Maxwell Wang Individual Project

In this project, we will attempt to reproduce the figures in the New York Times coronavirus map, for January 17, 2021. We will also comment briefly on the reproducibility of these figures. First, we will load in the required libraries.

```
# Load libraries
library(ggplot2)
library(tidyverse)
library(stringr)
library(zoo)
library(lubridate)
library(kableExtra)
```

Next, we will read in the CSVs required. c_data is a variable that stores county-by-county cases. For each county, it stores the fips (a county identifying code), as well as the TOTAL cases and deaths occuring by each date. s_data is a variable that stores the number of deaths, hospitalized, tests, etc., for each state.

```
c_data <- read.csv(file = ’us-counties.csv’)
s_data <- read.csv(file = ’all-states-history.csv’)
```

Figure 1)

We will now produce the rolling-average plot in Figure 1. Let’s use the c_data (county-by-county data) csv. One thing to note is that c_data’s ‘case’ field tracks the total cases, so the new cases for each day is the difference between the current day and the past day.

```
# For each date, sum up the total cases from all counties.
total_cases = c()
all_dates = unique(c_data$date)
for (date in all_dates){
  c = sum(na.omit(c_data[c_data$date == date,]$cases))
  total_cases = rbind(total_cases,c(date,as.numeric(c)))
}
```

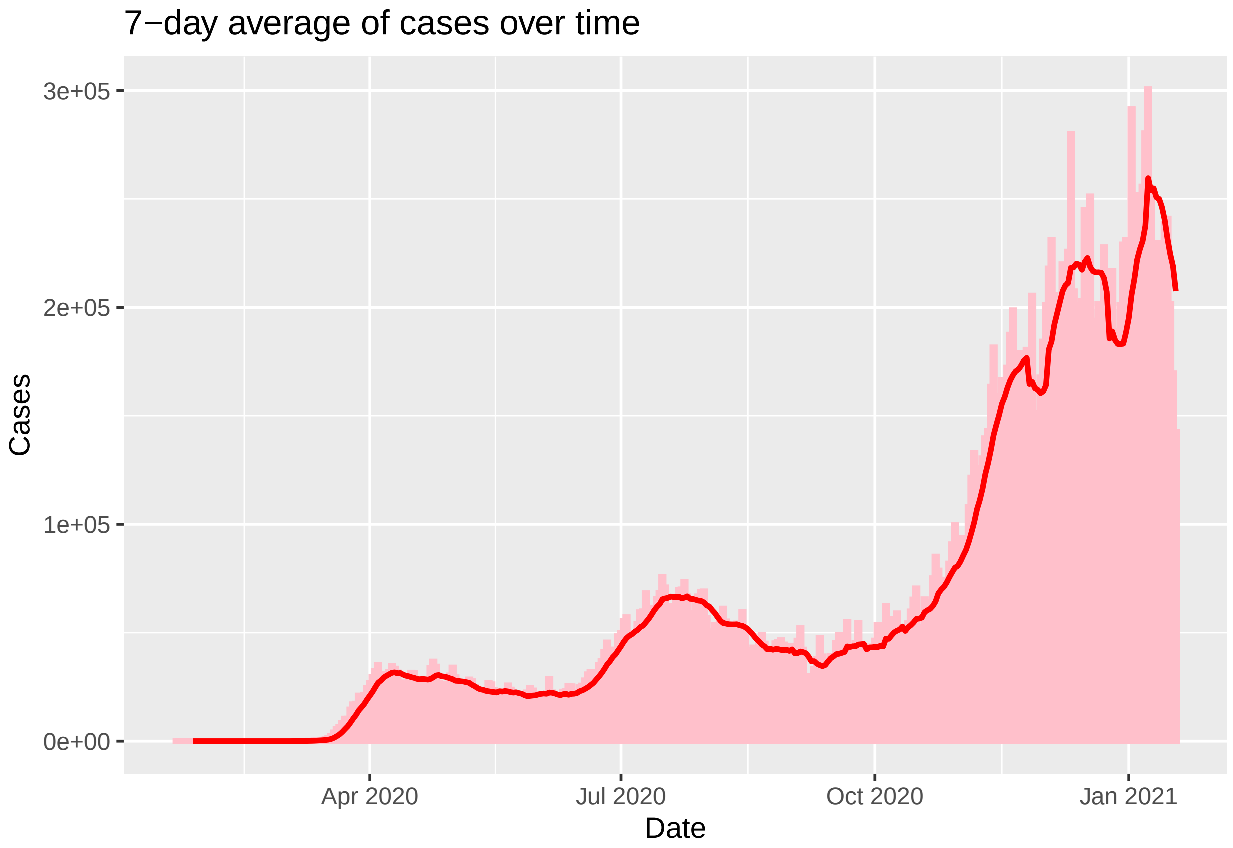
```
# Find the 7-day average of total cases for each date.
roll_avg_c = cbind(all_dates,rollmeanr(as.numeric(total_cases[,2]),k=7,fill=NA))

# Find the 7-day rolling average for new cases
new_cases = cbind(all_dates,as.numeric(total_cases[,2])-lag(as.numeric(total_cases[,2])))
# Holds the new national cases per day.
roll_avg_n = cbind(all_dates,rollmean(as.numeric(new_cases[,2]),k=7,fill=NA,align="right"))
# Holds the new rolling average (7-day) for each day.

new_cases = as.data.frame(cbind(new_cases,roll_avg_n[,2]))
#Create a dataframe that holds dates, total cases, and the rolling average.
```

```r
colnames(new_cases) = c("date","total_cases","roll_avg")
new_cases$date = as.Date(new_cases$date)

ggplot(data = new_cases) +
  geom_col(aes(x=as.Date(date),y=as.numeric(total_cases)),size = 1, color = "pink", fill = "white") +
  geom_line(aes(x = as.Date(date), y = as.numeric(roll_avg)), size = 1, color="red", group = 1) +
  ggtitle("7-day average of cases over time") + xlab("Date") + ylab("Cases")# Create our plot.
```



