

peers_activity

CarlosVargas

November 28, 2021

Peer-reviewed Exercise: The SARS-CoV-2 (Covid-19) epidemic

The goal of this project is to produce plots similar to those in the South China Morning Post (SCMP), showing the cumulative number of infected people for different countries. To do that, we need to perform several data processing steps to filter and format the data as required.

The environment

Those are the required libraries.

```
library(tidyverse)
library(ggplot2)
library(lubridate)
library(reshape2)
library(dplyr)
```

Creating a Dataframe from the data

The dataset (time_series_covid19_confirmed_global.csv) can be obtained from this source. The data is compiled by the Johns Hopkins University Center for Systems Science and Engineering JHU CSSE.

We create a new dataframe from the data file.

```
df <- read.csv("data/time_series_covid19_confirmed_global.csv",header=T,stringsAsFactors=FALSE)
```

Exploring the data

Let's gain some insight of the data.

```
ncol(df)
```

```
## [1] 679
```

```
nrow(df)
```

```
## [1] 280
```

```
colnames(df)[c(1:15)]
```

```
## [1] "Province.State" "Country.Region" "Lat" "Long"
## [5] "X1.22.20" "X1.23.20" "X1.24.20" "X1.25.20"
## [9] "X1.26.20" "X1.27.20" "X1.28.20" "X1.29.20"
## [13] "X1.30.20" "X1.31.20" "X2.1.20"
```

```
colnames(df)[c(ncol(df))]
```

```
## [1] "X11.26.21"
```

It seems we have 280 rows and 679 columns, most of the columns correspond to daily dates which range from 22/01/2020 to 26/11/2021.

We remove the Lat and Long columns as we are not interested on this information. Lets take a look to the first columns of the dataset.

```
df <- df[,c(1:2,5:ncol(df))]
df[c(1:10), c(1:5)]
```

	Province.State	Country.Region	X1.22.20	X1.23.20	X1.24.20
## 1		Afghanistan	0	0	0
## 2		Albania	0	0	0
## 3		Algeria	0	0	0
## 4		Andorra	0	0	0
## 5		Angola	0	0	0
## 6		Antigua and Barbuda	0	0	0
## 7		Argentina	0	0	0
## 8		Armenia	0	0	0
## 9	Australian Capital Territory	Australia	0	0	0
## 10	New South Wales	Australia	0	0	0

Data Transformation

We will focus on the following countries: Belgium, China, France, Germany, Italy, Japan, Korea South, Netherlands, Portugal, Spain, United Kingdom and US.

```
df <- df %>% filter(Country.Region %in% c("Belgium", "China", "France", "Germany", "Iran", "Italy", "Japan", "I
```

Lets take a look to what we got. We temporary omit the dates.

```
df[,c(1:2)]
```

	Province.State	Country.Region
## 1		Belgium
## 2	Anhui	China
## 3	Beijing	China
## 4	Chongqing	China
## 5	Fujian	China
## 6	Gansu	China

