
  
Frederik 9.  
1947 - 1972  
  
Christian 10.  
1912-1947  
  
Frederik 8.  
1906-1912  
  
Christian 9.  
1863-1906  
  
Frederik 7.  
1848-1863  
  
Christian 8.  
1839-1848  
  
Frederik 6.  
1808-1839  
  
Christian 7.  
1766-1808  
  
Frederik 5.  
1746-1766  
  
Christian 6.  
1730-1746  
  
Frederik 4.  
1699-1730  
  
Christian 5.  
1670-1699  
  
Frederik 3.  
1648-1670  
  
Christian 4.  
1588-1648  
  
Frederik 2.  
1559-1588  
  
Christian 3.  
1536-1559  
  
Interregnum  
1533-1536  
  
Frederik 1.  
1523-1533  
  
Christian 2.  
1513-1523  
  
Hans  
1482-1513  
  
Christian 1.  
1448-1481  
  
Christoffer 3. af Bayern  
1440-1448  
  
Erik 7. af Pommern  
1396-1439  
  
Margrete 1.  
1387-1396  
  
Oluf 2.  
1375-1387  
  
Valdemar 4. Atterdag  
1340-1375  
  
Interregnum  
1332-1340  
  
Christoffer 2.  
1329-1332  
  
Valdemar 3.  
1326-1329  
  
Christoffer 2.  
1319-1326  
  
Erik 6. Menved  
1286-1319  
  
Erik 5. Klipping  
1259-1286  
  
Christoffer 1.  
1252-1259  
  
Abel  
1250-1252  
  
Erik 4. Plovpenning  
1241-1250  
  
Valdemar 2. Sejr  
1202-1241  
  
Knud 4.  
1182-1202  
  
Valdemar 1. den Store  
1157-1182  
  
Svend 3., Knud 3., Valdemar 1.  
1146-1157  
  
Erik 3. Lam  
1137-1146  
  
Erik 2. Emune  
1134-1137  
  
Niels  
1104-1134  
  
Erik 1. Ejegod  
1095-1103  
  
Oluf 1. Hunger  
1086-1095  
  
Knud 2. den Hellige  
1080-1086  
  
Harald 3. Hen  
1074-1080  
  
Svend 2. Estridsen  
1047-1074  
  
Magnus den Gode  
1042-1047  
  
Hardeknud  
1035-1042  
  
Knud 1. den Store  
1018-1035  
  
Harald 2.  
1014-1018  
  
Svend 1. Tveskæg  
D. 1014  
  
Harald 1. Blåtand  
D. senest 987  
  
Gorm den Gamle.  
936, d. ca. 958")  
  
  
kings <- str\_split(kings,pattern = "\n")

We do a bit of cleaning magic.

df\_kings <- data.frame(Names = kings[[1]][seq(from = 1, to = length(kings[[1]]), by = 3 )], Year = kings[[1]][seq(from = 2, to = length(kings[[1]]), by = 3 )])  
df\_kings

## Names Year  
## 1 Margrethe 2. Siden 1972  
## 2 Frederik 9. 1947 - 1972  
## 3 Christian 10. 1912-1947  
## 4 Frederik 8. 1906-1912  
## 5 Christian 9. 1863-1906  
## 6 Frederik 7. 1848-1863  
## 7 Christian 8. 1839-1848  
## 8 Frederik 6. 1808-1839  
## 9 Christian 7. 1766-1808  
## 10 Frederik 5. 1746-1766  
## 11 Christian 6. 1730-1746  
## 12 Frederik 4. 1699-1730  
## 13 Christian 5. 1670-1699  
## 14 Frederik 3. 1648-1670  
## 15 Christian 4. 1588-1648  
## 16 Frederik 2. 1559-1588  
## 17 Christian 3. 1536-1559  
## 18 Interregnum 1533-1536  
## 19 Frederik 1. 1523-1533  
## 20 Christian 2. 1513-1523  
## 21 Hans 1482-1513  
## 22 Christian 1. 1448-1481  
## 23 Christoffer 3. af Bayern 1440-1448  
## 24 Erik 7. af Pommern 1396-1439  
## 25 Margrete 1. 1387-1396  
## 26 Oluf 2. 1375-1387  
## 27 Valdemar 4. Atterdag 1340-1375  
## 28 Interregnum 1332-1340  
## 29 Christoffer 2. 1329-1332  
## 30 Valdemar 3. 1326-1329  
## 31 Christoffer 2. 1319-1326  
## 32 Erik 6. Menved 1286-1319  
## 33 Erik 5. Klipping 1259-1286  
## 34 Christoffer 1. 1252-1259  
## 35 Abel 1250-1252  
## 36 Erik 4. Plovpenning 1241-1250  
## 37 Valdemar 2. Sejr 1202-1241  
## 38 Knud 4. 1182-1202  
## 39 Valdemar 1. den Store 1157-1182  
## 40 Svend 3., Knud 3., Valdemar 1. 1146-1157  
## 41 Erik 3. Lam 1137-1146  
## 42 Erik 2. Emune 1134-1137  
## 43 Niels 1104-1134  
## 44 Erik 1. Ejegod 1095-1103  
## 45 Oluf 1. Hunger 1086-1095  
## 46 Knud 2. den Hellige 1080-1086  
## 47 Harald 3. Hen 1074-1080  
## 48 Svend 2. Estridsen 1047-1074  
## 49 Magnus den Gode 1042-1047  
## 50 Hardeknud 1035-1042  
## 51 Knud 1. den Store 1018-1035  
## 52 Harald 2. 1014-1018  
## 53 Svend 1. Tveskæg D. 1014  
## 54 Harald 1. Blåtand D. senest 987  
## 55 Gorm den Gamle. 936, d. ca. 958

