Covid 19 analysis

Jie Shen

2023-09-24

File and Data

This is a R Markdown document for **COVID 19 project for China**. The data used in this project can be found at "https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series". Please visit the site for detailed data description.

The data I used are global cases and deaths. From the website "https://github.com/CSSEGISandData/COVID-19", Johns Hopkins Corona virus Resource Center ceased its collecting and reporting of global COVID-19 data on March 10, 2023. The global data is from World Health Organization (WHO) "https://www.who.int/".

Project goal

The project is to discover patterns and trends from COVID data in China. I want to explore things like the COVID cases and deaths trends over the years, and what states are best and worst.

Packages needed

Be sure the following packages are installed first:

- tidyverse
- ggplot2
- caret
- lubridate

Load Packages

```
library(tidyverse)
library(ggplot2)
library(forcats)
library(lubridate)
library(dplyr)
library(caret)
```

Import Data and clean up

```r
#Import data from webnsite
url_in<-"https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid
file_names<-c("time_series_covid19_confirmed_global.csv","time_series_covid19_deaths_global.csv")

urls=str_c(url_in, file_names)
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.
```

```r
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.
```

Now let's take a look and do some clean up

```r
# Take a look
head(global_cases)
```

```
## # A tibble: 6 x 1,147
##   `Province/State` `Country/Region`   Lat  Long `1/22/20` `1/23/20` `1/24/20`
##   <chr>            <chr>            <dbl> <dbl>     <dbl>     <dbl>     <dbl>
## 1 <NA>             Afghanistan       33.9 67.7          0         0         0
## 2 <NA>             Albania           41.2 20.2          0         0         0
## 3 <NA>             Algeria           28.0  1.66         0         0         0
## 4 <NA>             Andorra           42.5  1.52         0         0         0
## 5 <NA>             Angola           -11.2 17.9          0         0         0
## 6 <NA>             Antarctica       -71.9 23.3          0         0         0
## # i 1,140 more variables: `1/25/20` <dbl>, `1/26/20` <dbl>, `1/27/20` <dbl>,
## #   `1/28/20` <dbl>, `1/29/20` <dbl>, `1/30/20` <dbl>, `1/31/20` <dbl>,
## #   `2/1/20` <dbl>, `2/2/20` <dbl>, `2/3/20` <dbl>, `2/4/20` <dbl>,
## #   `2/5/20` <dbl>, `2/6/20` <dbl>, `2/7/20` <dbl>, `2/8/20` <dbl>,
## #   `2/9/20` <dbl>, `2/10/20` <dbl>, `2/11/20` <dbl>, `2/12/20` <dbl>,
## #   `2/13/20` <dbl>, `2/14/20` <dbl>, `2/15/20` <dbl>, `2/16/20` <dbl>,
## #   `2/17/20` <dbl>, `2/18/20` <dbl>, `2/19/20` <dbl>, `2/20/20` <dbl>, ...
```

```r
head(global_deaths)
```

```
## # A tibble: 6 x 1,147
##   `Province/State` `Country/Region`   Lat  Long `1/22/20` `1/23/20` `1/24/20`
##   <chr>            <chr>            <dbl> <dbl>     <dbl>     <dbl>     <dbl>
## 1 <NA>             Afghanistan       33.9 67.7          0         0         0
## 2 <NA>             Albania           41.2 20.2          0         0         0
## 3 <NA>             Algeria           28.0  1.66         0         0         0
## 4 <NA>             Andorra           42.5  1.52         0         0         0
## 5 <NA>             Angola           -11.2 17.9          0         0         0
## 6 <NA>             Antarctica       -71.9 23.3          0         0         0
## # i 1,140 more variables: `1/25/20` <dbl>, `1/26/20` <dbl>, `1/27/20` <dbl>,
## #   `1/28/20` <dbl>, `1/29/20` <dbl>, `1/30/20` <dbl>, `1/31/20` <dbl>,
## #   `2/1/20` <dbl>, `2/2/20` <dbl>, `2/3/20` <dbl>, `2/4/20` <dbl>,
## #   `2/5/20` <dbl>, `2/6/20` <dbl>, `2/7/20` <dbl>, `2/8/20` <dbl>,
## #   `2/9/20` <dbl>, `2/10/20` <dbl>, `2/11/20` <dbl>, `2/12/20` <dbl>,
## #   `2/13/20` <dbl>, `2/14/20` <dbl>, `2/15/20` <dbl>, `2/16/20` <dbl>,
## #   `2/17/20` <dbl>, `2/18/20` <dbl>, `2/19/20` <dbl>, `2/20/20` <dbl>, ...
```

```r
# Need to pivot dates to rows
global_cases<-global_cases %>%
  pivot_longer(cols= -c("Province/State", "Country/Region", Lat, Long),
                     names_to="date",
                     values_to="cases")
head(global_cases)
```

```
## # A tibble: 6 x 6
##   `Province/State` `Country/Region`   Lat  Long date    cases
##   <chr>            <chr>            <dbl> <dbl> <chr>   <dbl>
## 1 <NA>             Afghanistan       33.9  67.7 1/22/20     0
## 2 <NA>             Afghanistan       33.9  67.7 1/23/20     0
## 3 <NA>             Afghanistan       33.9  67.7 1/24/20     0
## 4 <NA>             Afghanistan       33.9  67.7 1/25/20     0
## 5 <NA>             Afghanistan       33.9  67.7 1/26/20     0
## 6 <NA>             Afghanistan       33.9  67.7 1/27/20     0
```

```r
# Do similar things to global deaths
global_deaths<-global_deaths %>%
  pivot_longer(cols= -c("Province/State", "Country/Region", Lat, Long),
                     names_to="date",
                     values_to="deaths")

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

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

