COVID19 Data from John Hopkins

Antony Sequeira

2/9/2022

Overview

For fulfillment of DTSA-5301 finals *COVID19 dataset from the Johns Hopkins github site* part of the assignment.

Following script installs the required libraries in Mac OS

This section can be copied to a file or input into R console.
You could also download it from my repo at https://github.com/asequeir-edu-2022/dtsa5301final

```
#!/usr/bin/env Rscript
r = getOption("repos")
r["CRAN"] = "http://cran.us.r-project.org"
options(repos = r)

print("Installing R libraries")
install.packages("chron")
install.packages("tidyverse")
install.packages("tinytex")

tinytex::install_tinytex()
```

Load libraries

```
knitr::opts_chunk$set(echo = TRUE, results = "hide", warning = FALSE)

library(tidyverse, warn.conflicts=F, quietly=T)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate, warn.conflicts=F, quietly=T)
library(chron, warn.conflicts=F, quietly=T)
```

```
knitr::opts_chunk$set(echo = TRUE, results = "echo")
```

Data source

The main data source is the github repository at https://github.com/CSSEGISandData/COVID-19

We will use the data from the folder at
https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series

Two of the time series files are for the US confirmed cases and deaths, reported at the county level.
They are `time_series_covid19_confirmed_US.csv` and `time_series_covid19_deaths_US.csv` respectively.

The report will focus on analyzing the data for United States only.

```
covid19_confirmed_url = "https://github.com/CSSEGISandData/COVID-19/raw/master/csse_covid_19_data/csse_
covid19_deaths_url = "https://github.com/CSSEGISandData/COVID-19/raw/master/csse_covid_19_data/csse_cov

covid19_confirmed_raw <- read_csv(covid19_confirmed_url)
covid19_deaths_raw <- read_csv(covid19_deaths_url)
```

Data description and cleaning

The `Province_State` column contains the name of the US State and the `Admin2` contains the County name for which daily data is collected.
These are the main location identifiers in this data.
The date columns are in long form and contain the cases and death counts.

We clean up the data mainly through the following three operations:
- variables to factor for appropriate columns
- date types from strings
- remove unneeded columns
- pivot from long date columns

Pivot long

This converts the long form date columns to a single date column with values.

```
covid19_confirmed <- covid19_confirmed_raw %>%
  pivot_longer(cols = -c(UID:Combined_Key),
                       names_to = "date",
                       values_to = "cases") %>%
  select(-c(Lat, Long_)) ## remove unwanted columns

covid19_deaths <- covid19_deaths_raw %>%
  pivot_longer(cols = -c(UID:Population),
                       names_to = "date",
```

```
                        values_to = "deaths") %>%
  select(-c(Lat, Long_)) ## remove unwanted columns
```

Join cases and deaths data and change column types (factor, date)

We change date strings to dates and identifier kind of string columns to factors.

```
covid19 <- covid19_confirmed %>%
  full_join(covid19_deaths) %>%
  mutate(iso3 = factor(iso3)) %>%
  mutate(Admin2 = factor(Admin2)) %>%
  mutate(Province_State = factor(Province_State)) %>%
  mutate(Country_Region = factor(Country_Region)) %>%
  mutate(date = mdy(date)) %>%
  select (-c(UID, iso2, code3, FIPS))
```

```
## Joining, by = c("UID", "iso2", "iso3", "code3", "FIPS", "Admin2", "Province_State", "Country_Region"
```

Missing data columns and plans to handle them

There is missing data in the following columns:

```
names(which(colSums(is.na(covid19)) > 0))
```

```
## [1] "Admin2"
```

I will ignore such data for this report by filtering.

```
covid19 <- covid19 %>%
  filter(iso3 == "USA") %>%  # filter non states for simplicity
  filter(!is.na(Admin2))     # ignore cruise ships
```