## **2**

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?”

df\_random <- read\_csv("SAFI\_openrefine.csv")

## Rows: 131 Columns: 62  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (45): interview\_date, province, district, ward, village, agr\_assoc, rem...  
## dbl (15): quest\_no, years\_farm, no\_membrs, \_members\_count, years\_liv, build...  
## dttm (2): start, end  
##   
## 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.

df\_random\_long <- df\_random %>%   
 separate\_rows(months\_no\_water, sep = ";") %>%   
 filter(months\_no\_water != "NULL")   
  
#Delete non-alphanumeric.  
df\_random\_long <- mutate(df\_random\_long, months\_no\_water = str\_replace\_all(df\_random\_long$months\_no\_water, "[^[:alnum:]]", ""))

df\_random\_long$months\_no\_water

## [1] "Aug" "Sept" "Aug" "Sept" "Oct" "Sept" "Oct" "Oct" "Nov" "Sept"  
## [11] "Oct" "Nov" "Aug" "Sept" "Oct" "Nov" "Oct" "Sept" "Oct" "Nov"   
## [21] "Oct" "Sept" "Oct" "Aug" "Sept" "Oct" "Nov" "Aug" "Sept" "Oct"   
## [31] "Nov" "Sept" "Oct" "Nov" "Sept" "Oct" "Aug" "Sept" "Oct" "Oct"   
## [41] "Nov" "Sept" "Oct" "Nov" "Sept" "Oct" "Sept" "Oct" "Aug" "Sept"  
## [51] "Sept" "Oct" "Aug" "Sept" "Oct" "Sept" "Oct" "Nov" "Sept" "Oct"   
## [61] "Sept" "Oct" "Nov" "Aug" "Sept" "Oct" "Aug" "Sept" "Oct" "Nov"   
## [71] "Dec" "Sept" "Oct" "Aug" "Sept" "Oct" "Aug" "Sept" "Oct" "Sept"  
## [81] "Oct" "Nov" "Aug" "Sept" "Oct" "Oct" "Nov" "Nov" "Dec" "Nov"   
## [91] "Dec" "Aug" "Sept" "Sept" "Oct" "Nov" "Nov" "Sept" "Oct" "Nov"   
## [101] "Oct" "Nov" "Oct" "Nov" "Aug" "Sept" "Oct" "Nov" "Sept" "Oct"   
## [111] "Sept" "Oct" "Nov" "Sept" "Oct" "Sept" "Oct" "Nov" "Sept" "Oct"   
## [121] "Nov" "Oct" "Nov" "Aug" "Sept" "Oct" "Nov" "Jan" "Dec" "Aug"   
## [131] "Sept" "Oct" "Nov" "Dec" "Aug" "Sept" "Oct" "Nov" "Sept" "Oct"   
## [141] "Nov" "Sept" "Oct" "Aug" "Sept" "Oct" "Sept" "Oct" "Sept" "Oct"   
## [151] "Sept" "Oct" "Nov" "Aug" "Sept" "Oct" "Nov" "Aug" "Sept" "Oct"   
## [161] "Aug" "Sept" "Sept" "Oct" "Nov" "Aug" "Sept" "Aug" "Sept" "Oct"   
## [171] "Nov" "Aug" "Sept" "Oct" "Nov" "Aug" "Sept" "Oct" "Nov" "Sept"  
## [181] "Oct" "Aug" "Sept" "Oct" "Nov" "Sept" "Oct" "Nov" "Sept" "Oct"   
## [191] "Nov" "Sept" "Oct" "Nov" "Apr" "May" "June" "July" "Aug" "Sept"  
## [201] "Oct" "Nov" "Jan" "Dec" "Aug" "Sept" "Oct" "Nov" "Dec" "Aug"   
## [211] "Sept" "Oct" "July" "Aug" "Sept" "Oct" "Nov" "Dec" "Sept" "Oct"   
## [221] "Nov" "Dec" "Oct" "Nov" "Aug" "Sept" "Oct" "Nov" "Dec" "Sept"  
## [231] "Oct" "Nov" "Sept" "Nov" "Sept" "Oct" "Nov" "Oct" "Nov" "Dec"   
## [241] "Aug" "Sept" "Oct" "Sept" "Oct" "Nov"

unique(df\_random\_long$months\_no\_water) #Sanity check to see if all is tidy.

## [1] "Aug" "Sept" "Oct" "Nov" "Dec" "Jan" "Apr" "May" "June" "July"

df\_random\_long %>%   
 filter(months\_no\_water != "NULL") %>%   
 count(months\_no\_water) %>%   
 arrange(desc(n))

## # A tibble: 10 × 2  
## months\_no\_water n  
## <chr> <int>  
## 1 Oct 74  
## 2 Sept 70  
## 3 Nov 51  
## 4 Aug 33  
## 5 Dec 11  
## 6 Jan 2  
## 7 July 2  
## 8 Apr 1  
## 9 June 1  
## 10 May 1

October and September is the two months reported the most times as being without water.

