STAT 413/613: HW on List Columns and COVID19

Yunting Chiu

2021-02-19

Instructions

1. Clone this homework repo to your homework directory as a new repo.
2. Rename the starter file under the analysis directory as `hw_01_yourname.Rmd` and use it for your solutions.

3. Modify the "author" field in the YAML header.

4. Stage and Commit R Markdown and HTML files (no PDF files).

5. **Push both .Rmd and HTML files to GitHub**.

- Make sure you have knitted to HTML prior to staging, committing, and pushing your final submission.

6. **Commit each time you answer a part of question, e.g. 1.1**

7. **Push to GitHub after each major question**

8. When complete, submit a response in Canvas

- Only include necessary code to answer the questions.
- Most of the functions you use should be from the tidyverse. Unnecessary Base R or other packages not covered in class will result in point deductions.
- Use Pull requests and or email to ask me any questions. If you email, please ensure your most recent code is pushed to GitHub.
- **Learning Outcome**
  - Use tidyverse functions to create, clean, tidy, and manipulate data frames in a list column
  - Apply purrr functions when working with list columns
  - Employ joins to manipulate data from multiple data frames
- **Context**
  - This assignment looks at COVID-19 data based on the most recent data as of the date you do the work.

Scoring Rubric

Load global and US confirmed cases and deaths data into a nested data frame

1. Create a variable called `url_in` to store this URL: "https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series"

- Revised on Nov. 8th: You may have noticed the URL in the homework has the web-page address not the Raw content address. Please use this URL for url_in: "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/"
- raw.githubusercontent.com returns the raw content of files stored in github, so they can be downloaded simply to your computer.
- If you instead download the file using the github.com link, you will actually be downloading a web page with buttons and comments and which displays your desired content in the middle – it's what you want to give to your web browser to get a nice page to look at but not to download.

```
# Installed library
library(tidyverse)
```

```
## -- Attaching packages ---- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
url_in <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_cov
```

2. Create a tibble named `df` with a variable called `file_names` with a row for each of the following four file names to be loaded from the URL:
   - time_series_covid19_confirmed_global.csv
   - time_series_covid19_deaths_global.csv
   - time_series_covid19_confirmed_US.csv
   - time_series_covid19_deaths_US.csv

```
df <- tibble(file_names = c("time_series_covid19_confirmed_global.csv",
                            "time_series_covid19_deaths_global.csv",
                            "time_series_covid19_confirmed_US.csv",
                            "time_series_covid19_deaths_US.csv")) -> df
```

3. Create a variable in the data frame called `url` that puts `url_in` on the front of each file_name to create a complete URL.

```
df %>%
  mutate(url = str_c(url_in, file_names, sep = "")) -> df
```

4. Use `mutate()` with `map()` to create a list column called `data` with each row holding the downloaded data frame for each file name.

```
df %>%
  mutate(data = map(url, ~read_csv(., na = ""))) -> df
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   `Province/State` = col_character(),
##   `Country/Region` = col_character()
## )

## See spec(...) for full column specifications.

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   `Province/State` = col_character(),
##   `Country/Region` = col_character()
## )

## See spec(...) for full column specifications.

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   iso2 = col_character(),
##   iso3 = col_character(),
##   Admin2 = col_character(),
##   Province_State = col_character(),
##   Country_Region = col_character(),
##   Combined_Key = col_character()
## )

## See spec(...) for full column specifications.

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   iso2 = col_character(),
##   iso3 = col_character(),
##   Admin2 = col_character(),
##   Province_State = col_character(),
##   Country_Region = col_character(),
##   Combined_Key = col_character()
## )

## See spec(...) for full column specifications.
```

5. Add a factor variable to `df` called "case_types" with the **unique** portions of the file names.

```r
df %>%
  mutate(case_types = as.factor(str_extract(file_names, "[:alpha:]*_[gU][:alpha:]*"))) ->
  df
# alpha = Any letter, [A-Za-z]
# reference: https://www.petefreitag.com/cheatsheets/regex/character-classes/
```

6. Remove any columns other than `case_types` and `data` from `df`.

- `df` should have four observations of two variables.

```r
df %>%
  select(case_types, data) -> df
```