Summary of the cleaned up data

```
summary(covid19)
```

```
##   iso3               Admin2         Province_State    Country_Region
##  ASM:      0   Unassigned:  38352   Texas    : 192512   US:2448512
##  GUM:      0   Washington:  23312   Georgia  : 121072
##  MNP:      0   Jefferson :  19552   Virginia: 101520
##  PRI:      0   Franklin  :  18800   Kentucky:  91744
##  USA:2448512   Jackson   :  18048   Missouri:  88736
##  VIR:      0   Lincoln   :  18048   Kansas   :  80464
##                (Other)   :2312400   (Other) :1772464
##  Combined_Key            date                cases           Population
##  Length:2448512     Min.   :2020-01-22   Min.   :      0   Min.   :       0
##  Class :character   1st Qu.:2020-07-27   1st Qu.:     78   1st Qu.:    9738
##  Mode  :character   Median :2021-01-31   Median :   1044   Median :   24528
```

```
##                     Mean   :2021-01-31   Mean   :   7265   Mean   :  100964
##                     3rd Qu.:2021-08-07   3rd Qu.:   4108   3rd Qu.:   65422
##                     Max.   :2022-02-11   Max.   :2752398   Max.   :10039107
##
##      deaths
##  Min.   :    0.0
##  1st Qu.:    1.0
##  Median :   19.0
##  Mean   :  125.2
##  3rd Qu.:   74.0
##  Max.   :29764.0
##
```

filter data

I will remove data with zero cases for simplicity.

```
covid19 <- covid19 %>%
  filter(cases > 0)
```

Data issues

There are deaths with population zero. This is because the deaths are recorded with unassigned county (Alaska for example). These will have to be removed if we need any division by population operation for any individual date. I do not plan to do that.

Visualization and Analysis

Generate necessary analysis ready data.

Generate per state aggregated data using grouping by location and summing the columns needed.

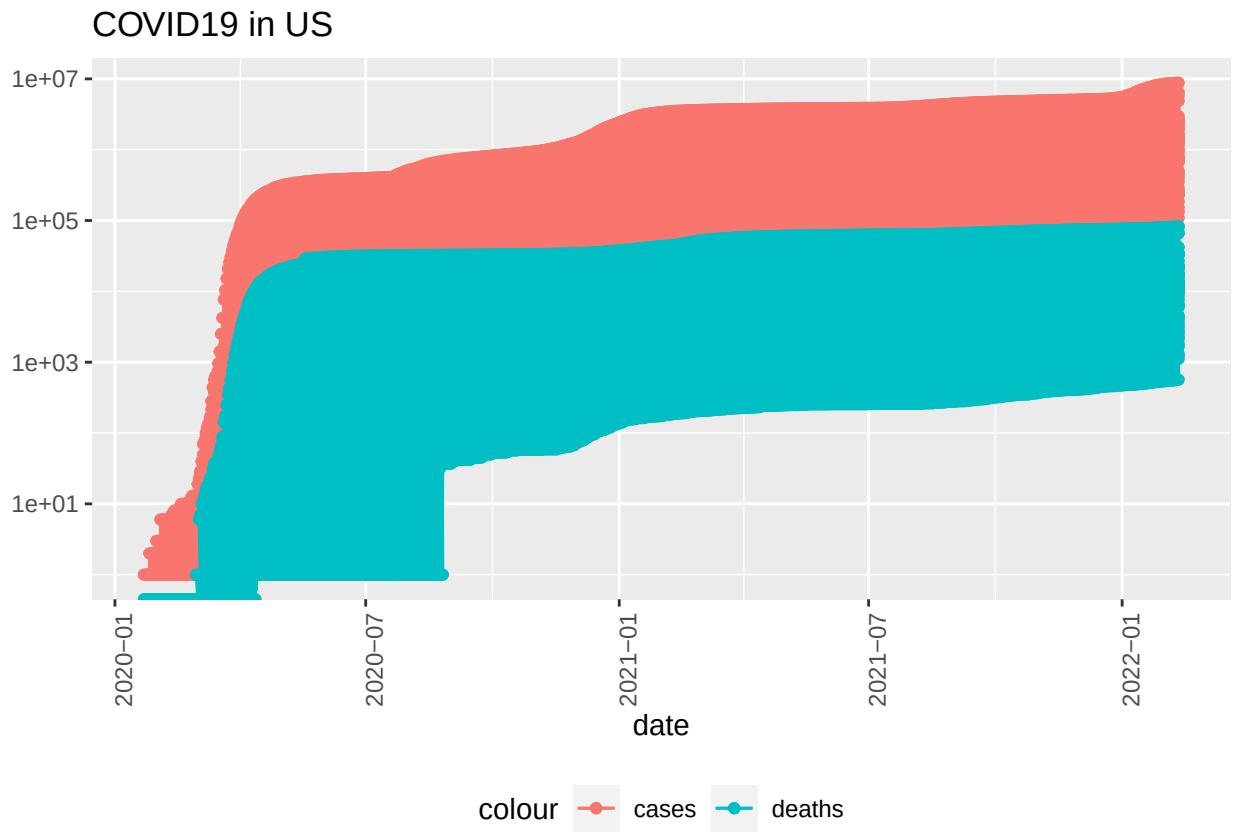
```
covid19_by_state <- covid19 %>%
  group_by(Province_State, Country_Region, date) %>%
  summarise(cases = sum(cases), deaths = sum(deaths), Population = sum(Population)) %>%
  mutate(deaths_per_million = deaths * 1000000 / Population) %>%
  select(Province_State, Country_Region, date,
         cases, deaths, deaths_per_million, Population) %>%
  ungroup()
```

