R Capstone Project: COVID-19 Data Part 2

Carl Klein

2023-07-14

```r
# load data from previous section
# df: the combined US county data from 20, 21, 22
# df_sums: the total counts, new counts, and rolling counts per date
# df_per_100: the total counts, new counts, and rolling counts per date per 100k
df <- read_csv('Data/df.csv')
us_population <- read_csv("data/fips_population_estimates.csv")
# df_sums <- read_csv('Data/df_sums.csv')
# df_per_100 <- read_csv('Data/df_per_100.csv')
```

Part 2 - US State Comparison While understanding the trends on a national level can be helpful in understanding how COVID-19 impacted the United States, it is important to remember that the virus arrived in the United States at different times. For the next part of your analysis, you will begin to look at COVID related deaths and cases at the state and county-levels.

Question 1 Your first task in Part 2 is to determine the top 10 states in terms of total deaths and cases between March 15, 2020, and December 31, 2021.

Once you have both lists, briefly describe your methodology and your results.

```r
# Determine the top 10 states in terms of total deaths and cases
# between March 15, 2020, and December 31, 2021. To do this, transform your
# combined COVID-19 data to summarize total deaths and cases by state up to
# December 31, 2021.

# this first table matches what was provided in the sample
df_totals <- df %>%
  filter(date == '2021-12-31') %>%
  group_by(state, date) %>%
  summarise(total_cases = sum(cases), total_deaths = sum(deaths)) %>%
  arrange(-total_cases, -total_deaths)
```

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

```r
head(df_totals, n = 10)
```

```
## # A tibble: 10 x 4
## # Groups:   state [10]
##    state          date       total_cases total_deaths
##    <chr>          <date>           <dbl>        <dbl>
##  1 California     2021-12-31     5515613        76709
##  2 Texas          2021-12-31     4574881        76062
##  3 Florida        2021-12-31     4166392       62504
##  4 New York       2021-12-31     3473970       58993
##  5 Illinois       2021-12-31     2154058       31017
##  6 Pennsylvania   2021-12-31     2036424       36705
##  7 Ohio           2021-12-31     2016095       29447
##  8 Georgia        2021-12-31     1798497       30283
##  9 Michigan       2021-12-31     1706355       28984
## 10 North Carolina 2021-12-31     1685504       19436
```

```r
# this will provide the list for the top 10 states by total deaths
top_10_total_deaths <- df %>%
  filter(date == '2021-12-31') %>%
  group_by(state) %>%
  summarise(total_deaths = sum(deaths)) %>%
  arrange(-total_deaths) %>%
  head(n = 10)

# this will provide the list for the top 10 states by total cases
top_10_total_cases <- df %>%
  filter(date == '2021-12-31') %>%
  group_by(state) %>%
  summarise(total_cases = sum(cases)) %>%
  arrange(-total_cases) %>%
  head(n = 10)
```

– Methodology, Results, and Interpretation – The dataset containing the combined three years of COVID-19 data in the US has the cumulative sums of the cases and deaths. Therefore, since we're concerned about the result with the maximum date of 12/31/2021, we only needed to filter these dates.

From here, the main task was to create lists for the top 10 states with the highest cases and highest deaths. By grouping by states, which had multiple observations due the counties, we could summarize the sum of either cases or deaths depending on which list we were creating. After that, arrange in descending order and take the top 10 of the list.

The lists are stored in variables, but the very first pipe code above was created to match the prompt sample. This includes the date variable and arranges the rows in descending order for both deaths and cases.

Question 2 Determine the top 10 states in terms of deaths per 100,000 people and cases per 100,000 people between March 15, 2020, and December 31, 2021.

Once you have both lists, briefly describe your methodology and your results. Do you expect the lists to be different than the one produced in Question 1? Which method, total or per 100,000 people, is a better method for reporting the statistics?

```r
# Determine the top 10 states in terms of deaths and cases per 100,000 people
# between March 15, 2020, and December 31, 2021. You should first tidy and
# transform the population estimates to include population totals by state.
# Use your relational data verbs (e.g. full_join()) to join the population
# estimates with the cases and death statistics using the state name as a key.
# Then, use case_when() and grepl() to add a population column to your table
# that only includes the estimated population for the associated year.
# Finally, mutate your table to calculate deaths and cases per 100,000 people
# and summarize by state.
# summarize the total population estimate by year and state
state_pop <- us_population %>%
  group_by(Year, STNAME) %>%
  summarise(state_pop = sum(Estimate))
```