Clean Data

1. Use `map()` to add the names from each of the four data frames to a new variable in `df` called `vars` and visually compare them to identify issues.

```
df %>%
  mutate(vars = map(df$data, names)) -> df
# map(df$vars, ~unlist(.)[1:15]) for checking
```

2. Take the following steps to fix any issues and create consistent data frames.

a. Create a short helper function called `fix_names()` which takes three arguments, a data frame, a string, and a replacement pattern. It should replace all occurrences of the string in the names of the variables in the data frame with the replacement pattern.
b. Convert “Province/State” and “Country/Region” to “Province_State” “Country_Region”.
c. Convert “admin2 to”County" and “Long_” to “Long”.
d. Remove the variables “UID”, “iso2”, “iso3”, “code3”, “FIPS”, and “Combined_Key” from the US data.
e. Add variables `Population` and `County` to the data frames where missing.
f. Add a variable called `Country_State` that combines the country with the province/state while keeping the original columns.
g. Update the values in `df$vars` when complete to check for consistency.

- Hint: Look at help for `map_if()`

```
# a
fix_names <- function(df, pattern, rePattern){
  stopifnot(is.data.frame(df), is.character(pattern), is.character(rePattern))
  names(df) <- str_replace_all(names(df), pattern, rePattern)
  return(df)
}

# b-f
df %>%
  mutate(data = map(data, ~fix_names(., "([ey])/", "\\1_")),
         data = map(data, ~fix_names(., "Admin2", "County")),
         data = map(data, ~fix_names(., "Long_", "Long")),
         data = map_if(data, str_detect(df$case_types, "US"),
                   ~select(., -c("UID", "iso2", "iso3",
                                 "code3", "FIPS", "Combined_Key"))),
         data = map_if(data, str_detect(df$case_types, "global"),
                      ~mutate(., County = "NA")),
         data = map_if(data, !str_detect(df$case_types, "deaths_US"),
                      ~mutate(., Population = 0)),
         data = map(data, ~unite(., "Country_State",
                                 c("Country_Region", "Province_State"),
                                 remove = FALSE, na.rm = TRUE,
                                 sep = "_"))
         ) -> df

# g
df %>%
  mutate(vars = map(df$data, names)) -> df # synchronize the vars correspondingly
# map(df$vars, ~unlist(.)) # for checking
```

Tidy each dataframe

1. Use `map()` along with pivot_longer to tidy each data frame and as part of the pivot, ensure the daily values are in a variable called "Date" and use a lubridate function inside the pivot to ensure it is of class date.
2. Save the new data frame to a variable called `df_long`

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
df %>%
  mutate(data = map(data, ~pivot_longer(data = ., cols = contains("/"),
                                        names_to = "Date",
                                        values_to = "dailyValues"))
         ) -> df
# df$vars <- map(df$data, names) # synchronize the vars correspondingly
# map(df$vars, ~unlist(.)) # for checking

# crate a function to fix in type of Date
mdyDate <- function(df, varsDate){
  stopifnot(is.data.frame(df), is.character(varsDate))
  df[[varsDate]] <- mdy(df[[varsDate]])
  return(df)
}

df %>%
  mutate(data = map(data, ~mdyDate(., "Date"))) -> df_long

# str(df_long) # check the data set
```

Add Continents

1. Use `map()` to add a new variable called `Continent` to each data frame.

- Hint: use the package {countrycode} to get the continents.
- If you don't have it already, use the console to install.
- Then load package countrycode and look at help for `countrycode::countrycode`
- You will get some warning messages about NAs which you will fix next.