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

Plot

This plot shows the accumulated cases and deaths over the whole of US. The plot looks thick since each state gets a point per day.

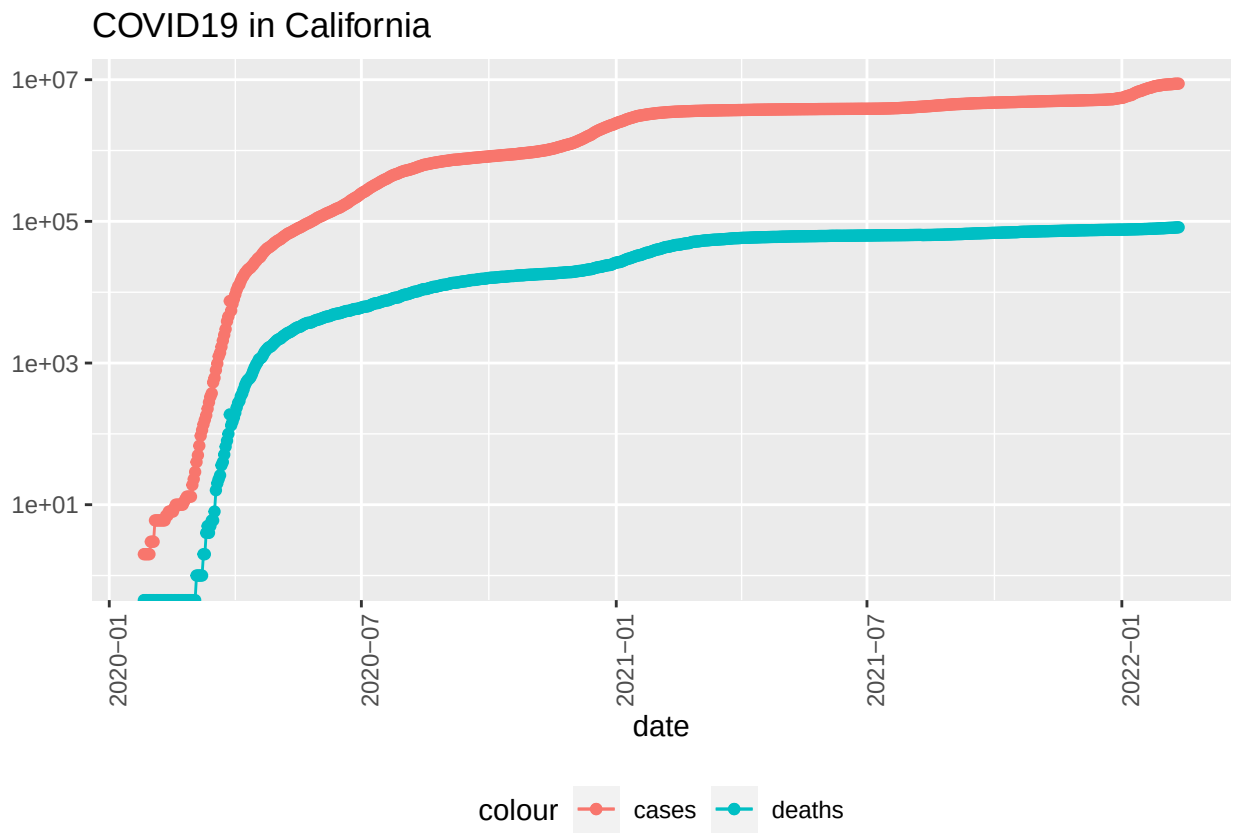
```
covid19_by_state %>%
  ggplot(aes(x = date, y = cases)) +

  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)
```