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

```
# pull out the years in the three combined years of COVID data, filter between
# desired dates, join with the state_pop dataset along state name and year
df_with_pop <- df %>%
  mutate(year = case_when(
    grepl('2020', date) ~ 2020,
    grepl('2021', date) ~ 2021)) %>%
  filter(date >= '2021/03/15') %>%
  filter(date <= '2021/12/31') %>%
  full_join(state_pop, join_by(state == STNAME, year == Year))

# recreate method from the first question, this time per 100,000 people:
# 1) recreate the sample dataframe
# 2) create the separate lists for top 10 states for deaths & cases

# recreate sample dataframe
df_totals_100 <- df_with_pop %>%
  filter(date == '2021/12/31') %>%
  mutate(deaths_per_100k = (deaths / state_pop) * 100000) %>%
  mutate(cases_per_100k = (cases / state_pop) * 100000) %>%
  select(state, date, deaths_per_100k, cases_per_100k) %>%
  arrange(-deaths_per_100k, -cases_per_100k)

head(df_totals_100, n = 100)
```

```
## # A tibble: 100 x 4
##    state                date       deaths_per_100k cases_per_100k
##    <chr>                <date>               <dbl>          <dbl>
##  1 Nevada               2021-12-31           206.          12027.
##  2 Rhode Island         2021-12-31           201.          12680.
##  3 Arizona              2021-12-31           188.          11970.
##  4 District of Columbia 2021-12-31           181.          14071.
##  5 New York             2021-12-31           178.           7811.
##  6 Delaware             2021-12-31           111.          10157.
##  7 Illinois             2021-12-31            99.0          6483.
##  8 Connecticut          2021-12-31            74.8          3454.
##  9 California           2021-12-31            70.4          4326.
## 10 Delaware             2021-12-31            67.9          4340.
## # i 90 more rows
```

```
# top 10 states for deaths per 100,000
top_10_deaths_100k <- df_with_pop %>%
  filter(date == '2021/12/31') %>%
  mutate(deaths_per_100k = (deaths / state_pop) * 100000) %>%
  select(state, deaths_per_100k) %>%
  arrange(-deaths_per_100k) %>%
  head(n = 10)
```

```r
# top 10 states for cases per 100,000
top_10_cases_100k <- df_with_pop %>%
  filter(date == '2021/12/31') %>%
  mutate(cases_per_100k = (cases / state_pop) * 100000) %>%
  select(state, cases_per_100k) %>%
  arrange(-cases_per_100k) %>%
  head(n = 10)
```

– Methodology, Results, and Interpretation – The initial setup for this problem, and hopefully useful for further questions, was to add population by state and year to the main dataset (the 3 years of combined COVID data).

After adding on the population by state and year, we again examined just the last date of the data we were interested in due to the cumulative format of the cases and deaths. From here, we were able to transform the data into the per 100,000 people columns.

It can be better to use a per 100,000 statistic for comparison purposes, especially if using state specific population. Doing so removes some volatility as we're comparing on more equal ground by approaching on essentially a per capita approach. It gives us better insights when states' populations are magnitudes apart. However, this doesn't account for situations such as population density, infrastructure, etc.

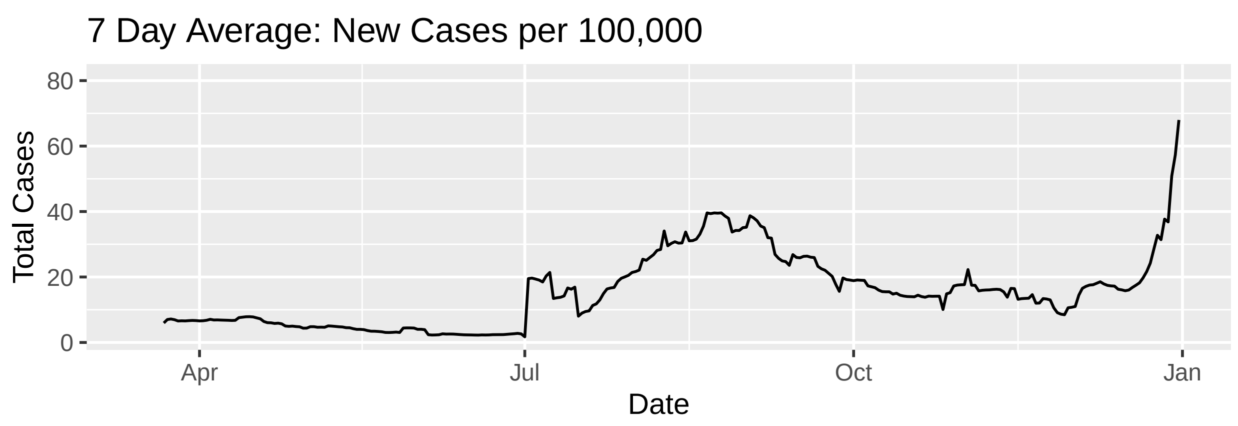
Question 3 Now, select a state and calculate the seven-day averages for new cases and deaths per 100,000 people. Once you have calculated the averages, create a visualization using ggplot2 to represent the data.