```
library(countrycode)
df_long %>%
  mutate(data = map(data, ~mutate(., Continent = countrycode(Country_Region,
                                                   origin = "country.name",
                                                   destination = "continent")))
         ) -> df_long
```

Fix NAs for Continents

- Use `map()` with `case_when()` to replace the NAs due to "Diamond Princess", "Kosovo", "MS Zaandam" with the most appropriate continent
- Use `map()` with `unique()` to confirm five continents in the global data frames and one in the US data frames

```
df_long %>%
  mutate(data = map(data, ~mutate(., Continent = case_when(
                                               Country_Region == "Diamond Princess" ~ "Asia",
                                               Country_Region == "Kosovo" ~ "Americas",
                                               Country_Region == "MS Zaandam" ~ "Europe",
                                               TRUE ~ Continent)
                                  ))) -> df_long

map(df_long$data, ~unique(.$Continent))
```

```
## [[1]]
## [1] "Asia"     "Europe"   "Africa"   "Americas" "Oceania"  NA
##
## [[2]]
## [1] "Asia"     "Europe"   "Africa"   "Americas" "Oceania"  NA
##
## [[3]]
## [1] "Americas"
##
## [[4]]
## [1] "Americas"
```

Unnest the Data Frames

1. Unnest and ungroup the data frame `df_long` and save into a new data frame called `df_all`
2. Remove original `df` and `df_long` dataframes from the environment
3. Remove the `vars` variable from `df_all`

```
# 1
df_long %>%
  unnest(cols = data) %>%
  ungroup() -> df_all

# 2
remove(df, df_long)

# 3
df_all %>%
  select(-vars) -> df_all
```

Get World Population Data

1. Read in World population data for 2019 into its own data frame called `df_pop`

- Use the provided CSV or you can go to the UN source
- The CSV has a few changes in country names to match the COVID data, e.g., US, and Iran.
- Note: the UN data is in thousands so it can have fractional values

2. Use a join to remove all Locations that are not in the `df_all` data frame.
3. Add the ranks for each location for population and population density to `df_pop`

```r
# 1
df_pop <- read_csv("./data/WPP2019_TotalPopulation.csv")
```

```
## Parsed with column specification:
## cols(
##   LocID = col_double(),
##   Location = col_character(),
##   PopTotal = col_double(),
##   PopDensity = col_double()
## )
```

```r
# summarize(df_pop, across(everything(), ~sum(is.na(.)))) # check NAs

# 2
semi_join(df_pop, df_all, by = c("Location" = "Country_Region")) -> df_pop

# 3
df_pop %>%
  mutate(rank_p = rank(-PopTotal, na.last = TRUE),
         rank_d = rank(-PopDensity, na.last = TRUE),
         PopTotal = (PopTotal*1000)) -> df_pop
```

Add Population Data to `df_all`

- Use a join to add the data from `df_pop` to `df_all`
- This means there will be two columns with population data:
  - Population for US Counties
  - PopTotal for the country level

```r
df_all %>%
  inner_join(df_pop, by = c("Country_Region" = "Location")) -> df_all

df_all
```

```
## # A tibble: 2,832,860 x 16
##    case_types Country_State Province_State Country_Region   Lat  Long County
##    <fct>      <chr>         <chr>          <chr>          <dbl> <dbl> <chr>
##  1 confirmed~ Afghanistan   <NA>           Afghanistan     33.9  67.7 NA
##  2 confirmed~ Afghanistan   <NA>           Afghanistan     33.9  67.7 NA
##  3 confirmed~ Afghanistan   <NA>           Afghanistan     33.9  67.7 NA
##  4 confirmed~ Afghanistan   <NA>           Afghanistan     33.9  67.7 NA
##  5 confirmed~ Afghanistan   <NA>           Afghanistan     33.9  67.7 NA
##  6 confirmed~ Afghanistan   <NA>           Afghanistan     33.9  67.7 NA
##  7 confirmed~ Afghanistan   <NA>           Afghanistan     33.9  67.7 NA
##  8 confirmed~ Afghanistan   <NA>           Afghanistan     33.9  67.7 NA
##  9 confirmed~ Afghanistan   <NA>           Afghanistan     33.9  67.7 NA
## 10 confirmed~ Afghanistan   <NA>           Afghanistan     33.9  67.7 NA
## # ... with 2,832,850 more rows, and 9 more variables: Population <dbl>,
## #   Date <date>, dailyValues <dbl>, Continent <chr>, LocID <dbl>,
## #   PopTotal <dbl>, PopDensity <dbl>, rank_p <dbl>, rank_d <dbl>
```

Analyse Data

1. Create a data frame by with data grouped by `Country_Region`, `Continent` `case_type`, `rank_p` and `rank_d` that summarizes the current totals and the totals as a percentage of total population.

- Be sure to look at how the data is reported so the numbers make sense.

2. What are the 20 Countries with the most confirmed cases and what is the percentage of their total population affected?
3. What are the 20 Countries with the most deaths and what is the percentage of their total population affected?
4. Try to interpret the results by just looking at the rankings for the totals with the rankings for total population and population density.

- interpretation: Although some countries are large and are not densely populated, they still report many cases on covid-19, such as the US, Brazil, which means it is hard to control covid in large countries even with relatively low population density.