Plotting for the state of California only.

```
covid19_by_state %>%
  filter(Province_State == "California") %>%
  ggplot(aes(x = date, y = cases)) +

  geom_line(aes(color = "cases")) +
  geom_point(aes(color = "cases")) +

  geom_line(aes(y = deaths, color = "deaths")) +
```

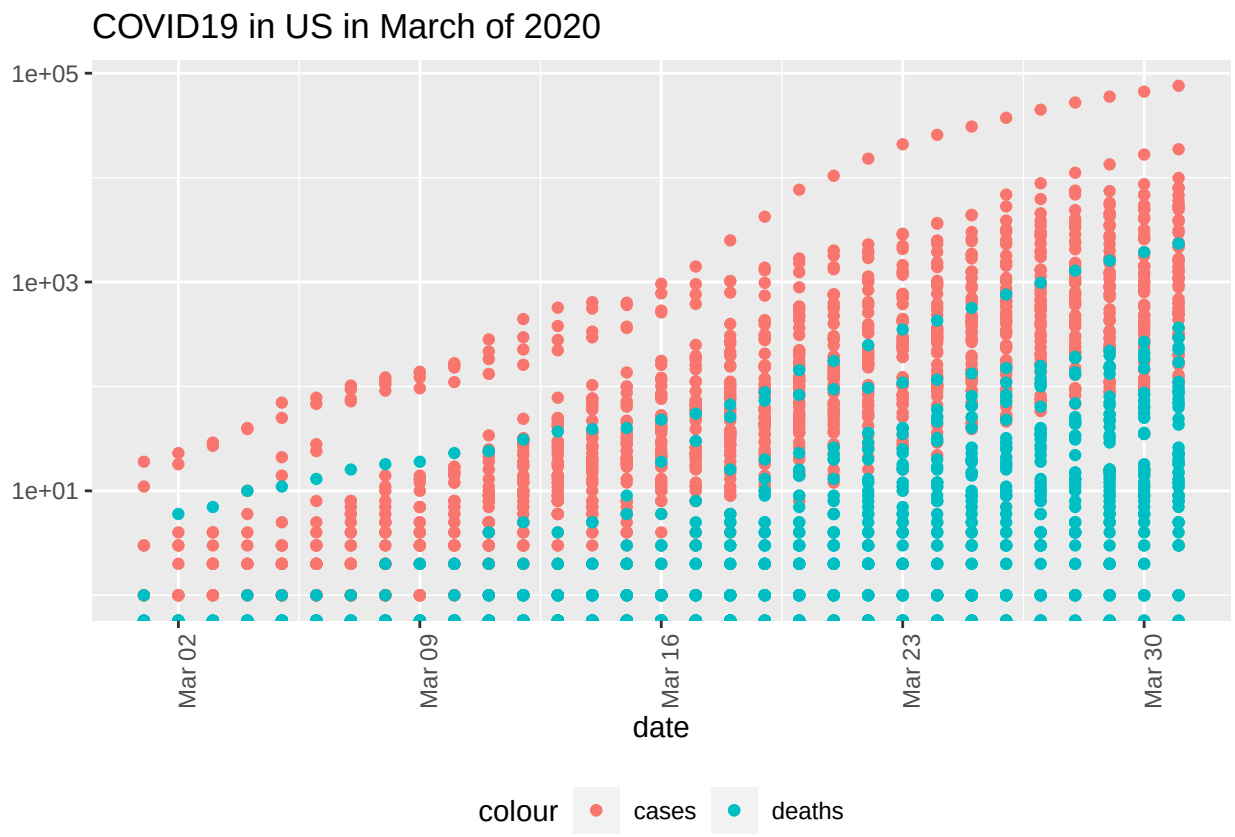
```
  geom_point(aes(y = deaths, color = "deaths")) +

  scale_y_log10() +

  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +

  labs(title = "COVID19 in California", y = NULL)
```



Plot for the month of March, 2020 only. This shows the range of values on a given day across all the US states.

```
covid19_by_state %>%
  filter (year(date) == 2020) %>%
  filter (month(date) == 3) %>%
  # filter (month(date) == 4 | month(date) == 5) %>%
  # filter (day(date) < 25) %>%
  ggplot(aes(x = date, y = cases)) +

  geom_point(aes(color = "cases")) +

  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 in March of 2020", y = NULL)
```