```r
# Summary statistics
summary(global)
```

```
##  Province_State     Country_Region          Lat               Long
##  Length:330327      Length:330327      Min.   :-71.950   Min.   :-178.12
##  Class :character   Class :character   1st Qu.:  3.934   1st Qu.: -42.60
##  Mode  :character   Mode  :character   Median : 21.513   Median :  20.94
##                                        Mean   : 19.719   Mean   :  22.18
##                                        3rd Qu.: 40.464   3rd Qu.:  90.36
##                                        Max.   : 71.707   Max.   : 178.06
##                                        NA's   :2286      NA's   :2286
##       date                cases               deaths
##  Min.   :2020-01-22   Min.   :        0   Min.   :      0
##  1st Qu.:2020-11-02   1st Qu.:      680   1st Qu.:      3
##  Median :2021-08-15   Median :    14429   Median :    150
##  Mean   :2021-08-15   Mean   :   959384   Mean   :  13380
##  3rd Qu.:2022-05-28   3rd Qu.:   228517   3rd Qu.:   3032
##  Max.   :2023-03-09   Max.   :103802702   Max.   :1123836
##
```

We can see the earliest date is 2020-01-22 and the latest is 2023-03-09.

Since it's unfair to compare the numbers from big population state to a small state, I also want to see cases and deaths per populations. I found the population data set on the same github website.

```r
# Import population data
uid_lookup_url="https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/UID_
uid=read_csv(uid_lookup_url)
```

```
## 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.
```

```r
# After looking through the columns, exclude unwanted columns
uid<-uid%>% select(-c(Lat, Long_, Combined_Key, iso2,  iso3,  code3,Admin2, UID, FIPS) )

# Add population column to global data
global<-global%>%
  full_join(uid, by=c("Province_State", "Country_Region"))

# Get China data
CN<-global%>%filter(Country_Region=="China")
head(CN)
```

```
## # A tibble: 6 x 8
##   Province_State Country_Region   Lat  Long date       cases deaths Population
##   <chr>          <chr>          <dbl> <dbl> <date>     <dbl>  <dbl>      <dbl>
```

```
## 1 Anhui          China          31.8  117. 2020-01-22     1      0   61027171
## 2 Anhui          China          31.8  117. 2020-01-23     9      0   61027171
## 3 Anhui          China          31.8  117. 2020-01-24    15      0   61027171
## 4 Anhui          China          31.8  117. 2020-01-25    39      0   61027171
## 5 Anhui          China          31.8  117. 2020-01-26    60      0   61027171
## 6 Anhui          China          31.8  117. 2020-01-27    70      0   61027171
```

Missing Values

Check missing value

```r
as.data.frame(colSums(CN%>%(is.na)))
```

```
##                  colSums(CN %>% (is.na))
## Province_State                         1
## Country_Region                         0
## Lat                                 1144
## Long                                1144
## date                                   1
## cases                                  1
## deaths                                 1
## Population                          1143
```

clean up missing values

```r
# I think we don't need Lat and Long columns, exclude them
CN<-CN%>%select(-c(Lat,Long))

# Since I will aggregate data by dates, I will just exclude rows with missing date. My guess is those a
CN <- CN %>% filter(!is.na(date))

# Check what Province_State those rows with missing Pupulation comes from
unique(CN %>% filter(is.na(Population)) %>% filter(!is.na(Province_State))) %>% distinct(Province_State)
```

```
## # A tibble: 1 x 1
##   Province_State
##   <chr>
## 1 Unknown
```

```r
# It seems those are all "unknow" states. I want to exclude them as well as cases 0 rows
CN <- CN %>% filter(cases>0 & Population >0)

# Take a look at missing values again
as.data.frame(colSums(CN%>%(is.na)))
```

```
##                  colSums(CN %>% (is.na))
## Province_State                         0
## Country_Region                         0
## date                                   0
## cases                                  0
## deaths                                 0
## Population                             0
```

Analysis

Get per state and total Country numbers

```
# China by state total cases, deaths, and death per million population
CN_by_state<-CN%>%
  group_by( Country_Region,Province_State, date) %>%
  summarise(cases=sum(cases), deaths=sum(deaths), Population = sum(Population)) %>%
  mutate(death_per_mill = deaths/Population*1000000) %>%
  ungroup()
```

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

```
#Take a look
tail(CN_by_state)
```

```
## # A tibble: 6 x 7
##   Country_Region Province_State date       cases deaths Population
##   <chr>          <chr>          <date>     <dbl>  <dbl>      <dbl>
## 1 China          Zhejiang       2023-03-04 11848      1   64567588
## 2 China          Zhejiang       2023-03-05 11848      1   64567588
## 3 China          Zhejiang       2023-03-06 11848      1   64567588
## 4 China          Zhejiang       2023-03-07 11848      1   64567588
## 5 China          Zhejiang       2023-03-08 11848      1   64567588
## 6 China          Zhejiang       2023-03-09 11848      1   64567588
## # i 1 more variable: death_per_mill <dbl>
```