## 7	Guangdong	China
## 8	Guangxi	China
## 9	Guizhou	China
## 10	Hainan	China
## 11	Hebei	China
## 12	Heilongjiang	China
## 13	Henan	China
## 14	Hong Kong	China
## 15	Hubei	China
## 16	Hunan	China
## 17	Inner Mongolia	China
## 18	Jiangsu	China
## 19	Jiangxi	China
## 20	Jilin	China
## 21	Liaoning	China
## 22	Macau	China
## 23	Ningxia	China
## 24	Qinghai	China
## 25	Shaanxi	China
## 26	Shandong	China
## 27	Shanghai	China
## 28	Shanxi	China
## 29	Sichuan	China
## 30	Tianjin	China
## 31	Tibet	China
## 32	Unknown	China
## 33	Xinjiang	China
## 34	Yunnan	China
## 35	Zhejiang	China
## 36	French Guiana	France
## 37	French Polynesia	France
## 38	Guadeloupe	France
## 39	Martinique	France
## 40	Mayotte	France
## 41	New Caledonia	France
## 42	Reunion	France
## 43	Saint Barthelemy	France
## 44	Saint Pierre and Miquelon	France
## 45	St Martin	France
## 46	Wallis and Futuna	France
## 47		France
## 48		Germany
## 49		Iran
## 50		Italy
## 51		Japan
## 52		Korea, South
## 53	Aruba	Netherlands
## 54	Bonaire, Sint Eustatius and Saba	Netherlands
## 55	Curacao	Netherlands
## 56	Sint Maarten	Netherlands
## 57		Netherlands
## 58		Portugal
## 59		Spain
## 60		US

```
## 61 Anguilla United Kingdom
## 62 Bermuda United Kingdom
## 63 British Virgin Islands United Kingdom
## 64 Cayman Islands United Kingdom
## 65 Channel Islands United Kingdom
## 66 Falkland Islands (Malvinas) United Kingdom
## 67 Gibraltar United Kingdom
## 68 Isle of Man United Kingdom
## 69 Montserrat United Kingdom
## 70 Saint Helena, Ascension and Tristan da Cunha United Kingdom
## 71 Turks and Caicos Islands United Kingdom
## 72 United Kingdom
```

Some countries (e.g. France, United Kingdom, etc) have several colony states that will not be considered in this study, so we filter out the rows containing some entry in `Province.State`. China is a special case, as it is listed by its provinces.

```
df <- df %>% filter(Province.State == "" | Country.Region == "China")
df[,c(1:2)]
```

```
## Province.State Country.Region
## 1 Belgium
## 2 Anhui China
## 3 Beijing China
## 4 Chongqing China
## 5 Fujian China
## 6 Gansu China
## 7 Guangdong China
## 8 Guangxi China
## 9 Guizhou China
## 10 Hainan China
## 11 Hebei China
## 12 Heilongjiang China
## 13 Henan China
## 14 Hong Kong China
## 15 Hubei China
## 16 Hunan China
## 17 Inner Mongolia China
## 18 Jiangsu China
## 19 Jiangxi China
## 20 Jilin China
## 21 Liaoning China
## 22 Macau China
## 23 Ningxia China
## 24 Qinghai China
## 25 Shaanxi China
## 26 Shandong China
## 27 Shanghai China
## 28 Shanxi China
## 29 Sichuan China
## 30 Tianjin China
## 31 Tibet China
## 32 Unknown China
```

```
## 33      Xinjiang      China
## 34      Yunnan      China
## 35      Zhejiang     China
## 36              France
## 37              Germany
## 38              Iran
## 39              Italy
## 40              Japan
## 41      Korea, South
## 42      Netherlands
## 43      Portugal
## 44      Spain
## 45      US
## 46      United Kingdom
```

The province of *HongKong* has to be considered as an independent country as it was treated this way by the SCMP. Thus, we modify the corresponding row.

```
df[df["Province.State"] == "Hong Kong", 2] <- "Hong Kong"
```

We get rid of the `Province.State` column as we do not need it anymore.

```
df <- df[,c(2:ncol(df))]
```

