Group Project: Covid Origins

Name: Sal Figueroa
Partner: Kylie Stearns

2025-04-01

Contents

Background

The World Health Organization has recently employed a new data science initiative, *CSIT-165*, that uses data science to characterize pandemic diseases. *CSIT-165* disseminates data driven analyses to global decision makers.

CSIT-165 is a conglomerate comprised of two fabricated entities: *Global Health Union (GHU)* and *Private Diagnostic Laboratories (PDL)*. Your and your partner's role is to play a data scientist from one of these two entities.

Data

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

Data for 2019 Novel Corona virus is operated by the John Hopkins University Center for Systems Science and Engineering (JHU CSSE). Data includes daily time series CSV summary tables, including confirmations, recoveries, and deaths. Country/region are countries/regions hat conform to World Health Organization (WHO). Lat and Long refer to coordinates references for the user. Date fields are stored in MM/DD/YYYY format.

Project Objectives

Objective 1

Predict where the origin started based on the area with the greatest number of confirmations and deaths on the first recorded day in the data set. Show this is the origin using an if statement.

```
#Most deaths and confirmed cases on x1.22.20 (date)
#find max value in column 5 (10.22.20) and check to see how many common max values there is.
max_value <- 0 #holds the updated max_value
RowMax <- 0 #this holds row (i) position in Data frame
```

```r
#For loop starting it column 5 to final column using nrow().
for(i in 1:nrow(deaths_global[,]))
{
num <- deaths_global[i,5] + confirmed_global[i,5]#Holds value to be tested against the current max valu

  if (num > max_value) #confitional if loop that updates the max value as iterates through column 5.
    {
    max_value <- num
    RowMax <- i #Holds the row position where the max value is located
    }
}

#holds info. for ob3
ob1Loc_PS <- deaths_global[RowMax,1]
ob1Loc_CR <- deaths_global[RowMax,2]
ob1Loc_Lat <- deaths_global[RowMax,3]
ob1Loc_Long  <- deaths_global[RowMax,4]

#Holds average
deathNconfimred <- deaths_global[RowMax,5] + confirmed_global[RowMax,5]
cat("Province/State:",deaths_global[RowMax,1],"Country/Region:",deaths_global[RowMax,2],"\n")
```

```
## Province/State: Hubei Country/Region: China
```

```r
cat("Latitude:",deaths_global[RowMax,3],",Longitude:",deaths_global[RowMax,4],"\n")
```

```
## Latitude: 30.9756 ,Longitude: 112.2707
```

```r
cat("Deaths:",deaths_global[RowMax,5],",confirmed cases:",confirmed_global[RowMax,5],"\n")
```

```
## Deaths: 17 ,confirmed cases: 444
```

```r
cat("Predicted area of Origin had the most initial deaths and confirmed cases (01.22.2020) at:",deathNc
```

```
## Predicted area of Origin had the most initial deaths and confirmed cases (01.22.2020) at: 461
```

Objective 2

Where is the most recent area to have a first confirmed case? To do this, you will need to use a for loop, if statement, and subsets.

```r
#find max value in column 5 (10.22.20) and check to see how many max location there is

RowMax <- nrow(confirmed_global[,]) #Qty of Rows 289
ColMax <- ncol(confirmed_global[,]) #QTy of Columns 1147
max_value <- as.numeric(ncol(confirmed_global[,]) * nrow(confirmed_global[,])) #serves as while loop li
num <- 0 #Holds the rotating confirmed_cases value in loop

ColName <- names(confirmed_global) #Loads the the column names into a vector

#initiates the loop counter and used for pulling data from data frame.
i <- 5 #Start the loop at column 5
j <- 1 #Start the loop at column 1 instead of zero, adjust later

#The while loop stops running when the num variable loads a value > 0
while(j < max_value && num == 0)
 {
```

```
  num <- confirmed_global[j,i]
     if( j %% RowMax == 0) #This conditional statement allows the while loop to irritate-
      {                    #through the each column before looping to next col.
      i <- i + 1 #Iterate through columns 1-1147
      }
   j <- j + 1  #Iterate through rows 1-289
 }

j <- j - 1 #This accounts for the final counter increment when exiting the while loop.

#holds info. for ob3
ob2Loc_PS <- confirmed_global[j,1]
ob2Loc_CR <- confirmed_global[j,2]
ob2Loc_Lat <- confirmed_global[j,3]
ob2Loc_Long  <- confirmed_global[j,4]



cat("Province/State: ",confirmed_global[j,1],"Province/State: ",confirmed_global[j,2],"\n")
```

```
## Province/State:  Anhui Province/State:  China
```

```
cat("Latitude: ",confirmed_global[j,3]," ,Longitude:",confirmed_global[j,4],"\n")
```

```
## Latitude:  31.8257  ,Longitude: 117.2264
```

```
cat("Predicted area of Origin had the most initial deaths on", ColName[i] ,"at:",confirmed_global[j,i],
```

```
## Predicted area of Origin had the most initial deaths on X1.22.20 at: 1
```