This plot we have reproduced looks pretty close to the plot in the NYT. We see that the pandemic starts a little before April, and that there are large bumps in late July, and a major ramp-up beginning around October. There are then a series of three sub-peaks between November and the present. The current trend is downwards, coming down from the highest point in early January. In addition, the "Total New Cases per Day" (the bar graph behind the red line graph) seems to follow the 7-day moving average quite closely, with some overshooting in the winter of 2020.

In terms of reproducibility, this figure from the NYT seems readily reproducible, at least in terms of the general trends. The data was easy to access, with pretty intuitive variable names and organization structure in us-counties.csv. There is a bit of lack of clarity about how exactly 7-day running averages are calculated (right aligned or centered), but I don't think this makes a large difference, since the running average usually doesn't change dramatically very quickly, so the trends will look the same regardless of which formula we use.

Table 2)

For the graph in Part 2), we need to extract 8 values. In terms of cases, we want Total Cases, New Cases on January 17, and the 14 day change in 7-day average in Cases. In terms of deaths, we want Total

Deaths, New Deaths on January 17, and the 14-day change in 7-day average in Deaths. Lastly, we want new hospitalizations on January 17, and the 14-day change in 7-day average in Hospitalizations.

```
total_h = c()
total_c = c()
total_d = c()

all_dates = unique(c_data$date)
for (date in all_dates){
  # Find total hospitalizations for each date
  h = sum(na.omit(s_data[s_data$date == date,]$hospitalizedCurrently))
  total_h = rbind(total_h,c(date,h))

  # Find total cases for each date
  c = sum(na.omit(c_data[c_data$date==date,]$cases))
  total_c = rbind(total_c,c(date,c))

  d = sum(na.omit(c_data[c_data$date==date,]$deaths))
  total_d = rbind(total_d,c(date,d))
}

t_d_avg = rollmean(as.numeric(total_d[,2]),k=7,fill=NA,align="right")

# Vectors that new cases/deaths
n_c = cbind(all_dates,as.numeric(total_c[,2])-lag(as.numeric(total_c[,2])))
n_d = cbind(all_dates,as.numeric(total_d[,2])-lag(as.numeric(total_d[,2])))


#Now find the difference in cases from two weeks ago.
roll_avg_c = cbind(all_dates,rollmean(as.numeric(n_c[,2]),k=7,fill=NA,align="right"))
diff_avg_c = as.numeric(roll_avg_c[363,2]) - as.numeric(roll_avg_c[349,2])
diff_in_case = diff_avg_c/as.numeric(roll_avg_c[349,2]) # This gives us percentage change.

#Now find the difference in deaths from two weeks ago.
roll_avg_d = cbind(all_dates,rollmean(as.numeric(n_d[,2]),k=7,fill=NA,align="right"))
diff_avg_d = as.numeric(roll_avg_d[363,2]) - as.numeric(roll_avg_d[349,2])
diff_in_deaths = diff_avg_d/as.numeric(roll_avg_d[349,2]) # This gives us percentage change.

#Now find the difference in hospitalizations from two weeks ago.
# One thing to note is that we use total_h here because of a difference in definition.
roll_avg_h = cbind(all_dates,rollmean(as.numeric(total_h[,2]),k=7,fill=NA,align="right"))
diff_avg_h = as.numeric(roll_avg_h[363,2]) - as.numeric(roll_avg_h[349,2])
diff_in_hos = diff_avg_h/as.numeric(roll_avg_h[349,2]) # This gives us percentage change.

# And print everything
today_date = ’2021-01-17’
sprintf("Total Cases: %i", as.numeric(total_c[363,2]))
```

```
## [1] "Total Cases: 23983607"
```

```
sprintf("Total New Cases on January 17: %i", as.numeric(n_c[363,2]))
```

```
## [1] "Total New Cases on January 17: 169641"
```

```
sprintf("14-day percent change in New Cases: %f", diff_in_case*100)
```

```
## [1] "14-day percent change in New Cases: 2.820180"
```

```
sprintf("Total Deaths: %i", as.numeric(total_d[363,2]))
```

```
## [1] "Total Deaths: 397612"
```

```
sprintf("Total New Deaths on January 17: %i", as.numeric(n_d[363,2]))
```

```
## [1] "Total New Deaths on January 17: 1730"
```

```
sprintf("14-day percent change in New Deaths: %f", diff_in_deaths*100)
```

```
## [1] "14-day percent change in New Deaths: 25.726681"
```

```
sprintf("Current Hospitalizations on January 17: %i", as.numeric(total_h[,2][363]))
```

```
## [1] "Current Hospitalizations on January 17: 124387"
```

```
sprintf("14-day percent change in current hospitalizations: %f", diff_in_hos*100)
```

```
## [1] "14-day percent change in current hospitalizations: 3.154048"
```

Overall, this looks pretty reproducible, and we successfully were able to reproduce all the values in the figure. For Cases, there are about 24 million total, 169641 on January 17, and a 2.8% 14-day change. For Deaths, there are 397612 total, 1730 on January 17, and a 25.72% 14-day change. And for Current Hospitalizations, there are 124387 on January 17, and a 3.15% 14-day change. These correspond to pretty much the exact same values in Table 2. The data was easy to access, though it may have been useful for the NYT to include the additonal hospitalization data in their github repository, so that we don't have to download it from another, separate website (of course, since that hospitalization data was not produced by the NYT, I'm not sure if there would have been ownership issues).

