Engineering the Future

Sample Report

Created by Diego Velandia
Last compile date: 14/02/2022

Code is available for free access through: https://github.com/Dialvec/r_shiny_latex_report # Introduction Sample datasets used as test subject for the present report describes evolution of confirmed positive cases, deceases and recovered patients along the current Covid-19 epidemics. Datasets are publicly available thanks to the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University.

Datasets come in separate .csv documents and they can be downloaded through CSSE public repository on github.

- Positive cases
- Confirmed deceases
- Recovered patients

R Markdown file is written in a way that users do not have to scroll into dense monolithic code to search for a specific code snippet to modify. Instead, individual file management allow users to quickly navigate, understand, modify or even change the coding structure to make a easy to maintain reporting tool.

```
source("RFunctions/DataSource.R")

df_positives_raw = get_positive_cases_df()
df_deceases_raw  = get_deceases_cases_df()
df_recovered_raw = get_recovered_cases_df()
```

R's source() function allows remote calling of functions, parameters and from .R and .Rmd

Data Exploration

Each dataset comes from a separate source. Therefore we begin by exploring the structure of each dataset.

Shape and variable type

A simple function to describe the size of the data frame can tell us valuable information regarding how to analize and understand the data frame structure.\

Positive Cases

```
## row count: 282
## column count: 758
##  -  character :  2
##  -  integer :  754
##  -  numeric :  2
```

Deceased Cases

```
## row count: 282
## column count: 755
##  -  character :  2
##  -  integer :  751
##  -  numeric :  2
```

Recovered Cases

```
## row count: 267
## column count: 755
##  -  character :  2
##  -  integer :  751
##  -  numeric :  2
```

I can be evidenced that all Datasets contain an amount of columns that can not be easily displayed. Using pre built descriptive methods like summary(df) might be overwhelming due to the high amount of columns in each data frame. Let's begin with the less common data types within the datasets.\

Character and numeric type columns

Since all Datasets share a common property of having 2 character and 2 numeric columns, it is possible that all Datasets have the same 4 columns (fields). We can verify these two columns on each case to verify the hypothesis and, then, start exploring the integer type columns.

```
## [1] "From positive cases dataset"

## columns of type: character
## Province.State
## Country.Region

## [1] "----------------------------"

## [1] "From deceases cases dataset"

## columns of type: character
## Province.State
## Country.Region

## [1] "----------------------------"

## [1] "From recovered cases dataset"

## columns of type: character
## Province.State
## Country.Region
```

Similar scenario can be expected regarding the numeric type variables.

```
## [1] "From positive cases dataset"
```

© 2022 Author(s). This is an open access article licensed under the Creative Commons Attribution License 4.0.
Code is available for public access through: https://github.com/Dialvec/r_shiny_latex_report

```
## columns of type: numeric
## Lat
## Long
```

```
## [1] "--------------------------"
```

```
## [1] "From deceases cases dataset"
```

```
## columns of type: numeric
## Lat
## Long
```

```
## [1] "--------------------------"
```

```
## [1] "From recovered cases dataset"
```

```
## columns of type: numeric
## Lat
## Long
```

From this structure we have two valuable insights: Geographic identifiers are suitable candidates as key for data joining; and time series are stored column-wise for each country and state, which are represented over rows.

Data completion

To verify data completion, we can use a function to verify either if a data frame column contains null values or empty strings. Keep in mind that empty strings are not considered as **na** values.

Positive Cases

```
##                 na_count
## Province.State       195
## Lat                    2
## Long                   2
```

Deceased Cases

```
##                 na_count
## Province.State       195
## Lat                    2
## Long                   2
```

Recovered Cases

```
##                 na_count
## Province.State       196
## Lat                    1
## Long                   1
```

As far as we have obtained insights from the data, **Province.State** is not available for all countries, which adds an extra challenge of analyze and properly aggregate countries with States or Provinces in order to standardize the geographic basis.

*Countries with **Province.State** variable not empty*

```
##        Positives       Deceases      Recovered
## 1      Australia      Australia      Australia
## 2         Canada         Canada          China
## 3          China          China       Denmark
## 4        Denmark        Denmark         France
## 5         France         France    Netherlands
## 6    Netherlands    Netherlands    New Zealand
## 7    New Zealand    New Zealand United Kingdom
## 8 United Kingdom United Kingdom      Australia
```

It is visible that all data frames share almost the same list of countries divided by State. Recovered cases data frame does not include Canada as a state-divided country.This requires a deeper understanding of the State structure for each country before merging them as a single country.

Data Transformation

From the data frames structure depicted through the data exploration phase, we have time series organized column-wise. Such format is quite handy for non-structured databases. In our case the structure shortage comes from the presence of states in some of the countries.

Let's start with exploring the names structure of some of the timestamp columns. Having them in **integer** type column helps us filtering data. Taking a random sample of column names of the timestamp, it can be noticed the way timestamps are stored.

```
## [1] "X11.9.21" "X1.21.22" "X11.4.21"
```

Dates are dot separated and include a **X** prefix that can be omitted to parse dates from **string** format to a manageable format.

```
## # A tibble: 5 x 4
##   Country.Region `8.4.20` `8.5.20` `8.6.20`
##   <chr>             <int>    <int>    <int>
## 1 Afghanistan       36833    36915    36982
## 2 Albania            5750     5889     6016
## 3 Algeria           32504    33055    33626
## 4 Andorra             939      939      944
## 5 Angola             1344     1395     1483
```

New format avoids unnecessary characters and column names can be called more easily.

Additional data preprocessing techniques can be used to manipulate data until the desired shape. Yet to be mentioned, that data cleaning is to be performed. So the current state of the dataset will be very likely to have data quality issues.

Exploratory plots

For a quick plot, let's show the last known state of recovered, deceased and recovered cases.

```
source("RFunctions/PlotData.R")

df_positives <- add_country_code_to_df(df_positives)
df_deceases <- add_country_code_to_df(df_deceases)
df_recovered <- add_country_code_to_df(df_recovered)
```

Here we perform a pretty fast aggregation.Highlighting the fact that results are merely demonstrative and still lack a

proper data quality assurance and aggregation for countries with states.

```
source("RFunctions/DataTransformation.R")

df_positives <- sum_up_countries(df_positives)
df_deceases <- sum_up_countries(df_deceases)
df_recovered <- sum_up_countries(df_recovered)
```

Finally. Make a plot of confirmed cases by 2/10/2022