List Columns and Nested Data Frames

Evan Bowman

2023-03-28

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://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/".

- This allows you to directly download the files at the John's Hopkins site: "https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series"

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

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

```
df <- tibble(file_names = c("time_series_covid19_confirmed_global.csv", "time_series_covid19_deaths_glo
  as_tibble()
```

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

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 <- df %>%
  mutate(data = map(url, ~ read_csv(., show_col_types = F)))
```

5. Add a factor variable `case_type` to `df` with the **unique** portions of the `file_names` as output from a {stringr} function.

```
df <- df %>%
  mutate(case_types = as.factor(str_extract(file_names, "[:alpha:]*_[gU][:alpha:]*")))
```

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

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

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

```
df
```

```
## # A tibble: 2 x 2
##   case_types       data
##   <fct>            <list>
## 1 confirmed_global <spc_tbl_ [289 x 1,147]>
## 2 deaths_global    <spc_tbl_ [289 x 1,147]>
```

Clean Data

1. Using a **single call to `map()`**, add only the first 15 names from each of the four data frames to a new variable in `df` called `vars`. Do NOT try to add to the individual data frames. `df` should now have three variables/columns and two observations

- We want to eventually combine all the data frames into one data frame so they should have the same column names.
- Visually compare them to identify any issues across the rows.

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

2. Inside a **single call to `mutate()`, make the changes in steps b through e** using {purrr} functions to fix any issues and create consistent data frames, Then use map to do f and g. (should be eight uses of a `map_*` function).

a. Create a short (4 lines) generic helper function called `fix_names()` which takes three arguments: a data frame, a string pattern, and a string "replacement pattern". It should replace all occurrences of the "string pattern" in the names of the columns in the data frame with the "replacement pattern". Include error checking to ensure the inputs are of the proper class. It should not know anything about the contents of the data frame argument.

b. Use your function with `map()` to convert `Province/State` to `Province_State` and `Country/Region` to `Country_Region`.

c. Use a {purrr} function to add variable called `Country_State` that unites `Country_Region` and `Province/State` while keeping the original columns and removing `NA`s.

d. Use a {purrr} function to remove the variables for `Lat` and `Long`

e. Use a {purrr} function with `select()` to reorder the variables in each data frame as follows: `Country_Region`, `Country_State`, `Province_State`, and then the remaining data columns.

f. Use `map()` to update the values in `df$vars` with the new first 15 names and show the values to check for consistency across data frames.

g. Use `map()` three times: show how many rows are in each data frame, show many columns are in each data frame, and show the name of the last column in each data frame. The last column should be the most recent date in the data frame as of the date the data was pulled.

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

df <- df %>%
  mutate(data = map(data, ~fix_names(., "([ey])/", "\\1_")),   #b
          data = map(data, ~unite(., "Country_State",
                                  c("Country_Region", "Province_State"),
                                  remove = FALSE, na.rm = TRUE,
                                  sep = "_")),                 #c
         data = map(data, ~select(., -c(Lat, Long))),      # d
         data = map(data, ~select(., c(Country_Region, Country_State, Province_State), everything()))) #e

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

```
## [[1]]
##  [1] "Country_Region" "Country_State"  "Province_State" "1/22/20"
##  [5] "1/23/20"        "1/24/20"        "1/25/20"        "1/26/20"
##  [9] "1/27/20"        "1/28/20"        "1/29/20"        "1/30/20"
## [13] "1/31/20"        "2/1/20"         "2/2/20"
##
## [[2]]
##  [1] "Country_Region" "Country_State"  "Province_State" "1/22/20"
##  [5] "1/23/20"        "1/24/20"        "1/25/20"        "1/26/20"
##  [9] "1/27/20"        "1/28/20"        "1/29/20"        "1/30/20"
## [13] "1/31/20"        "2/1/20"         "2/2/20"
```

```
# g
map(df$data, ~nrow(.))
```

```
## [[1]]
## [1] 289
##
## [[2]]
## [1] 289
```

```
map(df$data, ~ncol(.))
```

```
## [[1]]
## [1] 1146
##
## [[2]]
## [1] 1146
```

```
map(df$data, ~.[, ncol(.)])
```

```
## [[1]]
## # A tibble: 289 x 1
##    `3/9/23`
##       <dbl>
##  1   209451
##  2   334457
##  3   271496
##  4    47890
##  5   105288
##  6       11
##  7     9106
##  8 10044957
##  9   447308
## 10   232974
## # ... with 279 more rows
##
## [[2]]
## # A tibble: 289 x 1
##    `3/9/23`
##       <dbl>
##  1     7896
##  2     3598
##  3     6881
##  4      165
##  5     1933
##  6        0
##  7      146
##  8   130472
##  9     8727
## 10      228
## # ... with 279 more rows
```

Use {purrr} to Tidy Each Data Frame

1. Use `map()` along with `pivot_longer()` to tidy each data frame and then save the results **to a new tibble** called `df_long` as **its own tibble** (NOT a list column in the `df` tibble).

- `df` should still have 2 observations with three variables and the data frames in `data` should have 289 rows.
- `df_long` should have 2 observations with three variables but now the data frames in `data` should have at least 236,856 rows (as of 11/1/2022).

- **As part of the pivot**,
  - Put the daily totals in a variable called `Daily_Total`.
  - Put the dates in a variable called `Date`.
  - Use a {lubridate} function *inside the pivot* to ensure `Date` is of class `date`.

