COVID19 Analysis Final Project

Michael Grybko

2024-02-20

Introduction

The COVID-19 outbreak started in December of 2019 and was declared a pandemic by the World Health Organization on March 11th, 2020 (World Health Organization: WHO, 2024). As of August 2023, The World Health Organization (2023) estimated there were over 760 million cases and 6.9 million deaths recorded worldwide. The virus that causes COVID-19 is most often spread between people in close contact, and avoiding crowds and wearing mask is recommended to prevent the spread of COVID-19 (World Health Organization: WHO, 2023). Since close contact with others has been established as an important factor in the spread of COVID-19, it would be reasonable to assume areas with higher populations would be more adversely impacted by the disease. Here the relationship between population and COVID-19 cases and deaths will be examined.

Source and Description of the Data

The COVID19 data used for this research can be found at the CSSEGISandData/COVID-19 repository: https://github.com/CSSEGISandData/COVID-19/commits?author=CSSEGISandData

There are multiple data sets used in this research. Some of the data used in this research is a record of confirmed cases of COVID19 and deaths attributed to COVID19 by date. The data starts at the beginning of the pandemic through early 2023. There are data sets specific to the United States and there are global data sets. There are also data sets for population counts of the the areas represented in the COVID19 data.

Please be sure to install the necessary map packages for the map data to be properly displayed.

Libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(lubridate)
library(maps)
```

```
## Warning: package 'maps' was built under R version 4.2.3
```

```
##
## Attaching package: 'maps'
##
## The following object is masked from 'package:purrr':
##
##     map
```

Read in Data

```
url_in <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_cov

file_names <- c("time_series_covid19_confirmed_global.csv",
  "time_series_covid19_deaths_global.csv",
  "time_series_covid19_confirmed_US.csv",
  "time_series_covid19_deaths_US.csv")

urls <- str_c(url_in,file_names)
#urls
global_cases <- read_csv(urls[1])
```

```
## Rows: 289 Columns: 1147
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (2): Province/State, Country/Region
## dbl (1145): Lat, Long, 1/22/20, 1/23/20, 1/24/20, 1/25/20, 1/26/20, 1/27/20,...
##
## 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.
```

```
global_deaths <- read_csv(urls[2])
```

```
## Rows: 289 Columns: 1147
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (2): Province/State, Country/Region
## dbl (1145): Lat, Long, 1/22/20, 1/23/20, 1/24/20, 1/25/20, 1/26/20, 1/27/20,...
##
## 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.
```

```
us_cases <- read_csv(urls[3])
```

```
## Rows: 3342 Columns: 1154
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (1148): UID, code3, FIPS, Lat, Long_, 1/22/20, 1/23/20, 1/24/20, 1/25/20...
##
## 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.
```

```
us_deaths <- read_csv(urls[4])
```

```
## Rows: 3342 Columns: 1155
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (1149): UID, code3, FIPS, Lat, Long_, Population, 1/22/20, 1/23/20, 1/24...
##
## 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.
```

```
# global population data
uid_lookup_url <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/U

uid <- read_csv(uid_lookup_url) %>%
  select(-c(Lat, Long_, Combined_Key, code3, iso2, iso3, Admin2))
```

```
## Rows: 4321 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (7): iso2, iso3, FIPS, Admin2, Province_State, Country_Region, Combined_Key
## dbl (5): UID, code3, Lat, Long_, Population
##
## 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.
```

Tidy Global Data

```
global_cases <- global_cases %>%
  pivot_longer(cols = -c('Province/State',
                         'Country/Region', Lat, Long),
               names_to = "date",
               values_to = "cases") %>%
  select(-c(Lat,Long))

global_deaths <- global_deaths %>%
  pivot_longer(cols = -c('Province/State',
                         'Country/Region', Lat, Long),
               names_to = "date",
               values_to = "deaths") %>%
  select(-c(Lat,Long))
```

Join Golbal Cases and Deaths

```
global <- global_cases %>%
  full_join(global_deaths) %>%
  rename(Country_Region = `Country/Region`,
         Province_State = `Province/State`) %>%
  mutate(date = mdy(date))
```

```
## Joining with `by = join_by(`Province/State`, `Country/Region`, date)`
```

```
# add Combined_Key column
global <- global %>%
  unite("Combined_Key",
        c(Province_State, Country_Region),
```

```
        sep = ",",
        na.rm = TRUE,
        remove = FALSE)

# join population data with global covid data
global <- global %>%
  left_join(uid, by = c("Province_State", "Country_Region")) %>%
  select(-c(UID, FIPS)) %>%
  select(Province_State, Country_Region, date, cases, deaths, Population, Combined_Key)
```

Total Cases and Deaths by Country

```
global_totals_by_date <- global %>%
  group_by(Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths),
            Population = sum (Population)) %>%
  mutate(deaths_per_mill = deaths * 1000000 / Population) %>%
  mutate(cases_per_mill = cases * 1000000 / Population) %>%
  select(Country_Region, date, cases, deaths, cases_per_mill,
         deaths_per_mill, Population) %>%
  ungroup()
```

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

```
# total cases and deaths by country
world_country_totals <- global_totals_by_date %>%
  group_by(Country_Region) %>%
  summarize(deaths = max(deaths), cases = max(cases),
            population = max(Population),
            cases_per_thou = 1000 * cases / population,
            deaths_per_thou = 1000 * deaths / population) %>%
  filter(cases > 0, population > 0)
```

Tidy US Data

```
us_cases <- us_cases %>%
  pivot_longer(cols = -(UID:Combined_Key),
               names_to = "date",
               values_to = "cases") %>%
  select(Admin2:cases) %>%
  mutate(date = mdy(date)) %>%
  select(-c(Lat, Long_))

us_deaths <- us_deaths %>%
  pivot_longer(cols = -(UID:Population),
               names_to = "date",
               values_to = "deaths") %>%
  select(Admin2:deaths) %>%
  mutate(date = mdy(date)) %>%
  select(-c(Lat, Long_))
```

```
# merge US cases and deaths
us <- us_cases %>% full_join(us_deaths)
```

```
## Joining with `by = join_by(Admin2, Province_State, Country_Region,
## Combined_Key, date)`
```

Sum Cases by State and Country

