Covid-19-data Analysis

Daisaku Ikoma

2022-10-28

Objective of this analysis

I will analyze Covid-19 data. Data is read from the Johns Hopkins University Github site. I will visualize the data grouped by country and, for the US, by state. Then I am going to analyze correlations between Covid-19 cases and deaths. In addition, I will analyze autocorrelation and partial autocorrelation for new patients and try to fit them to a time-series model.

Clear memory

At first, I clear memories in advance.

```
rm(list=ls())
gc();gc()
```

```
##          used (Mb) gc trigger (Mb) max used (Mb)
## Ncells 472897 25.3    1029034   55   644200 34.5
## Vcells 858071  6.6    8388608   64  1635000 12.5
```

```
##          used (Mb) gc trigger (Mb) max used (Mb)
## Ncells 473292 25.3    1029034   55   644200 34.5
## Vcells 859004  6.6    8388608   64  1635000 12.5
```

Import libraries

I import libraries to use this analysis.

```
Sys.setenv(LANGUAGE="en")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
# If you haven't installed 'forecast' yet, please do so.
# >install.packages('forecast', dependencies = TRUE)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

Step 1 - Identify and import the data

I will start by reading in the data from the four main csv files.

```
# Get current Data in the four files
# They all begin the same way
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)
```

Let's read in the data and see what we have.

```
global_cases <- read_csv(urls[1])
global_deaths <- read_csv(urls[2])
US_cases <- read_csv(urls[3])
US_deaths <- read_csv(urls[4])
```

Step 2 - Tidy and Transform Data

After looking at global_cases and global_deaths, I would like to tidy those datasets and put each variable (date, cases, deaths) in their own colum. Also, I don't need Lat and Long for the analysis I am planning, so I will get rid of those and rename Region and State to be more R friendly.

```
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))

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

```
## Joining, by = c("Province/State", "Country/Region", "date")
```

```
global <- global %>% filter(cases > 0)

summary(global)
```

```
##  Province_state     Country_Region          date                cases
##  Length:268656      Length:268656      Min.   :2020-01-22   Min.   :       1
##  Class :character   Class :character   1st Qu.:2020-11-07   1st Qu.:     991
##  Mode  :character   Mode  :character   Median :2021-07-09   Median :   15550
##                                        Mean   :2021-07-05   Mean   :  853931
##                                        3rd Qu.:2022-03-06   3rd Qu.:  224610
##                                        Max.   :2022-10-27   Max.   :97409772
##      deaths
##  Min.   :      0
##  1st Qu.:      6
##  Median :    174
##  Mean   :  13119
##  3rd Qu.:   3138
##  Max.   :1070055
```

```
US_cases <- US_cases %>%
  pivot_longer(cols = -c(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_))

US <- US_cases %>%
  full_join(US_deaths)
```

```
## Joining, by = c("Admin2", "Province_State", "Country_Region", "Combined_Key",
## "date")
```

```
summary(US)
```

```
##     Admin2          Province_State     Country_Region     Combined_Key
##  Length:3375420     Length:3375420     Length:3375420     Length:3375420
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##       date                cases           Population           deaths
##  Min.   :2020-01-22   Min.   :  -3073   Min.   :       0   Min.   :  -82.0
##  1st Qu.:2020-09-30   1st Qu.:    227   1st Qu.:    9917   1st Qu.:    2.0
##  Median :2021-06-09   Median :   1846   Median :   24892   Median :   31.0
##  Mean   :2021-06-09   Mean   :  11974   Mean   :   99604   Mean   :  168.3
##  3rd Qu.:2022-02-17   3rd Qu.:   6809   3rd Qu.:   64979   3rd Qu.:  107.0
##  Max.   :2022-10-27   Max.   :3484615   Max.   :10039107   Max.   :33945.0
```

```
global <- global %>%
  unite("Combined_Key",
        c(Province_state, Country_Region),
        sep = ", ",
        na.rm = TRUE,
        remove = FALSE)

summary(global)
```

```
##  Combined_Key       Province_state     Country_Region          date
##  Length:268656      Length:268656      Length:268656      Min.   :2020-01-22
##  Class :character   Class :character   Class :character   1st Qu.:2020-11-07
##  Mode  :character   Mode  :character   Mode  :character   Median :2021-07-09
##                                                           Mean   :2021-07-05
##                                                           3rd Qu.:2022-03-06
##                                                           Max.   :2022-10-27
##      cases              deaths
##  Min.   :       1   Min.   :     0
##  1st Qu.:     991   1st Qu.:     6
##  Median :   15550   Median :   174
##  Mean   :  853931   Mean   : 13119
##  3rd Qu.:  224610   3rd Qu.:  3138
##  Max.   :97409772   Max.   :1070055
```

```
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.
```

Step 3 - Visualizing Data

I will visualize the data. First, draw a time series graph of cases and deaths across the US. Then plot the time series data for each state (here, New York and Colorado). I also shows a bar chart of total cases and deaths by state.

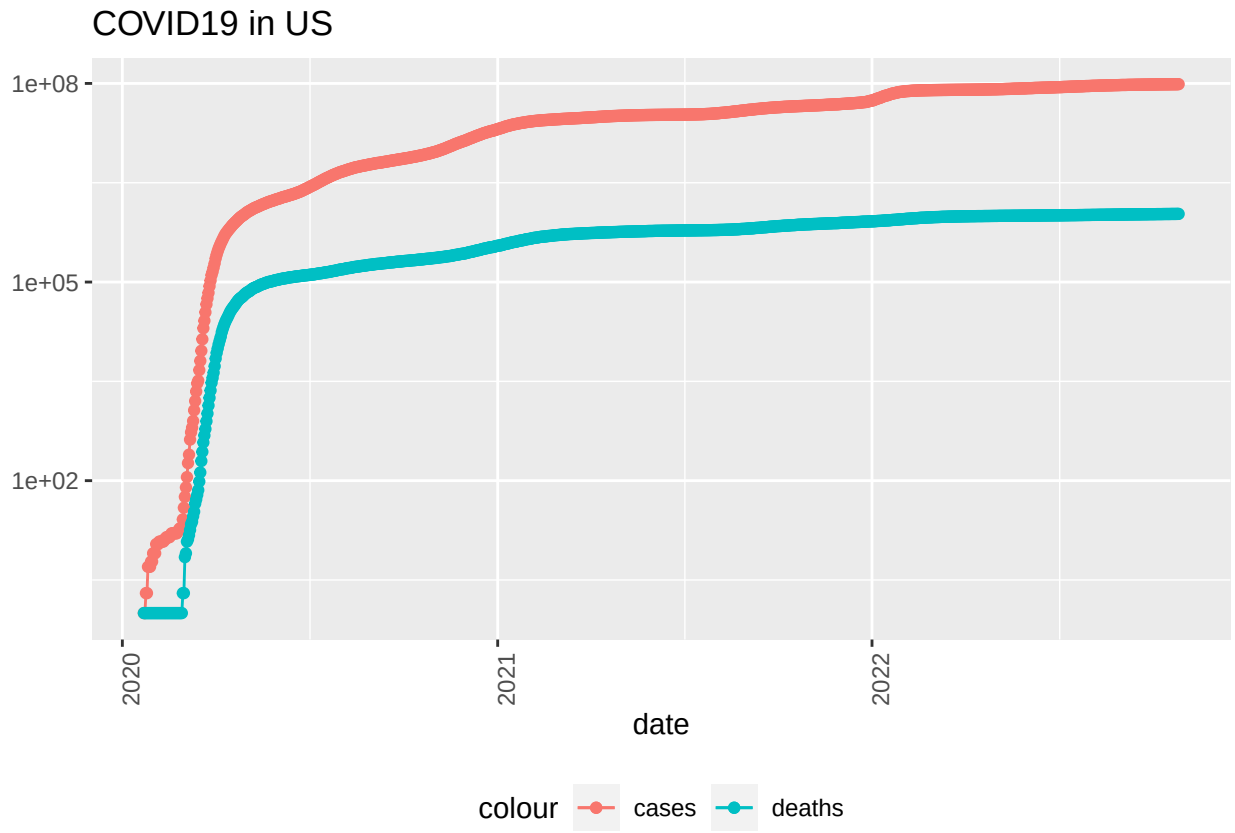
```
US_by_state <- US %>%
  group_by(Province_State, Country_Region, date) %>%
  summarise(cases = sum(cases), deaths = sum(deaths),
            Population = sum(Population)) %>%
  mutate(deaths_per_mill = deaths * 1000000 / Population) %>%
  select(Province_State, Country_Region, date,
         cases, deaths, deaths_per_mill, Population) %>%
  ungroup()
```

