```
>####################################### PART-B
##################################

>

> #------------------------TASK-1: DATA PREPARATION AND WRANGLING-----------------#

>

>

> rm(list=ls())

> install.packages('tidyverse')

Error in install.packages : Updating loaded packages

> install.packages("dplyr")

Error in install.packages : Updating loaded packages

> test <- require(tidyverse)

>

> library(modelr)

> library(broom)


Attaching package: ‘broom’


The following object is masked from ‘package:modelr’:


    bootstrap


> library(tidypredict)

Error in library(tidypredict) : there is no package called ‘tidypredict’

> library(dplyr)

> library(tidyverse)


Restarting R session...


> install.packages("tidyverse")
```

```
WARNING: Rtools is required to build R packages but is not currently installed.
Please download and install the appropriate version of Rtools before proceeding:


https://cran.rstudio.com/bin/windows/Rtools/

Installing package into ‘C:/Users/User/Documents/R/win-library/4.1’

(as ‘lib’ is unspecified)

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/tidyverse_1.3.1.zip'

Content type 'application/zip' length 430176 bytes (420 KB)

downloaded 420 KB


package ‘tidyverse’ successfully unpacked and MD5 sums checked


The downloaded binary packages are in

        C:\Users\User\AppData\Local\Temp\RtmpGkblVD\downloaded_packages

> library(tidyverse)

-- Attaching packages --------------------------------------------- tidyverse 1.3.1 --

v ggplot2 3.3.5     v purrr   0.3.4

v tibble  3.1.4     v dplyr   1.0.7

v tidyr   1.1.3     v stringr 1.4.0

v readr   2.0.1     v forcats 0.5.1

-- Conflicts ------------------------------------------------ tidyverse_conflicts() --

x dplyr::filter() masks stats::filter()

x dplyr::lag()    masks stats::lag()

> # Task 1: Data Preparation and Wrangling: (20 marks)

> # 1. Load and read the data from the CSV files and store them into dataframes
named

> # appropriately.

>

> countries <- read_csv("data/Countries.csv")

Rows: 208 Columns: 5
```

```
-- Column specification ----------------------------------------------------------------------
Delimiter: ","
chr (2): countryCode, Country
dbl (3): popData2018, GDP, GDP/capita

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.
>
> covid19 <- read_csv("data/Covid19.csv")
Rows: 15029 Columns: 6
-- Column specification ----------------------------------------------------------------------
Delimiter: ","
chr  (3): iso_code, location, continent
dbl  (2): new_cases, new_deaths
date (1): 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.
>
> recovered <- read_csv("data/Recovered.csv")
Rows: 185 Columns: 106
-- Column specification ----------------------------------------------------------------------
Delimiter: ","
chr   (1): Country.Region
dbl (105): 2020.01.22, 2020.01.23, 2020.01.24, 2020.01.25, 2020.01.26,
2020.01.27, 2020.01...

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.
>
```

```
> tests <- read_csv("data/Tests.csv")
Rows: 7859 Columns: 3
-- Column specification --------------------------------------------------------------------
Delimiter: ","
chr  (1): Country Code
dbl  (1): New Tests
date (1): 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.
> # 3. Change the column names in the dataframes were loaded from the following
files
> # accordingly.
> # File Name
> # Ordered New Column Names
> # Covid19.csv Code, Country, Continent, Date, NewCases,
> # NewDeaths
> # Tests.csv Code, Date, NewTests
> # Countries.csv Code, Country, Population, GDP, GDPCapita
> # Recovered.csv Country, Date, Recovered
>
>
> names(covid19) <- c('Code', 'Country', 'Continent', 'Date', 'NewCases',
'NewDeaths')
>
> names(tests) <- c('Code', 'Date', 'NewTests')
>
> names(countries) <- c('Code', 'Country', 'Population', 'GDP', 'GDPCapita')
>
> names(recovered) <- c('Country', 'Date', 'Recovered')
> # 4. Ensure that all dates variables are of date data type and with the same
format across the
> # dataframes.
> # 5. Considering the master dataframe is the one loaded from file “Covid19.csv”,
add new 5
> # variables to it from other files (Recovered.csv, Tests.csv, Countries.csv). The 5
new
> # added variables should be named (“Recovered”, “NewTests”, “Population”,
“GDP”,
> #                              “GDPCapita”) accordingly.
> # [Hint: you can use the merge function to facilitate the alignment of the data in
the different
> #  dataframes.]
>
> covid19 <- covid19 %>%
+   arrange(Code)
>
> covid19 <- merge(x=covid19, y=tests, by=c("Code","Date"), all.x = TRUE)
>
>
> str(recovered)
tibble [19,425 x 3] (S3: tbl_df/tbl/data.frame)
 $ Country  : chr [1:19425] "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan"
...
 $ Date     : chr [1:19425] "2020.01.22" "2020.01.23" "2020.01.24" "2020.01.25" ...
 $ Recovered: num [1:19425] 0 0 0 0 0 0 0 0 0 0 ...
>
> recovered$Date <- as.Date(recovered$Date, "%Y.%m.%d")
>
> str(recovered$Date)
 Date[1:19425], format: "2020-01-22" "2020-01-23" "2020-01-24" "2020-01-25"
"2020-01-26" "2020-01-27" ...
```

```
>
>
> covid19 <- merge(x=covid19, y=recovered, by=c("Country", "Date"),
all.x=TRUE)
>
>
>
> covid19 <- merge(x=covid19, y=countries, by=c("Code", "Country"), all.x =
TRUE)
>
>
> # 6. Check for Nas in all dataframes and change them to Zero.