```
# group US data
us_by_state <- us %>%
  group_by(Province_State, Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths),
            Population = sum(Population)) %>%
  mutate(deaths_per_mill = deaths *1000000 / Population) %>%
  mutate(cases_per_mill = cases *1000000 / Population) %>%
  select(Province_State, Country_Region, date, cases,
         deaths, cases_per_mill, deaths_per_mill, Population) %>%
  ungroup()
```

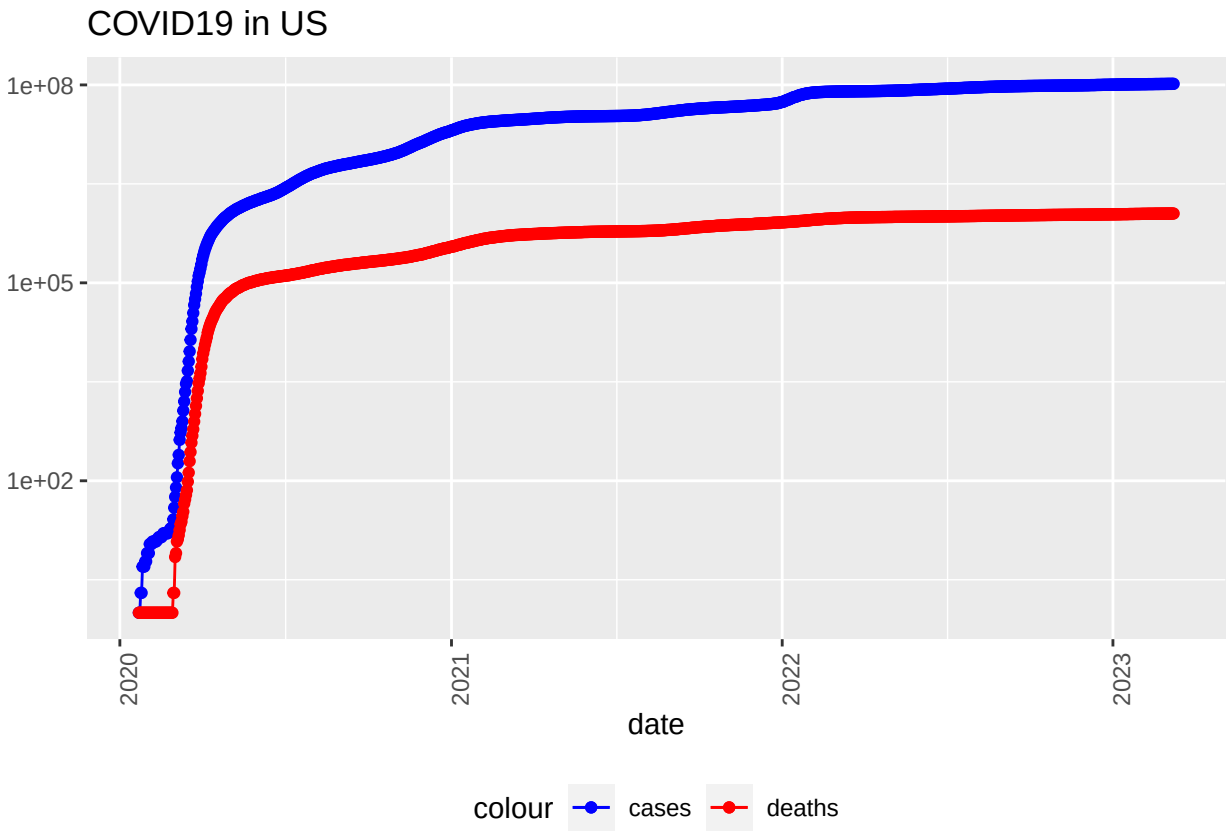
```
## `summarise()` has grouped output by 'Province_State', 'Country_Region'. You can
## override using the `.groups` argument.
```

```
us_totals <- us_by_state %>%
  group_by(Country_Region, date) %>%
  summarize(cases = sum(cases), deaths = sum(deaths),
            Population = sum (Population)) %>%
  mutate(deaths_per_mill = deaths * 1000000 / Population) %>%
  select(Country_Region, date, cases, deaths,
         deaths_per_mill, Population) %>%
  ungroup()
```

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

US Totals Graph Over Time

```
us_totals %>%
  filter(cases > 0) %>%
  ggplot(aes(x = date, y =cases)) +
  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +
  geom_line(aes(y = deaths, color = "deaths")) +
  geom_point(aes(y = deaths, color = "deaths")) +
  scale_y_log10() +
  scale_color_manual(values = c("cases" = "blue", "deaths" = "red")) +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = "COVID19 in US", y = NULL)
```



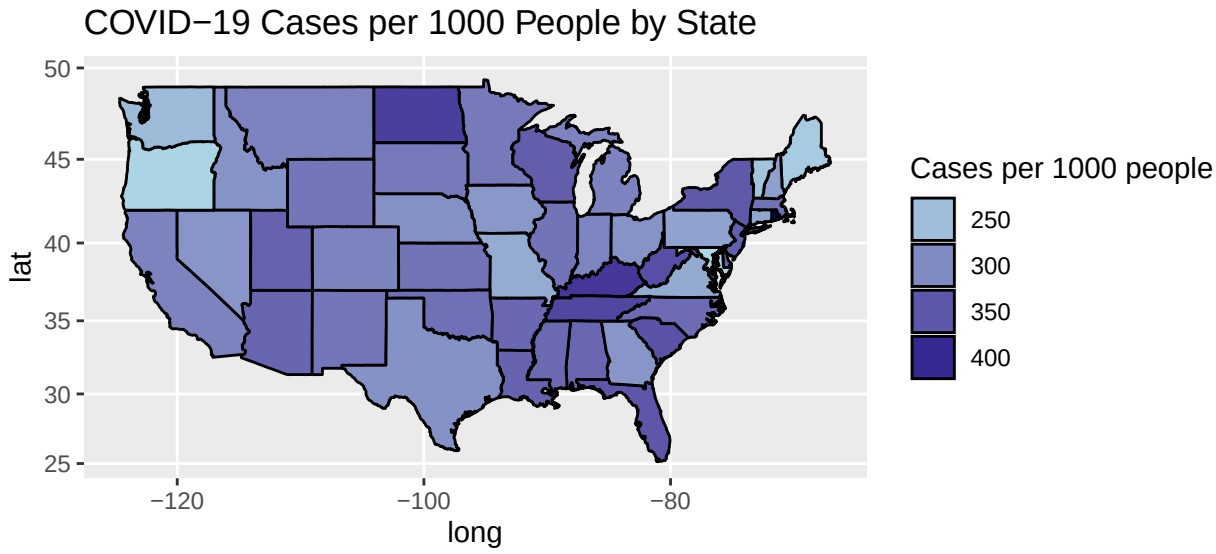
United States Cases and Deaths per Thousand by State