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

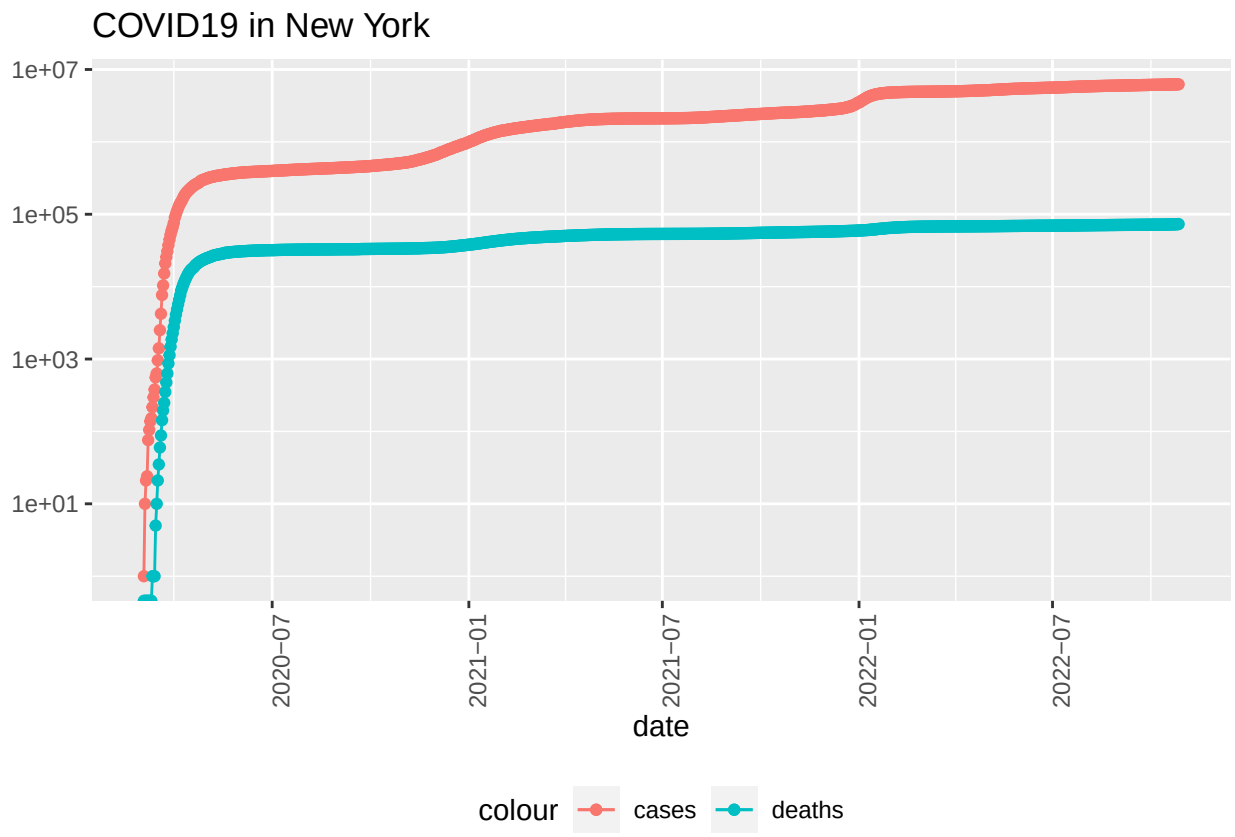
```
US_totals <- US_by_state %>%
  group_by(Country_Region, date) %>%
  summarise(cases = sum(cases), deaths = sum(deaths),
            Population = sum(Population)) %>%
  mutate(deaths_by_mill = deaths * 1000000 / Population) %>%
  ungroup()
```