>
> is.na(covid19)
         Code Country  Date Continent NewCases NewDeaths NewTests Recovered
Population   GDP
   [1,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
   [2,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
   [3,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
   [4,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
   [5,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
   [6,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
   [7,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
   [8,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
   [9,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
```

```
   [10,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [11,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [12,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [13,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [14,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [15,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [16,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [17,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [18,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [19,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [20,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [21,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [22,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [23,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [24,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [25,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [26,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
   [27,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
```

```
   [28,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [29,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [30,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [31,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [32,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [33,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [34,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [35,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [36,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [37,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [38,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [39,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [40,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [41,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [42,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [43,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [44,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE

   [45,] FALSE   FALSE FALSE    FALSE    FALSE    FALSE    TRUE     TRUE     FALSE
FALSE
```

```
  [46,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [47,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [48,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [49,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [50,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [51,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [52,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [53,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [54,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [55,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [56,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [57,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [58,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [59,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [60,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [61,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [62,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [63,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
```

```
  [64,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [65,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE      TRUE      FALSE
FALSE
  [66,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [67,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [68,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [69,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [70,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [71,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [72,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [73,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [74,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [75,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [76,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [77,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [78,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [79,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [80,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
  [81,] FALSE   FALSE FALSE    FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
```

```
   [82,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
   [83,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
   [84,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
   [85,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
   [86,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
   [87,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
   [88,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
   [89,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
   [90,] FALSE   FALSE FALSE     FALSE    FALSE     FALSE     TRUE     FALSE      FALSE
FALSE
         GDPCapita
    [1,]     FALSE
    [2,]     FALSE
    [3,]     FALSE
    [4,]     FALSE
    [5,]     FALSE
    [6,]     FALSE
    [7,]     FALSE
    [8,]     FALSE
    [9,]     FALSE
   [10,]     FALSE
   [11,]     FALSE
   [12,]     FALSE
   [13,]     FALSE
   [14,]     FALSE
```

```
[15,]   FALSE
[16,]   FALSE
[17,]   FALSE
[18,]   FALSE
[19,]   FALSE
[20,]   FALSE
[21,]   FALSE
[22,]   FALSE
[23,]   FALSE
[24,]   FALSE
[25,]   FALSE
[26,]   FALSE
[27,]   FALSE
[28,]   FALSE
[29,]   FALSE
[30,]   FALSE
[31,]   FALSE
[32,]   FALSE
[33,]   FALSE
[34,]   FALSE
[35,]   FALSE
[36,]   FALSE
[37,]   FALSE
[38,]   FALSE
[39,]   FALSE
[40,]   FALSE
[41,]   FALSE
[42,]   FALSE
[43,]   FALSE
[44,]   FALSE
```

```
[45,]    FALSE
[46,]    FALSE
[47,]    FALSE
[48,]    FALSE
[49,]    FALSE
[50,]    FALSE
[51,]    FALSE
[52,]    FALSE
[53,]    FALSE
[54,]    FALSE
[55,]    FALSE
[56,]    FALSE
[57,]    FALSE
[58,]    FALSE
[59,]    FALSE
[60,]    FALSE
[61,]    FALSE
[62,]    FALSE
[63,]    FALSE
[64,]    FALSE
[65,]    FALSE
[66,]    FALSE
[67,]    FALSE
[68,]    FALSE
[69,]    FALSE
[70,]    FALSE
[71,]    FALSE
[72,]    FALSE
[73,]    FALSE
[74,]    FALSE
```

```
  [75,]    FALSE
  [76,]    FALSE
  [77,]    FALSE
  [78,]    FALSE
  [79,]    FALSE
  [80,]    FALSE
  [81,]    FALSE
  [82,]    FALSE
  [83,]    FALSE
  [84,]    FALSE
  [85,]    FALSE
  [86,]    FALSE
  [87,]    FALSE
  [88,]    FALSE
  [89,]    FALSE
  [90,]    FALSE
 [ reached getOption("max.print") -- omitted 14939 rows ]
>
> covid19$NewTests[is.na(covid19$NewTests)]<-0
>
> covid19$Recovered[is.na(covid19$Recovered)]<-0
> # 7. Using existing “Date” variable; add month and week variables to the master dataframe.
> # [Hint: you may use functions from lubridate package]
> # [Hint: To ensure that this task has been finished correctly, when you run head(covid19_data), you
> #  should get results such as in the below image]
>
>
> library(lubridate)
```

```
Attaching package: ‘lubridate’


The following objects are masked from ‘package:base’:


    date, intersect, setdiff, union


>

> covid19 <- covid19 %>%

+

+   mutate(month= month(Date), week=week(Date))

>

> head(covid19)

  Code Country       Date     Continent NewCases NewDeaths NewTests Recovered
Population  GDP

1  ABW   Aruba 2020-03-13 North America        2         0        0         0     105845
2664

2  ABW   Aruba 2020-03-20 North America        2         0        0         0     105845
2664

3  ABW   Aruba 2020-03-24 North America        8         0        0         0     105845
2664

4  ABW   Aruba 2020-03-25 North America        5         0        0         0     105845
2664

5  ABW   Aruba 2020-03-26 North America        2         0        0         0     105845
2664

6  ABW   Aruba 2020-03-27 North America        9         0        0         0     105845
2664

  GDPCapita month week

1     25655     3   11

2     25655     3   12

3     25655     3   12

4     25655     3   13
```

```
5     25655     3   13
6     25655     3   13
> #-----------------------TASK-2: EXPLORATORY DATA ANALYSIS---------------------#
>
> # 1. Add four new variables to the master dataframe (“CumCases”,
“CumDeaths”,
> # “CumRecovered”, “CumTests”) These variables should reflect the cumulative
relevant
> # data up to the date of the observation, i.e CumCases for country “X” at Date
“Y” should
> # reflect the total number of cases in country “X” since the beginning of
recording data till
> # the date “Y”.
> # [Hint: first arrange by date and country, then for each new variable to be added
you need to
> #  group by country and mutate the new column using the cumsum function]
>
> covid19_cum <- covid19
>
> covid19_cum <- covid19_cum %>%
+   arrange(Date, Country) %>%
+   group_by(Country) %>%
+   mutate(CumCases=cumsum(NewCases), CumDeaths=cumsum(NewDeaths),
+          CumRecovered=cumsum(Recovered), CumTests=cumsum(NewTests))
>
> covid19_cum
# A tibble: 15,029 x 17
# Groups:   Country [208]
   Code  Country  Date       Continent NewCases NewDeaths NewTests Recovered
Population    GDP
   <chr> <chr>    <date>     <chr>        <dbl>     <dbl>    <dbl>     <dbl>
<dbl>  <dbl>
```

```
 1 AFG   Afghani~ 2020-01-01 Asia            0        0       0        0   37172386 2.20e4

 2 DZA   Algeria  2020-01-01 Africa          0        0       0        0   42228429 1.68e5

 3 ARM   Armenia  2020-01-01 Europe          0        0       0        0    2951776
1.15e4

 4 AUS   Austral~ 2020-01-01 Oceania         0        0       0        0   24992369
1.41e6

 5 AUT   Austria  2020-01-01 Europe          0        0       0        0    8847037 4.17e5

 6 AZE   Azerbai~ 2020-01-01 Europe          0        0       0        0    9942334
4.07e4

 7 BHR   Bahrain  2020-01-01 Asia            0        0       0        0    1569439 3.53e4

 8 BLR   Belarus  2020-01-01 Europe          0        0       0        0    9485386 5.44e4

 9 BEL   Belgium  2020-01-01 Europe          0        0       0        0   11422068
4.95e5

10 BRA   Brazil   2020-01-01 South Am~       0        0       0        0  209469333
2.06e6

# ... with 15,019 more rows, and 7 more variables: GDPCapita <dbl>, month
<dbl>, week <dbl>,

#   CumCases <dbl>, CumDeaths <dbl>, CumRecovered <dbl>, CumTests <dbl>

>

> # 2. Add two new variables to the master dataframe (“Active”, “FatalityRate”).