```
df_long <- df %>%
  mutate(data = map(data, ~pivot_longer(data = ., cols = contains("/"),
                                        names_to = "Date",
                                        values_to = "Daily_Total",
                                        names_transform = list(Date = mdy))))
str(df_long$data)
```

```
## List of 2
##  $ : tibble [330,327 x 5] (S3: tbl_df/tbl/data.frame)
##   ..$ Country_Region: chr [1:330327] "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
##   ..$ Country_State : chr [1:330327] "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
##   ..$ Province_State: chr [1:330327] NA NA NA NA ...
##   ..$ Date          : Date[1:330327], format: "2020-01-22" "2020-01-23" ...
##   ..$ Daily_Total   : num [1:330327] 0 0 0 0 0 0 0 0 0 0 ...
##  $ : tibble [330,327 x 5] (S3: tbl_df/tbl/data.frame)
##   ..$ Country_Region: chr [1:330327] "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
##   ..$ Country_State : chr [1:330327] "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
##   ..$ Province_State: chr [1:330327] NA NA NA NA ...
##   ..$ Date          : Date[1:330327], format: "2020-01-22" "2020-01-23" ...
##   ..$ Daily_Total   : num [1:330327] 0 0 0 0 0 0 0 0 0 0 ...
```

2. Use `map` to show how many rows are in each data frame in the `df_long$data` list column.

```
map(df_long$data, ~nrow(.))
```

```
## [[1]]
## [1] 330327
##
## [[2]]
## [1] 330327
```

3. Use `map` to show the last seven days of data in each data frame for the the United States (`Country_Region == "US"`). Use a sentence to describe what each row represents in each data frame.

```
last_seven <- map(df_long$data, ~filter(., Country_Region == "US"))
map(last_seven, ~tail(., 7))
```

```
## [[1]]
## # A tibble: 7 x 5
##   Country_Region Country_State Province_State Date       Daily_Total
##   <chr>          <chr>         <chr>          <date>           <dbl>
## 1 US             US            <NA>           2023-03-03   103648690
## 2 US             US            <NA>           2023-03-04   103650837
## 3 US             US            <NA>           2023-03-05   103646975
## 4 US             US            <NA>           2023-03-06   103655539
```

```
## 5 US             US            <NA>           2023-03-07   103690910
## 6 US             US            <NA>           2023-03-08   103755771
## 7 US             US            <NA>           2023-03-09   103802702
##
## [[2]]
## # A tibble: 7 x 5
##   Country_Region Country_State Province_State Date       Daily_Total
##   <chr>          <chr>         <chr>          <date>           <dbl>
## 1 US             US            <NA>           2023-03-03     1122165
## 2 US             US            <NA>           2023-03-04     1122172
## 3 US             US            <NA>           2023-03-05     1122134
## 4 US             US            <NA>           2023-03-06     1122181
## 5 US             US            <NA>           2023-03-07     1122516
## 6 US             US            <NA>           2023-03-08     1123246
## 7 US             US            <NA>           2023-03-09     1123836
```

Each row represents the cumulative sum of confirmed Covid cases and Covid deaths for the United States over the last seven days.

Add Continents

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

- Then load package {countrycode} and look at help for `countrycode::countrycode`
- You will get some warning messages about ambiguous values which you will fix in the next step.

```
library(countrycode)

  df_long <-  df_long %>%
  mutate(data = map(data, ~mutate(., Continent = countrycode(Country_Region,
                                            origin = "country.name",
                                            destination = "continent"),
                                ISO3_code = countrycode(Country_Region,
                                                        origin = "country.name",
                                                        destination = "iso3c"))))
```

```
## Warning in countrycode_convert(sourcevar = sourcevar, origin = origin, destination = dest, : Som

## Warning in countrycode_convert(sourcevar = sourcevar, origin = origin, destination = dest, : Som

## Warning in countrycode_convert(sourcevar = sourcevar, origin = origin, destination = dest, : Som

## Warning in countrycode_convert(sourcevar = sourcevar, origin = origin, destination = dest, : Som
```

2. Fix Ambiguous Values for Continents

- Use `map()` with `case_when()` (inside a `mutate()`) to replace the `NA`s due to Antarctica, Diamond Princess, Kosovo, Micronesia, MS Zaandam, Summer Olympics 2020, and Winter Olympics 2022 with the most appropriate continent.
- Use `map()` with `unique()` to confirm six continents in the global data frames

```
df_long <- df_long %>%
  mutate(data = map(data, ~mutate(., Continent = case_when(
                                Country_Region == " Antarctica" ~ " Antarctica",
                                Country_Region == "Diamond Princess" ~ "Asia",
                                Country_Region == "Kosovo" ~ "Europe",
                                Country_Region == "Micronesia" ~ "Oceania",
                                Country_Region == "MS Zaandam" ~ "Americas",
                                Country_Region == "Summer Olympics 2020" ~ "Asia",
                                Country_Region == "Winter Olympics 2022" ~ "Asia",
                                TRUE ~ Continent))))
map(df_long$data, ~unique(.$Continent))
```

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

Unnest the Data Frames

1. Unnest and ungroup the data frames in `df_long$data` and save the results into a new separate data frame called `df_all`. You should now have three distinct objects; `df`, `df_long`, and `df_all`