```
# China Totals
CN_totals<- CN%>%
  group_by( Country_Region, date) %>%
  summarise(cases=sum(cases), deaths=sum(deaths), Population = sum(Population)) %>%
  mutate(death_per_mill = deaths/Population*1000000) %>%
  arrange(death_per_mill) %>%
 ungroup()
```

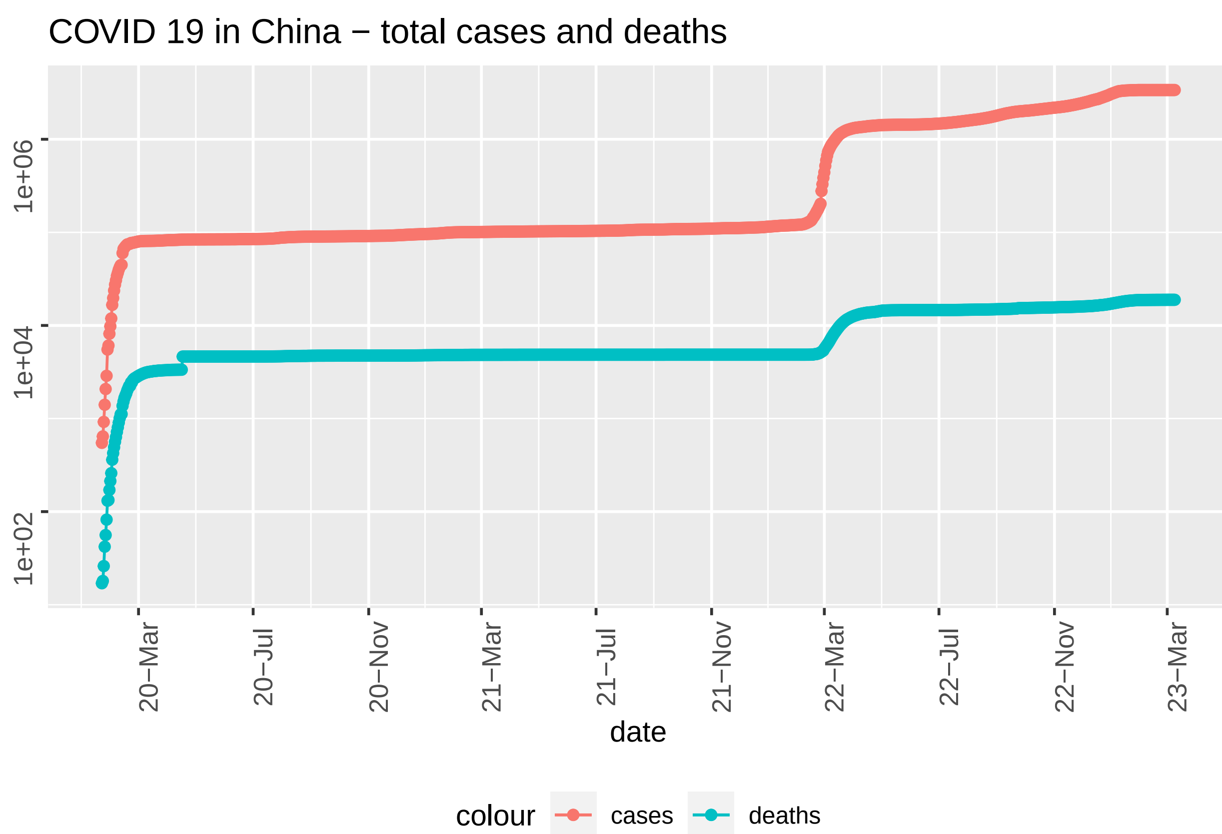
```
## `summarise()` has grouped output by 'Country_Region'. You can override using
## the `.groups` argument.
```

```
#Take a look
tail(CN_totals)
```

```
## # A tibble: 6 x 6
##   Country_Region date         cases deaths Population death_per_mill
##   <chr>          <date>       <dbl>  <dbl>      <dbl>          <dbl>
## 1 China          2023-03-04 3381708  18858 1417925054           13.3
## 2 China          2023-03-05 3381708  18859 1417925054           13.3
## 3 China          2023-03-06 3381708  18860 1417925054           13.3
## 4 China          2023-03-07 3381708  18860 1417925054           13.3
## 5 China          2023-03-08 3381708  18860 1417925054           13.3
## 6 China          2023-03-09 3381708  18861 1417925054           13.3
```

Visualization CN totals

```
# Visualize CN totals
options(repr.plot.width=30, repr.plot.height=10)
CN_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_x_date(date_labels = "%y-%b", date_breaks = "4 month") +
  theme(legend.position='bottom', axis.text=element_text(angle=90, size=10)) +
  labs(title="COVID 19 in China - total cases and deaths", y=NULL)
```



How about new cases and new deaths?

When looking at trends, it's good to see how many new cases and new deaths. Let's add those columns

```
# Add new cases columns to China data
CN_by_state<- CN_by_state%>% arrange(Country_Region, Province_State, date) %>%
      mutate(new_cases=cases-lag(cases), new_deaths=deaths-lag(deaths))
```

```r
CN_totals<- CN_totals%>% arrange(Country_Region, date) %>%
      mutate(new_cases=cases-lag(cases), new_deaths=deaths-lag(deaths))

# Take a look
tail(CN_by_state)
```

```
## # A tibble: 6 x 9
##   Country_Region Province_State date       cases deaths Population
##   <chr>          <chr>          <date>     <dbl>  <dbl>      <dbl>
## 1 China          Zhejiang       2023-03-04 11848      1   64567588
## 2 China          Zhejiang       2023-03-05 11848      1   64567588
## 3 China          Zhejiang       2023-03-06 11848      1   64567588
## 4 China          Zhejiang       2023-03-07 11848      1   64567588
## 5 China          Zhejiang       2023-03-08 11848      1   64567588
## 6 China          Zhejiang       2023-03-09 11848      1   64567588
## # i 3 more variables: death_per_mill <dbl>, new_cases <dbl>, new_deaths <dbl>
```

```r
tail(CN_totals)
```

```
## # A tibble: 6 x 8
##   Country_Region date         cases deaths Population death_per_mill new_cases
##   <chr>          <date>       <dbl>  <dbl>      <dbl>          <dbl>     <dbl>
## 1 China          2023-03-04 3381708  18858 1417925054           13.3         0
## 2 China          2023-03-05 3381708  18859 1417925054           13.3         0
## 3 China          2023-03-06 3381708  18860 1417925054           13.3         0
## 4 China          2023-03-07 3381708  18860 1417925054           13.3         0
## 5 China          2023-03-08 3381708  18860 1417925054           13.3         0
## 6 China          2023-03-09 3381708  18861 1417925054           13.3         0
## # i 1 more variable: new_deaths <dbl>
```

Visualize new cases and deaths in China