Let's check our dataset.

```
df[,c(1:5)]
```

```
##      Country.Region X1.22.20 X1.23.20 X1.24.20 X1.25.20
## 1      Belgium      0      0      0      0
## 2      China      1      9      15      39
## 3      China     14     22     36     41
## 4      China      6      9     27     57
## 5      China      1      5     10     18
## 6      China      0      2      2      4
## 7      China     26     32     53     78
## 8      China      2      5     23     23
## 9      China      1      3      3      4
## 10     China      4      5      8     19
## 11     China      1      1      2      8
## 12     China      0      2      4      9
## 13     China      5      5      9     32
## 14     Hong Kong      0      2      2      5
## 15     China    444    444    549    761
## 16     China      4      9     24     43
## 17     China      0      0      1      7
## 18     China      1      5      9     18
## 19     China      2      7     18     18
## 20     China      0      1      3      4
## 21     China      2      3      4     17
## 22     China      1      2      2      2
## 23     China      1      1      2      3
```

```
## 24      China      0      0      0      1
## 25      China      0      3      5     15
## 26      China      2      6     15     27
## 27      China      9     16     20     33
## 28      China      1      1      1      6
## 29      China      5      8     15     28
## 30      China      4      4      8     10
## 31      China      0      0      0      0
## 32      China      0      0      0      0
## 33      China      0      2      2      3
## 34      China      1      2      5     11
## 35      China     10     27     43     62
## 36      France      0      0      2      3
## 37      Germany     0      0      0      0
## 38      Iran        0      0      0      0
## 39      Italy       0      0      0      0
## 40      Japan       2      2      2      2
## 41      Korea, South 1      1      2      2
## 42      Netherlands 0      0      0      0
## 43      Portugal    0      0      0      0
## 44      Spain       0      0      0      0
## 45      US          1      1      2      2
## 46      United Kingdom 0      0      0      0
```

Next step is to group all the rows corresponding to *China* and add them up.

```
df <- df %>%
  group_by(Country.Region) %>%
  summarise(across(everything(), sum))
df[,c(1:5)]
```

```
## # A tibble: 14 x 5
##   Country.Region X1.22.20 X1.23.20 X1.24.20 X1.25.20
##   <chr>          <int>    <int>    <int>    <int>
## 1 Belgium            0        0        0        0
## 2 China           548      641      918     1401
## 3 France            0        0        2        3
## 4 Germany           0        0        0        0
## 5 Hong Kong         0        2        2        5
## 6 Iran              0        0        0        0
## 7 Italy             0        0        0        0
## 8 Japan            2        2        2        2
## 9 Korea, South      1        1        2        2
## 10 Netherlands      0        0        0        0
## 11 Portugal         0        0        0        0
## 12 Spain            0        0        0        0
## 13 United Kingdom   0        0        0        0
## 14 US               1        1        2        2
```

Now, we flatten the date columns to proceed with the plotting. We check with a subset.

```
melted_df <- melt(df, id.vars="Country.Region")
melted_df[c(1:20),]
```

```
##      Country.Region variable value
## 1      Belgium X1.22.20      0
## 2      China X1.22.20    548
## 3      France X1.22.20      0
## 4      Germany X1.22.20      0
## 5      Hong Kong X1.22.20      0
## 6      Iran X1.22.20      0
## 7      Italy X1.22.20      0
## 8      Japan X1.22.20      2
## 9      Korea, South X1.22.20      1
## 10     Netherlands X1.22.20      0
## 11     Portugal X1.22.20      0
## 12     Spain X1.22.20      0
## 13 United Kingdom X1.22.20      0
## 14      US X1.22.20      1
## 15     Belgium X1.23.20      0
## 16     China X1.23.20    641
## 17     France X1.23.20      0
## 18     Germany X1.23.20      0
## 19     Hong Kong X1.23.20      2
## 20     Iran X1.23.20      0
```

We need to format the dates correctly. We extract the useful information from the date names and an amount of years to each of them to get the real date.

```
melted_df$variable = as.Date(melted_df$variable, "X%m.%d.%Y") %m+% years(2000)
melted_df[c(1:10),]
```

