Lab 4: 2019 Novel Coronavirus

Name: Sal Figueroa

2025-04-18

Contents

Github Repository 1
Required data sets 1
Project Objectives 1
Objective 1 1
Objective 2 3
Objective 3 4
Objective 4 4
Objective 5 5

Github Repository

Holds all related files

github: Figgs0bit-Lab4 (https://github.com/Figgs0bit/CSIT165-Lab4.git)

Required data sets

2019 Novel Coronavirus COVID-19 (2019-nCoV) Data Repository by John Hopkins CSSE

The following data set is operated by the John Hopkins University Center for Systems Science and Engineering (JHU CSSE). Data sets include daily time series CSV summary tables for confirmations and deaths associated with COVID-19. Lat and Long refer to coordinate references for the data field. Date fields are stored in MM/DD/YYYY format. For this laboratory, we will be use confirmed and deaths data sets for the US.

Project Objectives

Before beginning your objectives in your final document, please state which day you downloaded the data sets on for analysis. For this laboratory, use the packages described in Module 10 and 11 to accomplish each objective where appropriate. The surgeon general for the United States recently created a new data science initiative, CSIT-165, that uses data science to characterize pandemic diseases. CSIT-165 disseminates data driven analyses to state governors. You are a data scientist for CSIT-165 and it is up to you and you alone to manipulate and visualize COVID-19 data for disease control.

Objective 1

Which state has the most confirmed cases of COVID-19? Which state has the most deaths?

```
#cat("Test VAR - Confirmed(row, col):",rowmax_Conf,",",colmax_Conf,", Deaths(row, col):",rowmax_Deaths,
#[j,6] County column
#[j,7] State column
#[j,1155] last column
#rowmax_Conf <- nrow(Confirmed_US[,]) #limit Qty of Rows 289
```

```
#colmax_Conf <- ncol(Confirmed_US[,]) #limit Qty of Rows 289
#rowmax_Deaths <- nrow(Deaths_US[,]) #limit Qty of Rows 289
#colmax_Deaths <- ncol(Deaths_US[,]) #limit Qty of Rows 289

#Deaths_US[i,j] #[3342, 1154]
#Confirmed_US[i,j] #[3342, 1155]


# removes NA values
Deaths_US[is.na(Deaths_US)] <- 0
Confirmed_US[is.na(Confirmed_US)] <- 0

V_State <- c() #Will Hold all US States and territories.
V_USconfirmed <- c() #Will Hold all entries of confirmed_US
V_USdeaths <- c() #Will Hold all entries of confirmed_US

sumC <- 0 #resets confirmed_global Sum C after each country multiple
sumD <- 0 #resets deaths_global Sum D after each country multiple
CountyCol <- 7


#creates a data frame with country and either deaths_global, confirmed_global values.
df_comfirmed <- c()
df_deaths <- c()
#Data is broken down by county or territory, script must now sort and consolidate common states.
#The following if else ladder sorts and filters the duplicate
#-counties and combines the data for a common region.
for(j in 1:rowmax) #Iterate through rows 3342
{
  #first if conditional is need as the the first row will
  #-not fall through the remaining if statements.


  if(j == 1)
    {
      #Adds the incremented value into each respective vector
      V_State <- c(V_State, Confirmed_US[j,CountyCol])#adds State or territory
      V_USconfirmed <- c(V_USconfirmed, Confirmed_US[j,colmax])#adds US confirmed
      V_USdeaths <- c(V_USdeaths, Deaths_US[j,colmax])#adds US Deaths

    }
  #The following (2) else if statements test and sort through duplicate country
  #-entries and then sums them together, respectively.
  else if(Confirmed_US[j,CountyCol] %in% Confirmed_US[j+1,CountyCol])#conditional for strings
    {
      #When a duplicate country entry is encountered values
      #-are summed up for Confirmed and death cases
      sumC <- sumC + Confirmed_US[j,colmax]
      sumD <- sumD + Deaths_US[j,colmax]

    }
  #conditional statement for strings
  else if((Confirmed_US[j,CountyCol] %in% Confirmed_US[j-1,CountyCol]) && (!(Confirmed_US[j,CountyCol])
```

```
    {
      #When a duplicate country entry is encountered values
      #-are summed up for Confirmed and death cases
      sumC <- sumC + Confirmed_US[j,colmax]
      sumD <- sumD + Deaths_US[j,colmax]

      #loads the shorten results with the common countries combined
      V_State <- c(V_State, Confirmed_US[j,CountyCol])#adds State or territory
      V_USconfirmed <- c(V_USconfirmed, sumC)#adds US confirmed
      V_USdeaths <- c(V_USdeaths, sumD)#adds US Deaths

      #resets the following sum variables to zero for the next duplicate country
      sumC <- 0
      sumD <- 0

    }
  #The last else statement is where countries without duplicates are loaded
  else
    {
      #adds deaths_global, confirmed_global to new vectors
      V_State <- c(V_State, Confirmed_US[j,CountyCol])#adds State or territory
      V_USconfirmed <- c(V_USconfirmed, Confirmed_US[j,colmax])#adds US confirmed
      V_USdeaths <- c(V_USdeaths, Deaths_US[j,colmax])#adds US Deaths

    }

}
#creates and loads data frame with the shorten and consolidated vectors
df_comfirmed <- data.frame(V_State, V_USconfirmed)
df_deaths <- data.frame(V_State, V_USdeaths)

#Reorders the Data-frame in decreasing order
df_C_decreasing <- df_comfirmed[order(df_comfirmed$V_USconfirmed, decreasing = TRUE),]
df_D_decreasing <- df_deaths[order(df_deaths$V_USdeaths, decreasing = TRUE),]

#View(df_C_decreasing)
#View(df_D_decreasing)
```