```
df_all <- df_long %>%
  unnest(cols = data) %>%
  ungroup()
head(df_all)
```

```
## # A tibble: 6 x 9
##   case_types    Count~1 Count~2 Provi~3 Date       Daily~4 Conti~5 ISO3_~6 vars
##   <fct>         <chr>   <chr>   <chr>   <date>       <dbl> <chr>   <chr>   <lis>
## 1 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-22       0 Asia    AFG     <chr>
## 2 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-23       0 Asia    AFG     <chr>
## 3 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-24       0 Asia    AFG     <chr>
## 4 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-25       0 Asia    AFG     <chr>
## 5 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-26       0 Asia    AFG     <chr>
## 6 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-27       0 Asia    AFG     <chr>
## # ... with abbreviated variable names 1: Country_Region, 2: Country_State,
## #   3: Province_State, 4: Daily_Total, 5: Continent, 6: ISO3_code
```

3. Remove the `vars` variable from `df_all` and save `df_all`.

```
df_all <- df_all %>%
  select(-vars)
head(df_all)
```

```
## # A tibble: 6 x 8
##   case_types       Country_~1 Count~2 Provi~3 Date       Daily~4 Conti~5 ISO3_~6
##   <fct>            <chr>      <chr>   <chr>   <date>       <dbl> <chr>   <chr>
## 1 confirmed_global Afghanist~ Afghan~ <NA>    2020-01-22       0 Asia    AFG
```

```
## 2 confirmed_global Afghanist~ Afghan~ <NA>    2020-01-23       0 Asia    AFG
## 3 confirmed_global Afghanist~ Afghan~ <NA>    2020-01-24       0 Asia    AFG
## 4 confirmed_global Afghanist~ Afghan~ <NA>    2020-01-25       0 Asia    AFG
## 5 confirmed_global Afghanist~ Afghan~ <NA>    2020-01-26       0 Asia    AFG
## 6 confirmed_global Afghanist~ Afghan~ <NA>    2020-01-27       0 Asia    AFG
## # ... with abbreviated variable names 1: Country_Region, 2: Country_State,
## #   3: Province_State, 4: Daily_Total, 5: Continent, 6: ISO3_code
```

- `df_all` should have at least 586,000 rows as of 11/1/2022.

```
nrow(df_all)
```

```
## [1] 660654
```

4. Save `df_all` as both a .csv file and a .rds file to a data directory. Compare their file sizes?

```
write_csv(df_all, "data/df_all.csv")
write_rds(df_all, "data/df_all.rds")
```

The file size of the csv file is around 31 mbs. The size of the rds file is roughly 53 mbs.

Get World Population Data

1.a. Use `vroom::vroom()` with a relative path to read in the zipped .csv file with World population data for 1950-2100 into its own data frame called `df_pop`.

```
df_pop <- vroom::vroom("data/WPP2022_TotalPopulationBySex.csv.zip", show_col_types = F)
```

```
## Multiple files in zip: reading 'WPP2022_TotalPopulationBySex.csv'
```

- The data is from the UN which uses different country names in many cases from the COVID data. It also uses a different structure for separating countries and territories.

- Note: the UN population data is in thousands so it can have fractional values.
- Filter the data to only those rows for 2022 where the scenario variant is "No change" and which have a valid `ISO3_code`,
- Select `Time`, `Location`, `ISO3_code`, `PopTotal`, `PopDensity` and save to `df_pop`.
  - You should have 237 rows remaining out of the 586,092 rows.

```
df_pop <- df_pop %>%
  filter(Time == "2022", Variant == "No change", !is.na(ISO3_code)) %>%
  select(Time, Location, ISO3_code, PopTotal, PopDensity)
head(df_pop)
```

```
## # A tibble: 6 x 5
##    Time Location ISO3_code PopTotal PopDensity
##   <dbl> <chr>    <chr>        <dbl>      <dbl>
## 1  2022 Burundi  BDI         12890.      497.
## 2  2022 Comoros  COM           837.      450.
## 3  2022 Djibouti DJI          1121.       48.4
## 4  2022 Eritrea  ERI          3684.       30.4
## 5  2022 Ethiopia ETH        123380.      123.
## 6  2022 Kenya    KEN         54027.       93.0
```

b. Show the countries (`Country_Region`) in the Covid data that are not in the population data - Use `ISO3_code`. How many are there?

```
covid_not_pop <- anti_join(df_all, df_pop, by = c("ISO3_code" = "ISO3_code"))

unique(covid_not_pop$Country_Region)
```

```
## [1] "Antarctica"           "Diamond Princess"     "Kosovo"
## [4] "MS Zaandam"           "Micronesia"           "Summer Olympics 2020"
## [7] "Winter Olympics 2022"
```

There are 7 "countries" that are in the Covid data that are not found in the population data.

c. Identify the countries in the population data that are not in the covid data. Use `ISO3_code` How many are there?