## 3

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)

df\_aarhus <- read\_csv("https://raw.githubusercontent.com/aarhusstadsarkiv/datasets/master/censuses/1801/census-1801-normalized.csv")

## Warning: One or more parsing issues, call `problems()` on your data frame for details,  
## e.g.:  
## dat <- vroom(...)  
## problems(dat)

## Rows: 44559 Columns: 17  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (11): sogn, amt, lokalitet, bygning, fnavn, enavn, koen, famstand, civil...  
## dbl (6): ft, id, loknr, famnr, alder, giftnr  
##   
## 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.

df\_aarhus %>%   
 filter(!is.na(erhverv)) %>%   
 count(erhverv) %>%   
 head()

## # A tibble: 6 × 2  
## erhverv n  
## <chr> <int>  
## 1 ?? 1  
## 2 [Snedker] 1  
## 3 [Vanfør, nyder Almisse] 1  
## 4 2. lectie hører 1  
## 5 3. lectie hører 1  
## 6 4. lectie hører 1

The data is still super messy so let us try and clean it up a bit.

df\_aarhus <- df\_aarhus %>%   
 filter(!is.na(erhverv))  
  
df\_aarhus %>%   
 mutate(erhverv = str\_to\_lower(gsub('[[:punct:]]', '', df\_aarhus$erhverv))) %>%  
 count(erhverv) %>%   
 filter(n > 3) %>%   
 arrange(desc(n)) %>%   
 head()

## # A tibble: 6 × 2  
## erhverv n  
## <chr> <int>  
## 1 bonde og gaardbeboer 2012  
## 2 huusmand med jord 749  
## 3 bonde og gårdbeboer 223  
## 4 soldat ved 1 jyske inf reg 136  
## 5 nyder ophold af gaarden 113  
## 6 nationalsoldat 110

We could now stem all words and use a danish stopword list to clean it even further. That requires a library and a stopword list that works with the danish languages.

## **Portfolio 3**

Portfolio 3

Sigurd Sørensen

2022-08-31

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.0 ✔ purrr 0.3.5   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.2.1 ✔ stringr 1.4.1   
## ✔ readr 2.1.3 ✔ forcats 0.5.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

Instructions: For this assignment, you need to answer a couple questions with code and then take a screenshot of your working environment. Submit the solutions including the URL to the screenshot in a doc/pdf to Brightspace.

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)

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)  
  
c(rooms > 2, complete.cases(rooms))

## [1] FALSE FALSE TRUE TRUE FALSE TRUE FALSE NA TRUE FALSE TRUE FALSE  
## [13] FALSE NA FALSE TRUE TRUE FALSE TRUE NA FALSE TRUE FALSE FALSE  
## [25] FALSE TRUE FALSE TRUE TRUE NA TRUE TRUE TRUE TRUE TRUE TRUE  
## [37] TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE  
## [49] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE

rooms[rooms >2 & !is.na(rooms)] %>%   
 length()

## [1] 12

rooms[rooms >2 & !is.na(rooms)] %>%   
 sum()

## [1] 55

**Answer:** 12 rooms > 2 and their sum is 55.

1. What type of data is in the ‘rooms’ vector?

class(rooms)

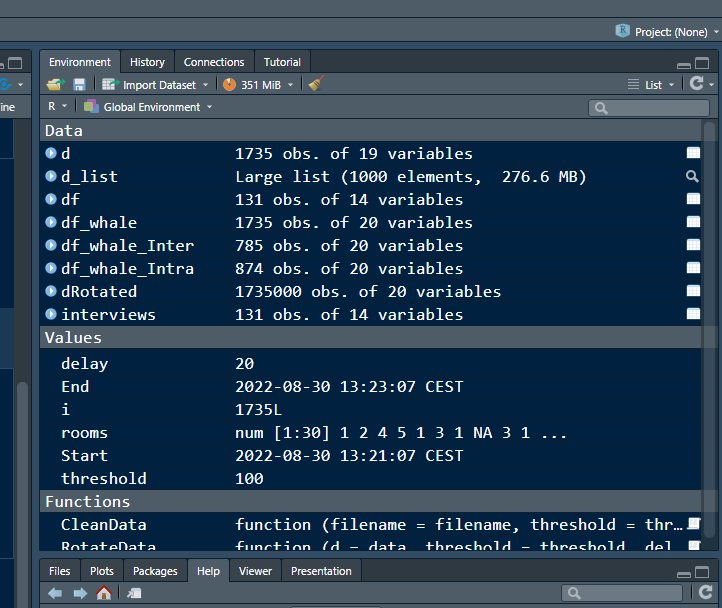
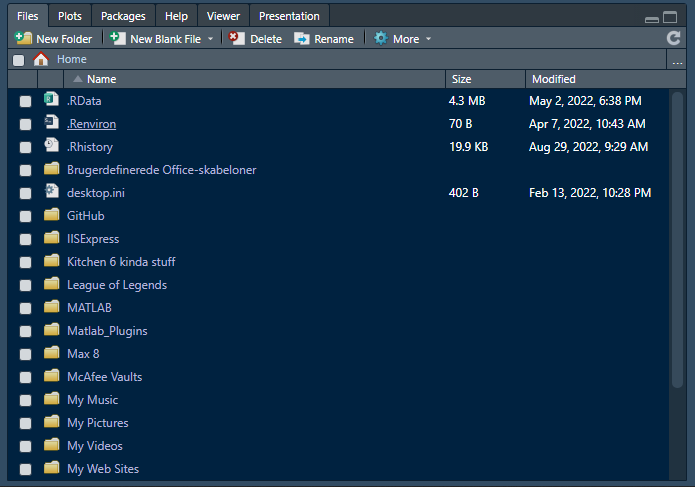
## [1] "numeric"

**Answer** Numeric but more specifically integers.

1. 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
2. the line of code you used to create the object,

interviews <- read\_csv("SAFI\_clean.csv")

## Rows: 131 Columns: 14  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (7): village, respondent\_wall\_type, memb\_assoc, affect\_conflicts, items...  
## dbl (6): key\_ID, no\_membrs, years\_liv, rooms, liv\_count, no\_meals  
## dttm (1): interview\_date  
##   
## 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.