There is one problem that can lead to confusion. In the chart, the 'On Jan 17' field for cases and deaths are the NEW cases and deaths occurring on January 17. However, for Hospitalizations, the 'On Jan 17' field is actually for all current hospitalizations on January 17, not just the new hospitalizations occurring on January 17. This also means that the 14-day change field has a different meaning depending on the row. For Cases and Deaths, the 14-day change is the change in new cases and deaths per day, over 14 days, but for Hospitalizations, the 14-day change is the change in total (not new) hospitalizations over 14 days. This can be pretty misleading, and is not actually stated in the figure. I would say this is a hindrance to reproducibility, and that the published figure should have some additional information to clarify this difference.

Figure 3)

Here, we will try to reproduce the heat-map for new cases. We can do some data-wrangling, since for the heatmap (and the state-by-state). We will also make use of the choroplethr package to get census data and also to make our heatmap.

```
library(choroplethr) # Need this to get the total pop per county based on fips
library(choroplethrMaps)
map_data = c()

# Make a vector of dates that we want to address.
#Note that we actually need 8 (not just 7) days,
#since we need to calculate the new cases on January 11 too.
week_dates = c("2021-01-10","2021-01-11","2021-01-12",
               "2021-01-13","2021-01-14","2021-01-15",
               "2021-01-16","2021-01-17")
all_fips = unique(c_data$fips)

# Add up the 7-day avg of new cases for each county.
for (f in all_fips){
  this_county_data = na.omit(c_data[c_data$fips == f,])
  this_county_week_data = this_county_data[this_county_data$date %in% week_dates,]
  # Find the new cases for each day from January 10 to January 17
  new_cases = as.numeric(this_county_week_data$cases)-lag(as.numeric(this_county_week_data$cases))
  new_cases[is.na(new_cases)] = 0
  # Find the average number of new cases over the last 7 days, omitting the first NA
  avg_new_cases = mean(new_cases[-1])
  map_data = rbind(map_data,c(f,this_county_data$state[1],avg_new_cases))
}
```

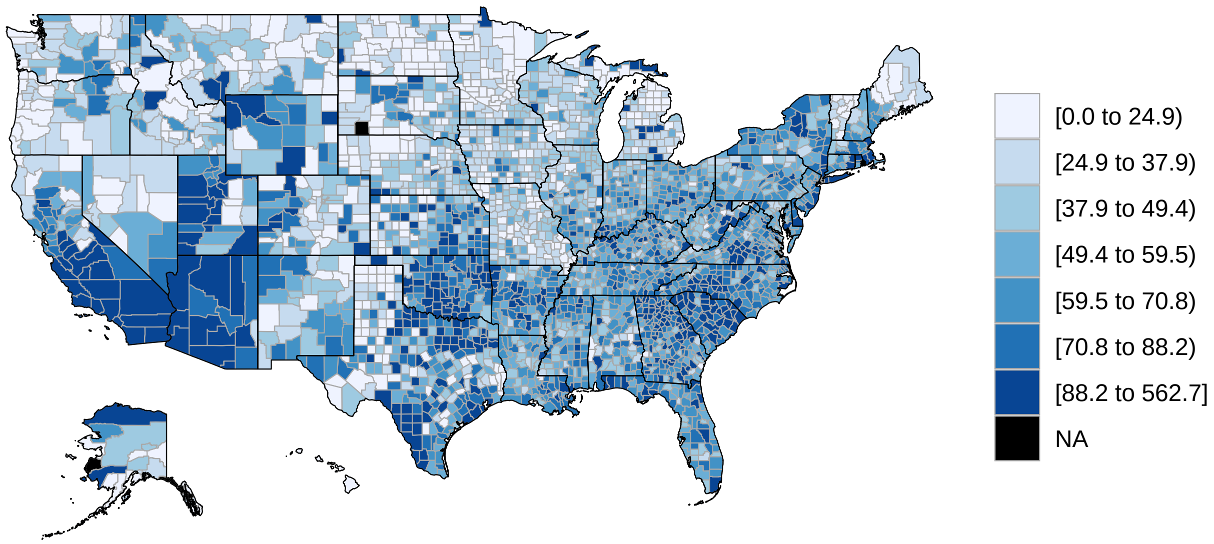
```
# mdf will contain the 7-day avg of new cases for each county.
mdf = as.data.frame(map_data)
colnames(mdf) = c("fips","state","avg_new_cases")
mdf = na.omit(mdf)
```