```
us_state_totals <- us_by_state %>%
  group_by(Province_State) %>%
  summarize(deaths = max(deaths), cases = max(cases),
            population = max(Population),
            cases_per_thou = 1000 * cases / population,
            deaths_per_thou = 1000 * deaths / population) %>%
  filter(cases > 0, population > 0)
```

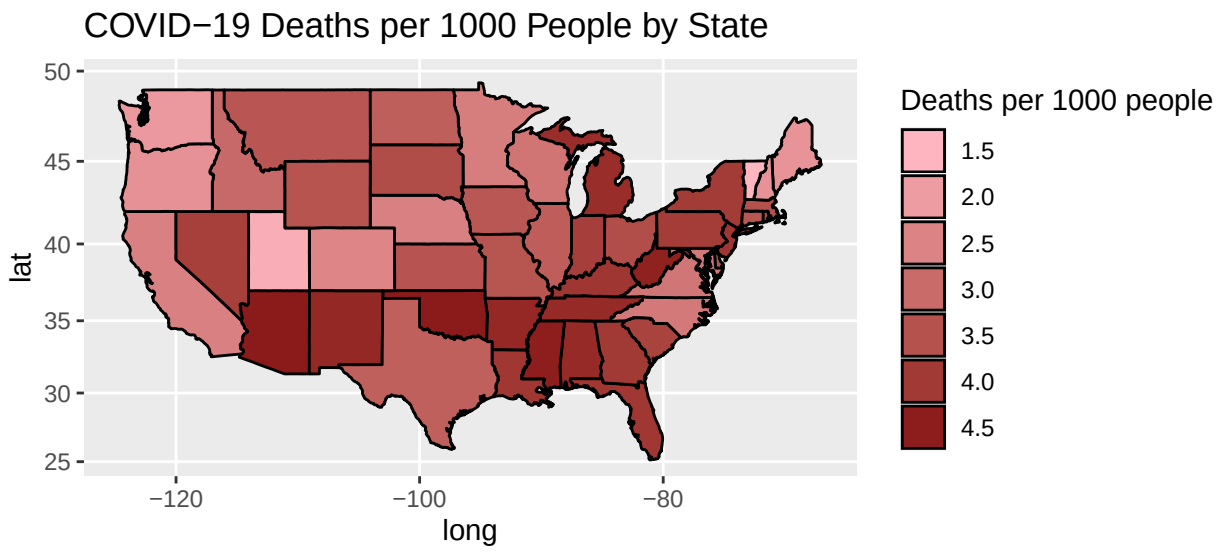
Maps of COVID-19 Cases and Deaths per Thousand

```
# load state map data
us_map_data <- map_data("state")

# rename the 'region' column to match the column name in us_state_totals
names(us_map_data)[names(us_map_data) == "region"] <- "Province_State"

# capitalize the first letter of each word in state names
us_map_data$Province_State <- tools::toTitleCase(us_map_data$Province_State)

# filter entries in us_state_totals not in state_map_summarized
us_state_totals_filtered <- us_state_totals %>%
  filter(Province_State %in% us_map_data$Province_State)
```

```r
# merge state map data with filtered
us_map_data <- merge(us_map_data, us_state_totals_filtered, by = "Province_State", all.x = TRUE)

# This data set is for the linear models.
us_map_data_aggregated <- us_map_data %>%
  group_by(Province_State) %>%
  summarise(mean_lat = mean(lat),
            mean_long = mean(long))

us_map_data_all <- merge(us_state_totals_filtered,
                     us_map_data_aggregated[, c("Province_State", "mean_lat", "mean_long")],
                     by = "Province_State",
                     all.x = TRUE)

# change state names back to original format to graph
us_map_data$Province_State <- tolower(us_map_data$Province_State)
colnames(us_map_data)[1] ="region"

us_cases_map <- ggplot(us_map_data, aes(long, lat, group = group, fill = cases_per_thou)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Cases per 1000 people", low = "lightblue", high = "navyblue", guide = "leg
  labs(title = "COVID-19 Cases per 1000 People by State")

# uncomment to save
#ggsave(us_cases_map, filename = "us_cases_map.jpeg", width = 10, height = 8, units = "in")

us_deaths_map <- ggplot(us_map_data, aes(long, lat, group = group, fill = deaths_per_thou)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Deaths per 1000 people", low = "lightpink", high = "firebrick4", guide =
  labs(title = "COVID-19 Deaths per 1000 People by State")

# uncomment to save
#ggsave(us_deaths_map, filename = "us_deaths_map.jpeg", width = 10, height = 8, units = "in")

us_cases_map
```




```
us_deaths_map
```



Closer Examination of States Differently Imacted by COVID-19: Colorado, Vermont and Kentucky

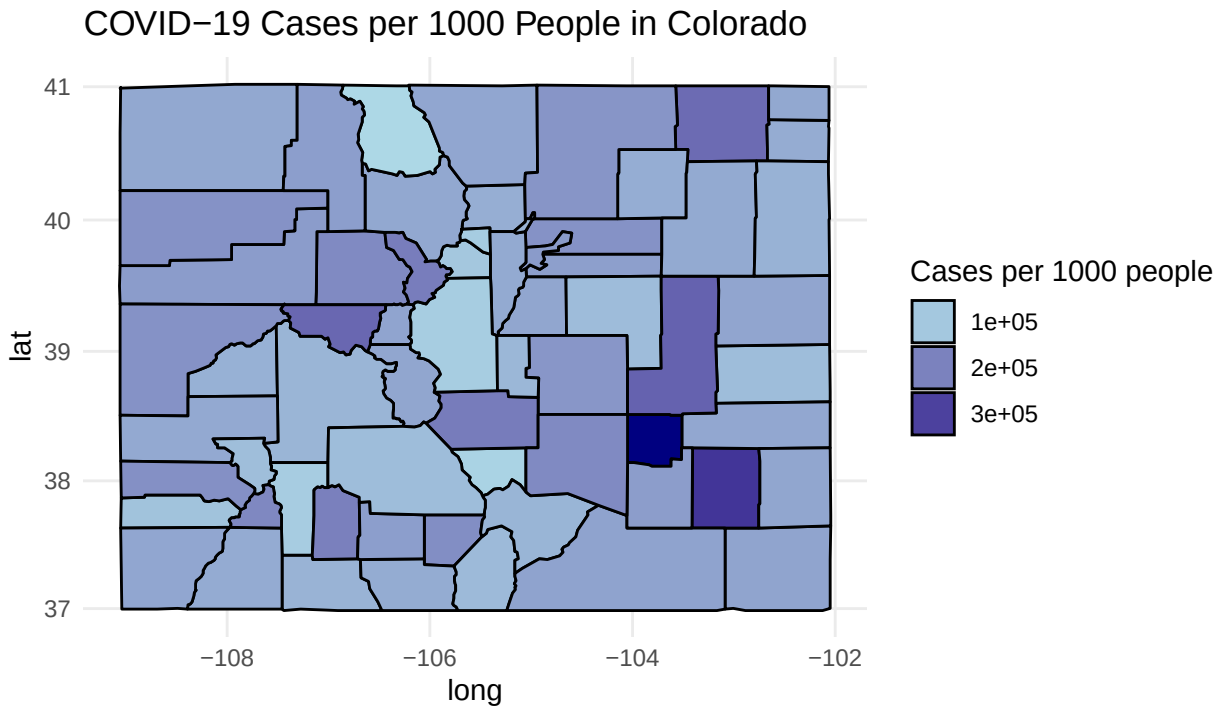
Colorado

```
# Colorado counties
co_county_map <- map_data("county", region = "colorado")