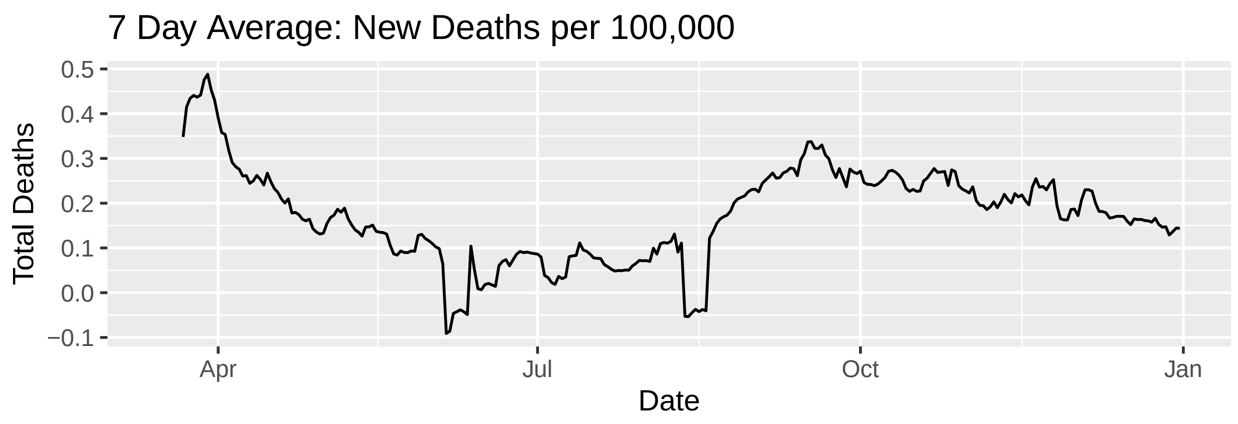
```r
# Select a state and then filter by state and date range your data from
# Question 1. Calculate the seven-day average following the same procedure
# as Part 1.

# Building from the dataset containing the states' populations we can find the
# new cases and deaths per day, and then find the 7 day averages per 100k

# remove county and fips, group by state and date
df_with_pop_totals <- df_with_pop %>%
  group_by(date, state, state_pop) %>%
  summarise(total_cases = sum(cases), total_deaths = sum(deaths))
```

```
## `summarise()` has grouped output by 'date', 'state'. You can override using the
## `.groups` argument.
```

```r
# select state
df_california <- df_with_pop_totals %>%
  filter(state == 'California')

# new cases
new_cases = c()
for (i in 1:length(df_california$total_cases) - 1) {
  new_cases[i] = df_california$total_cases[i + 1] - df_california$total_cases[i]
}
new_cases = c(0, new_cases)

# new deaths
new_deaths = c()
for (i in 1:length(df_california$total_deaths) - 1) {
  new_deaths[i] = df_california$total_deaths[i + 1] - df_california$total_deaths[i]
}
new_deaths = c(0, new_deaths)

# add on variables to the state specific dataframe
df_california <- df_california %>%
  cbind('new_cases' = new_cases, 'new_deaths' = new_deaths)

# 7 day rolling average new cases
cases_roll_7 <- df_california %>%
  rollmean(x = new_cases, k = 7, align = 'right', fill = NA) %>%
  lag(n = 1)

# 7 day rolling average new deaths
deaths_roll_7 <- df_california %>%
  rollmean(x = new_deaths, k = 7, align = 'right', fill = NA) %>%
  lag(n = 1)

df_california <- df_california %>%
  cbind('cases_roll_7' = cases_roll_7, 'deaths_roll_7' = deaths_roll_7)


# now create per 100k columns
df_california <- df_california %>%
  mutate(new_cases_100 = (new_cases / state_pop) * 100000) %>%
  mutate(new_deaths_100 = (new_deaths / state_pop) * 100000) %>%
  mutate(cases_roll_100 = (cases_roll_7 / state_pop) * 100000) %>%
  mutate(deaths_roll_100 = (deaths_roll_7 / state_pop) * 100000)
```

```r
# plot for the cases
plot1 <- ggplot(data = df_california) +