Summarize by monthly numbers

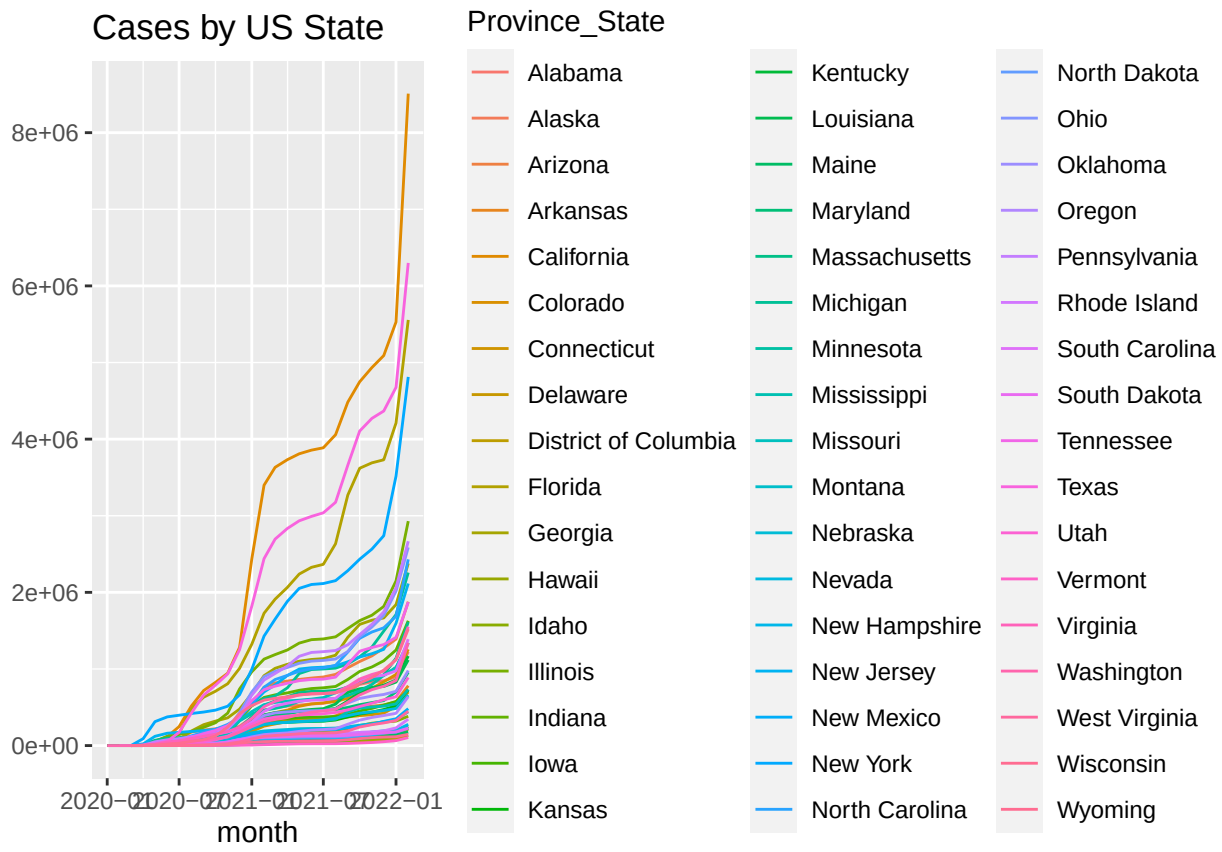
Aggregating the data by getting one point per month per state.

```
covid19_by_state_month <- covid19_by_state %>%
  group_by(Province_State,  month = lubridate::floor_date(date, "month")) %>%
  summarise(cases = min(cases), # use value from the 1st of the month
            deaths = min(deaths),
            Population=max(Population),
            cases_per_million = cases * 1000000 / Population,
            deaths_per_million = deaths * 1000000 / Population
            ) %>%  # occurrences by month
  ungroup()
```

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

Plotting this data shows the variations in the pandemic across the US states.

```
covid19_by_state_month %>%
  group_by(month) %>%
  ggplot(aes(x=month, y=cases, group=Province_State, color=Province_State)) +
  geom_line() +
  labs(title = "Cases by US State", y = NULL)
```