```
countries_pop <- anti_join(df_pop, df_all, by = c("ISO3_code" = "ISO3_code"))
unique(countries_pop$Location)
```

```
##  [1] "Mayotte"                          "Réunion"
##  [3] "Western Sahara"                   "Saint Helena"
##  [5] "Turkmenistan"                     "China, Hong Kong SAR"
##  [7] "China, Macao SAR"                 "Faroe Islands"
##  [9] "Guernsey"                         "Isle of Man"
## [11] "Jersey"                           "Gibraltar"
## [13] "Kosovo (under UNSC res. 1244)"    "Anguilla"
## [15] "Aruba"                            "Bonaire, Sint Eustatius and Saba"
## [17] "British Virgin Islands"           "Cayman Islands"
## [19] "Curaçao"                          "Guadeloupe"
## [21] "Martinique"                       "Montserrat"
## [23] "Puerto Rico"                      "Saint Barthélemy"
## [25] "Saint Martin (French part)"       "Sint Maarten (Dutch part)"
## [27] "Turks and Caicos Islands"         "United States Virgin Islands"
## [29] "Falkland Islands (Malvinas)"      "French Guiana"
## [31] "Bermuda"                          "Greenland"
## [33] "Saint Pierre and Miquelon"        "New Caledonia"
## [35] "Guam"                             "Micronesia (Fed. States of)"
## [37] "Northern Mariana Islands"         "American Samoa"
## [39] "Cook Islands"                     "French Polynesia"
## [41] "Niue"                             "Tokelau"
## [43] "Wallis and Futuna Islands"
```

There are 43 countries that are found in the UN's population data set that are not found in Johns Hopkins' Covid data.

d. What is the percentage of the world population contained in the countries not in the covid data?

```
sum(countries_pop$PopTotal) / sum(df_pop$PopTotal) * 100
```

```
## [1] 0.3089145
```

The percentage of the world population contained by the countries not found in the Covid data is .3089 percent.

2. Use a {dplyr} join to remove all Locations from `df_pop` that are not in the `df_all` data frame.

```
df_pop <- semi_join(df_pop, df_all, by = c("ISO3_code" = "ISO3_code"))
```

- Add variables to `df_pop` for the rank for each location for population (`rank_p`) and the rank for population density (`rank_d`).

```
df_pop <- df_pop %>%
  mutate(rank_p = min_rank(desc(PopTotal)),
         rank_d = min_rank(desc(PopDensity)))
head(df_pop)
```

```
## # A tibble: 6 x 7
##    Time Location ISO3_code PopTotal PopDensity rank_p rank_d
##   <dbl> <chr>    <chr>        <dbl>      <dbl>  <int>  <int>
## 1  2022 Burundi  BDI          12890.      497.      78     18
## 2  2022 Comoros  COM            837.      450.     159     20
## 3  2022 Djibouti DJI           1121.       48.4    157    137
## 4  2022 Eritrea  ERI           3684.       30.4    130    151
## 5  2022 Ethiopia ETH         123380.      123.      12     73
## 6  2022 Kenya    KEN          54027.       93.0     27     96
```

3. Show the countries with rank 1:10 for Total Population

- Then show the countries with rank 1:10 for Population Density.

```
# Rank_p
df_pop %>%
  select(Location, rank_p) %>%
  arrange(rank_p) %>%
  slice(1:10)
```

```
## # A tibble: 10 x 2
##    Location                rank_p
##    <chr>                    <int>
##  1 China                        1
##  2 India                        2
##  3 United States of America     3
```

```
##  4 Indonesia                     4
##  5 Pakistan                      5
##  6 Nigeria                       6
##  7 Brazil                        7
##  8 Bangladesh                    8
##  9 Russian Federation            9
## 10 Mexico                       10
```

```
# Rank_d
df_pop %>%
  select(Location, rank_d) %>%
  arrange(rank_d) %>%
  slice(1:10)
```

```
## # A tibble: 10 x 2
##    Location                        rank_d
##    <chr>                            <int>
##  1 Monaco                               1
##  2 Singapore                            2
##  3 Bahrain                              3
##  4 Maldives                             4
##  5 Malta                                5
##  6 Bangladesh                           6
##  7 Holy See                             7
##  8 State of Palestine                   8
##  9 China, Taiwan Province of China      9
## 10 Barbados                            10
```

Add Population Data to `df_all`

- Use a {dplyr} join to add the data from `df_pop` to `df_all` to create `df_allp` for only those countries in `df_all` even if they are not in `df_pop`.

```
df_allp <- left_join(df_all, df_pop, by = c("ISO3_code" = "ISO3_code")) %>%
  select(-Location)
head(df_allp)
```

```
## # A tibble: 6 x 13
##   case_types    Count~1 Count~2 Provi~3 Date       Daily~4 Conti~5 ISO3_~6  Time
##   <fct>         <chr>   <chr>   <chr>   <date>       <dbl> <chr>   <chr>   <dbl>
## 1 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-22       0 Asia    AFG      2022
## 2 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-23       0 Asia    AFG      2022
## 3 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-24       0 Asia    AFG      2022
## 4 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-25       0 Asia    AFG      2022
## 5 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-26       0 Asia    AFG      2022
## 6 confirmed_gl~ Afghan~ Afghan~ <NA>    2020-01-27       0 Asia    AFG      2022
## # ... with 4 more variables: PopTotal <dbl>, PopDensity <dbl>, rank_p <int>,
## #   rank_d <int>, and abbreviated variable names 1: Country_Region,
## #   2: Country_State, 3: Province_State, 4: Daily_Total, 5: Continent,
## #   6: ISO3_code
```

How many `Country_Regions` have Multiple `Country_States`?

- The data does not treat all countries the same with regard to reporting at the country level or at the province or region level.
- Some countries are reported with *totals only at the country level* (`Country_Region`),
- Some countries are reported with *totals only at the state/province level* (`Province_State`),
- Some countries are reported with *totals for the country and for separate state/provinces* for the country.
- We can use the `Country States` and `Country Region` variables to figure out what this means.

a. For each `Country Region` calculate the number of `Country States` for *distinct combinations of `Country States` and `Country Region`* and then,

- show in descending order the number of `Country_States`for each `Country_Region` where the number of `Country_States` is greater than 1.