  geom_line(aes(x = date, y = cases_roll_100)) +
  labs(title = '7 Day Average: New Cases per 100,000') +
  xlab('Date') +
  ylab('Total Cases') +
  scale_y_continuous(labels = scales::comma)

# plot for the total deaths
plot2 <- ggplot(data = df_california) +
  geom_line(aes(x = date, y = deaths_roll_100)) +
  labs(title = '7 Day Average: New Deaths per 100,000') +
  xlab('Date') +
  ylab('Total Deaths') +
  scale_y_continuous(labels = scales::comma)

# put the plots in a grid together (uses gridExtra)
grid.arrange(plot1, plot2)
```





```r
by_state <- function(df = df_with_pop_totals, state_name) {
  # filter by state
  df_state <- df %>%
  filter(state == state_name)

  # calculate new cases
  new_cases = c()
  for (i in 1:length(df_state$total_cases) - 1) {
    new_cases[i] = df_state$total_cases[i + 1] - df_state$total_cases[i]
  }
  new_cases = c(0, new_cases)

  # calculate new deaths
  new_deaths = c()
  for (i in 1:length(df_state$total_deaths) - 1) {
    new_deaths[i] = df_state$total_deaths[i + 1] - df_state$total_deaths[i]
  }
  new_deaths = c(0, new_deaths)

  # bind the new cases and deaths to the dataframe
  df_state <- df_state %>%
    cbind('new_cases' = new_cases, 'new_deaths' = new_deaths)


  # calculate the 7 day rolling averages for the new cases
  cases_roll_7 <- df_state %>%
    rollmean(x = new_cases, k = 7, align = 'right', fill = NA) %>%
    lag(n = 1)

  # calculate the 7 day rolling averages for the new deaths
  deaths_roll_7 <- df_state %>%
    rollmean(x = new_deaths, k = 7, align = 'right', fill = NA) %>%
    lag(n = 1)

  # bind the rolling cases and deaths to the dataframe
  df_state <- df_state %>%
    cbind('cases_roll_7' = cases_roll_7, 'deaths_roll_7' = deaths_roll_7)


  # add mutated versions of the 4 vectors created for the 100k by state result
  df_state <- df_state %>%
    mutate(new_cases_100 = (new_cases / state_pop) * 100000) %>%
    mutate(new_deaths_100 = (new_deaths / state_pop) * 100000) %>%
    mutate(cases_roll_100 = (cases_roll_7 / state_pop) * 100000) %>%
    mutate(deaths_roll_100 = (deaths_roll_7 / state_pop) * 100000)

  return(df_state)
}
```