Analyze a single day

Get data for a single day and show the top and bottom states by cases.

```
all_data = covid19_by_state_month %>%
                  filter(month == as.Date("2022-01-01")) %>%
                  select (-c(month, cases_per_million))
```

```
head(all_data %>% arrange(cases))
```

```
## # A tibble: 6 x 5
##   Province_State        cases deaths Population deaths_per_million
##   <fct>                 <dbl>  <dbl>      <dbl>              <dbl>
## 1 Vermont               64447    471     623989               755.
## 2 District of Columbia  94286   1211     705749              1716.
## 3 Wyoming              115638   1526     578759              2637.
## 4 Hawaii               115642   1094    1415786               773.
```

```
## 5 Maine                146736   1531    1344212              1139.
## 6 Alaska               157169    978     728809              1342.
```

```
tail(all_data %>% arrange(cases))
```

```
## # A tibble: 6 x 5
##   Province_State   cases deaths Population deaths_per_million
##   <fct>            <dbl>  <dbl>      <dbl>              <dbl>
## 1 Pennsylvania   2059613  36714   12801989              2868.
## 2 Illinois       2149548  30254   12671821              2388.
## 3 New York       3517696  59209   19453561              3044.
## 4 Florida        4209927  62504   21477737              2910.
## 5 Texas          4675575  75744   28995881              2612.
## 6 California     5528658  76520   39512223              1937.
```

Same data ordered by `cases_per_million` shows a different ordering.

```
all_data = covid19_by_state_month %>%
                  filter(month == as.Date("2022-01-01")) %>%
                  select (-c(month, deaths_per_million))
```

```
head(all_data %>% arrange(cases_per_million))
```

```
## # A tibble: 6 x 5
##   Province_State  cases deaths Population cases_per_million
##   <fct>           <dbl>  <dbl>      <dbl>             <dbl>
## 1 Hawaii         115642   1094    1415786             81680.
## 2 Oregon         421263   5655    4217737             99879.
## 3 Vermont         64447    471     623989            103282.
## 4 Maine          146736   1531    1344212            109161.
## 5 Washington     849075   9853    7614893            111502.
## 6 Maryland       700553  11758    6045680            115877.
```

```
tail(all_data %>% arrange(cases_per_million))
```

```
## # A tibble: 6 x 5
##   Province_State   cases deaths Population cases_per_million
##   <fct>            <dbl>  <dbl>      <dbl>             <dbl>
## 1 South Dakota    179204   2486     884659            202568.
## 2 Tennessee      1412302  20842    6829174            206804.
## 3 Alaska          157169    978     728809            215652.
## 4 Rhode Island    231096   3066    1059361            218147.
## 5 North Dakota    174626   2012     762062            229149.
## 6 Utah            636992   3787    2315956            275045.
```