Next, we will use the census data in choropethr to find the average new cases per 100k population.

```
county_cases_per_100k = c()
for (f in mdf$fips){
  data(df_pop_county)
  county_pop = df_pop_county[df_pop_county$region==f,][2] # Extract the population for the county.
  avg_new_cases_per_100k = as.numeric(mdf[mdf$fips==f,][3])*100000/county_pop
  county_cases_per_100k = rbind(county_cases_per_100k,c(f,avg_new_cases_per_100k))
}
```

Lastly, we will rearrange our data and create our plot.

```
ccper = as.data.frame(county_cases_per_100k)
ccper = na.omit(ccper)
colnames(ccper) = c("region","value") # Must rename our columns to use county_choropleth
ccper$value <- ifelse(ccper$value < 0, 0, ccper$value)
# Convert to numeric data.
ccper <- ccper %>%
    mutate(region = as.numeric(region))
ccper <- ccper %>%
    mutate(value = as.numeric(value))
county_choropleth(ccper,num_colors = 7)
```



In general, our heatmap looks pretty good. The main hotspots seem to be in the same places as the NYT article. Notably, Southern California, Arizona, Oklahoma, Northwest Wyoming, and the Southern Border of Texas seem especially notable for high number of new cases per population. There are also some isolated spots with many new cases in the East, such as portions of South Carolina, and central New York.

Thus, this plot does seem somewhat reproducible in terms of its overall trends. However, it's notable that census data was not included with the code (as far as I know), so I had to find a package that could give me that data. In addition, under my analysis, certain counties actually had less new cases than 14 days ago, and I had to set them to '0' new cases. While the NYT article does include the option of having "Few or No New Cases" as an option, it doesn't properly define what "Few" cases really entails, and doesn't actually seem to have an option for where number of new cases is negative (total cases declined), so I'm not too sure now the NYT handled the few counties where total cases decreased over the week.

Table 4)