Objective 3

How far away are the areas from objective 2 from where the first confirmed case(s) occurred? Please provide answer(s) in terms of miles. Use the function distm from the R package geosphere to calculate the distance between two coordinates in meters (geosphere::distm). You will need to convert the value returned by distm from meters to miles (this conversion is simple and can be found online). Please use a table or printed statement to describe what Province/State and Country/Region first confirmed cases occurred as well as the distance (in miles) away from the origin. Please print the following: {recent region} is {distance in miles} away from {origin city, origin country}.

```
#Calculate the distance using distm()
distance_meters <- distm(c(ob1Loc_Long, ob1Loc_Lat), c(ob2Loc_Long, ob2Loc_Lat), fun = distHaversine)
cat("From (ob1):\n")
```

```
## From (ob1):
```

```
cat("Province/State:",ob1Loc_PS,"Country/Region:",ob1Loc_CR,"\n")
```

```
## Province/State: Hubei Country/Region: China
```

```
cat("Latitude:",ob1Loc_Lat,",Longitude:",ob1Loc_Long,"\n\n")
```

```
## Latitude: 30.9756 ,Longitude: 112.2707
```

```
cat("To (ob2):\n")
```

```
## To (ob2):
```

```r
cat("Province/State:",ob2Loc_PS,"Country/Region:",ob2Loc_CR,"\n")
```

```
## Province/State: Anhui Country/Region: China
```

```r
cat("Latitude:",ob2Loc_Lat,",Longitude:",ob2Loc_Long,"\n\n")
```

```
## Latitude: 31.8257 ,Longitude: 117.2264
```

```r
cat("Distance:",round(distance_meters), "- meters\n") #Print the distance in meters
```

```
## Distance: 480239 - meters
```

```r
cat("Distance:",round(distance_meters/1000),"- Kilometers\n" ) #Print the distance in Kilometers
```

```
## Distance: 480 - Kilometers
```

```r
cat("Distance:",round(distance_meters/1609.34),"- miles\n" ) #1 mile = 1609.34 meters
```

```
## Distance: 298 - miles
```

Objective 4

CSIT-165 characterizes diseases using risk scores. Risk scores are calculated as the ratio of deaths to confirmations, that is Riskscore=100×deathsconfirmations. Risk scores equal to 100 indicate the highest risk while risk scores equal to 0 indicate the lowest risk. Areas are characterized as being especially vulnerable to loss if they have higher risk scores. For this assignment, exclude cruise ships (hint: they have lat and long coordinates of 0 or NA in this data set, filter this out before calculating risk scores).