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

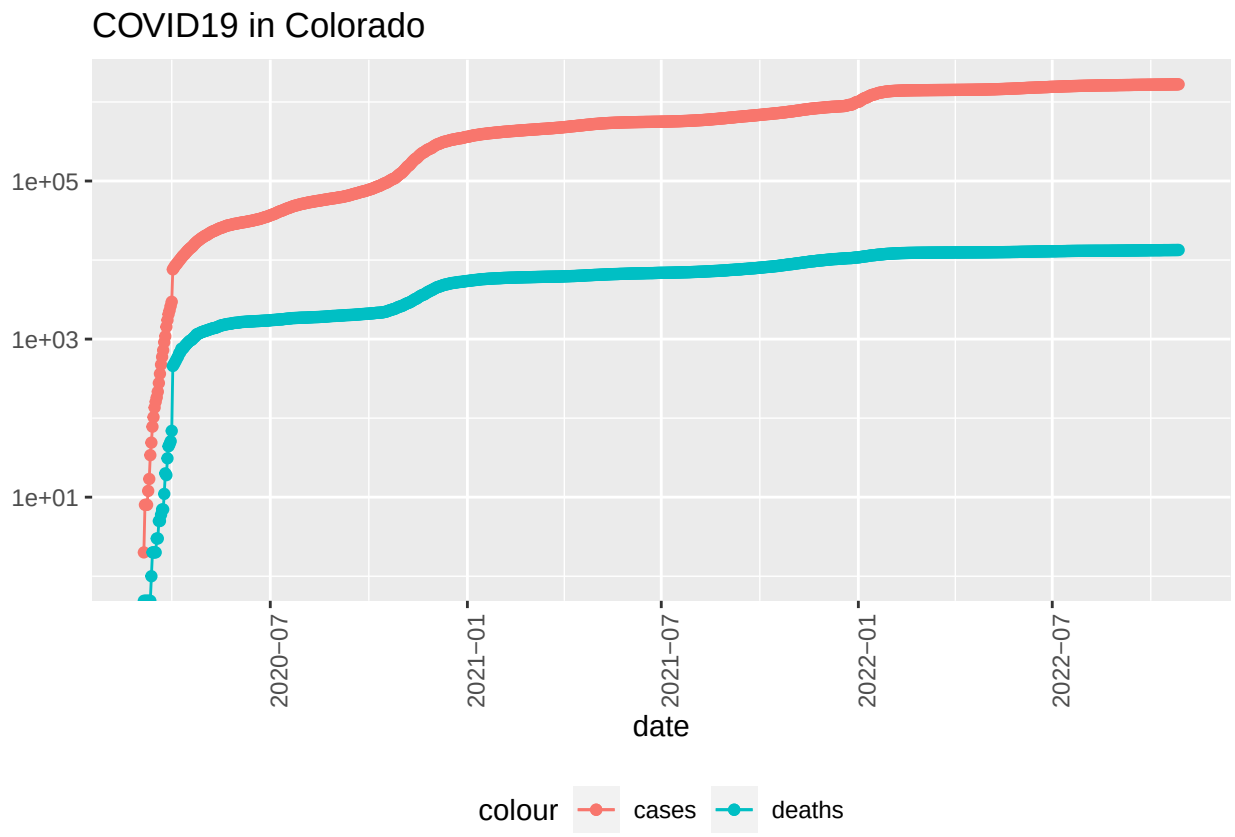
```
US_totals %>%
  filter(cases > 0) %>%
  ggplot(aes(x = date, y = cases)) +
  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +
  geom_line(aes(y = deaths, color = "deaths")) +
  geom_point(aes(y = deaths, color = "deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",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



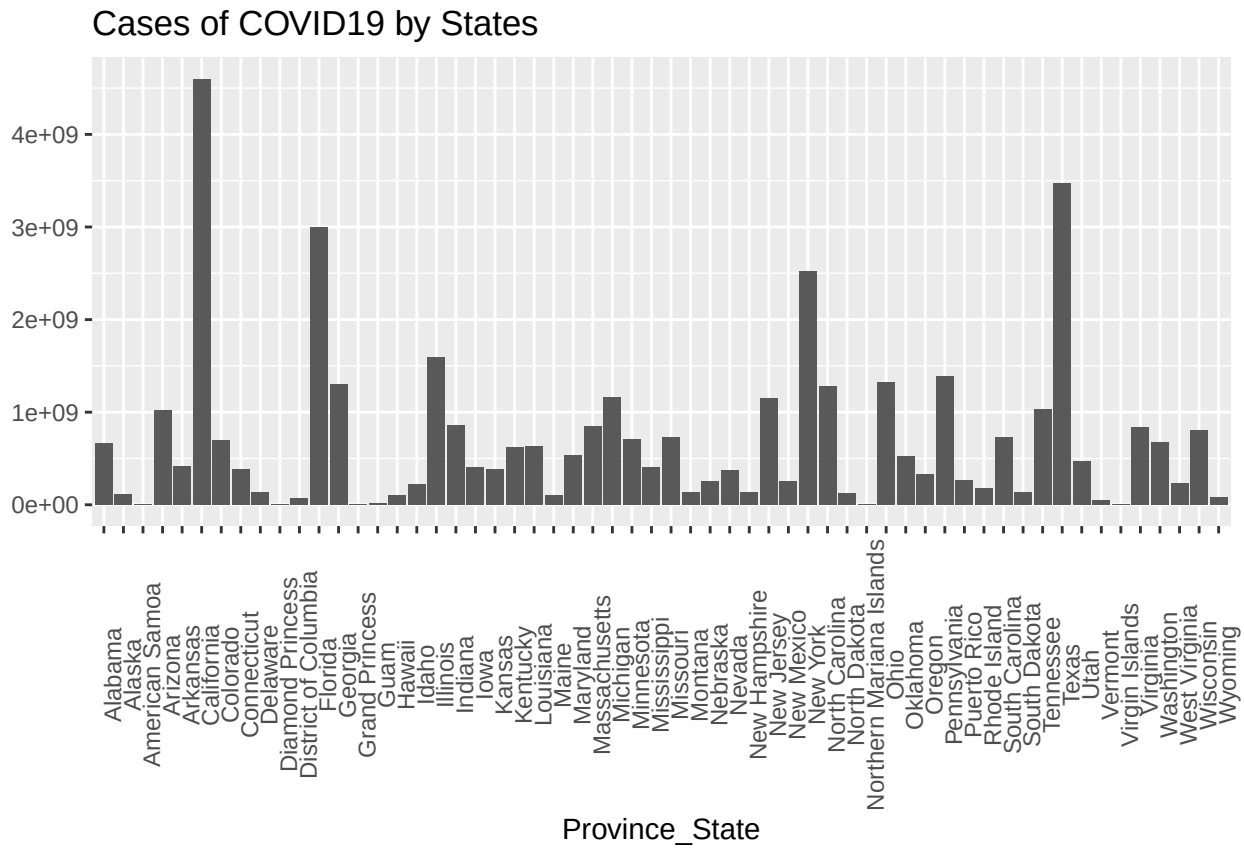