```
# 1
df_all %>%
 group_by(Country_Region, Continent, case_types, rank_p, rank_d) %>%
 summarise(ttlCases = max(dailyValues), ttlPerc = ttlCases/last(PopTotal)*100) %>%
  ungroup() -> tmp
```

```
## `summarise()` regrouping output by 'Country_Region', 'Continent', 'case_types', 'rank_p' (override wi
```

```
# 2
## Top 20 Countries with the most confirmed cases and the percentage effects
tmp %>%
filter(case_types == "confirmed_global") %>%
arrange(desc(ttlCases)) %>%
  head(20) -> confirmed20
confirmed20
```

```
## # A tibble: 20 x 7
##    Country_Region Continent case_types       rank_p rank_d ttlCases ttlPerc
##    <chr>          <chr>     <fct>             <dbl>  <dbl>    <dbl>   <dbl>
##  1 US             Americas  confirmed_global      3    131 27896042   8.48
##  2 India          Asia      confirmed_global      2     14 10963394   0.802
##  3 Brazil         Americas  confirmed_global      6    139 10030626   4.75
##  4 United Kingdom Europe    confirmed_global     18     29  4083242   6.05
##  5 France         Europe    confirmed_global     19     63  3517177   5.40
##  6 Spain          Europe    confirmed_global     24     83  3121687   6.68
##  7 Italy          Europe    confirmed_global     20     45  2765412   4.57
##  8 Turkey         Asia      confirmed_global     15     70  2616600   3.14
##  9 Germany        Europe    confirmed_global     14     33  2372209   2.84
## 10 Colombia       Americas  confirmed_global     23    125  2212525   4.40
## 11 Argentina      Americas  confirmed_global     25    156  2046795   4.57
## 12 Mexico         Americas  confirmed_global      9    107  2022662   1.59
## 13 Poland         Europe    confirmed_global     32     61  1614446   4.26
## 14 Iran           Asia      confirmed_global     16    118  1550142   1.87
## 15 South Africa   Africa    confirmed_global     21    121  1498766   2.56
## 16 Ukraine        Europe    confirmed_global     27     96  1333332   3.03
## 17 Indonesia      Asia      confirmed_global      4     53  1252685   0.463
## 18 Peru           Americas  confirmed_global     37    138  1252137   3.85
## 19 Czechia        Europe    confirmed_global     75     55  1123252  10.5
## 20 Netherlands    Europe    confirmed_global     59     13  1042674   6.10
```

```
# 3
## Top 20 Countries with the most died cases and the percentage effects
tmp %>%
filter(case_types == "deaths_global") %>%
  arrange(desc(ttlCases)) %>%
  head(20) -> deaths20
deaths20
```