Which area of the world currently has the lowest risk score (if more than one, display the one with the most confirmations)? Which area of the world currently has the highest risk score (if more than one, display the one with the most confirmations)? How do risk scores in these areas compare to global risk score? Why might it be helpful to calculate metrics like risk scores for different areas of the world and what would their limitations be (what assumptions does risk score make and what important variables might be left out)?

```r
#Row and Column variables
iCntLength <- ncol(deaths_global)#1147 number of columns
jCntLength <- nrow(deaths_global)#289 number of rows

#Variables for holding highest and lowest Risk score
low_value <- 10
high_value <- 0

#creates vectors to his and lows
VSUMdeaths_global <- 1:289
VSUMconfirmed_global <- 1:289
VSUMRiskScore <- 1:289

# removes NA values
deaths_global[is.na(deaths_global)] <- 0
confirmed_global[is.na(confirmed_global)] <- 0

#Running sum defaults
SUMdeaths_global <- 0
SUMconfirmed_global <- 0
SUMRiskScore  <- 0
emptyClr <- 0
```

```
#For loop iterates through the entire data frame
for(j in 1:jCntLength) #Iterate through rows 1-289
  {
      #Running sum defaults are reset every 1142 loops
      SUMdeaths_global <- 0
      SUMconfirmed_global <- 0
      SUMRiskScore  <- 0

  for(i in 5:iCntLength) #Iterate through columns 1-1147
    {
      #This variable filters out locations with zero coordinates.
      emptyClr <- (deaths_global[j,3] + deaths_global[j,4])

      if(emptyClr > 0 && confirmed_global[j,i] > 0)
        {
          SUMdeaths_global <- SUMdeaths_global + deaths_global[j,i] #holds the sum of every death per r
          SUMconfirmed_global <- SUMconfirmed_global + confirmed_global[j,i] #holds the sum of every co
          SUMRiskScore <- (SUMdeaths_global/SUMconfirmed_global)*100 #Riskscore eqn.=(deaths/confirmati
        }
      else if(confirmed_global[j,i] == 0)
        {SUMRiskScore <-0}
    }

     VSUMdeaths_global[j] <- SUMdeaths_global
     VSUMconfirmed_global[j] <- SUMconfirmed_global
     VSUMRiskScore[j] <- SUMRiskScore  #Vector length 1:289 Holds the risk score (all time) for each lo
}
```

Objective 4.1

```
low_value <- min(VSUMRiskScore)
high_value <- max(VSUMRiskScore)

HiMaxConfirmed <- 0
LoMaxConfirmed <- 1

for(j in 1:jCntLength) #Iterate through rows 1-289
{
      chkRiskScore <- VSUMRiskScore[j]

      if(chkRiskScore == high_value)
      {
        if(VSUMconfirmed_global[j] > HiMaxConfirmed)
        {
        HiMaxConfirmed <- VSUMconfirmed_global[j]
        HiColNum <- j
        }
      }

    if(chkRiskScore == low_value)
    {
    if(VSUMconfirmed_global[j] < LoMaxConfirmed)
```

```
        {
        LoMaxConfirmed <- VSUMconfirmed_global[j]
        LoColNum <- j
        }
    }
}
cat("Global Risk Score:",mean(VSUMRiskScore),",Max Risk Score:",max(VSUMRiskScore),",Min Risk Score:",m
```

Objective 4.2

```
## Global Risk Score: 2.895648 ,Max Risk Score: 600 ,Min Risk Score: 0
```

```
cat("Highest Risk Score:",VSUMRiskScore[HiColNum]," - Region(Row) w/ most confrimations:",HiColNum,"\n"
```

```
## Highest Risk Score: 600  - Region(Row) w/ most confrimations: 162
```

```
cat("Province/State:", deaths_global[HiColNum, 1],",Country/Region:",deaths_global[HiColNum, 2],"\n")
```

```
## Province/State:  ,Country/Region: Korea, North
```

```
cat("Lat:", deaths_global[HiColNum, 3],",Long:",deaths_global[HiColNum, 4],"\n\n")
```

```
## Lat: 40.3399 ,Long: 127.5101
```

```
cat("Lowest Risk Score:",VSUMRiskScore[LoColNum]," - Region(Row) w/ most confrimations:", LoColNum,"\n"
```

```
## Lowest Risk Score: 0  - Region(Row) w/ most confrimations: 6
```

```
cat("Province/State:", deaths_global[LoColNum, 1],",Country/Region:",deaths_global[LoColNum, 2],"\n")
```

```
## Province/State:  ,Country/Region: Antarctica
```

```
cat("Lat:", deaths_global[LoColNum, 3],",Long:",deaths_global[LoColNum, 4],"\n\n")
```

```
## Lat: -71.9499 ,Long: 23.347
```

Objective 5 *You are asked to make two tables with the top 5 countries that have the most COVID-19 related confirmations and and deaths. Make sure to include all of the counts for the country, not just the counts for one area in the country. To do this we will need to sum all of the values for each country, create new data frames from these values, and use the package kable to convert those data frames into tables.*