1. the ‘interviews’ object in the Environment, and 
2. 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. 
3. 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 monarchs ruling the longest. How many days did they rule (accounting for transition year?)

The data uses , delimeters so an easy solution would be to use read\_csv2() which be default uses , delimters.

## ℹ Using "','" as decimal and "'.'" as grouping mark. Use `read\_delim()` for more control.

## Rows: 47 Columns: 4  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ";"  
## chr (2): Kings, Yearasruler  
## dbl (2): Start\_date, End\_date  
##   
## 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.

But a way more fun and hacky way is this way too complicated work around. :)

## Rows: 47 Columns: 1  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (1): Kings;Start\_date;End\_date;Yearasruler  
##   
## 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.

## # A tibble: 6 × 1  
## `Kings;Start\_date;End\_date;Yearasruler`  
## <chr>   
## 1 "Gorm den Gamle;NA;NA;Unknown"   
## 2 "Harald 1. Bl\xe5tand ;NA;NA;Unknown"   
## 3 "Svend 1. Tvesk\xe6g ;NA;NA;Unknown"   
## 4 "Harald 2.;1014;1018;4"   
## 5 "Knud 1. den Store;1018;1035;17"   
## 6 "Hardeknud;1035;1042;7"

This is not readable or useful at the moment. We will have to clean it up.

newnames <- strsplit(colnames(df\_kings), split = ";")[[1]]  
  
df\_kings <- df\_kings %>%   
 separate(colnames(df\_kings)[1], newnames, sep = ";") %>%  
 filter(Yearasruler != "Unknown") %>%   
 mutate\_at(c("Start\_date", "End\_date", "Yearasruler"), as.numeric)  
  
head(df\_kings)

## # A tibble: 6 × 4  
## Kings Start\_date End\_date Yearasruler  
## <chr> <dbl> <dbl> <dbl>  
## 1 "Harald 2." 1014 1018 4  
## 2 "Knud 1. den Store" 1018 1035 17  
## 3 "Hardeknud" 1035 1042 7  
## 4 "Magnus den Gode " 1042 1047 5  
## 5 "Svend 2. Estridsen" 1047 1074 27  
## 6 "Harald 3. Hen " 1074 1080 6

Voila a few magic tricks and we get the data in tidyformat. We can now run our stats on it.

summary(df\_kings$Yearasruler)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.00 7.75 14.00 18.68 29.25 60.00

df\_kings %>%   
 arrange(desc(Yearasruler))

## # A tibble: 44 × 4  
## Kings Start\_date End\_date Yearasruler  
## <chr> <dbl> <dbl> <dbl>  
## 1 "Christian 4. " 1588 1648 60  
## 2 "Erik 7. af Pommern" 1396 1439 43  
## 3 "Christian 7. " 1766 1808 42  
## 4 "Valdemar 2. Sejr " 1202 1241 39  
## 5 "Erik 6. Menved" 1286 1319 35  
## 6 "Valdemar 4. Atterdag " 1340 1375 35  
## 7 "Chrstian 1." 1448 1481 33  
## 8 "Hans " 1482 1513 31  
## 9 "Frederik 4. " 1699 1730 31  
## 10 "Frederik 6. " 1808 1839 31  
## # … with 34 more rows

sort(df\_kings$Yearasruler, TRUE)[1:3]

## [1] 60 43 42

## **Portfolio 4**

5:W35: Managing Files on Steroids with Shell

**DESCRIPTION**

Your supervisor has shared a [folder of photos on Sciencedata.dk](https://sciencedata.dk/shared/9ae5ccab83576b9c38d5852e7aab65a3)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)

**Answer:**

$ wc -c \*.JPG | sort -rn | head -4

We need to use head -4 because the first line is the total size of all files.

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?

**Answer**

$ wc -c \*.JPG | sort -rn > ../size.txt

$ cat size.txt | grep -w 0 > empty\_files.txt

3) **Optional/Advanced:** Imagine you have a directory [goodphotos/](https://sciencedata.dk/shared/16112a12cc9f57ef697d4502448a3e60?download" \t "_blank) (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?

**Answer:**  We can use the find command to find all files in (.) current directory which match (-size) = 0c (0c denotes file size should be 0). And all the files will be deleted with the (-delete) command.

* find . -size 0c -delete
* find. -size -0c (to check if all the files were removed.

## **Portfolio 5**

HW5 - Make Data Move

05/10/2020

# Explore global development with R

Today, you will load a filtered gapminder dataset - with a subset of data on global development from 1952 - 2007 in increments of 5 years - to capture the period between the Second World War and the Global Financial Crisis.

**Your task: Explore the data and visualise it in both static and animated ways, providing answers and solutions to 7 questions/tasks below.**

## Get the necessary packages

First, start with installing the relevant packages ‘tidyverse’, ‘gganimate’, and ‘gapminder’.

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.0 ✔ purrr 0.3.5   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.2.1 ✔ stringr 1.4.1   
## ✔ readr 2.1.3 ✔ forcats 0.5.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

## Look at the data and tackle the tasks

First, see which specific years are actually represented in the dataset and what variables are being recorded for each country. Note that when you run the cell below, Rmarkdown will give you two results - one for each line - that you can flip between.

#Basic info on class of variables.  
str(gapminder)

## tibble [1,704 × 6] (S3: tbl\_df/tbl/data.frame)  
## $ country : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ year : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...  
## $ lifeExp : num [1:1704] 28.8 30.3 32 34 36.1 ...  
## $ pop : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867957 16317921 22227415 ...  
## $ gdpPercap: num [1:1704] 779 821 853 836 740 ...