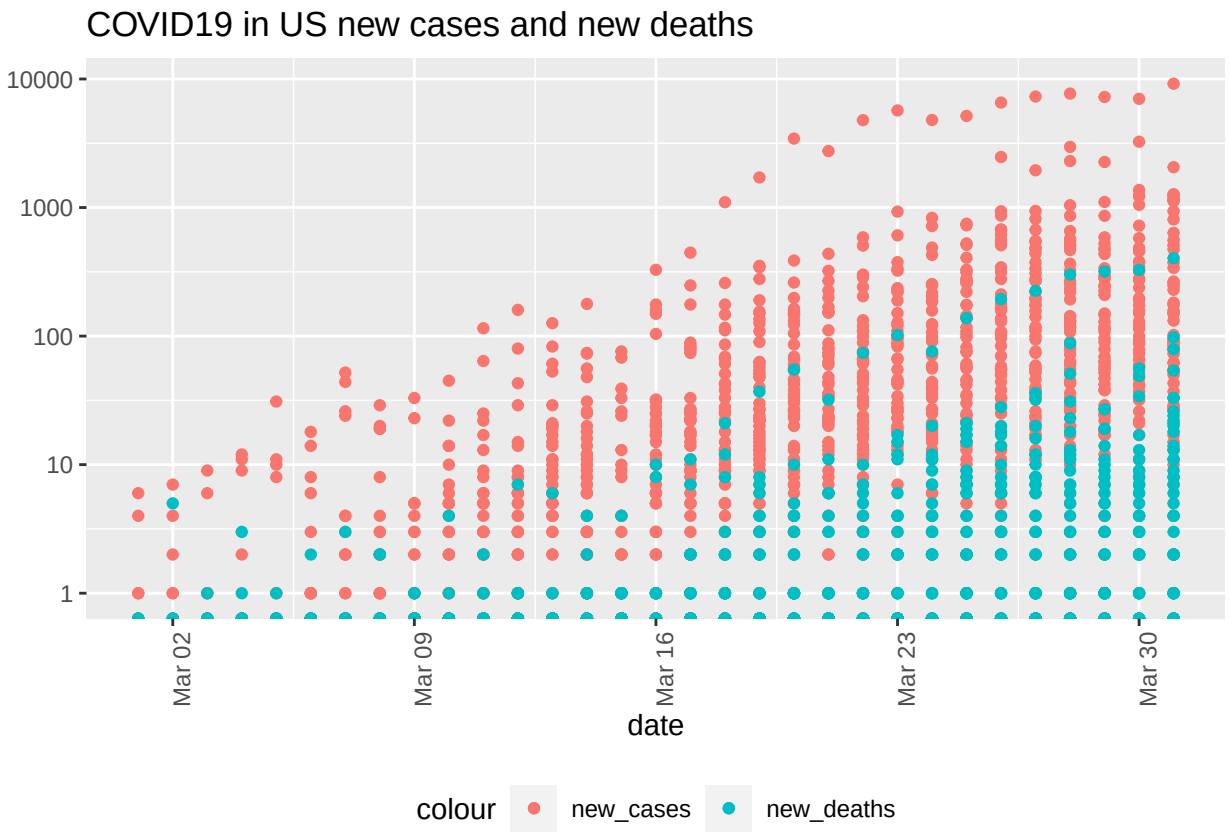
Data for model fitting

Generate daily cases from the cumulative.

```
covid19_by_state <- covid19_by_state %>%
  mutate(new_cases = cases - lag(cases),
         new_deaths = deaths - lag(deaths)
         )
```

Plot them for a day to see they look right.

```
covid19_by_state %>%
  filter (year(date) == 2020) %>%
  filter (month(date) == 3) %>%

  ggplot(aes(x = date, y = new_cases)) +

  geom_point(aes(color = "new_cases")) +

  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 = "COVID19 in US new cases and new deaths", y = NULL)
```



Filter data for California only for a better fit.

```
california_data = covid19_by_state_month %>%
  filter(cases > 0) %>%
  filter(Province_State == "California")
```

Model the data using linear model to fit deaths to cases.

```
model <- lm(deaths ~ cases, data = california_data)
summary(model)
```

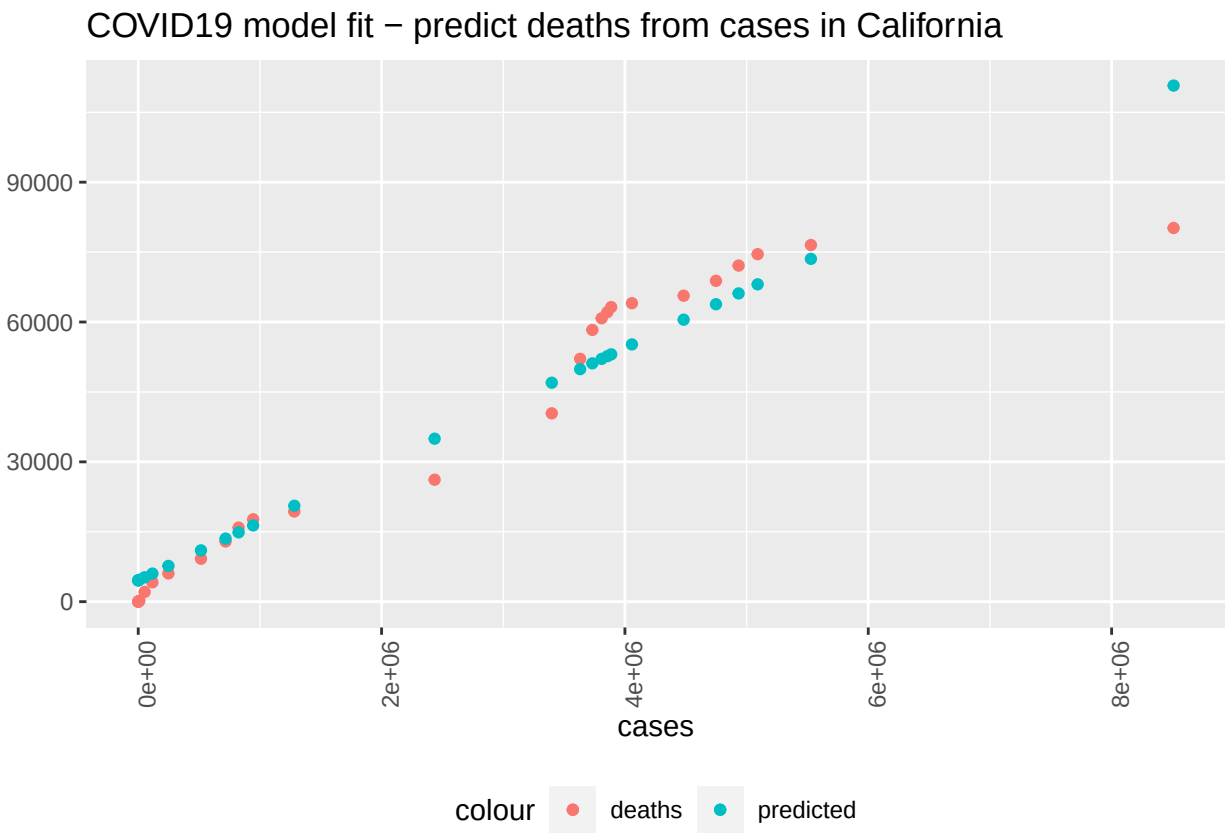
```
##
## Call:
## lm(formula = deaths ~ cases, data = california_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -30560.0  -4159.6    202.2   5769.1  10130.0
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.580e+03  2.496e+03   1.835   0.0789 .
## cases       1.247e-02  7.280e-04  17.135 5.79e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8422 on 24 degrees of freedom
## Multiple R-squared:  0.9244, Adjusted R-squared:  0.9213
## F-statistic: 293.6 on 1 and 24 DF,  p-value: 5.792e-15
```

```
california_data_pred  <- california_data %>%
  mutate(pred = predict(model))
```