Active

> # variable should reflect the infected cases that has not been closed yet (by
either recovery

> # or death), and it could be calculated from (CumCases – (CumDeaths +
CumRecovered)).

> # On the other hand, FatalityRate variable should reflect the percentages of
death to the

> # infected cases up to date and it could be calculated from (CumDeaths /
CumCases).

>

> library("dplyr")

>

> covid19_cum <- covid19_cum %>%

+   arrange(Date, Country) %>%
+   group_by(Country) %>%

+   mutate(Active=(CumCases - (CumDeaths + CumRecovered)),
FatalityRate=(CumDeaths / CumCases))

>

> covid19_cum

# A tibble: 15,029 x 19

# Groups:   Country [208]

   Code  Country  Date       Continent NewCases NewDeaths NewTests Recovered
Population    GDP

   <chr> <chr>    <date>     <chr>        <dbl>     <dbl>    <dbl>     <dbl>
<dbl>  <dbl>

 1 AFG   Afghani~ 2020-01-01 Asia             0         0        0         0   37172386 2.20e4

 2 DZA   Algeria  2020-01-01 Africa           0         0        0         0   42228429 1.68e5

 3 ARM   Armenia  2020-01-01 Europe           0         0        0         0    2951776
1.15e4

 4 AUS   Austral~ 2020-01-01 Oceania          0         0        0         0   24992369
1.41e6

 5 AUT   Austria  2020-01-01 Europe           0         0        0         0    8847037 4.17e5

 6 AZE   Azerbai~ 2020-01-01 Europe           0         0        0         0    9942334
4.07e4

 7 BHR   Bahrain  2020-01-01 Asia             0         0        0         0    1569439 3.53e4

 8 BLR   Belarus  2020-01-01 Europe           0         0        0         0    9485386 5.44e4

 9 BEL   Belgium  2020-01-01 Europe           0         0        0         0   11422068
4.95e5

10 BRA   Brazil   2020-01-01 South Am~        0         0        0         0  209469333
2.06e6

# ... with 15,019 more rows, and 9 more variables: GDPCapita <dbl>, month
<dbl>, week <dbl>,

#   CumCases <dbl>, CumDeaths <dbl>, CumRecovered <dbl>, CumTests <dbl>,
Active <dbl>,

#   FatalityRate <dbl>

>

> # 3. Add four new variables to the master dataframe (“Cases_1M_Pop”,
“Deaths_1M_Pop”,
> # “Recovered_1M_Pop”, “Tests_1M_Pop”) These variables should reflect the
cumulative

> # relevant rate per one million of the corresponding country population, (i.e
Cases_1M_Pop

> # for country “X” at Date “Y” should reflect the total number of new cases up to
date “Y”

> # per million people of country “X” population)

> # [Hint: Cases_1M_Pop = CumCases*(10^6) / Population)]

>

>

> covid19_cum <- covid19_cum %>%

+   arrange(Date, Country) %>%

+   group_by(Country) %>%

+   mutate(Cases_1M_Pop=(CumCases*(10^6) / Population),

+          Deaths_1M_Pop=(CumDeaths*(10^6) / Population),

+          Recovered_1M_Pop=(CumRecovered*(10^6) / Population),

+          Tests_1M_Pop=(CumTests*(10^6) / Population))

>

> covid19_cum

# A tibble: 15,029 x 23

# Groups:   Country [208]

   Code  Country  Date       Continent NewCases NewDeaths NewTests Recovered
Population    GDP

   <chr> <chr>    <date>     <chr>        <dbl>     <dbl>    <dbl>     <dbl>
<dbl>  <dbl>

 1 AFG   Afghani~ 2020-01-01 Asia             0         0        0         0   37172386 2.20e4

 2 DZA   Algeria  2020-01-01 Africa           0         0        0         0   42228429 1.68e5

 3 ARM   Armenia  2020-01-01 Europe           0         0        0         0    2951776
1.15e4

 4 AUS   Austral~ 2020-01-01 Oceania          0         0        0         0   24992369
1.41e6

 5 AUT   Austria  2020-01-01 Europe           0         0        0         0    8847037 4.17e5
 6 AZE   Azerbai~ 2020-01-01 Europe           0         0        0         0    9942334
4.07e4
 7 BHR   Bahrain  2020-01-01 Asia             0         0        0         0    1569439 3.53e4
 8 BLR   Belarus  2020-01-01 Europe           0         0        0         0    9485386 5.44e4
 9 BEL   Belgium  2020-01-01 Europe           0         0        0         0   11422068
4.95e5
10 BRA   Brazil   2020-01-01 South Am~        0         0        0         0  209469333
2.06e6
# ... with 15,019 more rows, and 13 more variables: GDPCapita <dbl>, month
<dbl>, week <dbl>,
#   CumCases <dbl>, CumDeaths <dbl>, CumRecovered <dbl>, CumTests <dbl>,
Active <dbl>,
#   FatalityRate <dbl>, Cases_1M_Pop <dbl>, Deaths_1M_Pop <dbl>,
Recovered_1M_Pop <dbl>,
#   Tests_1M_Pop <dbl>
>
> # 4. Find the day with the highest reported death toll across the world. Print the
date and the
> # death toll of that day.
>
> covid19_Max_cul_Deaths_toll <- covid19_cum %>%
+   group_by(Date) %>%
+   summarise(cul_deaths_toll_per_date=sum(CumDeaths)) %>%
+   summarise(Date = Date[which.max(cul_deaths_toll_per_date)],
+             max_cul_deaths_toll=max(cul_deaths_toll_per_date))
>
> covid19_Max_cul_Deaths_toll
# A tibble: 1 x 2
  Date       max_cul_deaths_toll
  <date>                   <dbl>
1 2020-05-05              250970
>
```

```
> # This is for calculating the highest deaths record date across the world
>
> covid19_Max_Deaths_toll <- covid19_cum %>%
+   group_by(Date) %>%
+   summarise(deaths_toll_per_date=sum(NewDeaths)) %>%
+   summarise(Date = Date[which.max(deaths_toll_per_date)],
+             max_deaths_toll_per_day=max(deaths_toll_per_date))
>
> covid19_Max_Deaths_toll
# A tibble: 1 x 2
  Date       max_deaths_toll_per_day
  <date>                       <dbl>
1 2020-04-16                   10520
> # 5. Build a graph to show how the cumulative data of (Infected Cases, Deaths,
Recovered,
> # Tests) change over the time for the whole world collectively.