#Unique years  
unique(gapminder$year)

## [1] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007

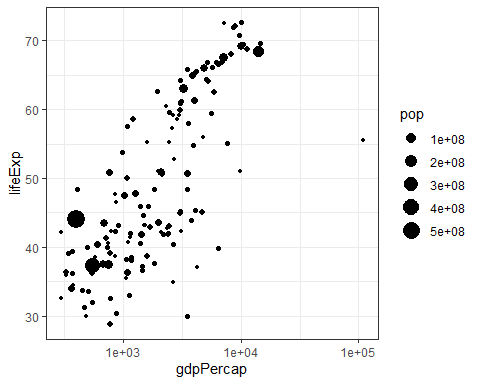
#Get an intuiton of the data structure  
head(gapminder)

## # A tibble: 6 × 6  
## country continent year lifeExp pop gdpPercap  
## <fct> <fct> <int> <dbl> <int> <dbl>  
## 1 Afghanistan Asia 1952 28.8 8425333 779.  
## 2 Afghanistan Asia 1957 30.3 9240934 821.  
## 3 Afghanistan Asia 1962 32.0 10267083 853.  
## 4 Afghanistan Asia 1967 34.0 11537966 836.  
## 5 Afghanistan Asia 1972 36.1 13079460 740.  
## 6 Afghanistan Asia 1977 38.4 14880372 786.

The dataset contains information on each country in the sampled year, its continent, life expectancy, population, and GDP per capita.

Let’s plot all the countries in 1952.

theme\_set(theme\_bw()) # set theme to white background for better visibility  
  
ggplot(subset(gapminder, year == 1952), aes(gdpPercap, lifeExp, size = pop)) +  
 geom\_point() +  
 scale\_x\_log10()



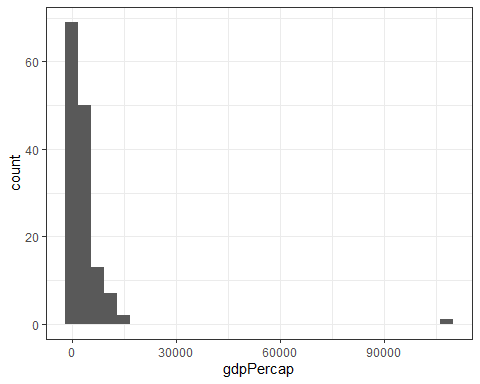
…

We see an interesting spread with an outlier to the right. Answer the following questions, please:

1. *Why does it make sense to have a log10 scale on x axis?*

#The best way to explain it is to visualize it.  
ggplot(subset(gapminder, year == 1952), aes(gdpPercap)) +  
 geom\_histogram()

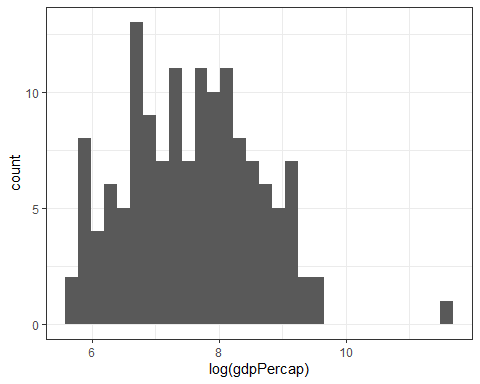
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

 We have a heavily skewed x (gdpPerCap) distribution. It almost looks like a log-normal distribution. (taking the log of the dist would make it normal).

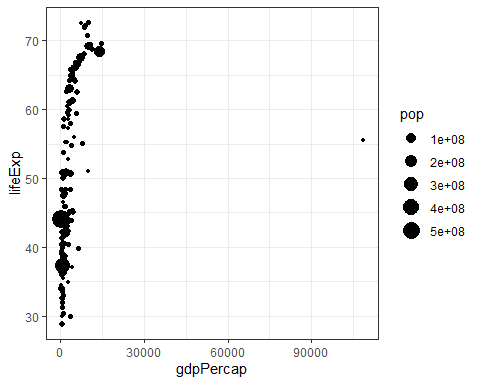
So both for visualization purposes but also for modelling purposes (if you wanna use a generalized linear regression model).

ggplot(subset(gapminder, year == 1952), aes(log(gdpPercap))) +  
 geom\_histogram()

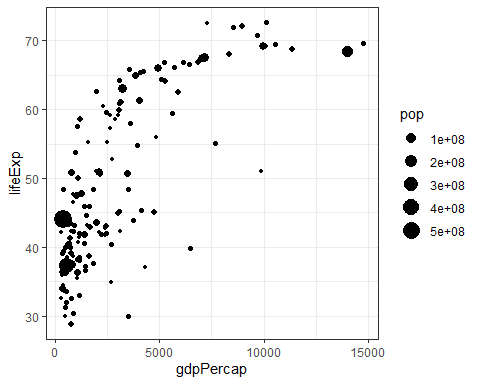
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



ggplot(subset(gapminder, year == 1952), aes(gdpPercap, lifeExp, size = pop)) +  
 geom\_point()



ggplot(subset(gapminder, year == 1952 & gdpPercap < 3e4), aes(gdpPercap, lifeExp, size = pop)) +  
 geom\_point()

 While removing the outlier does fix some of our issues it still shows the relationship to be fairly simialir to a logarithmic function. Which we can make linear by taking the log(x). It is indeed very smart.