```
state <- "New York"
US_by_state %>%
  filter(Province_State == state) %>%
  filter(cases > 0) %>%
  ggplot(aes(x = date, y = cases)) +
  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +
  geom_line(aes(y = deaths, color = "deaths")) +
  geom_point(aes(y = deaths, color = "deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("COVID19 in ", state), y = NULL)
```



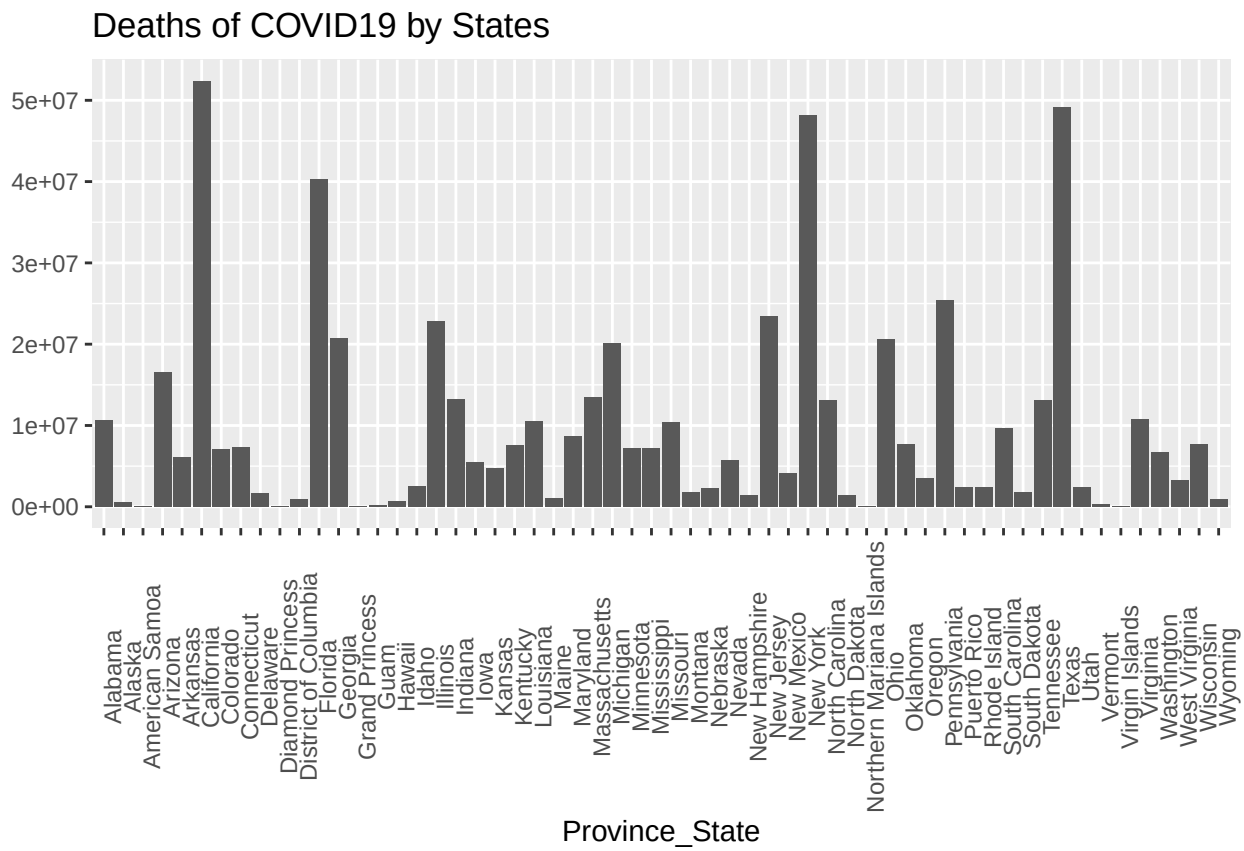

```
state <- "Colorado"
US_by_state %>%
  filter(Province_State == state) %>%
  filter(cases > 0) %>%
  ggplot(aes(x = date, y = cases)) +
  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +
  geom_line(aes(y = deaths, color = "deaths")) +
  geom_point(aes(y = deaths, color = "deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("COVID19 in ", state), y = NULL)
```




```
g_cases <- ggplot(US_by_state, aes(x = Province_State, y = cases)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(title = "Cases of COVID19 by States", y = NULL)
plot(g_cases)
```




```
g_deaths <- ggplot(US_by_state, aes(x = Province_State, y = deaths)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(title = "Deaths of COVID19 by States", y = NULL)
plot(g_deaths)
```



Step 4 - Analyzing Data

I will analyze new cases and deaths for the entire US and New York.

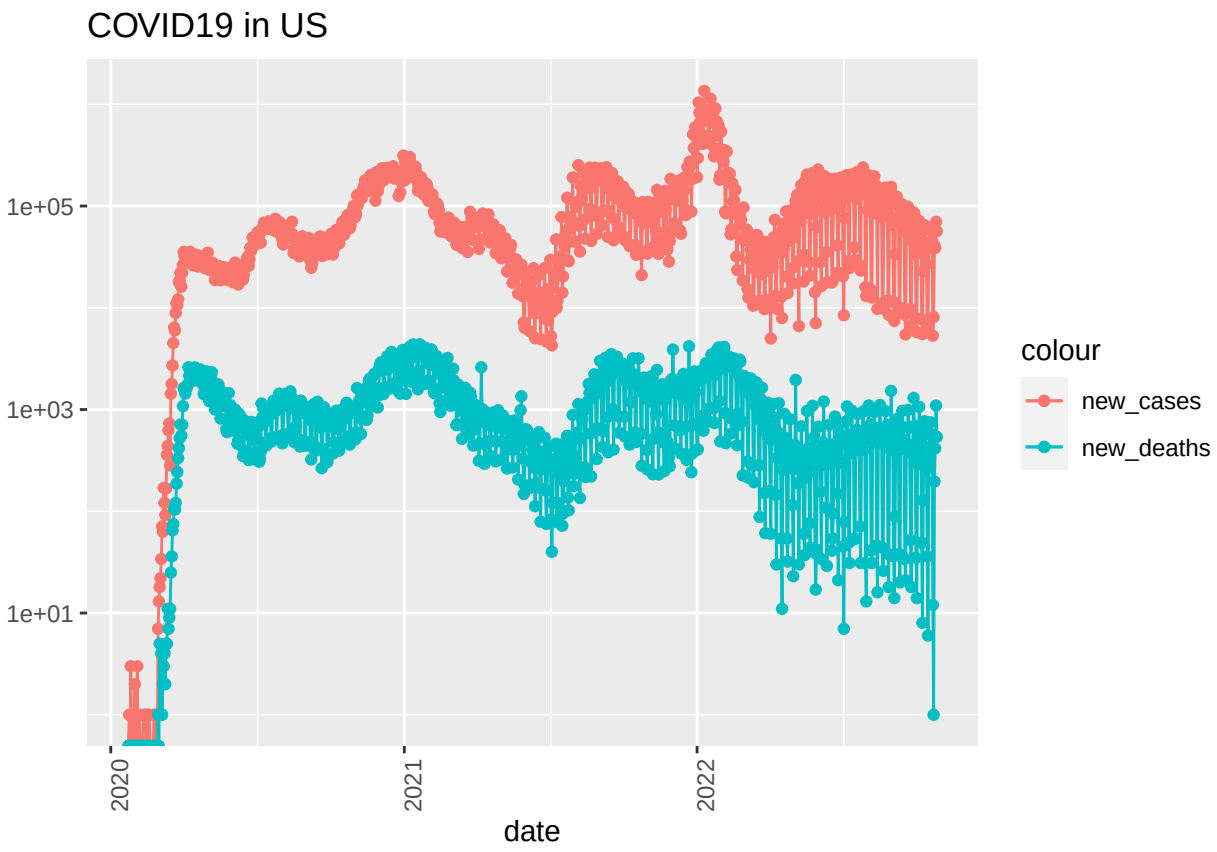
```
uid <- read_csv(uid_lookup_url) %>%
  select(-c(Lat, Long_, Combined_Key, code3, iso2, iso3, Admin2))
```