```r
# Visualize China totals
options(repr.plot.width=30, repr.plot.height=10)
CN_totals %>%
  filter(cases>0) %>%
  ggplot(aes(x=date, y=new_cases))  +
  geom_line(aes(color="new_cases")) +
  geom_point(aes(color="new_cases")) +
  geom_line(aes(y=deaths, color="new_deaths")) +
  geom_point(aes(y=deaths, color="new_deaths")) +
  scale_y_log10() +
  scale_x_date(date_labels = "%y-%b", date_breaks = "4 month") +
  theme(legend.position='bottom', axis.text=element_text(angle=90, size=10)) +
  labs(title="COVID 19 in China - new cases and deaths", y=NULL)
```

```
## Warning in self$trans$transform(x): NaNs produced
```

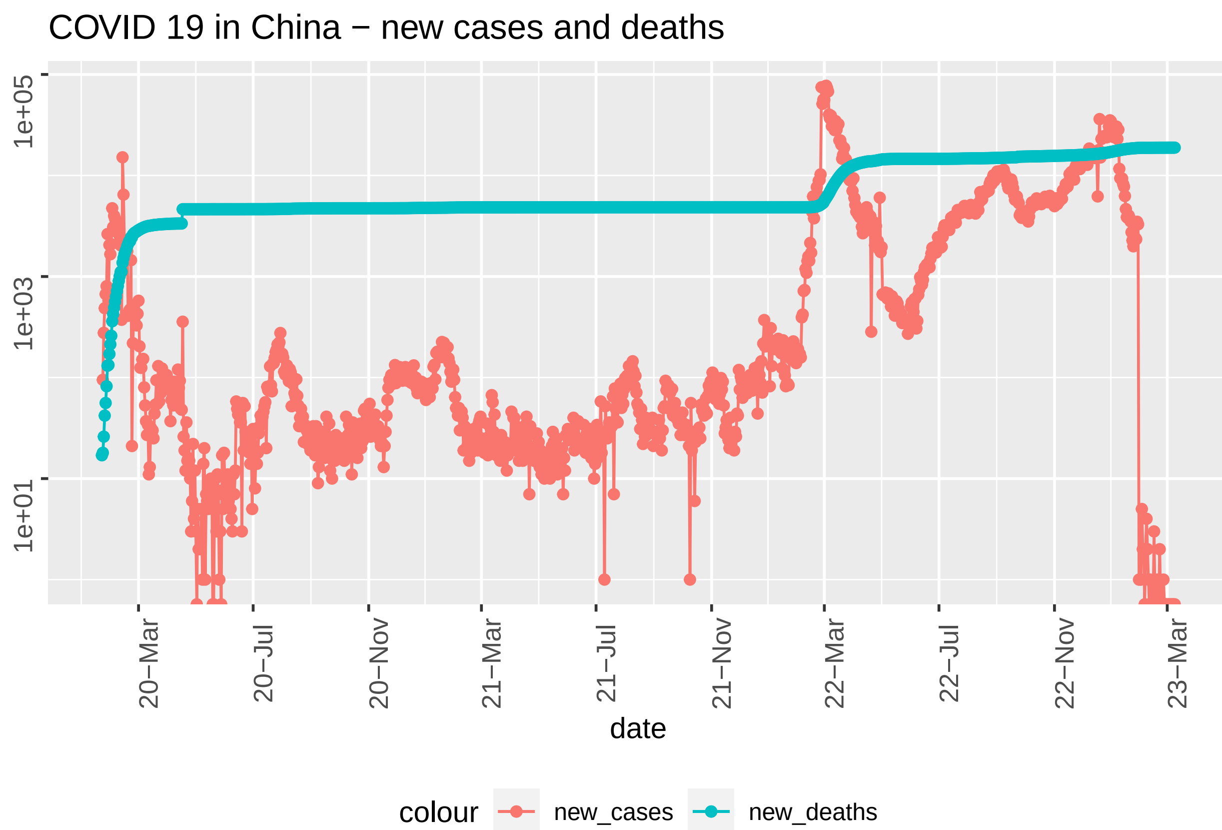
```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 1 row containing missing values (`geom_line()`).

## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



What are the worst and best states in China?

CN by states

Let's see which states are best/worst (in term of death/population)

```
CN_state_totals <- CN_by_state %>%
   group_by(Province_State) %>%
   summarize(cases=max(cases),
       deaths= max(deaths),
       Population=max(Population),
       cases_per_thou=1000*cases/Population,
       deaths_per_thou=1000*deaths/Population)
CN_state_totals %>% slice_min(deaths_per_thou,n=10)
```

```
## # A tibble: 10 x 6
##    Province_State cases deaths Population cases_per_thou deaths_per_thou
##    <chr>          <dbl>  <dbl>      <dbl>          <dbl>           <dbl>
##  1 Jiangsu         5075      0   84748016         0.0599       0
##  2 Ningxia         1276      0    7202654         0.177        0
##  3 Qinghai          782      0    5923957         0.132        0
##  4 Tibet           1647      0    3648100         0.451        0
##  5 Zhejiang       11848      1   64567588         0.183        0.0000155
##  6 Shanxi          7167      1   34915616         0.205        0.0000286
##  7 Guangxi        13371      2   50126804         0.267        0.0000399
##  8 Inner Mongolia  8847      1   24049155         0.368        0.0000416
##  9 Jiangxi         3423      2   45188635         0.0757       0.0000443
## 10 Liaoning        3547      2   42591407         0.0833       0.0000470
```

```r
CN_state_totals %>% slice_max(deaths_per_thou,n=10)
```