```
## # A tibble: 20 x 7
##    Country_Region Continent case_types    rank_p rank_d ttlCases ttlPerc
##    <chr>          <chr>     <fct>          <dbl>  <dbl>    <dbl>   <dbl>
##  1 US             Americas  deaths_global      3    131   493082  0.150
##  2 Brazil         Americas  deaths_global      6    139   243457  0.115
##  3 Mexico         Americas  deaths_global      9    107   178108  0.140
##  4 India          Asia      deaths_global      2     14   156111  0.0114
##  5 United Kingdom Europe    deaths_global     18     29   119387  0.177
##  6 Italy          Europe    deaths_global     20     45    94887  0.157
##  7 France         Europe    deaths_global     19     63    82975  0.127
##  8 Germany        Europe    deaths_global     14     33    67245  0.0805
##  9 Spain          Europe    deaths_global     24     83    66704  0.143
## 10 Iran           Asia      deaths_global     16    118    59264  0.0715
## 11 Colombia       Americas  deaths_global     23    125    58334  0.116
## 12 Argentina      Americas  deaths_global     25    156    50857  0.114
## 13 South Africa   Africa    deaths_global     21    121    48708  0.0832
## 14 Peru           Americas  deaths_global     37    138    44308  0.136
## 15 Poland         Europe    deaths_global     32     61    41582  0.110
## 16 Indonesia      Asia      deaths_global      4     53    33969  0.0126
## 17 Turkey         Asia      deaths_global     15     70    27821  0.0333
## 18 Ukraine        Europe    deaths_global     27     96    26191  0.0595
## 19 Belgium        Europe    deaths_global     69     19    21821  0.189
## 20 Chile          Americas  deaths_global     53    137    19798  0.104
```

Which countries in the top 20 for percentage of population affected are Not in the top 20 for the absolute number of cases and deaths?

- Try to interpret the results by just looking at the rankings for the totals with the rankings for total population and population density.
- Interpretation: These countries have low population, so the denominator is small, which brings up the percentage of population affected. On the other hand, the population density is not a critical element with this result.

```
tmp %>%
  arrange(desc(ttlPerc)) %>%
  head(20) -> perc20

perc20 %>%
  # anti_join() return all rows from x without a match in y.
  anti_join(confirmed20) %>%
  anti_join(deaths20) %>%
  select(Country_Region)
```