```
## 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.
```

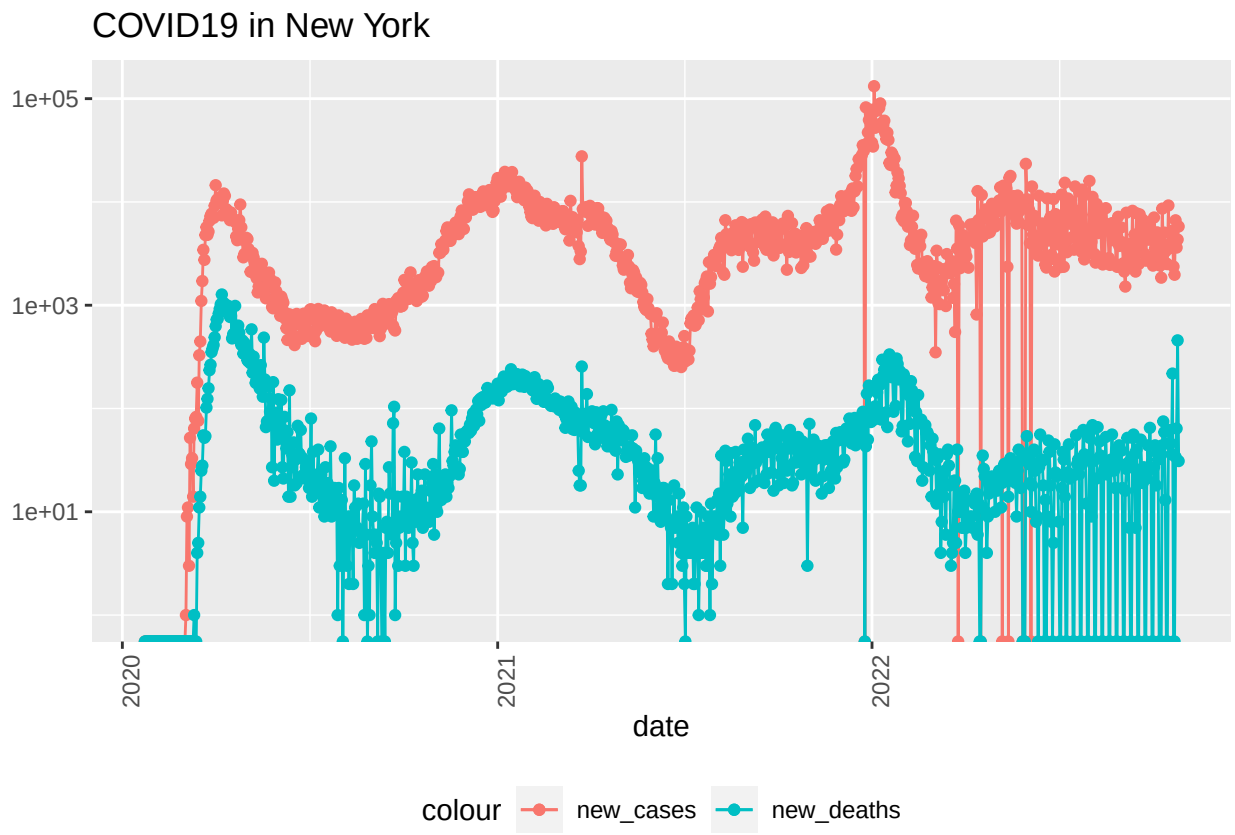
```
US_by_state <- US_by_state %>%
  mutate(new_cases = cases - lag(cases),
         new_deaths = deaths - lag(deaths))
US_totals <- US_totals %>%
  mutate(new_cases = cases - lag(cases),
         new_deaths = deaths - lag(deaths))

US_totals %>%
```

```
  ggplot(aes(x = date, y = new_cases)) +
  geom_line(aes(color = "new_cases")) +
  geom_point(aes(color = "new_cases")) +
  geom_line(aes(y = new_deaths, color = "new_deaths")) +
  geom_point(aes(y = new_deaths, color = "new_deaths")) +
  scale_y_log10() +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(title = "COVID19 in US", y = NULL)
```




```
state <- "New York"
US_by_state %>%
  filter(Province_State == state) %>%
  ggplot(aes(x = date, y = new_cases)) +
  geom_line(aes(color = "new_cases")) +
  geom_point(aes(color = "new_cases")) +
  geom_line(aes(y = new_deaths, color = "new_deaths")) +
  geom_point(aes(y = new_deaths, color = "new_deaths")) +
  scale_y_log10() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = str_c("COVID19 in ", state), y = NULL)
```




```
US_state_totals <- US_by_state %>%
  group_by(Province_State) %>%
  summarise(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)

US_state_totals %>%
  slice_min(deaths_per_thou, n = 10) %>%
  select(deaths_per_thou, cases_per_thou, everything())
```

```
## # A tibble: 10 x 6
##    deaths_per_thou cases_per_thou Province_State           deaths  cases Popul~1
##              <dbl>          <dbl> <chr>                     <dbl>  <dbl>   <dbl>
##  1           0.611           148. American Samoa               34 8.26e3   55641
##  2           0.725           239. Northern Mariana Islands     40 1.32e4   55144
##  3           1.16            217. Virgin Islands              124 2.33e4  107268
##  4           1.19            232. Vermont                     740 1.45e5  623989
##  5           1.20            256. Hawaii                     1704 3.62e5 1415872
##  6           1.40            261. Puerto Rico                5248 9.79e5 3754939
##  7           1.58            326. Utah                       5056 1.05e6 3205958
##  8           1.91            405. Alaska                     1413 3.00e5  740995
##  9           1.91            241. Washington                14550 1.84e6 7614893
## 10           1.97            221. Maine                      2642 2.97e5 1344212
## # ... with abbreviated variable name 1: Population
```

```
US_state_totals %>%
  slice_max(deaths_per_thou, n = 10) %>%
  select(deaths_per_thou, cases_per_thou, everything())
```

```
## # A tibble: 10 x 6
##    deaths_per_thou cases_per_thou Province_State deaths   cases Population
##              <dbl>          <dbl> <chr>           <dbl>   <dbl>      <dbl>
##  1            4.36           314. Mississippi     12968  933065    2976149
##  2            4.33           314. Arizona         31548 2287886    7278717
##  3            4.32           305. Oklahoma        17100 1208316    3956971
##  4            4.19           313. Alabama         20558 1534287    4903185
##  5            4.19           339. West Virginia    7513  607087    1792147
##  6            4.13           318. Arkansas        12462  959014    3017804
##  7            4.12           299. New Mexico       8633  626714    2096829
##  8            4.11           345. Tennessee       28074 2357243    6829174
##  9            3.93           289. Michigan        39250 2886176    9986857
## 10            3.93           314. New Jersey      34889 2789256    8882190
```

Step 5 - Modeling Data

I will do modeling. First, we perform a correlation analysis between cases and deaths. I also fit the correlation with a linear model. Next, I analyze the autocorrelations and partial autocorrelations of the time series data of new-affected individuals and fit them with a time series model (ARIMA model).