# filter data by county
colorado_data <- us %>%
  filter(Province_State == "Colorado")

# cases and deaths by county
co_county_cases <- colorado_data %>%
  group_by(Admin2, Population) %>%
```

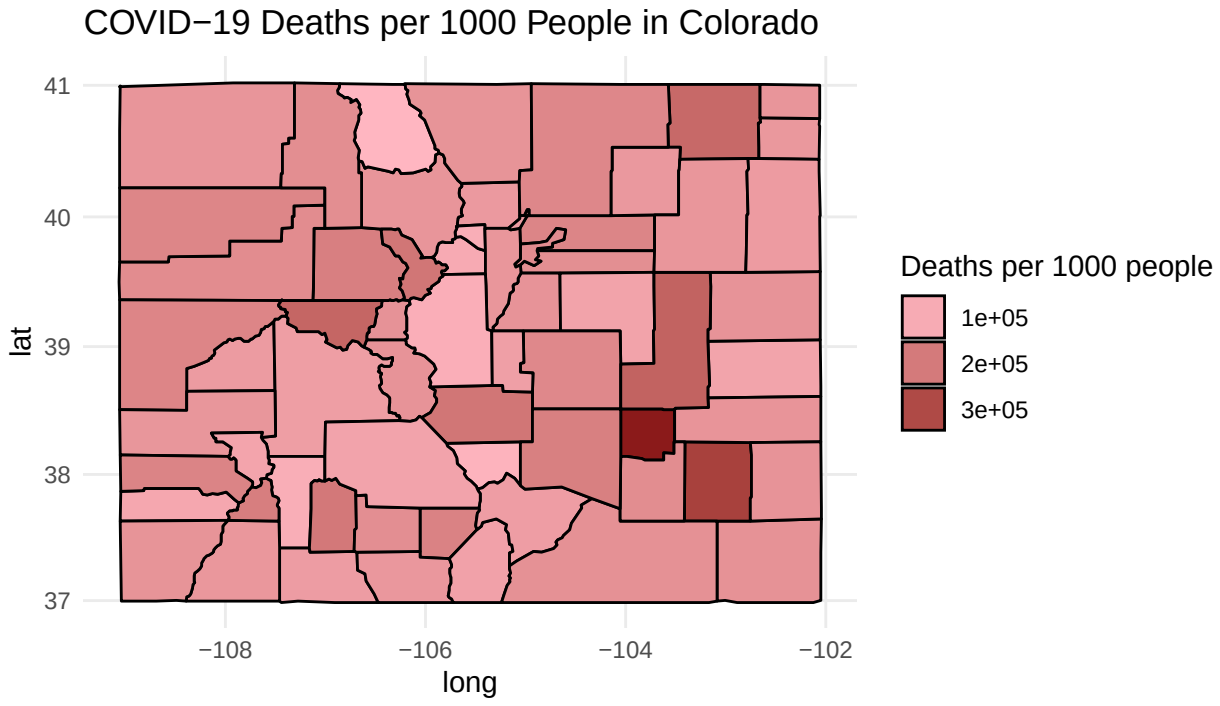
```
  summarise(total_cases = sum(cases, na.rm = TRUE),
            total_deaths = sum(deaths, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'Admin2'. You can override using the
## `.groups` argument.
```

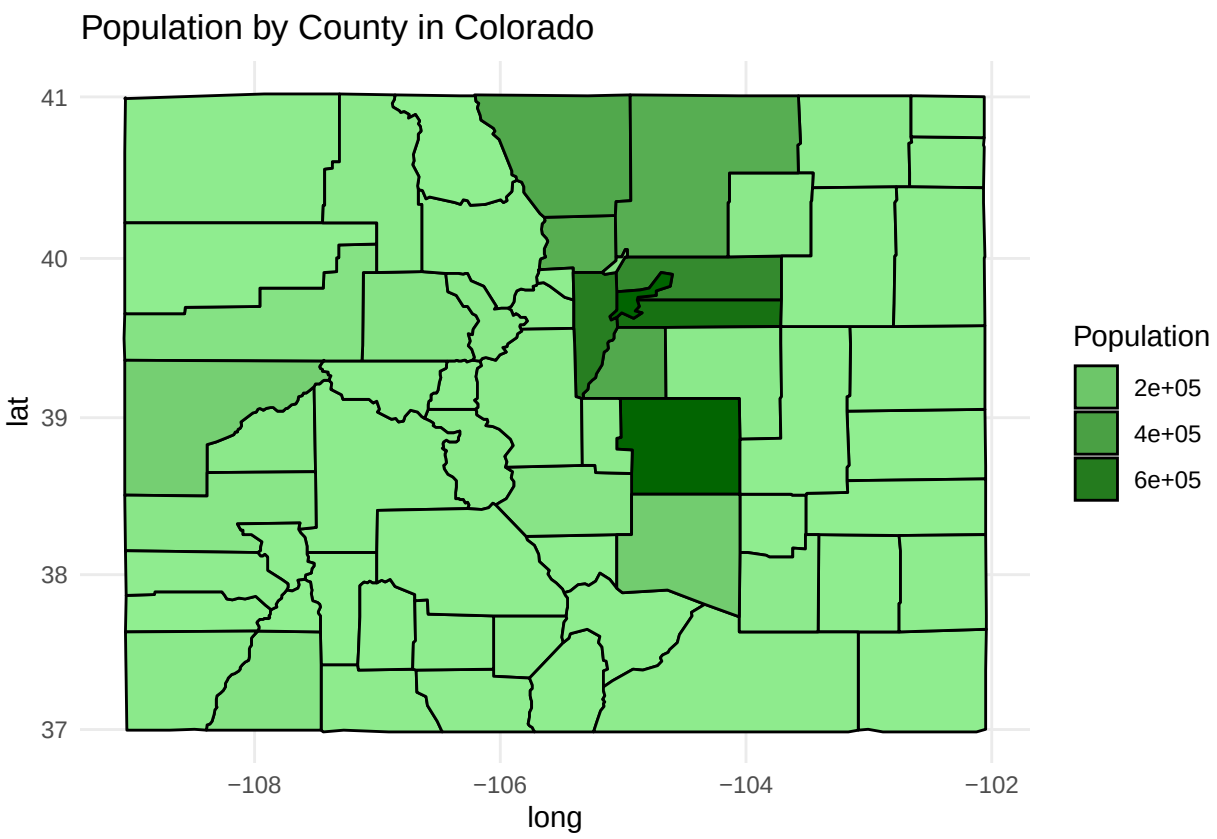
```
co_county_cases$Admin2 <- tolower(co_county_cases$Admin2)

# merge county boundary data with case data
co_county_map <- merge(co_county_map, co_county_cases, by.x = "subregion", by.y = "Admin2", all.x = TRU