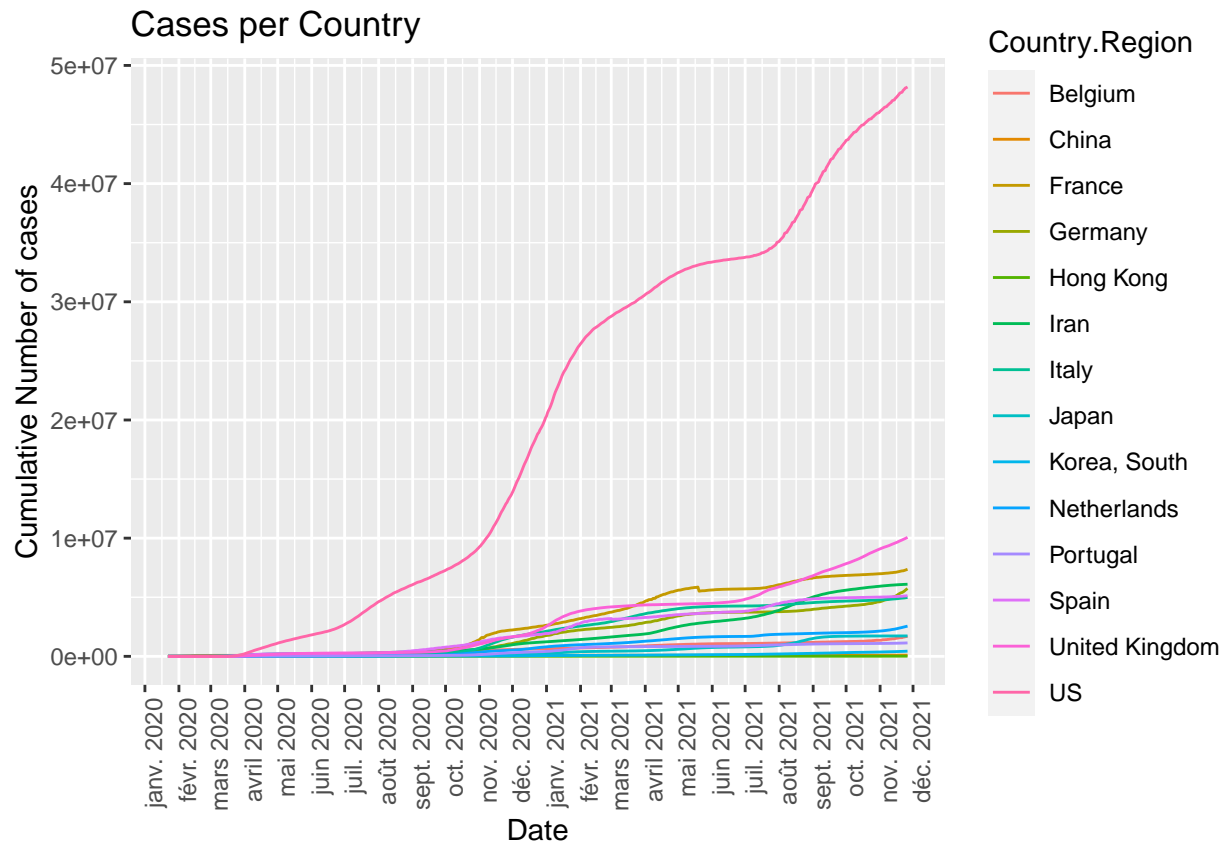
```
##      Country.Region variable value
## 1      Belgium 2020-01-22      0
## 2      China 2020-01-22    548
## 3      France 2020-01-22      0
## 4      Germany 2020-01-22      0
## 5      Hong Kong 2020-01-22      0
## 6      Iran 2020-01-22      0
## 7      Italy 2020-01-22      0
## 8      Japan 2020-01-22      2
## 9      Korea, South 2020-01-22      1
## 10     Netherlands 2020-01-22      0
```

Plotting

Linear Scale

Using the `melted_df` dataframe, we create a plot with the dates (`variable`) on the x axis and the cumulative number of cases (`value`) in the y axis. The dates are expressed per month and for the number of cases we used a linear scale.

```
ggplot(melted_df, aes(x = variable, y = value)) + geom_line(aes(color = Country.Region, group = Country
```



Logarithmic Scale

We recreate the same plot but using logarithmic scale this time.

```
ggplot(melted_df, aes(x = variable, y = value)) + geom_line(aes(color = Country.Region, group = Country
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```