> # [Hint: Use geom_line, use log for Y axis for better presentation, Use different
colour to
> #  distinguish between new cases, deaths, and recovered]
>
>
>
>
> max_cum_by_country_month <- covid19_cum %>%
+   group_by(month, Country) %>%
+   summarise(highest_cumcase=max(CumCases),
highest_cumdeaths=max(CumDeaths),
+             highest_cumrecovered=max(CumRecovered),
highest_cumtest=max(CumTests))
`summarise()` has grouped output by 'month'. You can override using the `.groups`
argument.
```

```
>

> max_cum_by_country_month

# A tibble: 747 x 6

# Groups:   month [5]

   month Country     highest_cumcase highest_cumdeaths highest_cumrecovered
highest_cumtest

   <dbl> <chr>                 <dbl>             <dbl>                <dbl>          <dbl>

 1     1 Afghanistan              0                 0                    0               0

 2     1 Algeria                0                 0                    0               0

 3     1 Armenia                  0                 0                    0               0

 4     1 Australia               7                 0                    2               0

 5     1 Austria                0                 0                    0               0

 6     1 Azerbaijan               0                 0                    0               0

 7     1 Bahrain                 0                 0                    0               0

 8     1 Belarus                 0                 0                    0               0

 9     1 Belgium                  0                 0                    0               0

10     1 Brazil                 0                 0                    0               0

# ... with 737 more rows

>

>

>

> covid_graph <- max_cum_by_country_month %>%

+   group_by(month) %>%

+   summarise(ww_cum_cases=sum(highest_cumcase),
ww_Deaths=sum(highest_cumdeaths),

+             ww_cum_Recovered=sum(highest_cumrecovered),
ww_cum_Tests=sum(highest_cumtest))

>

> covid_graph

# A tibble: 5 x 5

  month ww_cum_cases ww_Deaths ww_cum_Recovered ww_cum_Tests
 <dbl>       <dbl>     <dbl>         <dbl>      <dbl>
1   1       9826     213          222       5199
2   2      84498     2915         39772      159078
3   3     776582     37904         176549     5720246
4   4     3131210    227328        1012926     29230507
5   5     3543867    250970        1195347     33829516
>
>
> require(dplyr)
>
> require(scales)
Loading required package: scales

Attaching package: ‘scales’

The following object is masked from ‘package:purrr’:

    discard

The following object is masked from ‘package:readr’:

    col_factor

>
> covid_graph <- covid_graph %>%
+   gather(ww_cum_data, cum_value, -month)
>
> ggplot(covid_graph, aes(x=month, y=cum_value, group=ww_cum_data,
color=ww_cum_data)) +
+   theme_bw() +
+   geom_line() +

+   geom_point()+

+   ggtitle("The world wide cumulative data for cases, deaths, recovered and Tests
number") +

+   scale_y_log10()

> # Another way (Group by Date)

>

> max_cum_by_country_Date <- covid19_cum %>%

+   select(Date, Country, CumCases, CumDeaths, CumRecovered, CumTests) %>%

+   group_by(Date) %>%

+   summarise(case_date= sum(CumCases), test_date= sum(CumTests),

+             recovered_date=sum(CumRecovered), death_date=sum(CumDeaths))

>

> max_cum_by_country_Date

# A tibble: 126 x 5

   Date       case_date test_date recovered_date death_date

   <date>         <dbl>     <dbl>          <dbl>      <dbl>

 1 2020-01-01        27         5              0          0

 2 2020-01-02        27        21              0          0

 3 2020-01-03        44        39              0          0

 4 2020-01-04        44        45              0          0

 5 2020-01-05        59        59              0          0

 6 2020-01-06        59        88              0          0

 7 2020-01-07        59       115              0          0

 8 2020-01-08        59       131              0          0

 9 2020-01-09        59       176              0          0

10 2020-01-10        59       215              0          0

# ... with 116 more rows

>

>
```

```
> require(dplyr)

>

> require(scales)

>

> max_cum_by_country_Date <- max_cum_by_country_Date %>%

+   gather(ww_cum_data, cum_value, -Date)

>

> ggplot(max_cum_by_country_Date, aes(x=Date, y=cum_value,
group=ww_cum_data, color=ww_cum_data)) +

+   theme_bw() +

+   geom_line() +

+   ggtitle("The world wide cumulative data for cases, deaths, recovered and Tests
number (Date)") +

+   scale_y_log10()

Warning message:

Transformation introduced infinite values in continuous y-axis

> # 6. Extract the last day (05/05/2020) data and save it in a separate dataframe
called

> # “lastDay_data”.

> # [Hint: use filter function with Date = "2020-05-05"]

>

> Last_day_data <- covid19_cum %>%

+   filter(Date == "2020-05-05")

>

> Last_day_data

# A tibble: 207 x 23

# Groups:   Country [207]

   Code  Country  Date       Continent NewCases NewDeaths NewTests Recovered
Population    GDP

   <chr> <chr>    <date>     <chr>        <dbl>     <dbl>    <dbl>     <dbl>
<dbl>  <dbl>
```

```
 1 AFG   Afghani~ 2020-05-05 Asia          190        5       0       24   37172386
21992

 2 ALB   Albania  2020-05-05 Europe          8        0       0        0    2866376  13039

 3 DZA   Algeria  2020-05-05 Africa        174        2       0       69   42228429
167555

 4 AND   Andorra  2020-05-05 Europe          2        0       0       15      77006   3278

 5 AGO   Angola   2020-05-05 Africa          0        0       0        0   30809762 126505

 6 AIA   Anguilla 2020-05-05 North Am~       0        0       0        0      14731    311

 7 ATG   Antigua~ 2020-05-05 North Am~       0        0       0        1      96286
1248

 8 ARG   Argenti~ 2020-05-05 South Am~     104       14       0       30   44494502
637486

 9 ARM   Armenia  2020-05-05 Europe        121        4       0       40    2951776
11536

10 ABW   Aruba    2020-05-05 North Am~       0        0       0        0     105845
2664

# ... with 197 more rows, and 13 more variables: GDPCapita <dbl>, month <dbl>,
week <dbl>,

#   CumCases <dbl>, CumDeaths <dbl>, CumRecovered <dbl>, CumTests <dbl>,
Active <dbl>,

#   FatalityRate <dbl>, Cases_1M_Pop <dbl>, Deaths_1M_Pop <dbl>,
Recovered_1M_Pop <dbl>,

#   Tests_1M_Pop <dbl>

> # 7. Based on the last day data, extract the whole records of the top 10 countries
worldwide

> # that have current active cases, total confirmed cases, and fatality rate in
separate

> # dataframes (i.e. top10activeW, top10casesW, top10fatalityW, top10testsMW).