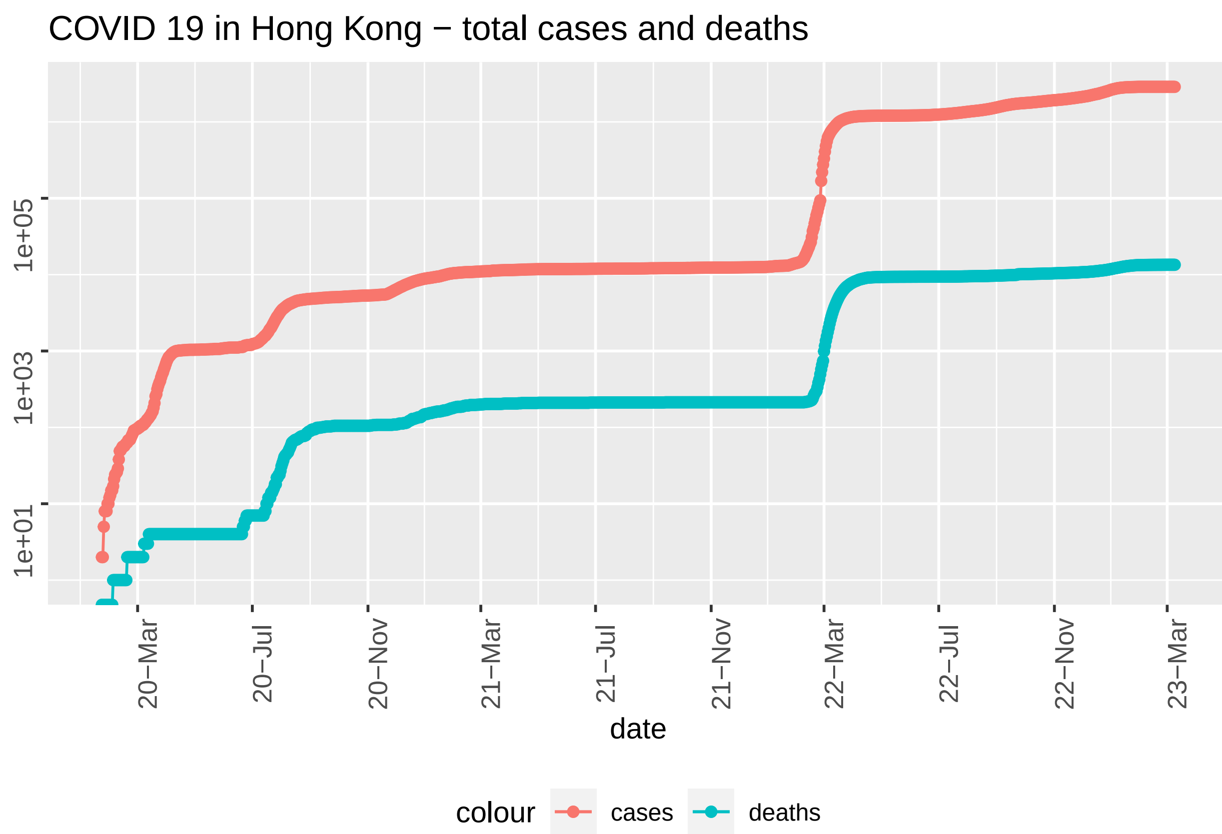
```
## # A tibble: 10 x 6
##    Province_State   cases deaths Population cases_per_thou deaths_per_thou
##    <chr>            <dbl>  <dbl>      <dbl>          <dbl>           <dbl>
##  1 Hong Kong      2876106  13467    7496988        384.            1.80
##  2 Macau             3547    121     649342          5.46          0.186
##  3 Hubei            72131   4515   57752557          1.25          0.0782
##  4 Shanghai         67040    595   24870895          2.70          0.0239
##  5 Beijing          40774     20   21893095          1.86          0.000914
##  6 Hainan           10483      6   10081232          1.04          0.000595
##  7 Heilongjiang      6603     18   31850088          0.207         0.000565
##  8 Chongqing        14715     11   32054159          0.459         0.000343
##  9 Henan             9948     23   99365519          0.100         0.000231
## 10 Tianjin           4392      3   13866009          0.317         0.000216
```

visualize state of interest

I want to visualize the top 3 worst states

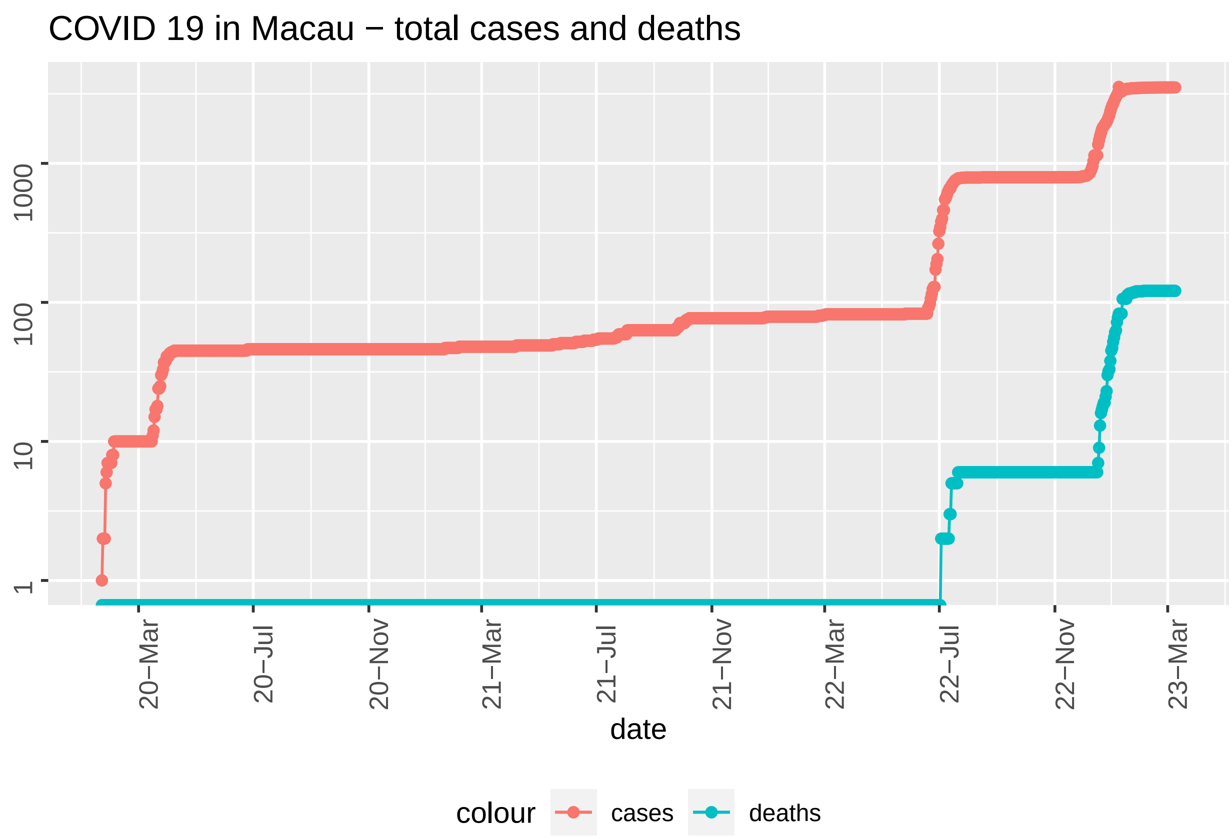
```r
state<- "Hong Kong"
CN_by_state %>%
  filter(Province_State==state) %>%
  ggplot(aes(x=date, y=cases))  +
  geom_line(aes(color="cases")) +
  geom_point(aes(color="cases")) +
  geom_line(aes(y=deaths, color="deaths")) +
  geom_point(aes(y=deaths, color="deaths")) +
  scale_y_log10() +
  scale_x_date(date_labels = "%y-%b", date_breaks = "4 month") +
  theme(legend.position='bottom', axis.text=element_text(angle=90, size=10)) +
  labs(title=str_c("COVID 19 in ", state," - total cases and deaths"), y=NULL)
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
## Transformation introduced infinite values in continuous y-axis
```