```
df_allp %>% group_by(Country_Region) %>%
  summarise(Country_States = length(unique(Country_State))) %>%
  filter(Country_States > 1) %>%
  arrange(desc(Country_States))
```

```
## # A tibble: 8 x 2
##   Country_Region Country_States
##   <chr>                   <int>
## 1 China                      34
## 2 Canada                     16
## 3 United Kingdom             15
## 4 France                     12
## 5 Australia                   8
## 6 Netherlands                 5
## 7 Denmark                     3
## 8 New Zealand                 3
```

b. For each `Country Region` calculate the number of `Country States` for distinct combinations of `Country States` and `Country Region` where the `Country_Region` does **not have matching entries in `Country_State`** and then,

- show in descending order the number of `Country_States` for each `Country_Region`, show where the number of `Country_States` is greater than 1.

```
df_allp %>% group_by(Country_Region) %>%
  summarise(unique_country_states = n_distinct(Country_State)) %>%
  filter(unique_country_states > 1) %>%
  arrange(desc(unique_country_states))
```

```
## # A tibble: 8 x 2
##   Country_Region unique_country_states
##   <chr>                          <int>
## 1 China                             34
## 2 Canada                            16
## 3 United Kingdom                    15
## 4 France                            12
```

```
## 5 Australia                          8
## 6 Netherlands                        5
## 7 Denmark                            3
## 8 New Zealand                        3
```

c. Explain what the difference between the two results suggests for future analysis of totals for each country represented in `Country_Region`.

I believe the first represents the countries that are regions of independently governed but that are in a commonwealth of some sort (i.e. territories, disputed claims, etc.)

Analyze Data

1. Use `df_allp` to create a new data frame 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**.

- Create grand totals for each of the two global case types for both `df_all` and your new data frame and compare them.
- Interpret the results. **Check a website to confirm if your numbers are reasonable and show the URL you checked and the numbers.**

```
percentages <- df_allp %>%
 group_by(Country_Region, Continent, case_types, rank_p, rank_d) %>%
 summarise(total_cases = max(Daily_Total), total_percent = total_cases/(last(PopTotal)*1000)*100) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'Country_Region', 'Continent',
## 'case_types', 'rank_p'. You can override using the `.groups` argument.
```

```
head(percentages)
```

```
## # A tibble: 6 x 7
##   Country_Region Continent case_types       rank_p rank_d total_cases total_pe~1
##   <chr>          <chr>     <fct>              <int>  <int>       <dbl>      <dbl>
## 1 Afghanistan    Asia      confirmed_global      36    123      209451     0.509
## 2 Afghanistan    Asia      deaths_global         36    123        7896     0.0192
## 3 Albania        Europe    confirmed_global     135     86      334457    11.8
## 4 Albania        Europe    deaths_global        135     86        3598     0.127
## 5 Algeria        Africa    confirmed_global      34    169      271496     0.605
## 6 Algeria        Africa    deaths_global         34    169        6881     0.0153
## # ... with abbreviated variable name 1: total_percent
```

Data was checked with information gathered from https://ourworldindata.org/covid-cases

2. What are the 20 Countries with the most confirmed cases and what is the percentage of their total population affected?

```
percentages %>%
  filter(case_types == "confirmed_global") %>%
  arrange(desc(total_cases)) %>%
  slice(1:20)
```

```
## # A tibble: 20 x 7
##    Country_Region Continent case_types       rank_p rank_d total_cases total_p~1
##    <chr>          <chr>     <fct>             <int>  <int>       <dbl>     <dbl>
##  1 US             Americas  confirmed_global      3    146   103802702     30.7
##  2 India          Asia      confirmed_global      2     19    44690738      3.15
##  3 France         Europe    confirmed_global     23     77    38618509     59.8
##  4 Germany        Europe    confirmed_global     19     42    38249060     45.9
##  5 Brazil         Americas  confirmed_global      7    161    37081209     17.2
##  6 Japan          Asia      confirmed_global     11     28    33320438     26.9
##  7 Korea, South   Asia      confirmed_global     29     16    30615522     59.1
##  8 Italy          Europe    confirmed_global     25     54    25603510     43.4
##  9 United Kingdom Europe    confirmed_global     21     34    24425309     36.2
## 10 Russia         Europe    confirmed_global      9    183    22075858     15.3
## 11 Turkey         Asia      confirmed_global     18     81    17042722     20.0
## 12 Spain          Europe    confirmed_global     30     94    13770429     29.0
## 13 Vietnam        Asia      confirmed_global     16     29    11526994     11.7
## 14 Argentina      Americas  confirmed_global     33    177    10044957     22.1
## 15 Taiwan*        Asia      confirmed_global     57      9     9970937     41.7
## 16 Netherlands    Europe    confirmed_global     71     17     8599981     49.0
## 17 Iran           Asia      confirmed_global     17    132     7572311      8.55
## 18 Mexico         Americas  confirmed_global     10    121     7483444      5.87
## 19 Indonesia      Asia      confirmed_global      4     64     6738225      2.45
## 20 Poland         Europe    confirmed_global     37     70     6444960     16.2
## # ... with abbreviated variable name 1: total_percent
```

3. What are the 20 Countries with the most deaths and what is the percentage of their total population affected?