1. *Who is the outlier (the richest country in 1952 - far right on x axis)?*

#Method1  
gapminder %>%   
 filter(year == 1952) %>%   
 arrange(desc(gdpPercap)) %>%   
 head(1)

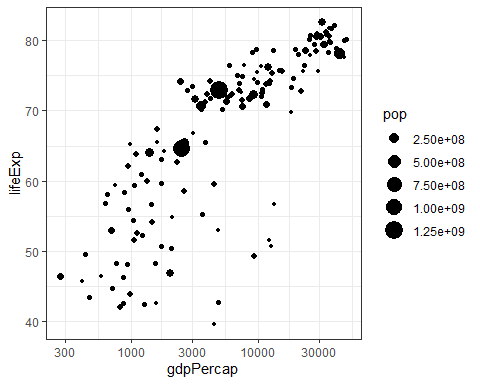
## # A tibble: 1 × 6  
## country continent year lifeExp pop gdpPercap  
## <fct> <fct> <int> <dbl> <int> <dbl>  
## 1 Kuwait Asia 1952 55.6 160000 108382.

#Method 2 (slighly more elegant and adaptable to other situations.)  
subset(gapminder, year == 1952)[which.max(subset(gapminder, year == 1952)$gdpPercap),]

## # A tibble: 1 × 6  
## country continent year lifeExp pop gdpPercap  
## <fct> <fct> <int> <dbl> <int> <dbl>  
## 1 Kuwait Asia 1952 55.6 160000 108382.

Next, you can generate a similar plot for 2007 and compare the differences

ggplot(subset(gapminder, year == 2007), aes(gdpPercap, lifeExp, size = pop)) +  
 geom\_point() +  
 scale\_x\_log10()



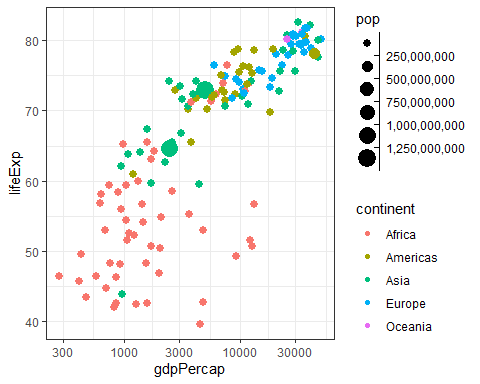
…

The black bubbles are a bit hard to read, the comparison would be easier with a bit more visual differentiation.

Tasks:

1. *Differentiate the* ***continents*** *by color, and fix the axis labels and units to be more legible (****Hint****: the 2.50e+08 is so called “scientific notation”, which you might want to eliminate)*

ggplot(subset(gapminder, year == 2007), aes(gdpPercap, lifeExp, size = pop, color = continent)) +  
 geom\_point() +  
 scale\_x\_log10() +   
 scale\_size\_binned(labels = scales::comma)



1. *What are the five richest countries in the world in 2007?*

gapminder %>%   
 filter(year == 2007) %>%   
 arrange(desc(gdpPercap)) %>%   
 head(5)

## # A tibble: 5 × 6  
## country continent year lifeExp pop gdpPercap  
## <fct> <fct> <int> <dbl> <int> <dbl>  
## 1 Norway Europe 2007 80.2 4627926 49357.  
## 2 Kuwait Asia 2007 77.6 2505559 47307.  
## 3 Singapore Asia 2007 80.0 4553009 47143.  
## 4 United States Americas 2007 78.2 301139947 42952.  
## 5 Ireland Europe 2007 78.9 4109086 40676.

## Make it move!

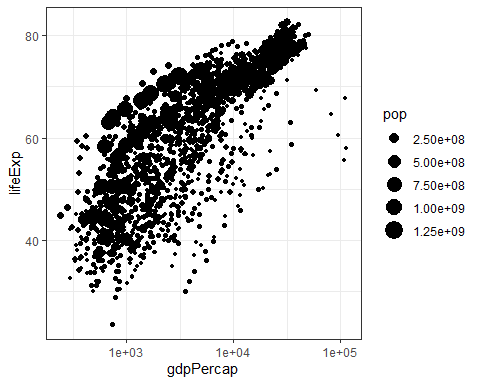
The comparison would be easier if we had the two graphs together, animated. We have a lovely tool in R to do this: the gganimate package. Beware that there may be other packages your operating system needs in order to glue interim images into an animation or video. Read the messages when installing the package.

Also, there are *two* ways of animating the gapminder ggplot.

### Option 1: Animate using transition\_states()

The first step is to create the object-to-be-animated

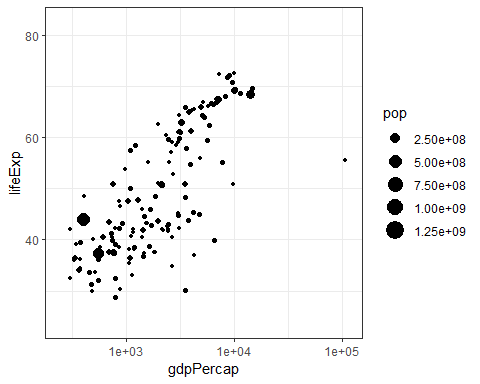
anim <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop)) +  
 geom\_point() +  
 scale\_x\_log10() # convert x to log scale  
anim



…

This plot collates all the points across time. The next step is to split it into years and animate it. This may take some time, depending on the processing power of your computer (and other things you are asking it to do). Beware that the animation might appear in the bottom right ‘Viewer’ pane, not in this rmd preview. You need to knit the document to get the visual inside an html file.

anim + transition\_states(year,   
 transition\_length = 1,  
 state\_length = 1)

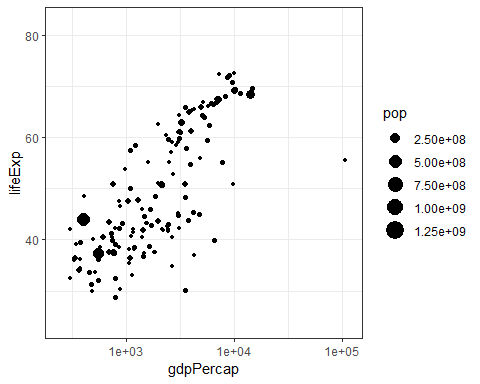
 …

Notice how the animation moves jerkily, ‘jumping’ from one year to the next 12 times in total. This is a bit clunky, which is why it’s good we have another option.