```
mod <- lm(deaths_per_thou ~ cases_per_thou, data = US_state_totals)
summary(mod)
```

```
##
## Call:
## lm(formula = deaths_per_thou ~ cases_per_thou, data = US_state_totals)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3470 -0.6060  0.1230  0.6736  1.1851
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.285191   0.716159  -0.398    0.692
## cases_per_thou  0.011205   0.002431   4.609 2.52e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8518 on 54 degrees of freedom
## Multiple R-squared:  0.2823, Adjusted R-squared:  0.269
## F-statistic: 21.24 on 1 and 54 DF,  p-value: 2.518e-05
```

```
US_state_totals %>% slice_min(cases_per_thou)
```

```
## # A tibble: 1 x 6
##   Province_State deaths cases Population cases_per_thou deaths_per_thou
##   <chr>           <dbl> <dbl>      <dbl>          <dbl>           <dbl>
## 1 American Samoa     34  8257      55641           148.           0.611
```

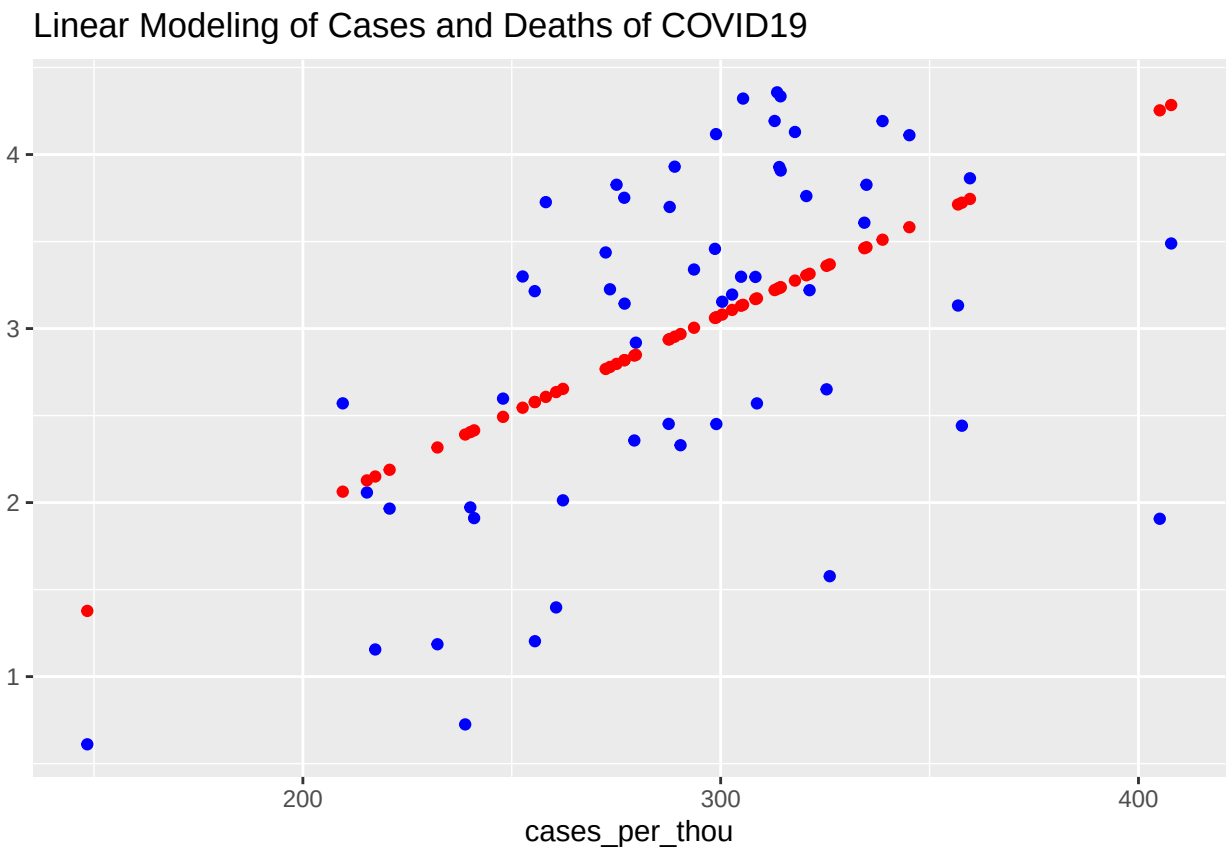
```
US_state_totals %>% slice_max(cases_per_thou)
```

```
## # A tibble: 1 x 6
##   Province_State deaths  cases Population cases_per_thou deaths_per_thou
##   <chr>           <dbl>  <dbl>      <dbl>          <dbl>           <dbl>
## 1 Rhode Island     3696 432042    1059361           408.            3.49
```

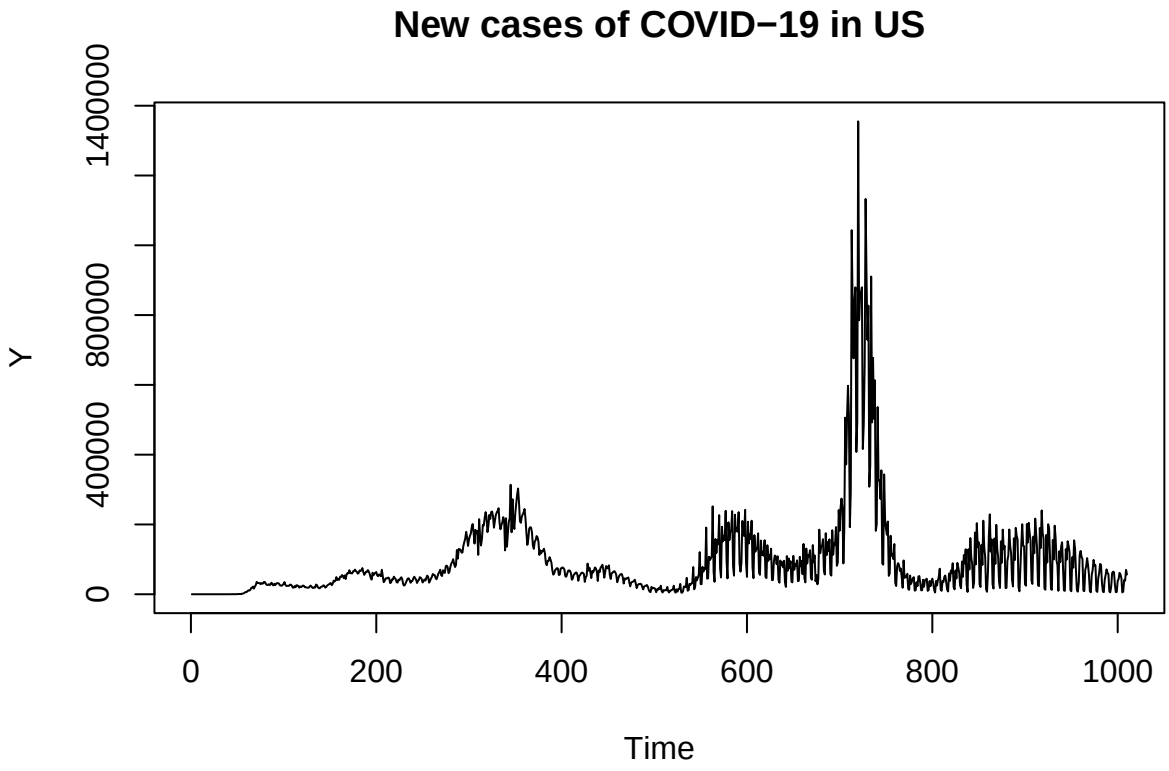
```
x_grid <- seq(1, 451)
new_df <- tibble(cases_per_thou = x_grid)
US_state_totals %>% mutate(pred = predict(mod))
```