# cases per thousand
co_county_map$cases_per_thou <- co_county_map$total_cases / co_county_map$Population * 1000

co_county_map$deaths_per_thou <- co_county_map$total_cases / co_county_map$Population * 1000
# CO heat map of cases per thou
co_county_cases_map <- ggplot(co_county_map, aes(long, lat, group = group, fill = cases_per_thou)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Cases per 1000 people", low = "lightblue", high = "navyblue", guide = "leg
  labs(title = "COVID-19 Cases per 1000 People in Colorado") +
  theme_minimal()

# uncomment to save
#ggsave(co_county_cases_map, filename = "co_county_cases_map.jpeg", width = 10, height = 8, units = "in

# CO heat map of deaths per thou
co_county_deaths_map <- ggplot(co_county_map, aes(long, lat, group = group, fill = deaths_per_thou)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Deaths per 1000 people", low = "lightpink", high = "firebrick4", guide =
  labs(title = "COVID-19 Deaths per 1000 People in Colorado") +
  theme_minimal()

# uncomment to save
#ggsave(co_county_deaths_map, filename = "co_county_deaths_map.jpeg", width = 10, height = 8, units = "

# CO heat map of population by county
co_county_pop_map <- ggplot(co_county_map, aes(long, lat, group = group, fill = Population)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Population", low = "lightgreen", high = "darkgreen", guide = "legend") +
  labs(title = "Population by County in Colorado") +
  theme_minimal()

# uncomment to save
#ggsave(co_county_pop_map, filename = "co_county_pop_map.jpeg", width = 10, height = 8, units = "in")

co_county_cases_map
```




```
co_county_deaths_map
```




```
co_county_pop_map
```



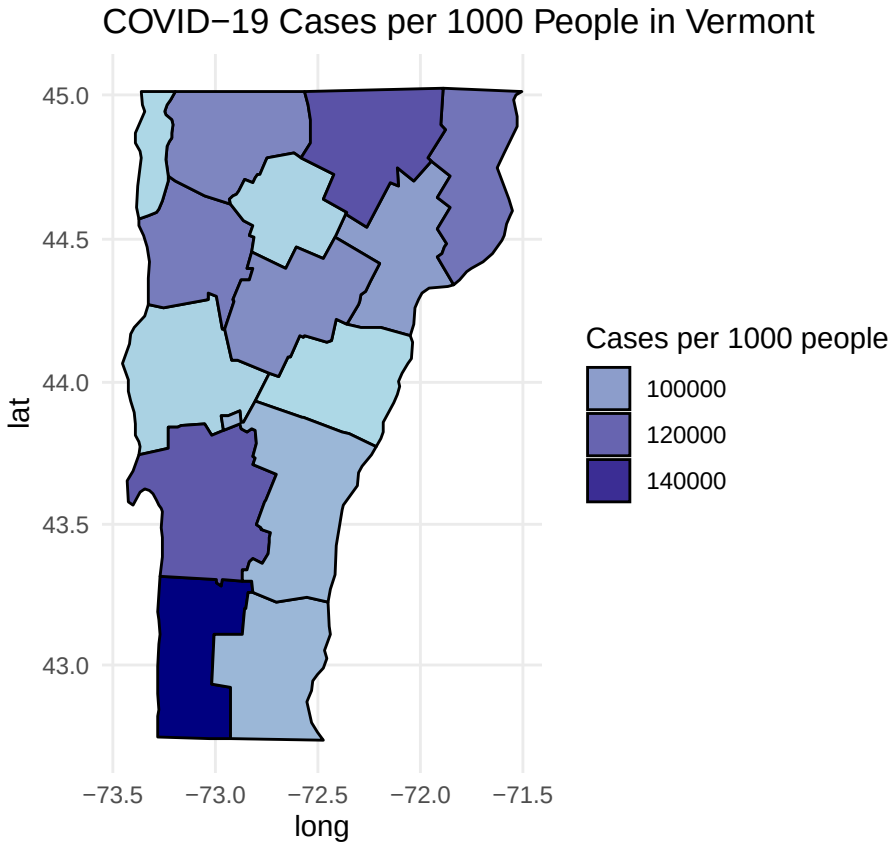
Vermont

```
# Vermont counties
vt_county_map <- map_data("county", region = "vermont")

# filter data by county
vermont_data <- us %>%
  filter(Province_State == "Vermont")

# cases and deaths by county
vt_county_cases <- vermont_data %>%
  group_by(Admin2, Population) %>%
  summarise(total_cases = sum(cases, na.rm = TRUE),
            total_deaths = sum(deaths, na.rm = TRUE))
```

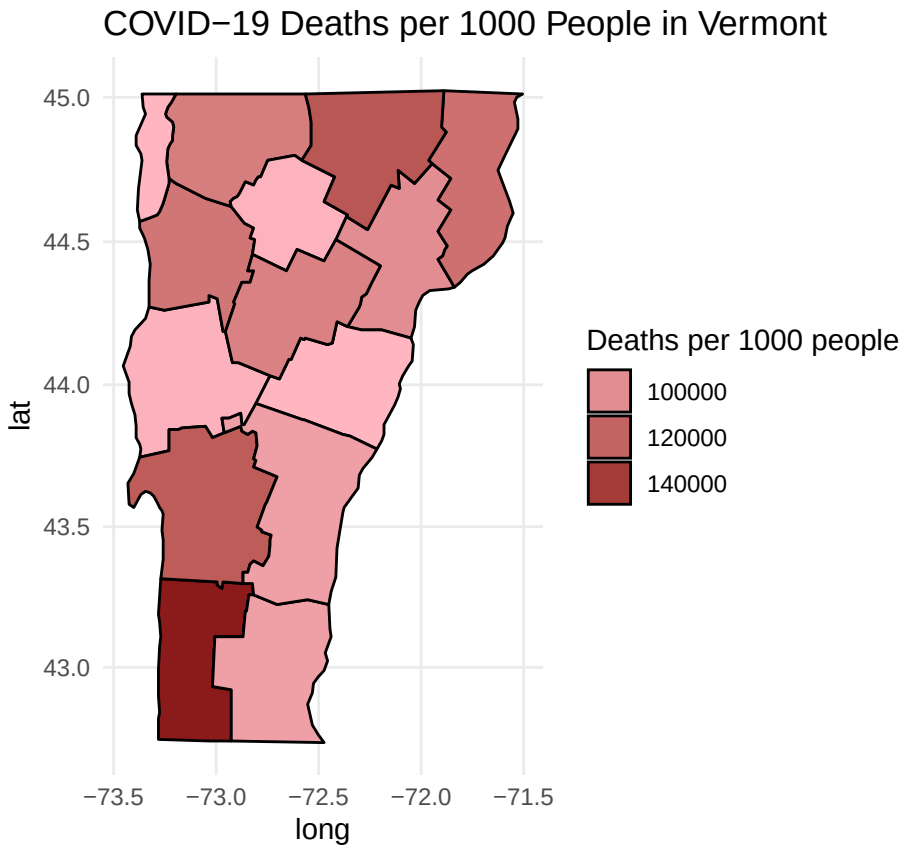
```
## `summarise()` has grouped output by 'Admin2'. You can override using the
## `.groups` argument.
```