```
percentages %>%
  filter(case_types == "deaths_global") %>%
  arrange(desc(total_cases)) %>%
  slice(1:20)
```

```
## # A tibble: 20 x 7
##    Country_Region Continent case_types    rank_p rank_d total_cases total_perc~1
##    <chr>          <chr>     <fct>          <int>  <int>       <dbl>        <dbl>
##  1 US             Americas  deaths_global      3    146     1123836      0.332
##  2 Brazil         Americas  deaths_global      7    161      699276      0.325
##  3 India          Asia      deaths_global      2     19      530779      0.0375
##  4 Russia         Europe    deaths_global      9    183      388478      0.268
##  5 Mexico         Americas  deaths_global     10    121      333188      0.261
##  6 United Kingdom Europe    deaths_global     21     34      219948      0.326
##  7 Peru           Americas  deaths_global     44    157      219539      0.645
##  8 Italy          Europe    deaths_global     25     54      188322      0.319
##  9 Germany        Europe    deaths_global     19     42      168935      0.203
## 10 France         Europe    deaths_global     23     77      161512      0.250
## 11 Indonesia      Asia      deaths_global      4     64      160941      0.0584
```

```
## 12 Iran           Asia      deaths_global     17    132      144933      0.164
## 13 Colombia       Americas  deaths_global     28    139      142339      0.274
## 14 Argentina      Americas  deaths_global     33    177      130472      0.287
## 15 Spain          Europe    deaths_global     30     94      119479      0.251
## 16 Ukraine        Europe    deaths_global     38    120      119283      0.300
## 17 Poland         Europe    deaths_global     37     70      119010      0.299
## 18 South Africa   Africa    deaths_global     24    136      102595      0.171
## 19 Turkey         Asia      deaths_global     18     81      101492      0.119
## 20 China          Asia      deaths_global      1     61       82195      0.00576
## # ... with abbreviated variable name 1: total_percent
```

4. Describe the results based on the totals with the rankings for total population and population density.

Both the results show the high transmission rates of COVID, as well as the difficulty in containing. Many countries on the list rank relatively low based on population density and highlight that COVID did not only affect high density countries.

High Percentage but Low Totals Countries

- Which countries in the top 20 for percentage of population for cases are **Not** in the top 20 for the absolute number of cases.

```
# Setting up Data frames for anti_join
percent20 <- percentages %>%
  filter(case_types == "confirmed_global") %>%
  select(Country_Region, total_cases, total_percent, rank_p, rank_d) %>%
  arrange(desc(total_percent)) %>%
  slice(1:20)

cases20 <- percentages %>%
  filter(case_types == "confirmed_global") %>%
  select(Country_Region, total_cases, total_percent, rank_p, rank_d) %>%
  arrange(desc(total_cases)) %>%
  slice(1:20)

anti_join(percent20, cases20)
```

```
## Joining, by = c("Country_Region", "total_cases", "total_percent", "rank_p",
## "rank_d")
```

```
## # A tibble: 17 x 5
##    Country_Region total_cases total_percent rank_p rank_d
##    <chr>                <dbl>         <dbl>  <int>  <int>
##  1 San Marino           23616          70.2    190     14
##  2 Austria            5961143          66.7     99     83
##  3 Slovenia           1331707          62.8    145     84
##  4 Brunei              279661          62.3    169    110
##  5 Andorra              47890          60.0    184     57
##  6 Denmark            3404407          57.9    113     66
##  7 Iceland             209137          56.1    172    191
##  8 Liechtenstein        21432          54.5    188     39
```

```
##  9 Portugal           5570473          54.2     92     79
## 10 Greece             5548487          53.4     90    108
## 11 Israel             4803824          53.1     98     22
## 12 Latvia              976255          52.8    148    152
## 13 Cyprus              650685          52.0    155     68
## 14 Switzerland        4413911          50.5    101     49
## 15 Luxembourg          317367          49.0    163     38
## 16 Georgia            1827537          48.8    129    133
## 17 Bahrain             710693          48.3    151      3
```

- Which countries in the top 20 for percentage of population for deaths are **Not** in the top 20 for the absolute number deaths?

```
percent20_deaths <- percentages %>%
  filter(case_types == "deaths_global") %>%
  select(Country_Region, total_cases, total_percent, rank_p, rank_d) %>%
  arrange(desc(total_percent)) %>%
  slice(1:20)

deaths20 <- percentages %>%
  filter(case_types == "deaths_global") %>%
  select(Country_Region, total_cases, total_percent, rank_p, rank_d) %>%
  arrange(desc(total_cases)) %>%
  slice(1:20)

anti_join(percent20_deaths, deaths20)
```

```
## Joining, by = c("Country_Region", "total_cases", "total_percent", "rank_p",
## "rank_d")
```