Hint: Sum each country's counts by subsetting the data frame using a list of countries available in the data set. Use a for loop to iterate through the data frame using the list of countries. For each country, calculate the count sum and assign this value to a list.

```
SumCountry <- c() #Empty 1D vector to append to risk score of each country.
SumCountryPos <- c() #Empty 1D vector to append each country in order
tmpsum <- 0 #Holds the summation of each country that has multiple cities
j <- 1


while(j <= jCntLength) #289 rows
{
chkVal <- VSUMRiskScore[j] # Loads the Risk Score Sum position
chkcity <- confirmed_global[j,2] #Data frame holds location, [row, Col]-confirmed_global[1,2]  == Afgha


if(j == jCntLength) #Accounts for the [j+1,2] loop feature when approaching the end of the loop.
        {
```

```
        SumCountry <- c(SumCountry, chkVal)
        SumCountryPos <- c(SumCountryPos,chkcity)
        }
else if(chkcity != confirmed_global[j+1,2])
        {
        SumCountry <- c(SumCountry, chkVal)
        SumCountryPos <- c(SumCountryPos,chkcity)
        }
    else
        {
          chkVal <- 0
          k <- j
          while(chkcity == confirmed_global[k+1,2])
            {
            #cat("Count:",k,",tmpsum:",tmpsum,"Country:",chkcity,"\n")
            tmpsum <- tmpsum + VSUMRiskScore[k]
            k <- k + 1
            }
              chkVal <- chkVal + tmpsum
              #cat("Count:",j,",",chkVal,"Country:",chkcity,"\n")
              SumCountry <- c(SumCountry, chkVal)
              SumCountryPos <- c(SumCountryPos,chkcity)
          j <- j + k

        }
j <- j + 1
#cat(j,":Leaving Loop\n")
}

#for(u in 1:length(SumCountry)) #289 rows)
#{
#cat("[j]:",u,"Country RiskScore:",SumCountry[u],"Country:",SumCountryPos[u],"\n")
#}

for(u in 1:jCntLength)
{
cat(u,":",VSUMRiskScore[u],":",deaths_global[u,2],"\n")

}
```

```
## 1 : 4.170705 : Afghanistan
## 2 : 1.339375 : Albania
## 3 : 2.682079 : Algeria
## 4 : 0.5181378 : Andorra
## 5 : 2.052195 : Angola
## 6 : 0 : Antarctica
## 7 : 0 : Antigua and Barbuda
## 8 : 0 : Argentina
## 9 : 1.998447 : Armenia
## 10 : 0.06533287 : Australia
## 11 : 0.14968 : Australia
## 12 : 0.07281039 : Australia
## 13 : 0.1218692 : Australia
## 14 : 0.1195694 : Australia
```

```
## 15 : 0.07925181 : Australia
## 16 : 0.2608789 : Australia
## 17 : 0.05848284 : Australia
## 18 : 0.62125 : Austria
## 19 : 1.267492 : Azerbaijan
## 20 : 0 : Bahamas
## 21 : 0.298588 : Bahrain
## 22 : 1.534912 : Bangladesh
## 23 : 0 : Barbados
## 24 : 0.7334421 : Belarus
## 25 : 1.12725 : Belgium
## 26 : 0 : Belize
## 27 : 0.703484 : Benin
## 28 : 0.03678493 : Bhutan
## 29 : 0 : Bolivia
## 30 : 4.179199 : Bosnia and Herzegovina
## 31 : 0.976428 : Botswana
## 32 : 0 : Brazil
## 33 : 0.1224292 : Brunei
## 34 : 3.348702 : Bulgaria
## 35 : 1.692555 : Burkina Faso
## 36 : 3.186785 : Burma
## 37 : 0.1020817 : Burundi
## 38 : 0 : Cabo Verde
## 39 : 2.19688 : Cambodia
## 40 : 1.630207 : Cameroon
## 41 : 0 : Canada
## 42 : 0 : Canada
## 43 : 0 : Canada
## 44 : 0 : Canada
## 45 : 0 : Canada
## 46 : 0 : Canada
## 47 : 0 : Canada
## 48 : 0 : Canada
## 49 : 0 : Canada
## 50 : 0 : Canada
## 51 : 0 : Canada
## 52 : 0 : Canada
## 53 : 0 : Canada
## 54 : 0 : Canada
## 55 : 0 : Canada
## 56 : 0 : Canada
## 57 : 0.8958715 : Central African Republic
## 58 : 3.12301 : Chad
## 59 : 0 : Chile
## 60 : 0.5176353 : China
## 61 : 0.2073052 : China
## 62 : 0.3507307 : China
## 63 : 0.04064397 : China
## 64 : 0.3791867 : China
## 65 : 0.0685692 : China
## 66 : 0.1314553 : China
## 67 : 0.510881 : China
## 68 : 0.3099867 : China
```