> # [Hint: you can use head(arranged_data, n=10) to get the top 10 records]

>

> # Current active case

>

> top10activeW <- Last_day_data %>%

+   arrange(desc(Active)) %>%
+   head(top10activeW, n=10)

>

> top10activeW

# A tibble: 10 x 23

# Groups:   Country [10]

   Code  Country  Date       Continent NewCases NewDeaths NewTests Recovered
Population    GDP

   <chr> <chr>    <date>     <chr>        <dbl>     <dbl>    <dbl>     <dbl>
<dbl>  <dbl>

 1 USA   United ~ 2020-05-05 North Am~    22593      1252        0      2611
327167434 1.95e7

 2 GBR   United ~ 2020-05-05 Europe        3985       288    69839        16   66488991
2.63e6

 3 RUS   Russia   2020-05-05 Europe       10581        76   157114      1770
144478050 1.53e6

 4 ITA   Italy    2020-05-05 Europe        1221       195    55263      2352   60431283
1.94e6

 5 ESP   Spain    2020-05-05 Europe         545       164        0      2143   46723749
1.31e6

 6 FRA   France   2020-05-05 Europe         576       306        0      1366   66987244
2.58e6

 7 BRA   Brazil   2020-05-05 South Am~     6633       296        0      2406  209469333
2.06e6

 8 TUR   Turkey   2020-05-05 Asia          1614        64    33283      5119   82319724
8.52e5

 9 NLD   Netherl~ 2020-05-05 Europe         199        26        0         1   17231017
8.31e5

10 IND   India    2020-05-05 Asia          3900       195    84713      1295 1352617328
2.58e6

# ... with 13 more variables: GDPCapita <dbl>, month <dbl>, week <dbl>,
CumCases <dbl>,

#   CumDeaths <dbl>, CumRecovered <dbl>, CumTests <dbl>, Active <dbl>,
FatalityRate <dbl>,

#   Cases_1M_Pop <dbl>, Deaths_1M_Pop <dbl>, Recovered_1M_Pop <dbl>,
Tests_1M_Pop <dbl>
```

```
>
> # Total confirmed cases
>
> top10casesW <- Last_day_data %>%
+   arrange(desc(CumCases)) %>%
+   head(top10casesW, n=10)
>
> top10casesW
# A tibble: 10 x 23
# Groups:   Country [10]
   Code  Country  Date       Continent NewCases NewDeaths NewTests Recovered
Population    GDP
   <chr> <chr>    <date>     <chr>        <dbl>     <dbl>    <dbl>     <dbl>
<dbl>  <dbl>
 1 USA   United ~ 2020-05-05 North Am~    22593      1252        0      2611
327167434 1.95e7
 2 ESP   Spain    2020-05-05 Europe         545       164        0      2143   46723749
1.31e6
 3 ITA   Italy    2020-05-05 Europe        1221       195    55263      2352   60431283
1.94e6
 4 GBR   United ~ 2020-05-05 Europe        3985       288    69839        16   66488991
2.63e6
 5 DEU   Germany  2020-05-05 Europe         685       139        0      2400   82927922
3.69e6
 6 RUS   Russia   2020-05-05 Europe       10581        76   157114      1770
144478050 1.53e6
 7 FRA   France   2020-05-05 Europe         576       306        0      1366   66987244
2.58e6
 8 TUR   Turkey   2020-05-05 Asia          1614        64    33283      5119   82319724
8.52e5
 9 BRA   Brazil   2020-05-05 South Am~     6633       296        0      2406  209469333
2.06e6
10 IRN   Iran     2020-05-05 Asia          1223        74    11255      1096   81800269
4.61e5
```

```
# ... with 13 more variables: GDPCapita <dbl>, month <dbl>, week <dbl>,
CumCases <dbl>,
#   CumDeaths <dbl>, CumRecovered <dbl>, CumTests <dbl>, Active <dbl>,
FatalityRate <dbl>,
#   Cases_1M_Pop <dbl>, Deaths_1M_Pop <dbl>, Recovered_1M_Pop <dbl>,
Tests_1M_Pop <dbl>
>
> # Fatality rate
>
> top10fatalityW <- Last_day_data %>%
+   arrange(desc(FatalityRate)) %>%
+   head(top10fatalityW, n=10)
>
> top10fatalityW
# A tibble: 10 x 23
# Groups:   Country [10]
   Code  Country  Date       Continent NewCases NewDeaths NewTests Recovered
Population    GDP
   <chr> <chr>    <date>     <chr>        <dbl>     <dbl>    <dbl>     <dbl>
<dbl>  <dbl>
 1 NIC   Nicarag~ 2020-05-05 North Am~        1         0        0         0    6465513
1.38e4
 2 COM   Comoros  2020-05-05 Africa           1         1        0         0     832322 6.48e2
 3 FRA   France   2020-05-05 Europe         576       306        0      1366   66987244
2.58e6
 4 SXM   Sint Ma~ 2020-05-05 North Am~        0         0        0         0      41486
1.06e3
 5 YEM   Yemen    2020-05-05 Asia             2         0        0         0   28498687 2.80e4
 6 BEL   Belgium  2020-05-05 Europe         361        80        0        63   11422068
4.95e5
 7 GBR   United ~ 2020-05-05 Europe        3985       288    69839        16   66488991
2.63e6
 8 VGB   British~ 2020-05-05 North Am~        1         0        0         0      29802
9.02e2
```

```
 9 MNP   Norther~ 2020-05-05 Oceania          0         0        0         0      56882 1.39e3

10 ITA   Italy    2020-05-05 Europe        1221       195    55263      2352   60431283
1.94e6

# ... with 13 more variables: GDPCapita <dbl>, month <dbl>, week <dbl>,
CumCases <dbl>,

#   CumDeaths <dbl>, CumRecovered <dbl>, CumTests <dbl>, Active <dbl>,
FatalityRate <dbl>,

#   Cases_1M_Pop <dbl>, Deaths_1M_Pop <dbl>, Recovered_1M_Pop <dbl>,
Tests_1M_Pop <dbl>

>

> # Total tests

>

> top10testsW <- Last_day_data %>%

+   arrange(desc(CumTests)) %>%

+   head(top10testsW, n=10)

>

> top10testsW

# A tibble: 10 x 23

# Groups:   Country [10]

   Code  Country  Date       Continent NewCases NewDeaths NewTests Recovered
Population    GDP

   <chr> <chr>    <date>     <chr>        <dbl>     <dbl>    <dbl>     <dbl>
<dbl>  <dbl>

 1 USA   United ~ 2020-05-05 North Am~    22593      1252        0      2611
327167434 1.95e7

 2 RUS   Russia   2020-05-05 Europe       10581        76   157114      1770
144478050 1.53e6

 3 DEU   Germany  2020-05-05 Europe         685       139        0      2400   82927922
3.69e6

 4 ITA   Italy    2020-05-05 Europe        1221       195    55263      2352   60431283
1.94e6

 5 ESP   Spain    2020-05-05 Europe         545       164        0      2143   46723749
1.31e6
```

```
 6 TUR   Turkey   2020-05-05 Asia          1614        64    33283      5119   82319724
8.52e5

 7 IND   India    2020-05-05 Asia          3900       195    84713      1295 1352617328
2.58e6

 8 GBR   United ~ 2020-05-05 Europe        3985       288    69839        16   66488991
2.63e6

 9 CAN   Canada   2020-05-05 North Am~     1298       172    21199       976
37058856 1.65e6

10 FRA   France   2020-05-05 Europe         576       306        0      1366   66987244
2.58e6

# ... with 13 more variables: GDPCapita <dbl>, month <dbl>, week <dbl>,
CumCases <dbl>,

#   CumDeaths <dbl>, CumRecovered <dbl>, CumTests <dbl>, Active <dbl>,
FatalityRate <dbl>,

#   Cases_1M_Pop <dbl>, Deaths_1M_Pop <dbl>, Recovered_1M_Pop <dbl>,
Tests_1M_Pop <dbl>

> # 8. Based on the last day data, print the up to date confirmed, death, recovered
cases as well

> # as the tests for every continent.