– Methodology, Results, and Interpretation – Borrowed some of the code from the solution in part 1 of this analysis. The main difference being we also needed to include state specification in this portion, whereas before we were performing calculations on total populations by date.

Additionally, I created a function which will return our requested statistics. I looked ahead and saw we had to perform these calculations for a few more states, so I thought this would streamline that process.

Question 4 Using the same state, identify the top 5 counties in terms of deaths and cases per 100,000 people.

```r
# Using the same state as Question 2, filter your state and date range from the
# combined data set from Part 1 and summarize cases and deaths.
# Produce two lists arranged by deaths and cases. When transforming the data,
# be sure to include the "fips" column as you will need this to complete Question 5.
# California counties arranged by descending cases
cali_county_cases <- df_with_pop %>%
  filter(date == '2021-12-31') %>%
  filter(state == 'California') %>%
  group_by(county, date, fips, state_pop) %>%
  summarise(total_cases = sum(cases), total_deaths = sum(deaths)) %>%
  mutate(total_cases_100 = (total_cases / state_pop) * 100000) %>%
  mutate(total_deaths_100 = (total_deaths / state_pop) * 100000) %>%
  arrange(-total_cases_100)
```

```
## `summarise()` has grouped output by 'county', 'date', 'fips'. You can override
## using the `.groups` argument.
```

```r
# California counties arranged by descending deaths
cali_county_deaths <- df_with_pop %>%
  filter(date == '2021-12-31') %>%
  filter(state == 'California') %>%
  group_by(county, date, fips, state_pop) %>%
  summarise(total_cases = sum(cases), total_deaths = sum(deaths)) %>%
  mutate(total_cases_100 = (total_cases / state_pop) * 100000) %>%
  mutate(total_deaths_100 = (total_deaths / state_pop) * 100000) %>%
  arrange(-total_deaths_100)
```

```
## `summarise()` has grouped output by 'county', 'date', 'fips'. You can override
## using the `.groups` argument.
```

```
head(cali_county_cases, n = 5)
```

```
## # A tibble: 5 x 8
## # Groups:   county, date, fips [5]
##   county     date       fips  state_pop total_cases total_deaths total_cases_100
##   <chr>      <date>     <chr>     <dbl>       <dbl>        <dbl>           <dbl>
## 1 Los Angel~ 2021-12-31 06037  39237836     1697286        27637           4326.
## 2 San Diego  2021-12-31 06073  39237836      451082         4469           1150.
## 3 Riverside  2021-12-31 06065  39237836      415024         5585           1058.
## 4 San Berna~ 2021-12-31 06071  39237836      399021         6051           1017.
## 5 Orange     2021-12-31 06059  39237836      358569         5890            914.
## # i 1 more variable: total_deaths_100 <dbl>
```

```
head(cali_county_deaths, n = 5)
```

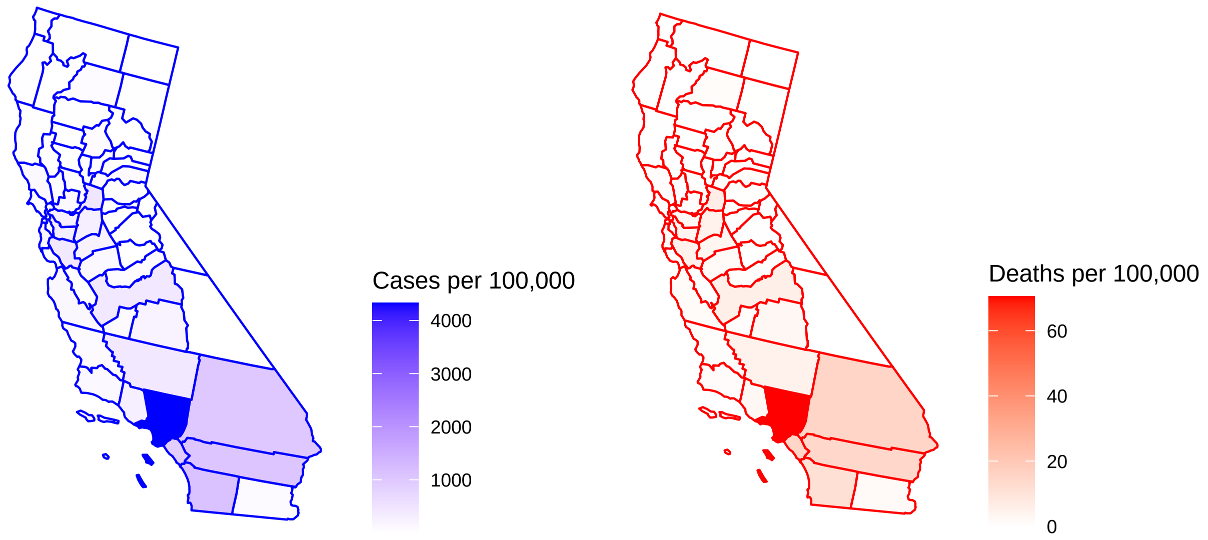
```
## # A tibble: 5 x 8
## # Groups:   county, date, fips [5]
##   county     date       fips  state_pop total_cases total_deaths total_cases_100
##   <chr>      <date>     <chr>     <dbl>       <dbl>        <dbl>           <dbl>
## 1 Los Angel~ 2021-12-31 06037  39237836     1697286        27637           4326.
## 2 San Berna~ 2021-12-31 06071  39237836      399021         6051           1017.
## 3 Orange     2021-12-31 06059  39237836      358569         5890            914.
## 4 Riverside  2021-12-31 06065  39237836      415024         5585           1058.
## 5 San Diego  2021-12-31 06073  39237836      451082         4469           1150.
## # i 1 more variable: total_deaths_100 <dbl>
```