```
## # A tibble: 17 x 5
##    Country_Region         total_cases total_percent rank_p rank_d
##    <chr>                        <dbl>         <dbl>  <int>  <int>
##  1 Bulgaria                     38228         0.564    107    125
##  2 Bosnia and Herzegovina       16280         0.503    134    124
##  3 Hungary                      48762         0.489     94     82
##  4 North Macedonia               9662         0.462    147    102
##  5 Georgia                      16971         0.453    129    133
##  6 Montenegro                    2808         0.448    164    140
##  7 Croatia                      17987         0.446    128    117
##  8 Czechia                      42491         0.405     88     67
##  9 Slovakia                     21035         0.373    114     78
## 10 Moldova                      12003         0.367    133     90
## 11 San Marino                     122         0.362    190     14
## 12 Lithuania                     9596         0.349    138    141
## 13 Romania                      67736         0.345     64    101
## 14 Latvia                        6269         0.339    148    152
## 15 Greece                       34779         0.335     90    108
## 16 Slovenia                      7078         0.334    145     84
## 17 Chile                        64273         0.328     65    159
```

- Describe the results based on the per population results with the rankings for total population and population density.

All of the results of both confirmed cases and deaths highlight that low populated countries were affected more on the proportional level. In these measurements, population density does not really factor in.

Plotting the Data

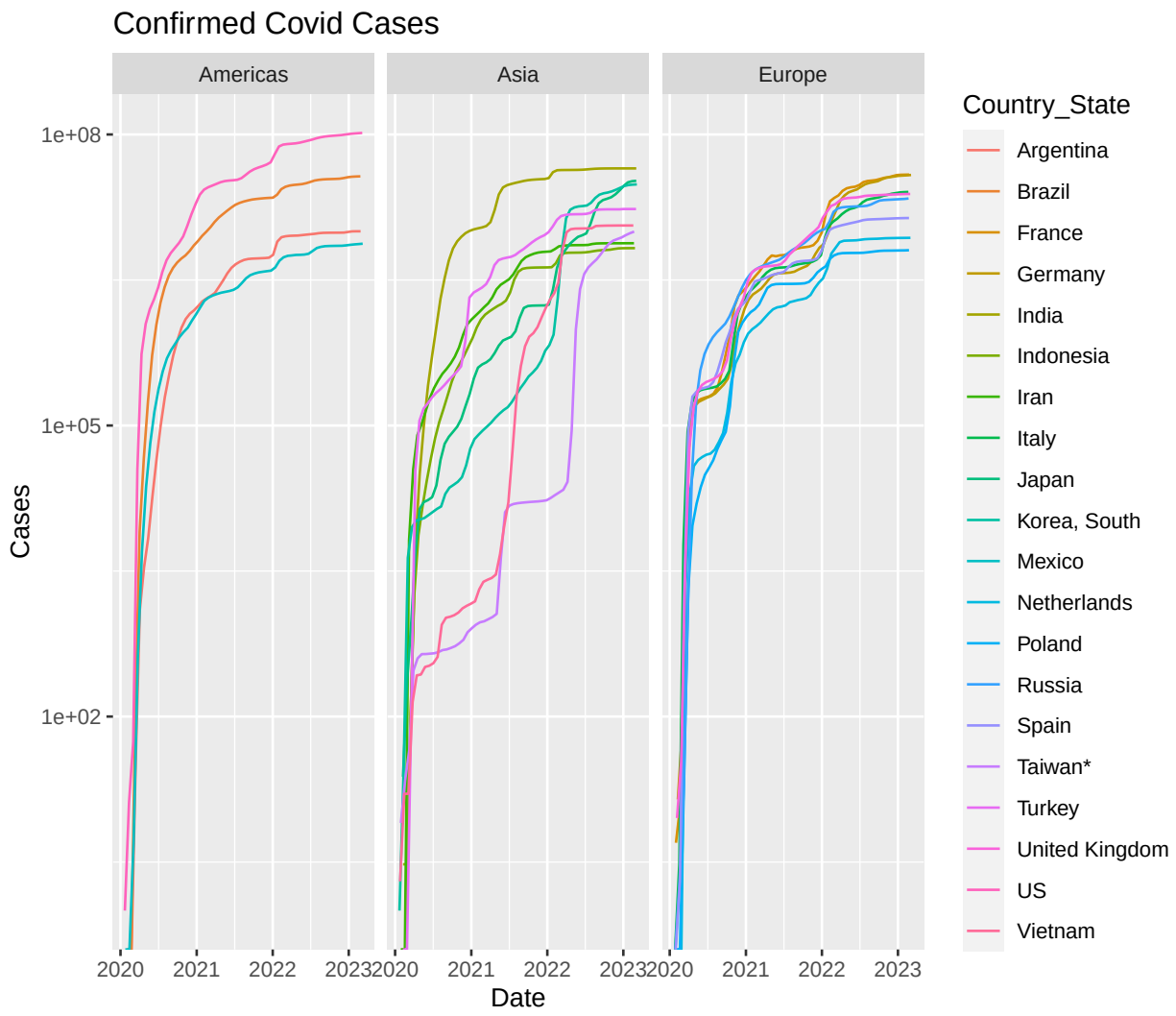
- Create two plots, one for the number of cases, and one for the number of deaths, showing the changes **over time**.
- Limit to the top 20 `Country_Region` for highest cases and then for highest deaths
- Show each country and facet by continent with the same scale for the y axis.
- Use a log scale for the y axis.
- Interpret each plot with respect to the total cases (or deaths) and the path of cases (or deaths) across and within different continents.