```
## 69 : 0.4888163 : China
## 70 : 0.6094539 : China
## 71 : 0.8379553 : China
## 72 : 0.5804656 : China
## 73 : 6.44638 : China
## 74 : 0.2493576 : China
## 75 : 0.05736921 : China
## 76 : 0 : China
## 77 : 0.08686147 : China
## 78 : 0.02524587 : China
## 79 : 0.1996877 : China
## 80 : 1.988029 : China
## 81 : 0 : China
## 82 : 0 : China
## 83 : 0.1595991 : China
## 84 : 0.4053658 : China
## 85 : 0.8580658 : China
## 86 : 0.006555796 : China
## 87 : 0.1351026 : China
## 88 : 0.2736905 : China
## 89 : 0 : China
## 90 : 0 : China
## 91 : 0.2787427 : China
## 92 : 0.1218325 : China
## 93 : 0.0384527 : China
## 94 : 0 : Colombia
## 95 : 2.356721 : Comoros
## 96 : 1.557864 : Congo (Brazzaville)
## 97 : 1.74318 : Congo (Kinshasa)
## 98 : 0 : Costa Rica
## 99 : 0.8931843 : Cote d'Ivoire
## 100 : 1.553784 : Croatia
## 101 : 0 : Cuba
## 102 : 0.2453864 : Cyprus
## 103 : 1.158523 : Czechia
## 104 : 0.08618094 : Denmark
## 105 : 0.1587067 : Denmark
## 106 : 0.2709848 : Denmark
## 107 : 0 : Diamond Princess
## 108 : 1.216625 : Djibouti
## 109 : 0 : Dominica
## 110 : 0 : Dominican Republic
## 111 : 0 : Ecuador
## 112 : 5.155079 : Egypt
## 113 : 0 : El Salvador
## 114 : 1.221697 : Equatorial Guinea
## 115 : 0.8685246 : Eritrea
## 116 : 0.5386537 : Estonia
## 117 : 2.186777 : Eswatini
## 118 : 1.56746 : Ethiopia
## 119 : 1.267893 : Fiji
## 120 : 0.5246554 : Finland
## 121 : 0 : France
## 122 : 0.9348516 : France
```

```
## 123 : 0 : France
## 124 : 0 : France
## 125 : 0.6047549 : France
## 126 : 0.5444128 : France
## 127 : 0.2282857 : France
## 128 : 0 : France
## 129 : 0 : France
## 130 : 0 : France
## 131 : 0 : France
## 132 : 0.7111761 : France
## 133 : 0.6402048 : Gabon
## 134 : 0 : Gambia
## 135 : 1.069879 : Georgia
## 136 : 0.7018741 : Germany
## 137 : 0.8442973 : Ghana
## 138 : 0.9110151 : Greece
## 139 : 0 : Grenada
## 140 : 0 : Guatemala
## 141 : 1.070151 : Guinea
## 142 : 0 : Guinea-Bissau
## 143 : 0 : Guyana
## 144 : 0 : Haiti
## 145 : 0 : Holy See
## 146 : 0 : Honduras
## 147 : 2.641821 : Hungary
## 148 : 0.1036653 : Iceland
## 149 : 1.252685 : India
## 150 : 2.717352 : Indonesia
## 151 : 2.181827 : Iran
## 152 : 1.189683 : Iraq
## 153 : 0.6844148 : Ireland
## 154 : 0.3401495 : Israel
## 155 : 1.268816 : Italy
## 156 : 0 : Jamaica
## 157 : 0.3048672 : Japan
## 158 : 0.9538799 : Jordan
## 159 : 1.405154 : Kazakhstan
## 160 : 1.77439 : Kenya
## 161 : 0 : Kiribati
## 162 : 600 : Korea, North
## 163 : 0.1325111 : Korea, South
## 164 : 1.521447 : Kosovo
## 165 : 0.4656868 : Kuwait
## 166 : 1.538654 : Kyrgyzstan
## 167 : 0.3506685 : Laos
## 168 : 0.834904 : Latvia
## 169 : 1.030158 : Lebanon
## 170 : 0 : Lesotho
## 171 : 0 : Liberia
## 172 : 1.350405 : Libya
## 173 : 0.6549655 : Liechtenstein
## 174 : 0.9239703 : Lithuania
## 175 : 0.5669939 : Luxembourg
## 176 : 0 : MS Zaandam
```

```