Objective 2

Create a new table called current_tally by merging confirmed and deaths containing only the columns iso2, Province_State, Admin2, Lat, Long_, and today's count for confirmed and deaths. Rename current_tally's columns as Country, State, City, Lat, Long, Confirmed, and Deaths respectively. Filter current_tally for data points where the country is the US. Please print current_tally as a tibble.

```
#current_tally <- rbind(Confirmed_US, Deaths_US)

current_tally <- Confirmed_US %>%
                 merge(Deaths_US, by = c("iso2", "Province_State", "Admin2", "Lat", "Long_", "X2.29.20")

current_tally <- tibble(current_tally)
#current_tally <- data.table(current_tally)

#print(current_tally)
```

```
#View(current_tally)
```

Objective 3

In current_tally, create a new variable, unaccounted, as the absolute difference between confirmed and deaths for each city. Determine the number of unaccounted for each state using grouped operation.

```
#print(colnames(current_tally))

#current_tally <- current_tally %>%
#  mutate(unaccounted = X3.9.23.x - X3.9.23.y)


#current_tally %>% relocate(unaccounted)

#View(current_tally)

current_tally %>%
        group_by(X3.9.23.x, X3.9.23.y) %>%
        summarize(unaccounted = abs(X3.9.23.x - X3.9.23.y))
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `summarise()` has grouped output by 'X3.9.23.x', 'X3.9.23.y'. You can override
## using the `.groups` argument.
```

```
## # A tibble: 3,325 x 3
## # Groups:   X3.9.23.x, X3.9.23.y [3,232]
##    X3.9.23.x X3.9.23.y unaccounted
##        <int>     <int>       <int>
##  1         0         0           0
##  2         0         0           0
##  3         0         0           0
##  4         0         0           0
##  5         0         0           0
##  6         0         0           0
##  7         0         0           0
##  8         0         0           0
##  9         0         0           0
## 10         0         0           0
## # i 3,315 more rows
```

Objective 4

Using the package kable, create a table showing the top 5 unnacounted states.

```
current_tally %>%
        group_by(X3.9.23.x, X3.9.23.y) %>%
        summarize(unaccounted = abs(X3.9.23.x - X3.9.23.y))
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `summarise()` has grouped output by 'X3.9.23.x', 'X3.9.23.y'. You can override
## using the `.groups` argument.
```

```
## # A tibble: 3,325 x 3
## # Groups:   X3.9.23.x, X3.9.23.y [3,232]
##    X3.9.23.x X3.9.23.y unaccounted
##        <int>     <int>       <int>
##  1         0         0           0
##  2         0         0           0
##  3         0         0           0
##  4         0         0           0
##  5         0         0           0
##  6         0         0           0
##  7         0         0           0
##  8         0         0           0
##  9         0         0           0
## 10         0         0           0
## # i 3,315 more rows
```

```
#knitr::kable(current_tally, "pipe", align=c("l","c","c"))
```

Objective 5

Create any visualization you like using the data sets used in this laboratory. Provide your visualization with a title and custom axis labels. Describe how this visualization conveys information that is not readily apparent from looking at the data alone.