>

> up_to_date_data <- Last_day_data %>%

+   group_by(Continent) %>%

+   summarise(utd_confirmed_cases=sum(CumCases),
utd_deaths=sum(CumDeaths),

+             utd_recovered=sum(CumRecovered), utd_tests=sum(CumTests))

>

>

> up_to_date_data

# A tibble: 6 x 5

  Continent     utd_confirmed_cases utd_deaths utd_recovered utd_tests

  <chr>                       <dbl>      <dbl>         <dbl>     <dbl>

1 Africa                      47124       1845         16317    618154

2 Asia                       567862      19991        313323   6010340
```

```
3 Europe                  1406374     141780        537696  17013488
4 North America            1290176      75981        238452   8447832
5 Oceania                     8579        122          7313    820684
6 South America             223752      11251         82246    919018
> # 9. Build a graph to show the total number of cases over the time for the top 10 countries that
> # have been obtained in question 7 (Use log for Y axis for better presentation).
> # [Hint: first you need to get the data of the top-10 countries and then plot their lines]
>
>
> top10casesW$Country
 [1] "United States of America" "Spain"                    "Italy"
 [4] "United Kingdom"           "Germany"                  "Russia"
 [7] "France"                   "Turkey"                   "Brazil"
[10] "Iran"
>
> all_time_top10_data <- covid19_cum %>%
+   filter(Country %in% top10casesW$Country)
>
> all_time_top10_data
# A tibble: 1,186 x 23
# Groups:   Country [10]
   Code  Country  Date       Continent NewCases NewDeaths NewTests Recovered Population    GDP
   <chr> <chr>    <date>     <chr>        <dbl>     <dbl>    <dbl>     <dbl>      <dbl>  <dbl>
 1 BRA   Brazil   2020-01-01 South Am~        0         0        0         0  209469333 2.06e6
 2 FRA   France   2020-01-01 Europe           0         0        0         0   66987244 2.58e6
 3 DEU   Germany  2020-01-01 Europe           0         0        0         0   82927922 3.69e6
 4 IRN   Iran     2020-01-01 Asia            0         0        0         0   81800269 4.61e5

 5 ITA   Italy    2020-01-01 Europe          0         0        0         0   60431283 1.94e6

 6 RUS   Russia   2020-01-01 Europe          0         0        0         0  144478050
1.53e6

 7 ESP   Spain    2020-01-01 Europe          0         0        0         0   46723749 1.31e6

 8 GBR   United ~ 2020-01-01 Europe          0         0        0         0   66488991
2.63e6

 9 USA   United ~ 2020-01-01 North Am~        0         0        0         0  327167434
1.95e7

10 BRA   Brazil   2020-01-02 South Am~        0         0        0         0  209469333
2.06e6

# ... with 1,176 more rows, and 13 more variables: GDPCapita <dbl>, month
<dbl>, week <dbl>,

#   CumCases <dbl>, CumDeaths <dbl>, CumRecovered <dbl>, CumTests <dbl>,
Active <dbl>,

#   FatalityRate <dbl>, Cases_1M_Pop <dbl>, Deaths_1M_Pop <dbl>,
Recovered_1M_Pop <dbl>,

#   Tests_1M_Pop <dbl>

>

> unique(all_time_top10_data[c("Country")])

# A tibble: 10 x 1

# Groups:   Country [10]

   Country

   <chr>

 1 Brazil

 2 France

 3 Germany

 4 Iran

 5 Italy

 6 Russia

 7 Spain

 8 United Kingdom
 9 United States of America

10 Turkey

>

> all_time_top10_cases <- all_time_top10_data[ , c("Country", "Date", "NewCases",
"CumCases", "month")]

>

> all_time_top10_cases <- all_time_top10_cases %>%

+   group_by(month, Country) %>%

+   summarise(highest_cumcases_top10=max(CumCases),
newcases_top10=sum(NewCases))

`summarise()` has grouped output by 'month'. You can override using the `.groups`
argument.