```
vt_county_cases$Admin2 <- tolower(vt_county_cases$Admin2)

# merge county boundary data with case data
vt_county_map <- merge(vt_county_map, vt_county_cases, by.x = "subregion", by.y = "Admin2", all.x = TRU

# cases per thousand
vt_county_map$cases_per_thou <- vt_county_map$total_cases / vt_county_map$Population * 1000

vt_county_map$deaths_per_thou <- vt_county_map$total_cases / vt_county_map$Population * 1000
```

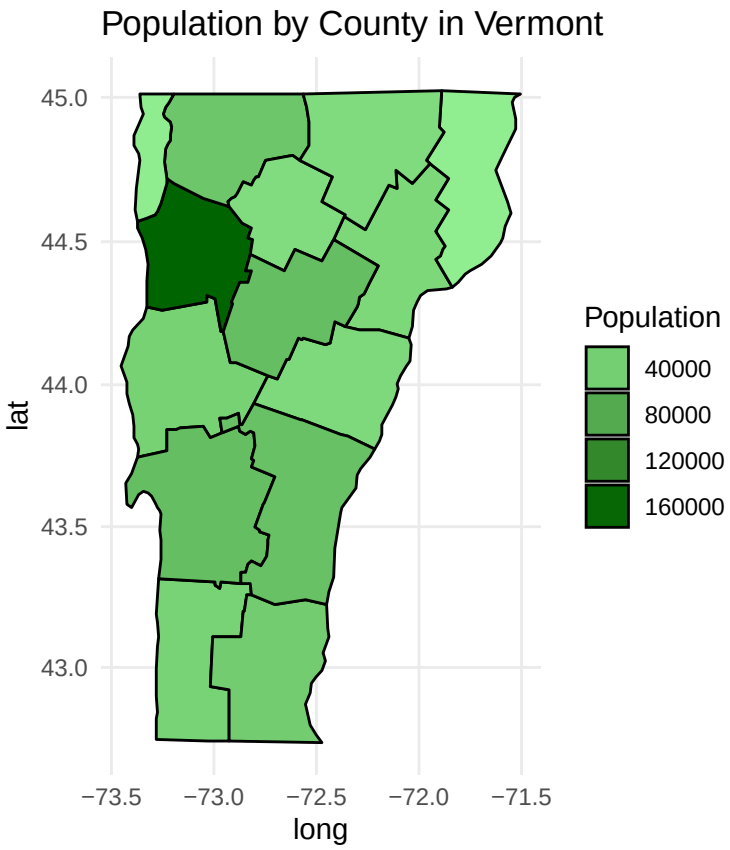
```
# FL heat map of cases per thou
ggplot(vt_county_map, aes(long, lat, group = group, fill = cases_per_thou)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Cases per 1000 people", low = "lightblue", high = "navyblue", guide = "leg
  labs(title = "COVID-19 Cases per 1000 People in Vermont") +
  theme_minimal()
```




```
# FL heat map of deaths per thou
ggplot(vt_county_map, aes(long, lat, group = group, fill = deaths_per_thou)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Deaths per 1000 people", low = "lightpink", high = "firebrick4", guide =
  labs(title = "COVID-19 Deaths per 1000 People in Vermont") +
  theme_minimal()
```




```
# CO heat map of population by county
ggplot(vt_county_map, aes(long, lat, group = group, fill = Population)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Population", low = "lightgreen", high = "darkgreen", guide = "legend") +
  labs(title = "Population by County in Vermont") +
  theme_minimal()
```



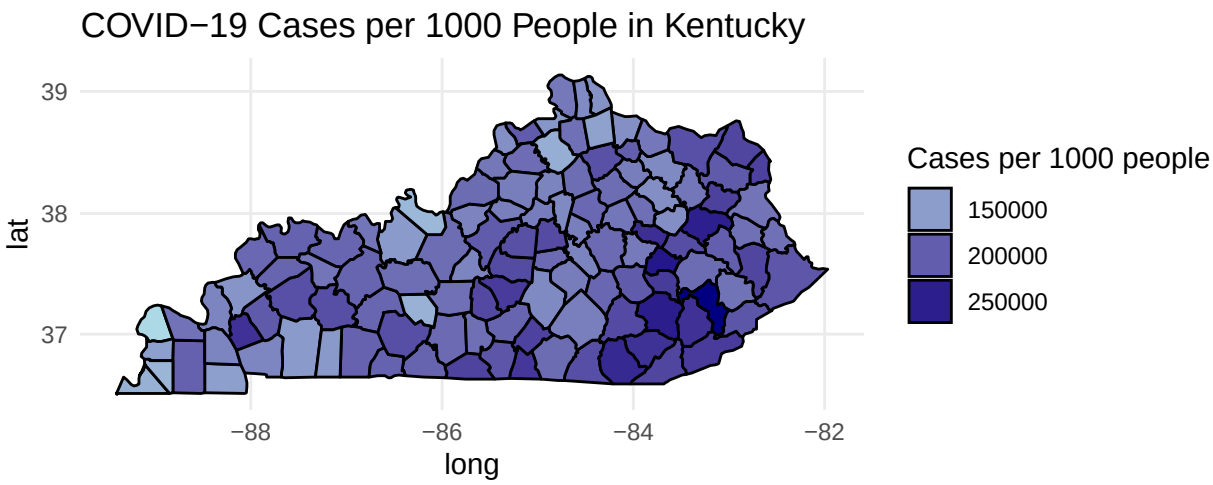
Kentucky

```
# Kentucky counties
ky_county_map <- map_data("county", region = "kentucky")

# filter data by county
kentucky_data <- us %>%
  filter(Province_State == "Kentucky")

# cases and deaths by county
ky_county_cases <- kentucky_data %>%
  group_by(Admin2, Population) %>%
  summarise(total_cases = sum(cases, na.rm = TRUE),
            total_deaths = sum(deaths, na.rm = TRUE))
```

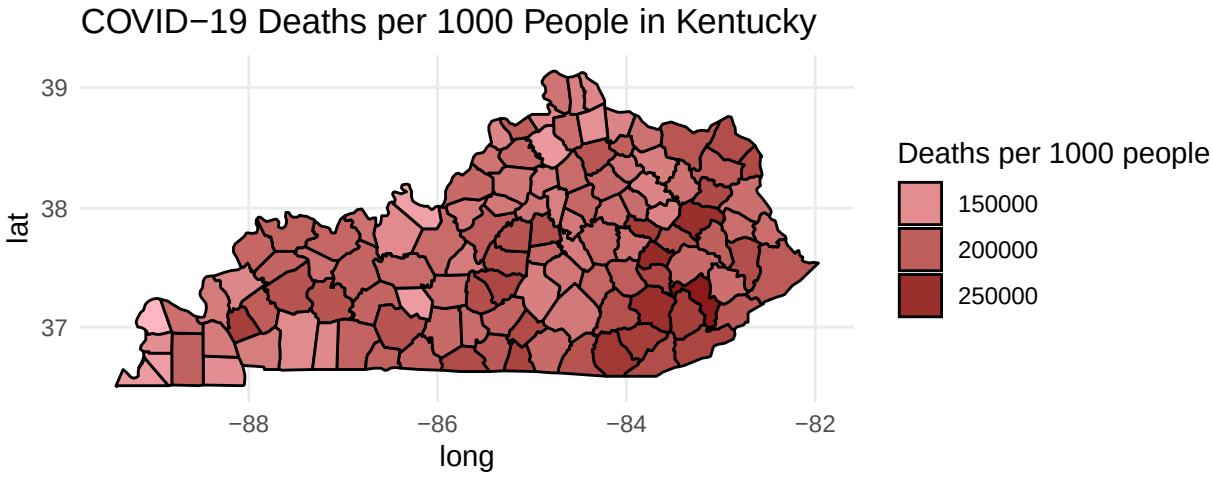
```
## `summarise()` has grouped output by 'Admin2'. You can override using the
## `.groups` argument.
```