```
## # A tibble: 56 x 7
##    Province_State       deaths    cases Population cases_per_thou deaths~1  pred
##    <chr>                 <dbl>    <dbl>      <dbl>          <dbl>    <dbl> <dbl>
##  1 Alabama               20558  1534287    4903185           313.    4.19   3.22
##  2 Alaska                 1413   300177     740995           405.    1.91   4.25
##  3 American Samoa           34     8257      55641           148.    0.611  1.38
##  4 Arizona               31548  2287886    7278717           314.    4.33   3.24
##  5 Arkansas              12462   959014    3017804           318.    4.13   3.28
##  6 California            96886 11362031   39512223           288.    2.45   2.94
##  7 Colorado              13415  1672312    5758736           290.    2.33   2.97
##  8 Connecticut           11462   910919    3565287           255.    3.21   2.58
##  9 Delaware               3136   312850     973764           321.    3.22   3.31
## 10 District of Columbia   1392   169436     705749           240.    1.97   2.40
## # ... with 46 more rows, and abbreviated variable name 1: deaths_per_thou
```

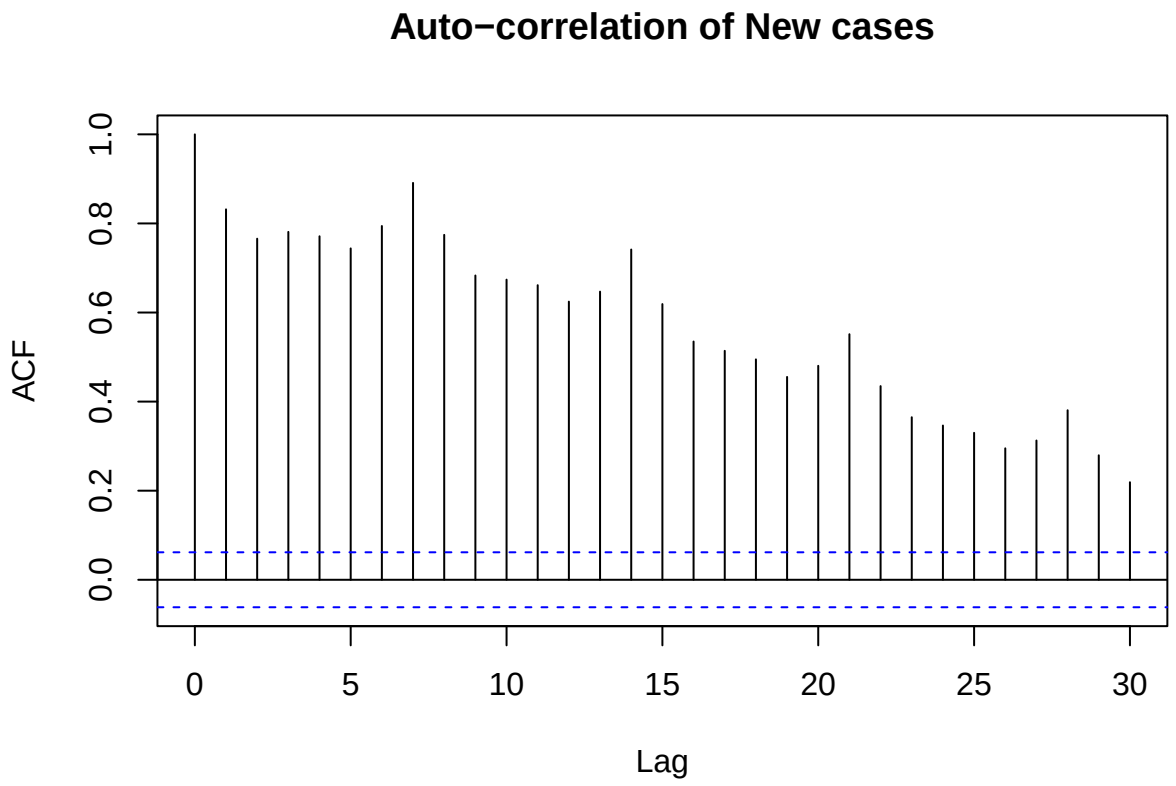
```
US_tot_w_pred <- US_state_totals %>% mutate(pred = predict(mod))
US_tot_w_pred %>% ggplot() +
  geom_point(aes(x = cases_per_thou, y = deaths_per_thou), color = "blue") +
  geom_point(aes(x = cases_per_thou, y = pred), color = "red") +
  labs(title = "Linear Modeling of Cases and Deaths of COVID19", y = NULL)
```



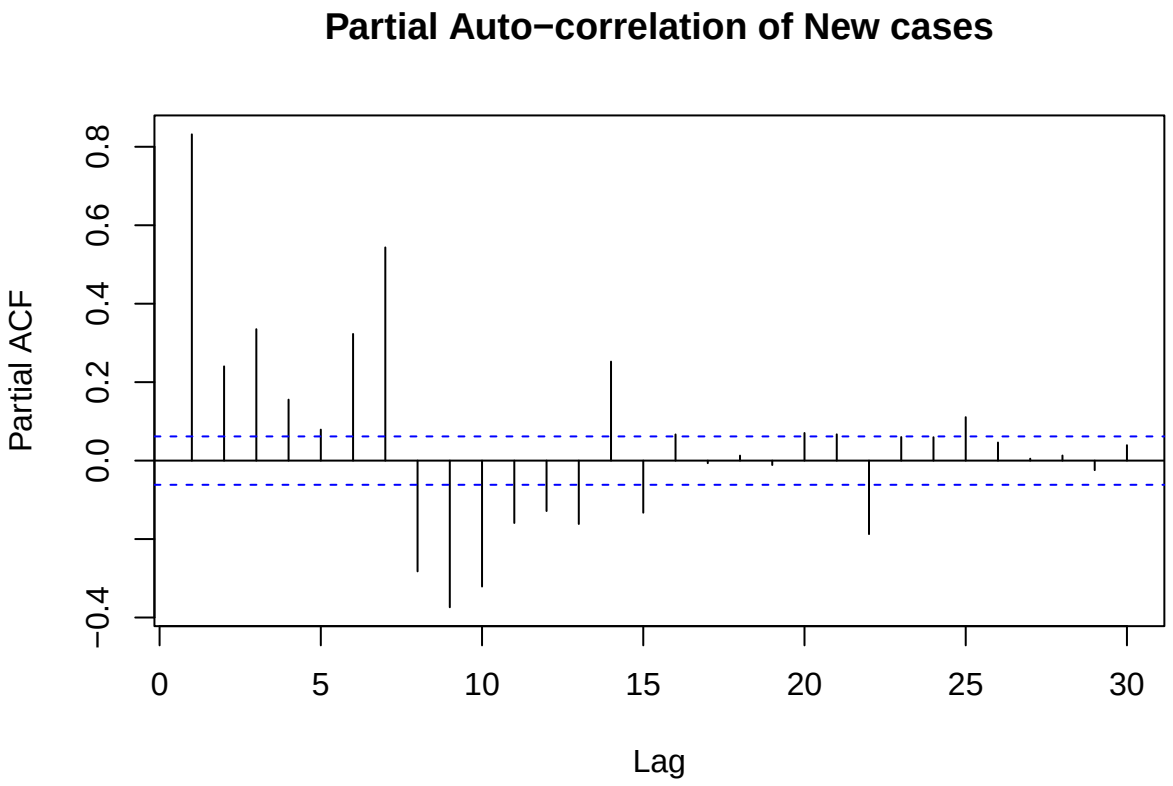