>

> all_time_top10_cases

# A tibble: 48 x 4

# Groups:   month [5]

   month Country                  highest_cumcases_top10 newcases_top10

   <dbl> <chr>                                     <dbl>          <dbl>

 1     1 Brazil                                        0              0

 2     1 France                                        6              6

 3     1 Germany                                       5              5

 4     1 Iran                                          0              0

 5     1 Italy                                         3              3

 6     1 Russia                                        0              0

 7     1 Spain                                         0              0

 8     1 United Kingdom                                2              2

 9     1 United States of America                      6              6

10     2 Brazil                                        1              1

# ... with 38 more rows

>
```

```
> ggplot(all_time_top10_cases, aes(x=month, y=highest_cumcases_top10,
group=Country, color=Country)) +
+   theme_bw() +
+   geom_line() +
+   geom_point() +
+   ggtitle("The world wide cumulative cases for top 10 countries") +
+   scale_y_log10()
Warning messages:
1: Transformation introduced infinite values in continuous y-axis
2: Transformation introduced infinite values in continuous y-axis
> ggplot(all_time_top10_cases, aes(x=month, y=newcases_top10, group=Country,
color=Country)) +
+   theme_bw() +
+   geom_line() +
+   geom_point() +
+   ggtitle("The world wide new cases for top 10 countries") +
+   scale_y_log10()
Warning messages:
1: Transformation introduced infinite values in continuous y-axis
2: Transformation introduced infinite values in continuous y-axis
> # 10. Build a graph for the top 10 countries with current highest active cases
which was
> # obtained previously in question 7. The graph should have one subgraph (i.e.
using facet
> # function) for each of these countries, every subgraph should show how the new
cases, new
> # deaths, and new recovered cases were changing over time (Use log for Y axis
for better
> # presentation, Use different colour to distinguish between new cases, deaths,
and
> # recovered).
> # [hint: geom_line function with date on x_axis and each of the values of the
variables in y_axis]
```

```
>
> top10activeW$Country
 [1] "United States of America" "United Kingdom"           "Russia"
 [4] "Italy"                    "Spain"                    "France"
 [7] "Brazil"                   "Turkey"                   "Netherlands"
[10] "India"
>
> all_time_top10active_data <- covid19_cum %>%
+   filter(Country %in% top10activeW$Country)
>
> all_time_top10active_data
# A tibble: 1,185 x 23
# Groups:   Country [10]
   Code  Country  Date       Continent NewCases NewDeaths NewTests Recovered
Population    GDP
   <chr> <chr>    <date>     <chr>        <dbl>     <dbl>    <dbl>     <dbl>
<dbl>  <dbl>
 1 BRA   Brazil   2020-01-01 South Am~        0         0        0         0  209469333
2.06e6
 2 FRA   France   2020-01-01 Europe           0         0        0         0   66987244 2.58e6
 3 IND   India    2020-01-01 Asia             0         0        0         0 1352617328 2.58e6
 4 ITA   Italy    2020-01-01 Europe           0         0        0         0   60431283 1.94e6
 5 NLD   Netherl~ 2020-01-01 Europe           0         0        0         0   17231017
8.31e5
 6 RUS   Russia   2020-01-01 Europe           0         0        0         0  144478050
1.53e6
 7 ESP   Spain    2020-01-01 Europe           0         0        0         0   46723749 1.31e6
 8 GBR   United ~ 2020-01-01 Europe           0         0        0         0   66488991
2.63e6
 9 USA   United ~ 2020-01-01 North Am~        0         0        0         0  327167434
1.95e7
10 BRA   Brazil   2020-01-02 South Am~        0         0        0         0  209469333
2.06e6
```

```
# ... with 1,175 more rows, and 13 more variables: GDPCapita <dbl>, month
<dbl>, week <dbl>,

#   CumCases <dbl>, CumDeaths <dbl>, CumRecovered <dbl>, CumTests <dbl>,
Active <dbl>,

#   FatalityRate <dbl>, Cases_1M_Pop <dbl>, Deaths_1M_Pop <dbl>,
Recovered_1M_Pop <dbl>,

#   Tests_1M_Pop <dbl>

>

> all_time_top10active_cases <- all_time_top10active_data[ , c("Country", "Date",
"NewCases", "NewDeaths", "Recovered","month")]

>

> all_time_top10active_cases <- all_time_top10active_cases %>%

+   group_by(month, Country) %>%

+   summarise(newcases_top10active=sum(NewCases),
newdeaths_top10active=sum(NewDeaths),

+             recovered_top10active=sum(Recovered))

`summarise()` has grouped output by 'month'. You can override using the `.groups`
argument.

>

> all_time_top10active_cases <- all_time_top10active_cases %>%

+   gather(Top10active_Data, value, -month, -Country)

>

> all_time_top10active_cases

# A tibble: 144 x 4

# Groups:   month [5]

  month Country              Top10active_Data     value

  <dbl> <chr>                <chr>                <dbl>

 1     1 Brazil              newcases_top10active     0

 2     1 France              newcases_top10active     6

 3     1 India               newcases_top10active     1

 4     1 Italy               newcases_top10active     3

 5     1 Netherlands         newcases_top10active     0
 6     1 Russia                   newcases_top10active     0
 7     1 Spain                    newcases_top10active     0
 8     1 United Kingdom           newcases_top10active     2
 9     1 United States of America newcases_top10active     6
10     2 Brazil                   newcases_top10active     1
# ... with 134 more rows
>
> top10activeW %>%
+   ggplot(aes(x=Country, y=Active, group=Country, color=Country, fill=Country)) +
+   geom_bar(stat='identity')+
+   theme_bw()
> all_time_top10active_cases %>%
+   ggplot(aes(x=month, y=value, group=Top10active_Data, color=Top10active_Data)) +
+   geom_line() +
+   theme_bw() +
+   facet_wrap(~Country, scale="free") +
+   scale_y_log10()
Warning message:
Transformation introduced infinite values in continuous y-axis
> #------------------------TASK-3:DATA-DRIVEN MODELLING--------------------------#
> library(modelr)
> library(broom)

Attaching package: ‘broom’

The following object is masked from ‘package:modelr’:

    bootstrap
```

```
> # 1. Based on the data of the last day, that you have extracted in the previous
task, create a

> # separate dataframe named "cor_data" with the data of these variables

> # (CumCases, CumTests, Population, GDP, GDPCapita).

> # [Hint: you can use select function on the lastday_data dataframe]

> cor_data <- Last_day_data[ , c("CumCases", "CumTests", "Population", "GDP",
"GDPCapita")]

> cor_data

# A tibble: 207 x 5

   CumCases CumTests Population    GDP GDPCapita

      <dbl>    <dbl>      <dbl>  <dbl>     <dbl>

 1     2894        0   37172386  21992       619

 2      803        0    2866376  13039      4450

 3     4648        0   42228429 167555      4055

 4      750        0      77006   3278     39153

 5       35        0   30809762 126505      4247

 6        3        0      14731    311     29493

 7       25        0      96286   1248     14803

 8     4874    58685   44494502 637486     14400

 9     2507        0    2951776  11536      3937

10      100        0     105845   2664     25655

# ... with 197 more rows

> # 2. Compute the correlation matrix between the variables of the “cor_data” and
visualise

> # this correlation matrix.