– Methodology, Results, and Interpretation – Using a similar process as a previous question, we can use the original dataset containing the 3 years of combine COVID-19 data, merged with population by state and year, and filter using our end date of interest due to the cumulative format of cases and deaths.

Question 5 Modify the code below for the map projection to plot county-level deaths and cases per 100,000 people for your state.

```
plot1 <- plot_usmap(regions = "county",
           include="CA",
           data = cali_county_cases,
           values = "total_cases_100",
           color = "blue") +
  scale_fill_continuous(
    low = "white",
    high = "blue",
    name = "Cases per 100,000") +
  theme(legend.position = 'right')
```

```r
plot2 <- plot_usmap(regions = "county",
          include="CA",
          data = cali_county_deaths,
          values = "total_deaths_100",
          color = "red") +
  scale_fill_continuous(
    low = "white",
    high = "red",
    name = "Deaths per 100,000") +
  theme(legend.position = 'right')

grid.arrange(plot1, plot2, ncol = 2)
```



– Methodology, Results, and Interpretation – This is a great visual to show not only where the higher concentrations of cases and deaths were, but also the relationship between cases and deaths. The plots make a case for counties with a higher concentration of cases also saw a higher concentration of deaths.

Question 6 Finally, select three other states and calculate the seven-day averages for new deaths and cases per 100,000 people for between March 15, 2020, and December 31, 2021.

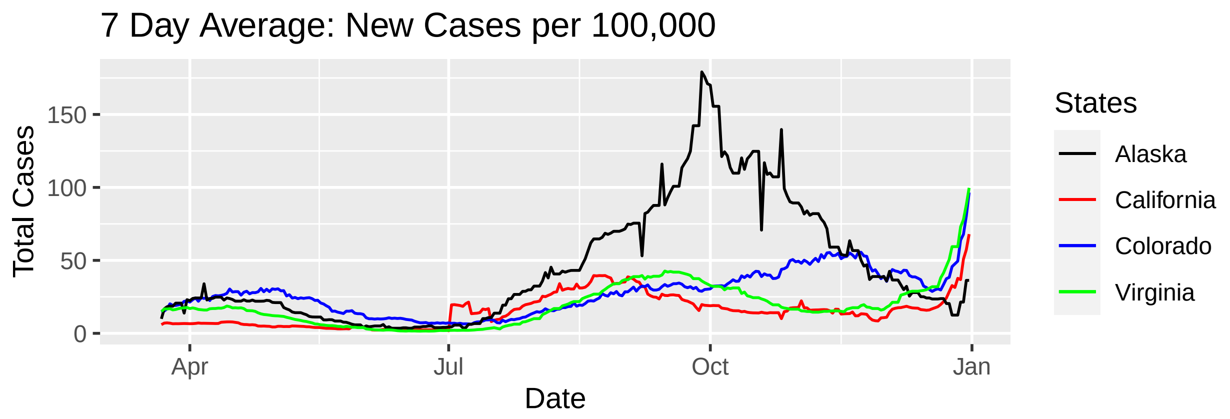
```r
# using the function we created above, we'll calculate the datasets for the states:
df_california <- by_state(df = df_with_pop_totals, 'California')
df_colorado <- by_state(df = df_with_pop_totals, 'Colorado')
```