## 177 : 2.04806 : Madagascar
## 178 : 3.170555 : Malawi
## 179 : 0.8304661 : Malaysia
## 180 : 0.1975122 : Maldives
## 181 : 2.736946 : Mali
## 182 : 0.8526933 : Malta
## 183 : 0.1104576 : Marshall Islands
## 184 : 1.807855 : Mauritania
## 185 : 0.448603 : Mauritius
## 186 : 0 : Mexico
## 187 : 0.2583486 : Micronesia
## 188 : 2.201371 : Moldova
## 189 : 0.5543743 : Monaco
## 190 : 0.2392813 : Mongolia
## 191 : 1.178988 : Montenegro
## 192 : 1.430659 : Morocco
## 193 : 1.030714 : Mozambique
## 194 : 0 : Namibia
## 195 : 0.02118301 : Nauru
## 196 : 1.22742 : Nepal
## 197 : 0 : Netherlands
## 198 : 0 : Netherlands
## 199 : 0 : Netherlands
## 200 : 0 : Netherlands
## 201 : 0.4391417 : Netherlands
## 202 : 0 : New Zealand
## 203 : 0 : New Zealand
## 204 : 0.1053048 : New Zealand
## 205 : 0 : Nicaragua
## 206 : 3.535608 : Niger
## 207 : 1.275687 : Nigeria
## 208 : 3.04525 : North Macedonia
## 209 : 0.2899227 : Norway
## 210 : 1.164864 : Oman
## 211 : 2.064219 : Pakistan
## 212 : 0.1276598 : Palau
## 213 : 0 : Panama
## 214 : 1.422087 : Papua New Guinea
## 215 : 0 : Paraguay
## 216 : 0 : Peru
## 217 : 1.637521 : Philippines
## 218 : 2.080355 : Poland
## 219 : 0.7053922 : Portugal
## 220 : 0.1817156 : Qatar
## 221 : 2.381577 : Romania
## 222 : 2.088974 : Russia
## 223 : 1.153073 : Rwanda
## 224 : 0 : Saint Kitts and Nevis
## 225 : 0 : Saint Lucia
## 226 : 0 : Saint Vincent and the Grenadines
## 227 : 0 : Samoa
## 228 : 0.9193081 : San Marino
## 229 : 1.32343 : Sao Tome and Principe
## 230 : 1.334926 : Saudi Arabia
```

```
## 231 : 2.329556 : Senegal
## 232 : 0.8060222 : Serbia
## 233 : 0.3929509 : Seychelles
## 234 : 0 : Sierra Leone
## 235 : 0.09553223 : Singapore
## 236 : 0.9422909 : Slovakia
## 237 : 0.802928 : Slovenia
## 238 : 0.7366341 : Solomon Islands
## 239 : 5.025287 : Somalia
## 240 : 0 : South Africa
## 241 : 0.9287509 : South Sudan
## 242 : 1.196025 : Spain
## 243 : 2.348825 : Sri Lanka
## 244 : 7.408334 : Sudan
## 245 : 0 : Summer Olympics 2020
## 246 : 0 : Suriname
## 247 : 1.023725 : Sweden
## 248 : 0.5507781 : Switzerland
## 249 : 5.858412 : Syria
## 250 : 0.1868293 : Taiwan*
## 251 : 0.7144485 : Tajikistan
## 252 : 2.281723 : Tanzania
## 253 : 0.7471902 : Thailand
## 254 : 0.5566195 : Timor-Leste
## 255 : 0.8204214 : Togo
## 256 : 0 : Tonga
## 257 : 0 : Trinidad and Tobago
## 258 : 2.886126 : Tunisia
## 259 : 0.7104461 : Turkey
## 260 : 0 : Tuvalu
## 261 : 0 : US
## 262 : 2.114614 : Uganda
## 263 : 2.21652 : Ukraine
## 264 : 0.2642034 : United Arab Emirates
## 265 : 0 : United Kingdom
## 266 : 0 : United Kingdom
## 267 : 0 : United Kingdom
## 268 : 0 : United Kingdom
## 269 : 0 : United Kingdom
## 270 : 0 : United Kingdom
## 271 : 0.793977 : United Kingdom
## 272 : 0.2743974 : United Kingdom
## 273 : 0.4000545 : United Kingdom
## 274 : 0.3533462 : United Kingdom
## 275 : 0 : United Kingdom
## 276 : 0 : United Kingdom
## 277 : 0 : United Kingdom
## 278 : 0 : United Kingdom
## 279 : 1.334083 : United Kingdom
## 280 : 0 : Uruguay
## 281 : 0.6947751 : Uzbekistan
## 282 : 0.1305705 : Vanuatu
## 283 : 0 : Venezuela
## 284 : 0.4915959 : Vietnam
```

```
## 285 : 0.9101361 : West Bank and Gaza
## 286 : 0 : Winter Olympics 2022
## 287 : 19.23293 : Yemen
## 288 : 1.343033 : Zambia
## 289 : 2.465156 : Zimbabwe
```