>

> cor(cor_data)

            CumCases  CumTests  Population       GDP   GDPCapita

CumCases   1.0000000 0.8897823  0.23201137 0.8515318  0.14270183

CumTests   0.8897823 1.0000000  0.23525764 0.7318026  0.14273118
```

```
Population 0.2320114 0.2352576  1.00000000 0.5642929 -0.07901111
GDP        0.8515318 0.7318026  0.56429288 1.0000000  0.12807801
GDPCapita  0.1427018 0.1427312 -0.07901111 0.1280780  1.00000000
> library(GGally)
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
> ggcorr(cor_data, label=TRUE, label_alpha = TRUE)
> # 3. Divide the cor_data into training and testing, where training data represent
65%
> # of the number of rows.
>
> sample_idx <- sample(c(TRUE, FALSE), nrow(cor_data), replace = T, prob =
c(0.65,0.35))
> train <- cor_data[sample_idx, ]
> test  <- cor_data[!sample_idx, ]
>
> train
# A tibble: 137 x 5
   CumCases CumTests Population    GDP GDPCapita
      <dbl>    <dbl>      <dbl>  <dbl>     <dbl>
 1      750        0      77006   3278     39153
 2       35        0   30809762 126505      4247
 3        3        0      14731    311     29493
 4       25        0      96286   1248     14803
 5     4874    58685   44494502 637486     14400
 6     2507        0    2951776  11536      3937
 7      100        0     105845   2664     25655
 8    15621   285883    8847037 416835     47718
 9     1984        0    9942334  40748      4146
```

```
10       83        0     385640  11791     29825
# ... with 127 more rows
>
> test
# A tibble: 70 x 5
   CumCases CumTests Population     GDP GDPCapita
      <dbl>    <dbl>      <dbl>   <dbl>     <dbl>
 1     2894        0   37172386   21992       619
 2      803        0    2866376   13039      4450
 3     4648        0   42228429  167555      4055
 4     6825   664756   24992369 1408675     57613
 5     3533   155501    1569439   35325     23688
 6       82        0     286641    4353     16494
 7      115        0      63968    5601    102192
 8   107780        0  209469333 2055512      9821
 9        7        0      29802     902     31917
10     1652    50303    7024216   58222      8218
# ... with 60 more rows
> # 4. Train a linear regression model to predict cumulative cases from the GDP of the
> # countries. Then, evaluate this model on the test data and print the root mean
> # square error value.
>
> single_model <- lm(CumCases ~ GDP, data = train)
> print(single_model)


Call:
lm(formula = CumCases ~ GDP, data = train)


Coefficients:
```

```
(Intercept)          GDP
 -1.559e+03    5.746e-02

>
> summary(single_model)

Call:
lm(formula = CumCases ~ GDP, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-263165    -315    1294    1590  144053

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.559e+03  2.827e+03  -0.551    0.582
GDP          5.746e-02  1.576e-03  36.447   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 32330 on 135 degrees of freedom
Multiple R-squared:  0.9077,       Adjusted R-squared:  0.9071
F-statistic:  1328 on 1 and 135 DF,  p-value: < 2.2e-16

>
>
> #Test model on the test set
>
> test$Predicted_1 <- predict(single_model, test)
>
```

```
> # Compute the residual mean square error (RMSE) to evaluate the output of the
model.

>

> actuals <- test$CumCases

> predictions <- test$Predicted_1

>

> sqrt(mean((predictions-actuals)^2))

[1] 75431.51

>

>

> plot(single_model)

Hit <Return> to see next plot: plot(single_model)

Hit <Return> to see next plot: plot(single_model)

Hit <Return> to see next plot: plot(single_model)

Hit <Return> to see next plot: plot(single_model)

> # 5. Train another linear regression model to predict cumulative cases from all
the

> # other variables. Then, evaluate this model on the test data and print the root

> # mean square error value.

>

> multi_model <- lm(CumCases ~ ., data = train)

> print(multi_model)


Call:

lm(formula = CumCases ~ ., data = train)


Coefficients:

(Intercept)     CumTests   Population          GDP    GDPCapita

 -6.707e+02    4.822e-02   -9.689e-05    4.199e-02   -5.008e-02
```

```
>
> summary(multi_model)

Call:
lm(formula = CumCases ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-183526     102    1400    2832  104289 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -6.707e+02  2.706e+03  -0.248    0.805    
CumTests     4.822e-02  5.258e-03   9.169 8.22e-16 ***
Population  -9.689e-05  1.842e-05  -5.259 5.68e-07 ***
GDP          4.199e-02  2.328e-03  18.042  < 2e-16 ***
GDPCapita   -5.008e-02  9.029e-02  -0.555    0.580    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 24410 on 132 degrees of freedom
Multiple R-squared:  0.9486,	Adjusted R-squared:  0.947 
F-statistic: 608.9 on 4 and 132 DF,  p-value: < 2.2e-16

>
> # Test the second model on the testing data and evaluate its performance using RMSE metrics
>
> test$predicted_2 <- predict(multi_model, test)
>
```

```
> actuals <- test$CumCases
> predictions <- test$predicted_2
>
> sqrt(mean((predictions-actuals)^2))
[1] 39875.78
>
> plot(multi_model)
Hit <Return> to see next plot: plot(multi_model)
Hit <Return> to see next plot: plot(multi_model)
Hit <Return> to see next plot: plot(multi_model)
Hit <Return> to see next plot: plot(multi_model)
> ggplot(cor_data, aes(GDP)) +
+   geom_histogram(aes(y = ..density..), fill = "aquamarine3") +
+   geom_density(color = "red")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
> ggplot(cor_data, aes(GDPCapita)) +
+   geom_histogram(aes(y = ..density..), fill = "aquamarine3") +
+   geom_density(color = "red")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
> ggplot(cor_data, aes(CumTests)) +
+   geom_histogram(aes(y = ..density..), fill = "aquamarine3") +
+   geom_density(color = "red")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
> ggplot(cor_data, aes(Population)) +
+   geom_histogram(aes(y = ..density..), fill = "aquamarine3") +
+   geom_density(color = "red")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
> # For analysis:
>
> top10casesW %>%
+   ggplot(aes(x=Country, y=CumCases, group=Country, color=Country, fill=Country)) +
+   geom_bar(stat='identity')+
+   theme_bw()
> top10casesW %>%
+   ggplot(aes(x=Country, y=CumCases, group=Country, color=Country, fill=Country)) +
+   geom_bar(stat='identity')+
+   theme_bw()
> top10fatalityW %>%
+   ggplot(aes(x=Country, y=FatalityRate, group=Country, color=Country, fill=Country)) +
+   geom_bar(stat='identity')+
+   theme_bw()
> top10testsW %>%
+   ggplot(aes(x=Country, y=CumTests, group=Country, color=Country, fill=Country)) +
+   geom_bar(stat='identity')+
+   theme_bw()
>
```