For Table 4, we're just taking all the counties from each state and summing up the total. This is quite self-explanatory, and we do it below.

```
state_data = c()
all_states = unique(mdf$state)
for (s in all_states){
  state_week = c()
  # For each day in the past week, we will sum up the total cases in the state
  for (d in week_dates){
    today_data = c_data[c_data$date == d,]
    state_today = sum(as.numeric(today_data[today_data$state == s,]$cases))
```

```r
    state_week = c(state_week,state_today)
  }
  # We can then find c_s, the average new cases over the last 7 days.
  new_cases_week = state_week - lag(state_week)
  c_s = mean(new_cases_week[-1])

  # We also want total cases on January 17, which is easy to find.
  state_entries = c_data[c_data$state==s,]
  state_entries_today = state_entries[state_entries$date == "2021-01-17",]
  state_total_cases = sum(as.numeric(state_entries_today$cases))
  state_data = rbind(state_data,c(s,state_total_cases,c_s))
}

# Now, name our columns and display our dataframe.
state_data = as.data.frame(state_data)
colnames(state_data) = c("state","total_cases","avg_new_cases")
state_data
```

```
##                         state total_cases     avg_new_cases
## 1                  Washington      293296  2423.28571428571
## 2                    Illinois     1071307              5717
## 3                  California     3006583  39579.7142857143
## 4                     Arizona      673882  7905.14285714286
## 5               Massachusetts      470140  5335.57142857143
## 6                   Wisconsin      568166              2558
## 7                       Texas     2127334  22782.1428571429
## 8                    Nebraska      182139  848.428571428571
## 9                        Utah      323837  2548.28571428571
## 10                     Oregon      133205  1074.57142857143
## 11                    Florida     1571271             13467
## 12                    Georgia      791322  8457.28571428571
## 13              New Hampshire       56864               752
## 14             North Carolina      675272  7395.57142857143
## 15                 New Jersey      627221  6056.14285714286
## 16                   New York     1242818  15281.2857142857
## 17                   Colorado      376921              1986
## 18                   Maryland      326648  2827.85714285714
## 19                     Nevada      261847  1869.85714285714
## 20                  Tennessee      672292  4452.71428571429
## 21                     Hawaii       24309  144.142857142857
## 22                    Indiana      593262  3798.85714285714
## 23                   Kentucky      329816  3284.71428571429
## 24                  Minnesota      446448  1401.14285714286
## 25                   Oklahoma      354979  3373.85714285714
## 26               Pennsylvania      772747  6748.71428571429
## 27             South Carolina      388184  4808.42857142857
## 28       District of Columbia       33851  294.285714285714
## 29                     Kansas      259226  1956.71428571429
## 30                   Missouri      468106              2460
## 31                    Vermont       10057  155.714285714286
## 32                   Virginia      439305  5778.42857142857
## 33                Connecticut      223422  2489.71428571429
## 34                       Iowa      305151  1209.85714285714
## 35                Louisiana      368980  3164.42857142857
## 36                     Ohio      826754  7098.42857142857
## 37                 Michigan      580394  2845.57142857143
## 38             South Dakota      105544               318
## 39                 Arkansas      271154  2296.85714285714
## 40                 Delaware       70294  717.285714285714
## 41              Mississippi      252475  1913.28571428571
## 42               New Mexico      163637  1068.57142857143
## 43             North Dakota       95886               166
## 44                  Wyoming       49363  361.571428571429
## 45                   Alaska       51630  242.142857142857
## 46                    Maine       33559  608.714285714286
## 47                  Alabama      422598  2956.85714285714
## 48                    Idaho      155617  839.285714285714
## 49                  Montana       89393  438.428571428571
## 50            West Virginia      108821              1087
## 51             Rhode Island      104443  975.571428571429
## 52           Virgin Islands        2260  31.1428571428571
## 53 Northern Mariana Islands         128 0.428571428571429
```

It seems that our results are pretty much exactly what the NYT has on their graphic, for both the fields we looked at. For example, we can see that our table has California with 3006583 cases on January 17, and 39579.7 as the daily average in the past 7 days, which is exactly what the NYT has too. This Table seems quite reproducible to me; the data was readily available, and the results were pretty easy to calculate once I knew what I was looking for, since we're mostly just taking a sum over all the cases in each state, and then a 7-day average in new cases. Since the formula for these calculations is relatively simple, I was able to figure them out myself, but I think for more complex calculations, code should be included so that we can see exactly what is being calculated for these values. I used the us-counties.csv data from the NYT github repository, but this process is actually much easier if you used their us-states.csv (and the results turn out to be the same). This is nice, since this us-states.csv provided me with a way to check results, and can also be used for an easier calculation of these values.

Conclusion

Overall, I would say that the NYT figures are quite reproducible. The data is pretty accessible for the most part, and while there are a few calculations that might have been a bit vague (such as the calculation of 7-day averages), most of the calculations are simple enough that I was able to reproduce them without too much of an issue. The organization of the data was also quite reasonable, with self-explanatory file and variable names. For the tables (2 and 4), I was able to reproduce the values pretty much exactly. For the figures (1 and 3), the overall trends matched very well. In general, most of the figures and values here were easy to calculate (averages, differences in averages, etc.), so we really didn't need any additional code/explanation to create these figures. I would personally give the reproducibility of this article an A grade, since the reproduction of these figures seems quite intuitive and easy.