### Option 2 Animate using transition\_time()

This option smoothes the transition between different ‘frames’, because it interpolates and adds transitional years where there are gaps in the timeseries data.

anim2 <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop)) +  
 geom\_point() +  
 scale\_x\_log10() + # convert x to log scale  
 transition\_time(year)  
anim2

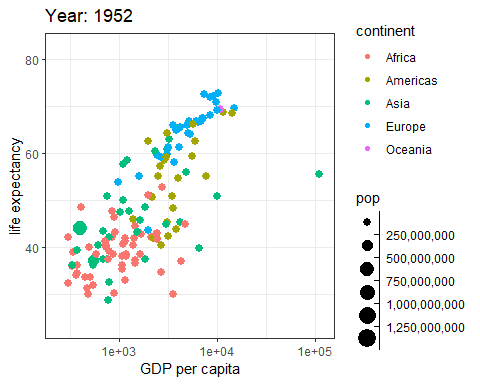


The much smoother movement in Option 2 will be much more noticeable if you add a title to the chart, that will page through the years corresponding to each frame.

Now, choose one of the animation options and get it to work. You may need to troubleshoot your installation of gganimate and other packages

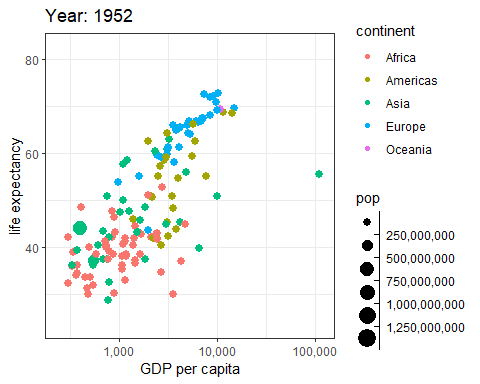
1. *Can you add a title to one or both of the animations above that will change* *in sync with the animation?* *(****Hint****: search labeling for transition\_states() and transition\_time() functions respectively)*

ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, color = continent)) +  
 scale\_x\_log10() +   
 geom\_point() +  
 labs(title = 'Year: {frame\_time}', x = 'GDP per capita', y = 'life expectancy') +  
 scale\_size\_binned(labels = scales::comma) +  
 transition\_time(year)



1. *Can you made the axes’ labels and units more readable? Consider expanding the abreviated lables as well as the scientific notation in the legend and x axis to whole numbers.*

ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, color = continent)) +  
 scale\_x\_log10(labels = scales::comma) +   
 geom\_point() +  
 labs(title = 'Year: {frame\_time}', x = 'GDP per capita', y = 'life expectancy') +  
 scale\_size\_binned(labels = scales::comma) +  
 transition\_time(year)



1. *Come up with a question you want to answer using the gapminder data and write it down. Then, create a data visualisation that answers the question and explain how your visualization answers the question. (Example: you wish to see what was mean life expectancy across the continents in the year you were born versus your parents’ birth years). [Hint: if you wish to have more data than is in the filtered gapminder, you can load either the gapminder\_unfiltered dataset and download more at* [*https://www.gapminder.org/data/*](https://www.gapminder.org/data/) *]*

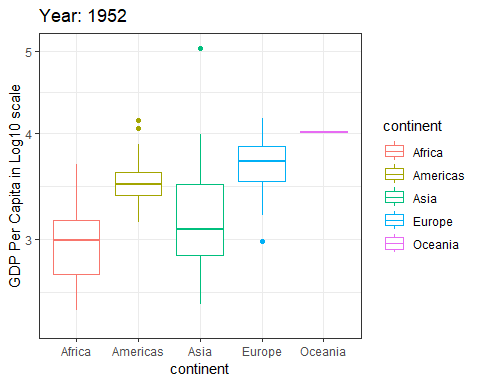
gapminder

## # A tibble: 1,704 × 6  
## country continent year lifeExp pop gdpPercap  
## <fct> <fct> <int> <dbl> <int> <dbl>  
## 1 Afghanistan Asia 1952 28.8 8425333 779.  
## 2 Afghanistan Asia 1957 30.3 9240934 821.  
## 3 Afghanistan Asia 1962 32.0 10267083 853.  
## 4 Afghanistan Asia 1967 34.0 11537966 836.  
## 5 Afghanistan Asia 1972 36.1 13079460 740.  
## 6 Afghanistan Asia 1977 38.4 14880372 786.  
## 7 Afghanistan Asia 1982 39.9 12881816 978.  
## 8 Afghanistan Asia 1987 40.8 13867957 852.  
## 9 Afghanistan Asia 1992 41.7 16317921 649.  
## 10 Afghanistan Asia 1997 41.8 22227415 635.  
## # … with 1,694 more rows