```
confirmed <- cases20$Country_Region

df_all %>%
  filter(case_types == "confirmed_global", Country_State == confirmed) %>%
  ggplot() +
  geom_line(mapping = aes(x = Date, y = Daily_Total, color = Country_State)) +
  facet_wrap(~Continent) +
  scale_y_log10() +
  ylab("Cases") +
  ggtitle("Confirmed Covid Cases")
```

```
## Warning in Country_State == confirmed: longer object length is not a multiple of
## shorter object length
```

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



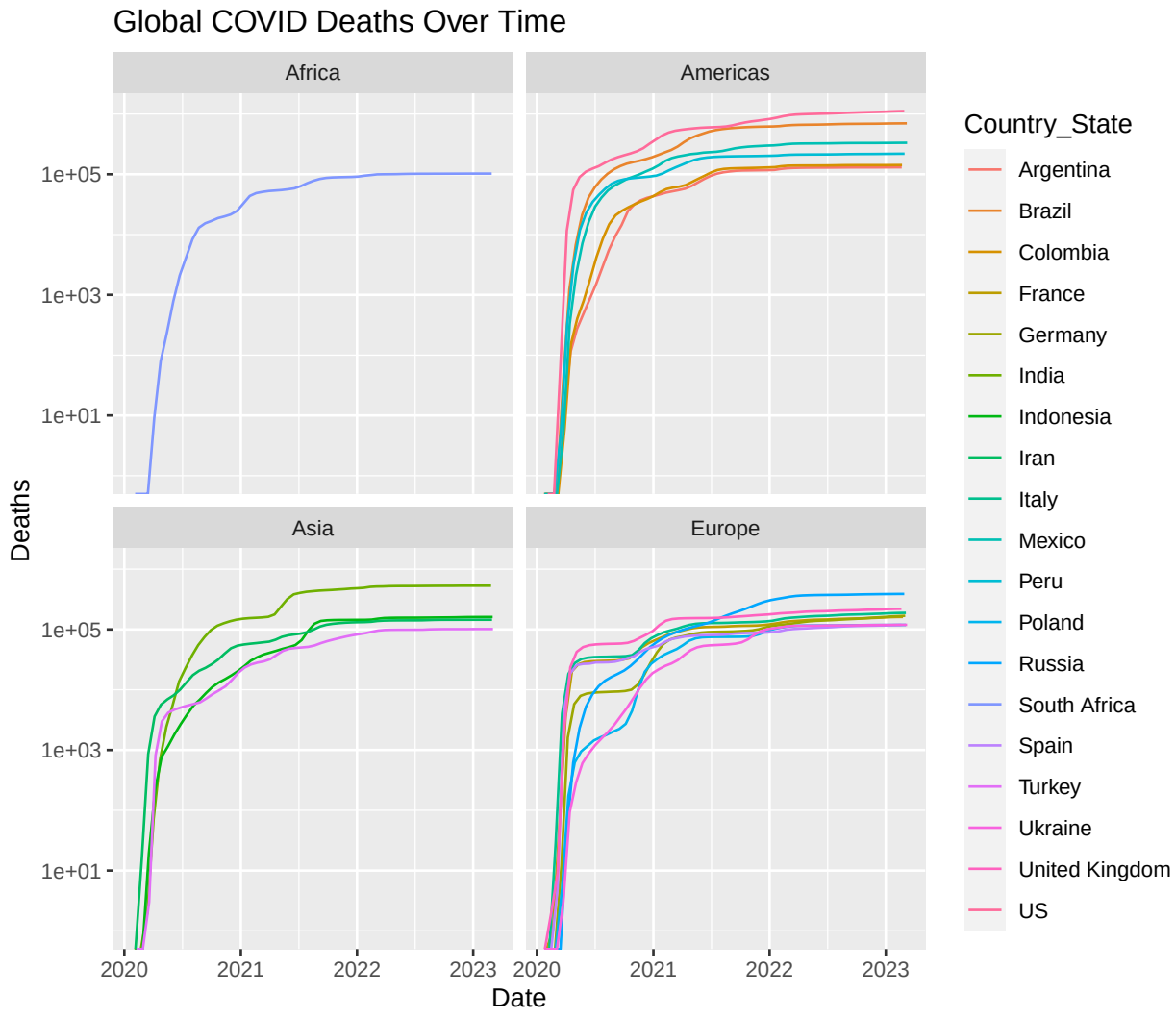
Most cases follow a major increase during the first year of the pandemic and begin to increase with a curvilinear pattern around the beginning of 2021. All seem to have areas where they flatten out but then exponentially increasing again which represents the different waves of the pandemic.

```
# Deaths
deaths <- deaths20$Country_Region

df_all %>%
  filter(case_types == "deaths_global", Country_State == deaths) %>%
  ggplot() +
  geom_line(mapping = aes(x = Date, y = Daily_Total, color = Country_State)) +
  facet_wrap(~Continent) +
  scale_y_log10() +
  ylab("Deaths") +
  ggtitle("Global COVID Deaths Over Time")
```

```
## Warning in Country_State == deaths: longer object length is not a multiple of
## shorter object length
```

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



The trends of global deaths follows the same trend as the confirmed cases in the initial part of the pandemic. However, as it flattens out, there is less occurrences of spikes. This could be explained by the success of the vaccines in decreasing the odds of death from Covid.