```
ky_county_cases$Admin2 <- tolower(ky_county_cases$Admin2)

# merge county boundary data with case data
ky_county_map <- merge(ky_county_map, ky_county_cases, by.x = "subregion", by.y = "Admin2", all.x = TRU

# cases per thousand
ky_county_map$cases_per_thou <- ky_county_map$total_cases / ky_county_map$Population * 1000
```

```r
ky_county_map$deaths_per_thou <- ky_county_map$total_cases / ky_county_map$Population * 1000

# FL heat map of cases per thou
ggplot(ky_county_map, aes(long, lat, group = group, fill = cases_per_thou)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Cases per 1000 people", low = "lightblue", high = "navyblue", guide = "leg
  labs(title = "COVID-19 Cases per 1000 People in Kentucky") +
  theme_minimal()
```

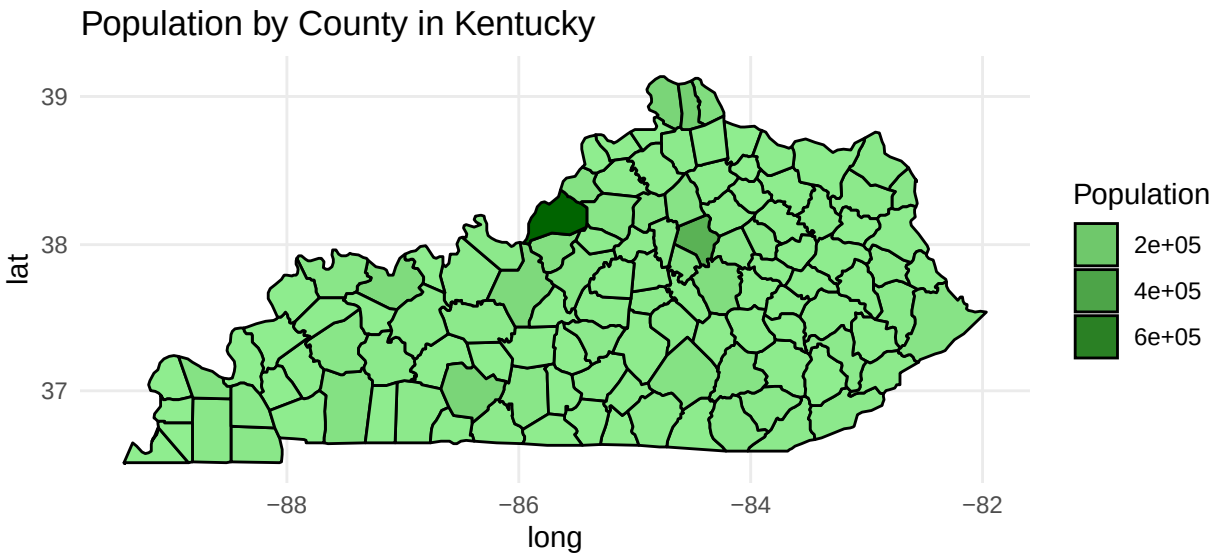



```r
# FL heat map of deaths per thou
ggplot(ky_county_map, aes(long, lat, group = group, fill = deaths_per_thou)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Deaths per 1000 people", low = "lightpink", high = "firebrick4", guide =
  labs(title = "COVID-19 Deaths per 1000 People in Kentucky") +
  theme_minimal()
```




```r
# CO heat map of population by county
ggplot(ky_county_map, aes(long, lat, group = group, fill = Population)) +
  geom_polygon(color = "black") +
  coord_map() +
  scale_fill_gradient(name = "Population", low = "lightgreen", high = "darkgreen", guide = "legend") +
```

```
  labs(title = "Population by County in Kentucky") +
  theme_minimal()
```



Statistical Models

Linear Models United States

Cases per thousand as the response variable with population as the lone predictor.

```
mod_us_pop_cases <- lm(cases_per_thou ~ population, data = us_state_totals)
summary(mod_us_pop_cases)
```

```
##
## Call:
## lm(formula = cases_per_thou ~ population, data = us_state_totals)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -156.496  -29.550    4.457   31.208  128.454
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.060e+02  8.765e+00  34.912   <2e-16 ***
## population  4.010e-07  9.426e-07   0.425    0.672
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50.44 on 54 degrees of freedom
## Multiple R-squared:  0.00334,    Adjusted R-squared:  -0.01512
## F-statistic: 0.181 on 1 and 54 DF,  p-value: 0.6722
```

Deaths per thousand as the response variable with population as the lone predictor.

```
mod_us_pop_deaths <- lm(deaths_per_thou ~ population, data = us_state_totals)
summary(mod_us_pop_deaths)
```

```
##
## Call:
## lm(formula = deaths_per_thou ~ population, data = us_state_totals)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3521 -0.6953  0.3244  0.7287  1.4675
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.962e+00  1.761e-01  16.821   <2e-16 ***
## population  2.850e-08  1.894e-08   1.505    0.138
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.013 on 54 degrees of freedom
## Multiple R-squared:  0.04026,    Adjusted R-squared:  0.02249
## F-statistic: 2.265 on 1 and 54 DF,  p-value: 0.1381
```

Cases per thousand as the response variable with latitude, longitude, and population as predictors.

```
mod_us_cases_long_lat <- lm(cases_per_thou ~ mean_lat + mean_long + population, data = us_map_data_all)
summary(mod_us_cases_long_lat)
```

```
##
## Call:
## lm(formula = cases_per_thou ~ mean_lat + mean_long + population,
##     data = us_map_data_all)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -90.003 -28.887   4.056  21.503 122.185
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.250e+02  6.228e+01   6.824 1.86e-08 ***
## mean_lat    -2.412e+00  1.340e+00  -1.800   0.0786 .
## mean_long    1.610e-01  4.067e-01   0.396   0.6941
## population  -5.163e-07  8.736e-07  -0.591   0.5575
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 41.84 on 45 degrees of freedom
## Multiple R-squared:  0.07473,    Adjusted R-squared:  0.01304
## F-statistic: 1.211 on 3 and 45 DF,  p-value: 0.3165
```

Deaths per thousand as the response variable with latitude, longitude, population, and csaes per thousand predictors.