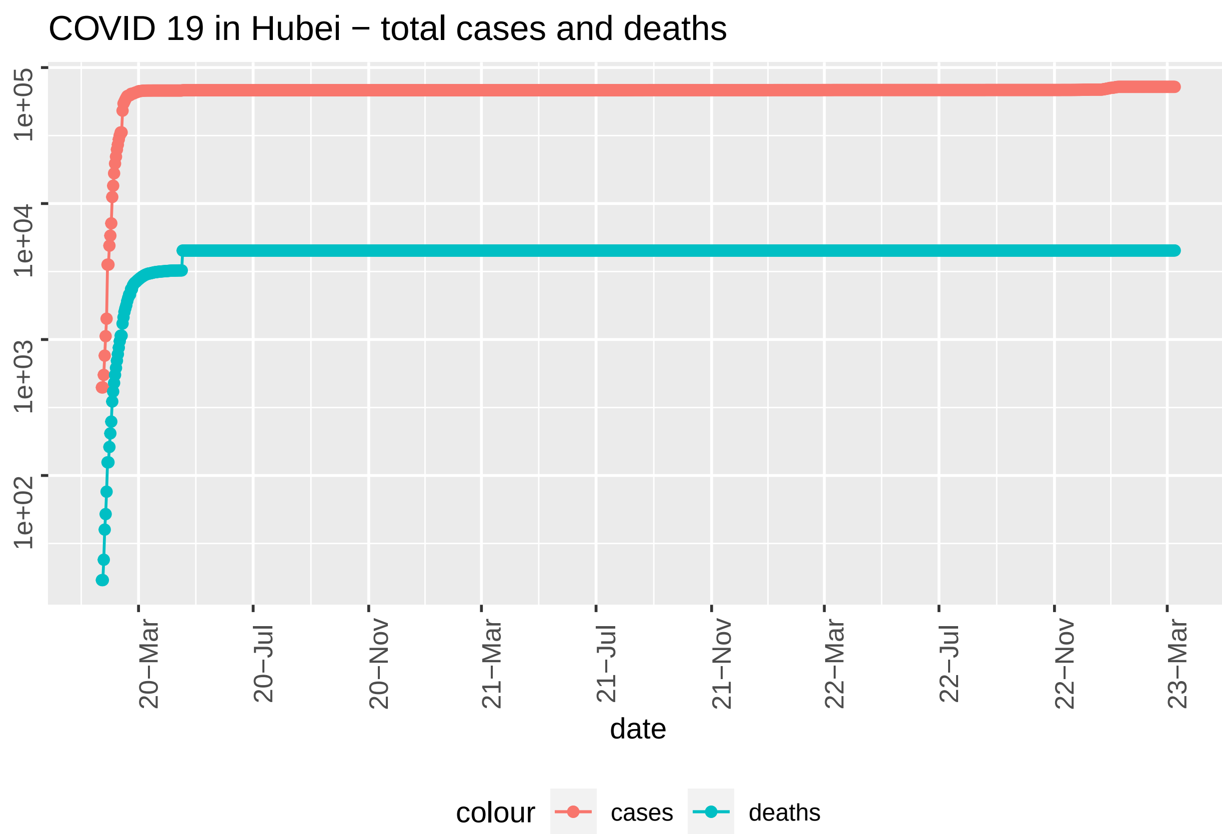



```r
state<- "Macau"
CN_by_state %>%
  filter(Province_State==state) %>%
  ggplot(aes(x=date, y=cases))  +
  geom_line(aes(color="cases")) +
  geom_point(aes(color="cases")) +
  geom_line(aes(y=deaths, color="deaths")) +
  geom_point(aes(y=deaths, color="deaths")) +
  scale_y_log10() +
  scale_x_date(date_labels = "%y-%b", date_breaks = "4 month") +
  theme(legend.position='bottom', axis.text=element_text(angle=90, size=10)) +
  labs(title=str_c("COVID 19 in ", state," - total cases and deaths"), y=NULL)
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
## Transformation introduced infinite values in continuous y-axis
```




```
state<- "Hubei"
CN_by_state %>%
  filter(Province_State==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() +
  scale_x_date(date_labels = "%y-%b", date_breaks = "4 month") +
  theme(legend.position='bottom', axis.text=element_text(angle=90, size=10)) +
  labs(title=str_c("COVID 19 in ", state," - total cases and deaths"), y=NULL)
```



Modeling

Modeling isn't the focus of this project. But I just want to do a couple of quick ones.