```
## Joining, by = c("Country_Region", "Continent", "case_types", "rank_p", "rank_d", "ttlCases", "ttlPer
## Joining, by = c("Country_Region", "Continent", "case_types", "rank_p", "rank_d", "ttlCases", "ttlPer
## # A tibble: 15 x 1
##    Country_Region
##    <chr>
##  1 Andorra
##  2 Montenegro
##  3 San Marino
##  4 Slovenia
##  5 Israel
##  6 Luxembourg
##  7 Panama
##  8 Portugal
##  9 Bahrain
## 10 Lithuania
## 11 Georgia
## 12 Liechtenstein
## 13 Belgium
## 14 Switzerland
## 15 Sweden
```

Create two plots, one for the number of cases and one for the number of deaths over time for the top 20 country/states faceting by continent.

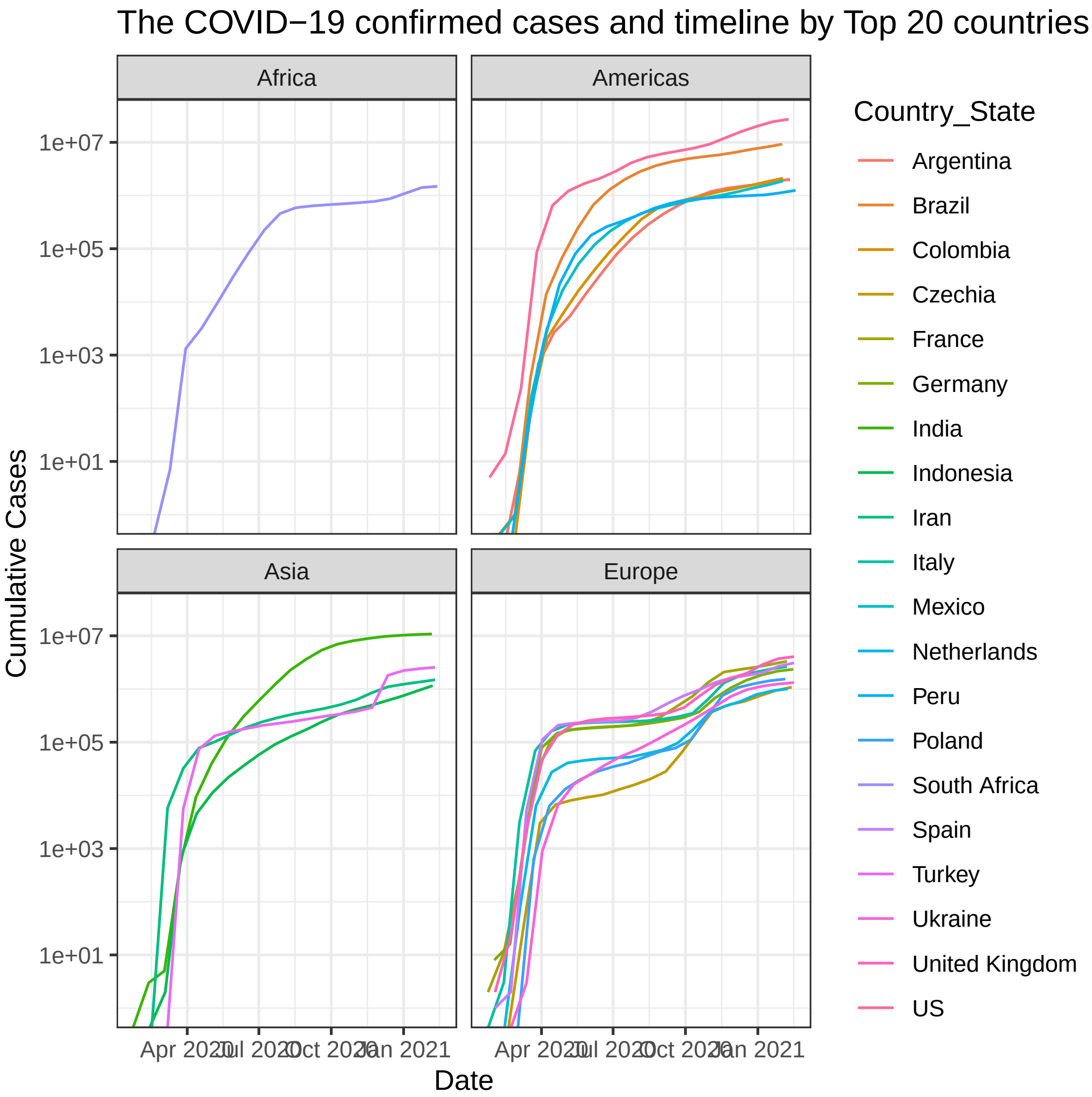
- Use appropriate scales for the axes.
- Create two sets of plots
- Interpret each plot.

1. The outbreaks are still in the Americas and Europe. Covid has increased more rapidly around the world since April, and the Coronavirus cases still hit daily high.