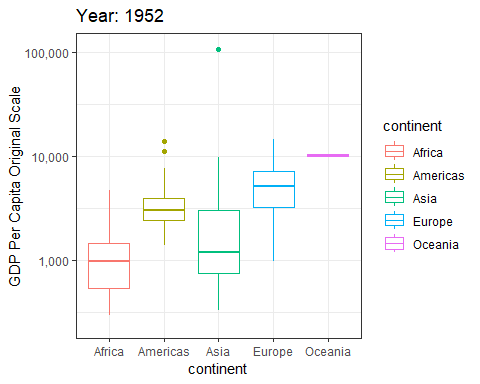
#### Question:

How has GDP Per Capita changed in the different continents over time.

ggplot(gapminder, aes(y = log10(gdpPercap), x = continent, col = continent)) +   
 geom\_boxplot() +   
 scale\_y\_log10(labels = scales::comma) +  
 labs(title = 'Year: {frame\_time}', y = "GDP Per Capita in Log10 scale") +  
 transition\_time(year)



ggplot(gapminder, aes(y = gdpPercap, x = continent, col = continent)) +   
 geom\_boxplot() +   
 scale\_y\_log10(labels = scales::comma) +  
 labs(title = 'Year: {frame\_time}', y = "GDP Per Capita Original Scale") +  
 transition\_time(year)



## **Portfolio 6:**

Portfolio 6

Sigurd Fyhn Sørensen

2022-10-30

# 1

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.

library(gapminder)  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.0 ✔ purrr 0.3.5   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.2.1 ✔ stringr 1.4.1   
## ✔ readr 2.1.3 ✔ forcats 0.5.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

df <- gapminder::gapminder  
  
GDP\_func <- function(pop, GDPpercap){  
 stopifnot(is.numeric(pop))  
 stopifnot(is.numeric(GDPpercap))  
   
 GDP <- pop \* GDPpercap  
   
 if (sum(is.na(GDP))> 0){ warning("Some GDP values returned NA")}  
 if (sum(GDP <= 0)> 0){ warning("GDP can't be negative check your the level of Population & GDP Per Capita")}  
   
 return(GDP)  
   
}  
  
df <- df %>%   
 mutate(GDP = GDP\_func(df$pop, df$gdpPercap))  
  
  
df %>%   
 filter(country == "Denmark" & year %in% c( 1967, 1977, 1987, 1997, 2007,2017))

## # A tibble: 5 × 7  
## country continent year lifeExp pop gdpPercap GDP  
## <fct> <fct> <int> <dbl> <int> <dbl> <dbl>  
## 1 Denmark Europe 1967 73.0 4838800 15937. 77116977700.  
## 2 Denmark Europe 1977 74.7 5088419 20423. 103920280028.  
## 3 Denmark Europe 1987 74.8 5127024 25116. 128771236166.  
## 4 Denmark Europe 1997 76.1 5283663 29804. 157476118456.  
## 5 Denmark Europe 2007 78.3 5468120 35278. 192906627081.

# 2

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)

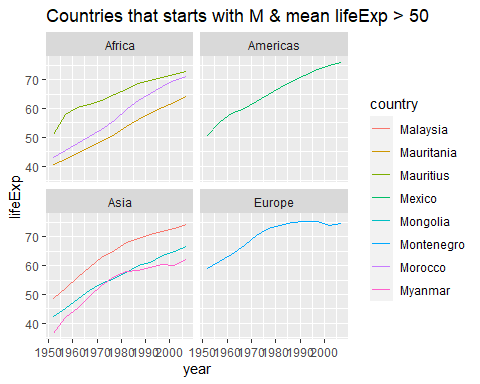
df\_unique <- df %>%   
 distinct(country)  
  
data.frame(Country = df$country , year = df$year,   
 Beings\_with\_b = grepl(df$country, pattern = "^B"),  
 Life\_exp\_bin = if\_else(df$lifeExp > 70, ">70",if\_else(df$lifeExp < 50, "<50", "50-70"))) %>%   
 sample\_n(20) #Just to show a subset

## Country year Beings\_with\_b Life\_exp\_bin  
## 1 Dominican Republic 1952 FALSE <50  
## 2 Finland 2007 FALSE >70  
## 3 Mali 1962 FALSE <50  
## 4 Niger 1957 FALSE <50  
## 5 Japan 1957 FALSE 50-70  
## 6 Japan 2007 FALSE >70  
## 7 Austria 1962 FALSE 50-70  
## 8 Kenya 2002 FALSE 50-70  
## 9 Liberia 2007 FALSE <50  
## 10 Puerto Rico 2002 FALSE >70  
## 11 Costa Rica 2002 FALSE >70  
## 12 Angola 1967 FALSE <50  
## 13 Netherlands 1967 FALSE >70  
## 14 Burkina Faso 1952 TRUE <50  
## 15 Poland 1987 FALSE >70  
## 16 Congo, Rep. 2007 FALSE 50-70  
## 17 Gabon 2002 FALSE 50-70  
## 18 Greece 1962 FALSE 50-70  
## 19 Yemen, Rep. 1992 FALSE 50-70  
## 20 Rwanda 1987 FALSE <50

# 3

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.

df %>%   
 filter(grepl(country, pattern = "^M") == T) %>%  
 group\_by(country) %>%   
 filter(mean(lifeExp) > 50) %>%   
 ggplot(aes(x = year, y = lifeExp, Group = country, color = country)) + geom\_line() + facet\_wrap(~continent) + labs(title = "Countries that starts with M & mean lifeExp > 50")



df %>%   
 filter(grepl(country, pattern = "^M") == T) %>%  
 group\_by(country) %>%   
 filter(mean(lifeExp) < 50) %>%   
 ggplot(aes(x = year, y = lifeExp, Group = country, color = country)) + geom\_line() + facet\_wrap(~continent) + labs(title = "Countries that starts with M & mean lifeExp < 50")