```
Y <- US_totals$new_cases
Y[is.na(Y)] <- 0
ts.plot(Y, main = "New cases of COVID-19 in US")
```




```
acf(Y, main = "Auto-correlation of New cases")
```




```
pacf(Y, main = "Partial Auto-correlation of New cases")
```




```
auto.arima(Y, ic="aic", stepwise=T, trace=T)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,1,2) with drift         : 24991.5
##  ARIMA(0,1,0) with drift         : 25550.64
##  ARIMA(1,1,0) with drift         : 25455.35
##  ARIMA(0,1,1) with drift         : 25254.16
##  ARIMA(0,1,0)                    : 25548.64
##  ARIMA(1,1,2) with drift         : 25212.4
##  ARIMA(2,1,1) with drift         : 25204.45
##  ARIMA(3,1,2) with drift         : Inf
##  ARIMA(2,1,3) with drift         : 25043.07
##  ARIMA(1,1,1) with drift         : 25240.45
##  ARIMA(1,1,3) with drift         : 25043.83
##  ARIMA(3,1,1) with drift         : 25206.84
##  ARIMA(3,1,3) with drift         : 24994.04
##  ARIMA(2,1,2)                    : 24989.5
##  ARIMA(1,1,2)                    : 25210.41
##  ARIMA(2,1,1)                    : 25202.45
##  ARIMA(3,1,2)                    : Inf
##  ARIMA(2,1,3)                    : 25041.08
##  ARIMA(1,1,1)                    : 25238.46
##  ARIMA(1,1,3)                    : 25041.83
```

```
##  ARIMA(3,1,1)                    : 25204.85
##  ARIMA(3,1,3)                    : 24992.04
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(2,1,2)                    : 25013.06
##
##  Best model: ARIMA(2,1,2)

## Series: Y
## ARIMA(2,1,2)
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       0.9311  -0.5715  -1.6357  0.9001
## s.e.  0.0290   0.0335   0.0129  0.0161
##
## sigma^2 = 3.387e+09:  log likelihood = -12501.53
## AIC=25013.06   AICc=25013.12   BIC=25037.64
```

Conclusions

- US COVID-19 cases are increasing exponentially, and so are the deaths. Since the middle of 2020, it has been increasing continuously, although not exponentially.
- I looked at the cases in New York and Colorado, the trend is similar across the US.
- By state, California, Texas, New York, and Florida have the most cases and deaths.
- Focusing on new cases and deaths, both will level off from the middle of 2020.
- There is a loose positive correlation between the cases and the deaths.
- From ACF, the order of the MR model is likely to be more than 30 days, and the order of the AR model is likely to be more than half a month. However, the best ARIMA model was (2, 1, 2).

Bias include the distribution of PCR test kits. In developed countries such as the US, it is possible to receive sufficient PCR tests and measure the number of infected people, but in developing countries, there is a possibility that accurate numbers cannot be obtained due to the lack of PCR test kits. Also, within the US, it is conceivable that the status of COVID-19 testing will change as a function of income. A certain number of low-income people are likely to be sick or die without being tested for COVID-19.

Therefore, for a more objective analysis, it is necessary to correct the bias while looking at trends in the number of deaths other than COVID-19 in the future.

```
sessionInfo()
```

```
## R version 4.2.1 (2022-06-23 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 22621)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Japanese_Japan.utf8  LC_CTYPE=Japanese_Japan.utf8
## [3] LC_MONETARY=Japanese_Japan.utf8 LC_NUMERIC=C
## [5] LC_TIME=Japanese_Japan.utf8
##
```

```
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] forecast_8.18   lubridate_1.8.0 forcats_0.5.2   stringr_1.4.1
##  [5] dplyr_1.0.9     purrr_0.3.4     readr_2.1.2     tidyr_1.2.0
##  [9] tibble_3.1.8    ggplot2_3.3.6   tidyverse_1.3.2
##
## loaded via a namespace (and not attached):
##  [1] tseries_0.10-52     httr_1.4.4          bit64_4.0.5
##  [4] vroom_1.5.7         jsonlite_1.8.0      modelr_0.1.9
##  [7] assertthat_0.2.1    TTR_0.24.3          highr_0.9
## [10] googlesheets4_1.0.1 cellranger_1.1.0    yaml_2.3.5
## [13] pillar_1.8.1        backports_1.4.1     lattice_0.20-45
## [16] glue_1.6.2          quadprog_1.5-8      digest_0.6.29
## [19] rvest_1.0.3         colorspace_2.0-3    htmltools_0.5.3
## [22] timeDate_4021.106   pkgconfig_2.0.3     broom_1.0.1
## [25] haven_2.5.1         scales_1.2.1        tzdb_0.3.0
## [28] googledrive_2.0.0   farver_2.1.1        generics_0.1.3
## [31] ellipsis_0.3.2      withr_2.5.0         urca_1.3-3
## [34] nnet_7.3-17         cli_3.3.0           quantmod_0.4.20
## [37] magrittr_2.0.3      crayon_1.5.1        readxl_1.4.1
## [40] evaluate_0.16       fs_1.5.2            fansi_1.0.3
## [43] nlme_3.1-157        xts_0.12.1          xml2_1.3.3
## [46] tools_4.2.1         hms_1.1.2           gargle_1.2.0
## [49] lifecycle_1.0.1     munsell_0.5.0       reprex_2.0.2
## [52] compiler_4.2.1      rlang_1.0.4         grid_4.2.1
## [55] rstudioapi_0.14     labeling_0.4.2      rmarkdown_2.16
## [58] gtable_0.3.0        fracdiff_1.5-1      DBI_1.1.3
## [61] curl_4.3.2          R6_2.5.1            zoo_1.8-11
## [64] knitr_1.40          bit_4.0.4           fastmap_1.1.0
## [67] utf8_1.2.2          stringi_1.7.8       parallel_4.2.1
## [70] Rcpp_1.0.9          vctrs_0.4.1         dbplyr_2.2.1
## [73] tidyselect_1.1.2    xfun_0.32           lmtest_0.9-40
```