```
confirmed <- confirmed20$Country_Region

df_all %>%
  filter(case_types == "confirmed_global", Country_State == confirmed) %>%
  ggplot() +
  geom_line(mapping = aes(x = Date, y = dailyValues, color = Country_State)) +
  facet_wrap(~Continent) +
  scale_y_log10() +
  theme_bw() +
  ylab("Cumulative Cases") +
  ggtitle("The COVID-19 confirmed cases and timeline by Top 20 countries")
```

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

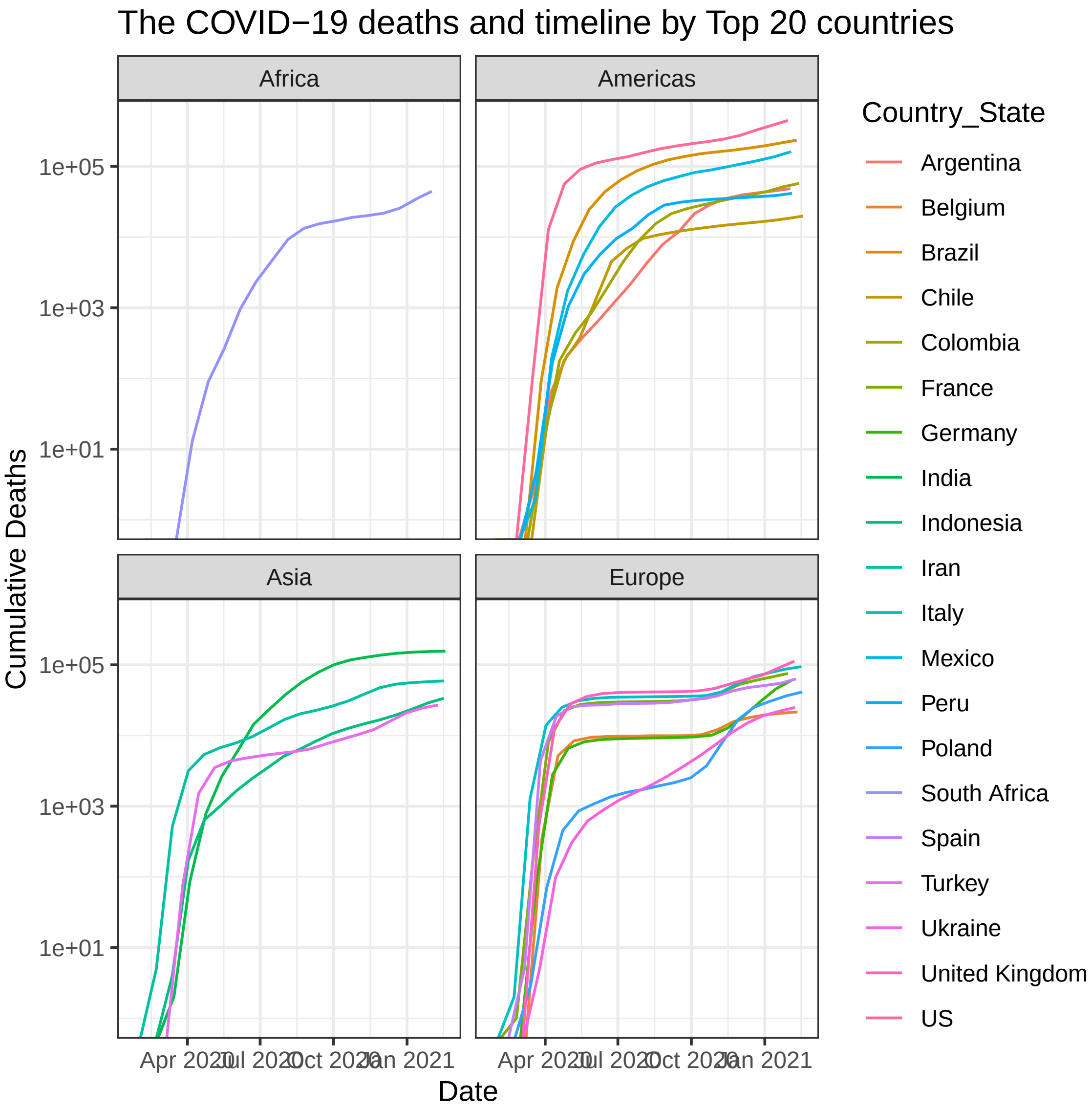


2. Based on the plot of confirmed cases, we can see the deaths have positive association with confirmed cases of covid. It is note worthy that Turkey and Ecuador have a large number of deaths but they don't have high confirmed cases.

```
deaths <- deaths20$Country_Region

df_all %>%
  filter(case_types == "deaths_global", Country_State == deaths) %>%
  ggplot() +
  geom_line(mapping = aes(x = Date, y = dailyValues, color = Country_State)) +
  facet_wrap(~Continent) +
  scale_y_log10() +
  theme_bw() +
  ylab("Cumulative Deaths") +
  ggtitle("The COVID-19 deaths and timeline by Top 20 countries")
```

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


The COVID−19 deaths and timeline by Top 20 countries
Africa
Americas
Asia
Europe
Cumulative Deaths
1e+05
1e+03
1e+01
1e+05
1e+03
1e+01
Apr 2020
Jul 2020
Oct 2020
Jan 2021
Apr 2020
Jul 2020
Oct 2020
Jan 2021
Date
Country_State
Argentina
Belgium
Brazil
Chile
Colombia
France
Germany
India
Indonesia
Iran
Italy
Mexico
Peru
Poland
South Africa
Spain
Turkey
Ukraine
United Kingdom
US