Linear Regression

```
# fit model
mod<-lm(deaths_per_thou ~ cases_per_thou, data = CN_state_totals)
summary(mod)
```

```
##
## Call:
## lm(formula = deaths_per_thou ~ cases_per_thou, data = CN_state_totals)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.014039 -0.007515 -0.006868 -0.006497  0.154572
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.252e-03  5.520e-03   1.133    0.266
## cases_per_thou 4.672e-03  8.264e-05  56.531   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03117 on 31 degrees of freedom
## Multiple R-squared:  0.9904, Adjusted R-squared:  0.9901
## F-statistic:  3196 on 1 and 31 DF,  p-value: < 2.2e-16
```

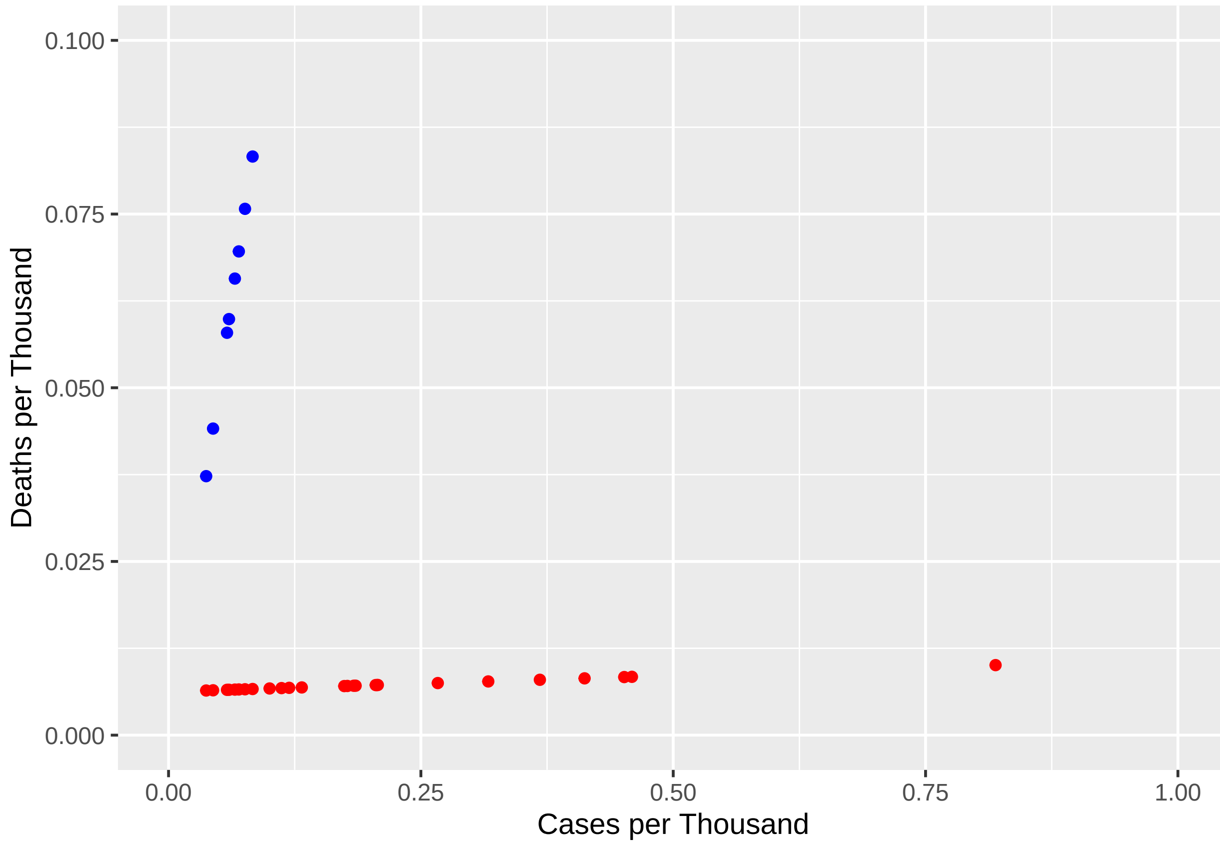
```r
# add predicted results column
CN_tot_w_pred <- CN_state_totals %>% mutate(pred = predict(mod))
head(CN_tot_w_pred)
```

```
## # A tibble: 6 x 7
##   Province_State  cases deaths Population cases_per_thou deaths_per_thou    pred
##   <chr>           <dbl>  <dbl>      <dbl>          <dbl>           <dbl>   <dbl>
## 1 Anhui            2275      7   61027171         0.0373       0.000115  0.00643
## 2 Beijing         40774     20   21893095         1.86         0.000914  0.0150
## 3 Chongqing       14715     11   32054159         0.459        0.000343  0.00840
## 4 Fujian          17122      2   41540086         0.412        0.0000481 0.00818
## 5 Gansu            1742      2   25019831         0.0696       0.0000799 0.00658
## 6 Guangdong      103248     10  126012510         0.819        0.0000794 0.0101
```

```r
# visualize results
CN_tot_w_pred %>% ggplot() +
  geom_point(aes(x = cases_per_thou, y = cases_per_thou), color = "blue") +
  geom_point(aes(x = cases_per_thou, y = pred), color = "red") +
  xlim(0,1)+
  ylim(0,0.1) +
  labs(x = "Cases per Thousand", y = "Deaths per Thousand")  # Customize axis labels
```

```
## Warning: Removed 25 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 7 rows containing missing values (`geom_point()`).
```