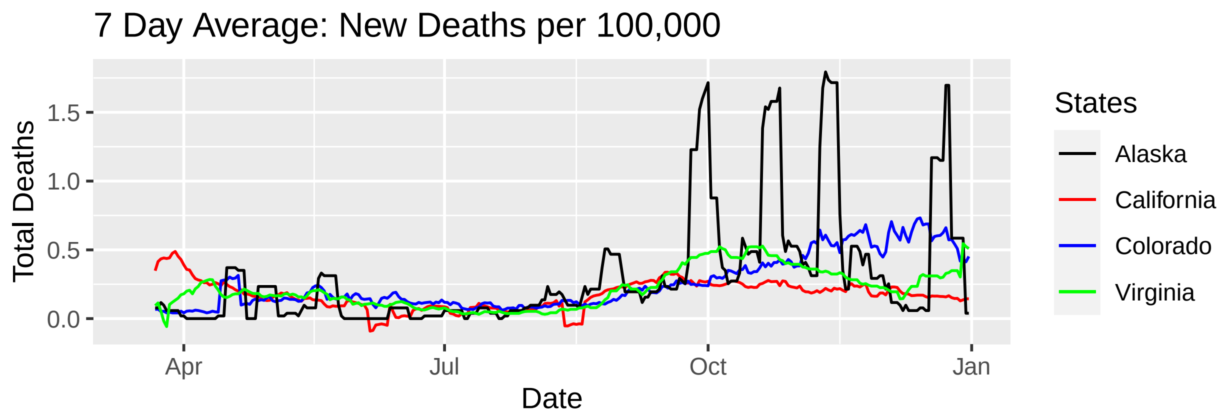
```r
df_alaska <- by_state(df = df_with_pop_totals, 'Alaska')
df_virginia <- by_state(df = df_with_pop_totals, 'Virginia')

# combine the rolling cases
combined_cases <- list(df_california[c('date', 'cases_roll_100')],
                       df_colorado[c('date', 'cases_roll_100')],
                       df_alaska[c('date', 'cases_roll_100')],
                       df_virginia[c('date', 'cases_roll_100')]) %>%
  reduce(full_join, by = 'date') %>%
  rename(c('California' = 'cases_roll_100.x',
           'Colorado' = 'cases_roll_100.y',
           'Alaska' = 'cases_roll_100.x.x',
           'Virginia' = 'cases_roll_100.y.y'))

# combine the rolling deaths
combined_deaths <- list(df_california[c('date', 'deaths_roll_100')],
                       df_colorado[c('date', 'deaths_roll_100')],
                       df_alaska[c('date', 'deaths_roll_100')],
                       df_virginia[c('date', 'deaths_roll_100')]) %>%
  reduce(full_join, by = 'date') %>%
  rename(c('California' = 'deaths_roll_100.x',
           'Colorado' = 'deaths_roll_100.y',
           'Alaska' = 'deaths_roll_100.x.x',
           'Virginia' = 'deaths_roll_100.y.y'))
```

– Methodology, Results, and Interpretation – We created a function when we were initially tasked with finding this information for a single state. Using that function across the states, we can then combine the necessary data together.

Question 7 Create a visualization comparing the seven-day averages for new deaths and cases per 100,000 people for the four states you selected.

```r
# plot for the cases
plot1 <- ggplot(data = combined_cases) +
  geom_line(aes(x = date, y = California, color = 'California')) +
  geom_line(aes(x = date, y = Colorado, color = 'Colorado')) +
  geom_line(aes(x = date, y = Alaska, color = 'Alaska')) +
  geom_line(aes(x = date, y = Virginia, color = 'Virginia')) +
  labs(title = '7 Day Average: New Cases per 100,000') +
  xlab('Date') +
  ylab('Total Cases') +
  scale_color_manual(name = 'States',
                     values = c('California' = 'red',
                                'Colorado' = 'blue',
                                'Alaska' = 'black',
                                'Virginia' = 'green'))

# plot for the total deaths
plot2 <- ggplot(data = combined_deaths) +
  geom_line(aes(x = date, y = California, color = 'California')) +
  geom_line(aes(x = date, y = Colorado, color = 'Colorado')) +
  geom_line(aes(x = date, y = Alaska, color = 'Alaska')) +
  geom_line(aes(x = date, y = Virginia, color = 'Virginia')) +
  labs(title = '7 Day Average: New Deaths per 100,000') +
  xlab('Date') +
  ylab('Total Deaths') +
  scale_color_manual(name = 'States',
                     values = c('California' = 'red',
                                'Colorado' = 'blue',
                                'Alaska' = 'black',
                                'Virginia' = 'green'))

# put the plots in a grid together (uses gridExtra)
grid.arrange(plot1, plot2)
```





– Methodology, Results, and Interpretation – Using different plots for cases and deaths due to the magnitude of difference between their values (even at the per 100,000 people level), we can plot all of the states' values per category.

Visually, Alaska stands out with higher cases and deaths and very sharp spikes.