Plot the model predictions

Plot the model data with predicted values.

```
california_data_pred %>%
  ggplot(aes(x = cases, y = deaths)) +

  geom_point(aes(color = "deaths")) +

  geom_point(aes(y = pred, color = "predicted")) +

  # scale_y_log10() +

  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90)) +

  labs(title = "COVID19 model fit - predict deaths from cases in California", y = NULL)
```



Bias

There could be multiple sources of bias in the COVID19 data:

- The data collection is likely biased since this data is collected over many jurisdictions with possibly different rules of reporting.
- The data also spans a time window that 2 years where data collection process may not be uniform.
- I have picked California to fit a model. It is possible that a linear model may not fit for all states.

Session info

```
## R version 4.1.2 (2021-11-01)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Monterey 12.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1-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] chron_2.3-56    lubridate_1.8.0 forcats_0.5.1   stringr_1.4.0
##  [5] dplyr_1.0.7     purrr_0.3.4     readr_2.1.1     tidyr_1.1.4
##  [9] tibble_3.1.6    ggplot2_3.3.5   tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.8       assertthat_0.2.1 digest_0.6.29    utf8_1.2.2
##  [5] R6_2.5.1         cellranger_1.1.0 backports_1.4.1  reprex_2.0.1
##  [9] evaluate_0.14    highr_0.9        httr_1.4.2       pillar_1.6.4
## [13] rlang_0.4.12     curl_4.3.2       readxl_1.3.1     rstudioapi_0.13
## [17] rmarkdown_2.11   labeling_0.4.2   bit_4.0.4        munsell_0.5.0
## [21] broom_0.7.11     compiler_4.1.2   modelr_0.1.8     xfun_0.29
## [25] pkgconfig_2.0.3  htmltools_0.5.2  tidyselect_1.1.1 fansi_1.0.2
## [29] crayon_1.4.2     tzdb_0.2.0       dbplyr_2.1.1     withr_2.4.3
## [33] grid_4.1.2       jsonlite_1.7.2   gtable_0.3.0     lifecycle_1.0.1
## [37] DBI_1.1.2        magrittr_2.0.1   scales_1.1.1     cli_3.1.0
## [41] stringi_1.7.6    vroom_1.5.7      farver_2.1.0     fs_1.5.2
## [45] xml2_1.3.3       ellipsis_0.3.2   generics_0.1.1   vctrs_0.3.8
## [49] tools_4.1.2      bit64_4.0.5      glue_1.6.0       hms_1.1.1
## [53] parallel_4.1.2   fastmap_1.1.0    yaml_2.2.1       colorspace_2.0-2
## [57] rvest_1.0.2      knitr_1.37       haven_2.4.3
```