```
mod_us_deaths_long_lat <- lm(deaths_per_thou ~ mean_lat + mean_long + population + cases_per_thou, data
summary(mod_us_deaths_long_lat)
```

```
##
```

```
## Call:
## lm(formula = deaths_per_thou ~ mean_lat + mean_long + population +
##     cases_per_thou, data = us_map_data_all)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.01819 -0.34324  0.03487  0.46580  1.33417
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.580e+00  1.362e+00   2.628 0.011773 *
## mean_lat       -6.815e-02  2.127e-02  -3.204 0.002525 **
## mean_long       8.386e-04  6.247e-03   0.134 0.893825
## population     -1.056e-08  1.345e-08  -0.785 0.436449
## cases_per_thou  8.397e-03  2.286e-03   3.673 0.000646 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6416 on 44 degrees of freedom
## Multiple R-squared:  0.4282, Adjusted R-squared:  0.3762
## F-statistic: 8.237 on 4 and 44 DF,  p-value: 4.758e-05
```

World Linear Models

Cases per thousand as the response variable with population as the lone predictor.

```
mod_world_pop_cases <- lm(cases_per_thou ~ population, data = world_country_totals)
summary(mod_world_pop_cases)
```

```
##
## Call:
## lm(formula = cases_per_thou ~ population, data = world_country_totals)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -174.51 -156.18  -72.00   97.86  518.93
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.769e+02  1.410e+01  12.553   <2e-16 ***
## population  -1.820e-07  1.252e-07  -1.453    0.148
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 188 on 192 degrees of freedom
## Multiple R-squared:  0.01088,    Adjusted R-squared:  0.00573
## F-statistic: 2.112 on 1 and 192 DF,  p-value: 0.1478
```

Deaths per thousand as the response variable with population as the lone predictor.

```
mod_world_pop_deaths <- lm(deaths_per_thou ~ population, data = world_country_totals)
summary(mod_world_pop_deaths)
```

```
##
```

```
## Call:
## lm(formula = deaths_per_thou ~ population, data = world_country_totals)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2571 -1.1078 -0.4923  0.7840  5.4173
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.257e+00  1.001e-01  12.563   <2e-16 ***
## population  -4.833e-10  8.890e-10  -0.544    0.587
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.334 on 192 degrees of freedom
## Multiple R-squared:  0.001537,   Adjusted R-squared:  -0.003663
## F-statistic: 0.2956 on 1 and 192 DF,  p-value: 0.5873
```

Conclusion and Examination of Bias

Here the relationship between population and the severity of the COVID-19 pandemic was examined. Interestingly, although close contact with others has been established as an important factor in the spread of COVID-19, population was not found to be a significant factor in cases or deaths in the United States or worldwide. This trend was visualized in the state maps, which showed the counties that were most heavily impacted by COVID-19 were not the most populated. Linear models that included only population as the predictor resulted in p-values for population well above any acceptable significance factor. The only linear model that performed reasonably well had deaths per thousand as the response variable and included the predictors for geospatial data. In this model the predictor for latitude had a significant negative relationship with deaths per thousand. Meaning as there is a relationship between decreasing latitude, in the United States that translates to moving more southernly, and an increase in deaths related to COVID-19.

The pandemic became a polarizing and politicized issue in the United States and in other areas of the world. Allcott et al. (2020) found that there were sharp differences in how Republicans and Democrats viewed the pandemic and these differences resulted in measurable behavioral changes such as social distancing practices. Hart et al. (2020) went on to find that some of these differences can be attributed to divergent information regarding the pandemic from right and left leaning media outlets and politicians. Therefore, one must be cautious of bias when analyzing and interpreting data related to COVID-19. For example, I could try to explain the geospatial trend seen in one of the models as being the result of political affiliation influencing behavior. Although this may be true, it would not be appropriate because these types of models do not determine causality.

The R-squared value for the most relevant model was 0.3762, indicating there is still a large amount of variability in the deaths per thousand response that is not explained by the model. The World Health Organization (2023) listed several other factors that influence the severity of COVID-19 infections, such as age and pre-existing conditions like heart disease and diabetes. Based on the analysis done in this research and expert domain knowledge, a data set with more relevant features is needed to build a more accurate model of COVID-19 cases and deaths.

References

Allcott H., Boxell L., Conway J. C., Gentzkow M., Thaler M., Yang D. Y. (2020). Polarization and public health: Partisan differences in social distancing during the coronavirus pandemic. National Bureau of Economic Research. https://doi.org/10.3386/w26946

Hart, P. S., Chinn, S., & Soroka, S. (2020). Politicization and polarization in COVID-19 news coverage. Science Communication, 42(5), 679–697. https://doi.org/10.1177/1075547020950735

World Health Organization: WHO. (2024, February 20). Coronavirus disease (COVID-19) pandemic. https://www.who.int/europe/emergencies/situations/covid-19

World Health Organization: WHO. (2023, August 9). Coronavirus disease (COVID-19). https://www.who.int/news-room/fact-sheets/detail/coronavirus-disease-(covid-19)

Session Information

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS 14.6.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] maps_3.4.2      lubridate_1.9.2 forcats_1.0.0   stringr_1.5.0
##  [5] dplyr_1.1.2     purrr_1.0.1     readr_2.1.4     tidyr_1.3.0
##  [9] tibble_3.2.1    ggplot2_3.4.2   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] highr_0.10        pillar_1.9.0      compiler_4.2.2    tools_4.2.2
##  [5] bit_4.0.5         digest_0.6.35     timechange_0.2.0  evaluate_0.23
##  [9] lifecycle_1.0.3   gtable_0.3.3      pkgconfig_2.0.3   rlang_1.1.1
## [13] cli_3.6.1         rstudioapi_0.14   mapproj_1.2.11    curl_5.2.1
## [17] parallel_4.2.2    yaml_2.3.7        xfun_0.39         fastmap_1.1.1
## [21] withr_2.5.0       knitr_1.42        generics_0.1.3    vctrs_0.6.5
## [25] hms_1.1.3         bit64_4.0.5       grid_4.2.2        tidyselect_1.2.0
## [29] glue_1.6.2        R6_2.5.1          fansi_1.0.4       vroom_1.6.3
## [33] rmarkdown_2.21    farver_2.1.1      tzdb_0.4.0        magrittr_2.0.3
## [37] scales_1.3.0      htmltools_0.5.8.1 colorspace_2.1-0  labeling_0.4.2
## [41] utf8_1.2.3        stringi_1.7.12    munsell_0.5.0     crayon_1.5.2
```