GitHub Log

```
git log --pretty=format:"%nSubject: %s%nAuthor: %aN%nDate: %aD%nBody: %b"
```

```
##
## Subject: Finished code, now adding comments and double checking
## Author: Sal - Figgs0bit
## Date: Tue, 1 Apr 2025 11:34:12 -0700
## Body:
##
## Subject: Working on ob5
## Author: Sal - Figgs0bit
## Date: Mon, 31 Mar 2025 21:23:53 -0700
## Body:
##
## Subject: Finished ob4.1, ob4.2
## Author: Sal - Figgs0bit
## Date: Mon, 31 Mar 2025 13:56:47 -0700
## Body:
##
## Subject: Finished ob3 coordinates, all comments up to date. Ob4.1 & ob4.2 need corrections.
## Author: Sal - Figgs0bit
## Date: Sun, 30 Mar 2025 18:06:18 -0700
## Body:
##
## Subject: code and Comments for ob2 updated
## Author: Sal - Figgs0bit
## Date: Sun, 30 Mar 2025 08:09:05 -0700
## Body:
##
## Subject: Updateding notes, ob3 and ob 5 needed.
## Author: Sal - Figgs0bit
## Date: Sun, 30 Mar 2025 05:49:15 -0700
## Body:
##
## Subject: Finished Code for Ob2, need code comments
## Author: Sal - Figgs0bit
## Date: Fri, 28 Mar 2025 12:23:15 -0700
## Body:
##
## Subject: Ob1 code done, need to update code comments
## Author: Sal - Figgs0bit
## Date: Fri, 28 Mar 2025 03:30:43 -0700
## Body:
##
## Subject: R ob1 chunk
## Author: Sal - Figgs0bit
## Date: Fri, 28 Mar 2025 02:12:28 -0700
```

```
## Body:
##
## Subject: Objective 1, Deaths finished. Need Confirmation for loops.
## Author: Sal - Figgs0bit
## Date: Mon, 24 Mar 2025 20:39:21 -0700
## Body:
##
## Subject: Merge branch 'main' of github.com:Figgs0bit/CSIT165-CovidGroupProj
## Author: Sal - Figgs0bit
## Date: Mon, 24 Mar 2025 17:34:18 -0700
## Body:
##
## Subject: added {r Data setup} chunk. Downloads csv from repository to dataframe
## Author: Sal - Figgs0bit
## Date: Mon, 24 Mar 2025 17:34:05 -0700
## Body:
##
## Subject: Uploaded data
## Author: Figgs0bit
## Date: Mon, 24 Mar 2025 17:14:29 -0700
## Body: time_series_covid19 data (global deaths, global recovered)
##
## Subject: Merge branch 'main' of github.com:Figgs0bit/CSIT165-CovidGroupProj
## Author: Sal - Figgs0bit
## Date: Mon, 24 Mar 2025 17:12:01 -0700
## Body:
##
## Subject: Template knited to PDF w/o (\usepackage{tabu}, library(kableExtra))
## Author: Sal - Figgs0bit
## Date: Mon, 24 Mar 2025 17:11:54 -0700
## Body:
##
## Subject: Update README.md
## Author: Figgs0bit
## Date: Mon, 24 Mar 2025 17:10:19 -0700
## Body: Contains copy of Group Contract
##
## Subject: First Commit: setting up Github repository need to add template
## Author: Sal - Figgs0bit
## Date: Mon, 24 Mar 2025 17:05:25 -0700
## Body:
##
## Subject: Initial commit
## Author: Figgs0bit
## Date: Mon, 24 Mar 2025 17:00:21 -0700
## Body:
```