KNN

```
### fit model
mod <- train(deaths_per_thou ~ cases_per_thou, data = CN_state_totals, method = "knn")
print(mod)
```

```
## k-Nearest Neighbors
##
## 33 samples
##  1 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 33, 33, 33, 33, 33, 33, ...
## Resampling results across tuning parameters:
##
##   k  RMSE       Rsquared   MAE
##   5  0.2211850  0.5503169  0.06949801
##   7  0.2225704  0.5425577  0.06942550
##   9  0.2234967  0.5325535  0.06939834
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 5.
```

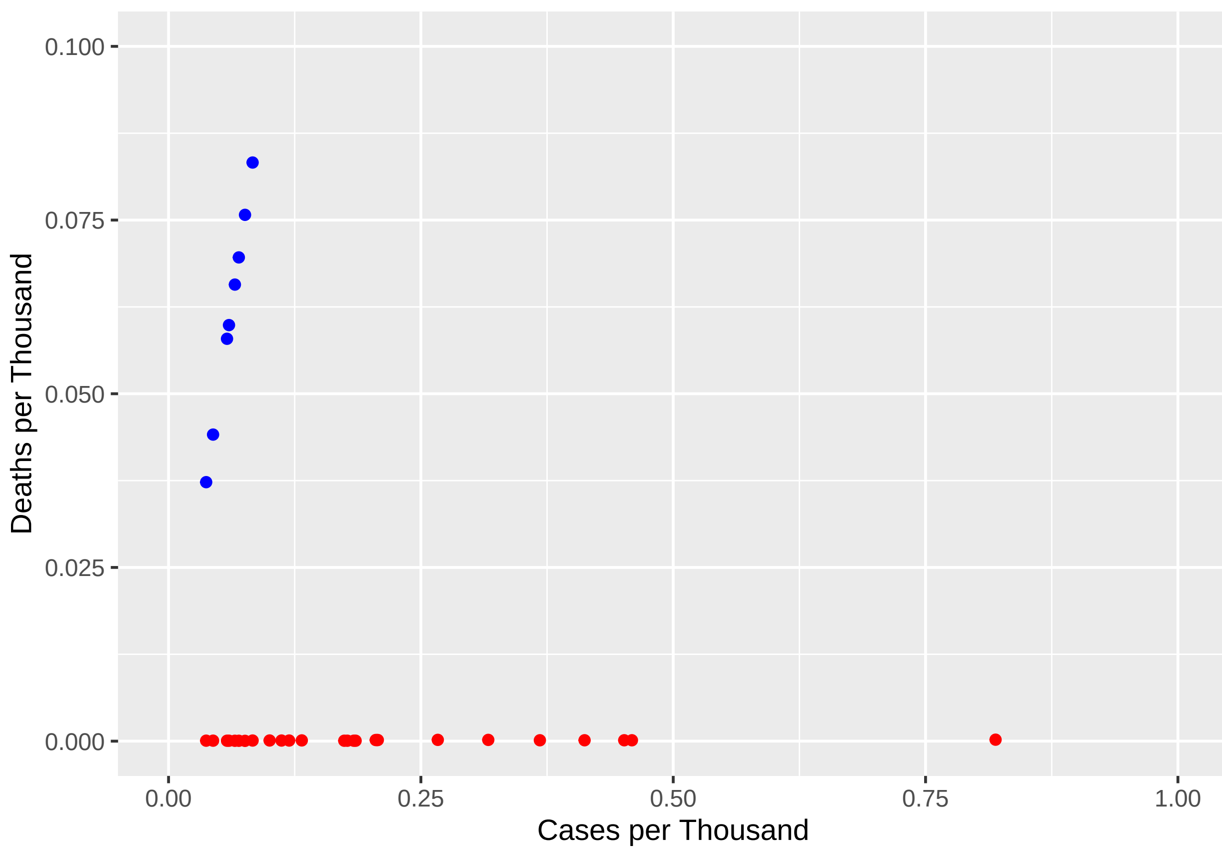
```
# add predicted results column
CN_tot_w_pred <- CN_state_totals %>% mutate(pred = predict(mod))
head(CN_tot_w_pred)
```

```
## # A tibble: 6 x 7
##   Province_State  cases deaths Population cases_per_thou deaths_per_thou    pred
##   <chr>           <dbl>  <dbl>      <dbl>          <dbl>           <dbl>   <dbl>
## 1 Anhui            2275      7   61027171         0.0373       0.000115  7.18e-5
## 2 Beijing         40774     20   21893095         1.86         0.000914  2.08e-2
## 3 Chongqing       14715     11   32054159         0.459        0.000343  1.30e-4
## 4 Fujian          17122      2   41540086         0.412        0.0000481 1.30e-4
## 5 Gansu            1742      2   25019831         0.0696       0.0000799 5.49e-5
## 6 Guangdong      103248     10  126012510         0.819        0.0000794 2.13e-4
```

```
# visuailze results
CN_tot_w_pred %>% ggplot() +
  geom_point(aes(x = cases_per_thou, y = cases_per_thou), color = "blue") +
  geom_point(aes(x = cases_per_thou, y = pred), color = "red") +
  xlim(0,1)+
  ylim(0,0.1) +
  labs(x = "Cases per Thousand", y = "Deaths per Thousand")  # Customize axis labels)
```

```
## Warning: Removed 25 rows containing missing values (`geom_point()`).

## Warning: Removed 7 rows containing missing values (`geom_point()`).
```



Bias

Bias can come from different sources.

First, I want to consider the quality of data. How is data collected? Are all cases and deaths captured? Why are there some unknown states with unknown populations? Since I excluded those rows with missing population and states, the total population of China won't be accurate.

Second, for modeling, it seems my predictions are far away from the actual. It seems the bias are large there. What causes this? I think maybe over the years the deaths per thousand and cases per thousand changes so much. It won't be good to just use deaths per thousand to predict cases per thousand. Maybe my modeling methods are wrong and my data doesn't fit well with the model assumptions.

Third, when I draw the modeling graphs, I eliminated some outliers. This may give the wrong impression of the range of deaths per thousand and cases per thousand.