Covid

Klaus Watschinger

11/25/2021

Load Data

For this report we use data from the Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University. We load for csv files.

```
## Get current Data in the four files which are all in the same folder
url_in <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_cov
file_names <-
  c("time_series_covid19_confirmed_US.csv",
    "time_series_covid19_confirmed_global.csv",
    "time_series_covid19_deaths_US.csv",
    "time_series_covid19_deaths_global.csv")
urls <- str_c(url_in,file_names)
```

With those 4 urls we now read in the data.

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

Clean up Data

My initial review of the data leads to the following next steps:

- Tidy data, values for each date should go into a row
- Remove Lat and Long
- Rename Region and State columns
- Convert strings to date
- Join deaths and cases for US and Global data sets.

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

```
#take a look at the data after the join, renaming and tidying up
summary(global)
```

```
##  Province_State     Country_Region          date                cases
##  Length:189840      Length:189840      Min.   :2020-01-22   Min.   :       0
##  Class :character   Class :character   1st Qu.:2020-07-09   1st Qu.:     173
##  Mode  :character   Mode  :character   Median :2020-12-25   Median :    3147
##                                        Mean   :2020-12-25   Mean   :  347610
##                                        3rd Qu.:2021-06-13   3rd Qu.:   68149
##                                        Max.   :2021-11-29   Max.   :48437955
##      deaths
##  Min.   :     0
##  1st Qu.:     1
##  Median :    48
##  Mean   :  7745
##  3rd Qu.:  1171
##  Max.   :778601
```

Now tidying the US data sets.

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

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

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

We now retrieve ISO and FIPS country codes to complete our work on global 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: 4214 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.
```

Finally, we perform a left join (all records in global are matched with the ISO FIPS codes) and finalize our global data set.

```
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)
```

Charting

Preparing the tibbles

Now we are going to visualize the US data. We want to see the data grouped on state level. (per date)

```
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, deaths_per_mill, cases_per_mill, Populati
  ungroup()
```

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

```
#Check the latest records
tail(US_by_state)
```

```
## # A tibble: 6 x 8
##   Province_State Country_Region date        cases deaths deaths_per_mill
##   <chr>          <chr>          <date>      <dbl>  <dbl>           <dbl>
## 1 Wyoming        US             2021-11-24 110264   1347           2327.
## 2 Wyoming        US             2021-11-25 110264   1347           2327.
```

```
## 3 Wyoming        US             2021-11-26 110264   1347           2327.
## 4 Wyoming        US             2021-11-27 110264   1347           2327.
## 5 Wyoming        US             2021-11-28 110264   1347           2327.
## 6 Wyoming        US             2021-11-29 110824   1347           2327.
## # ... with 2 more variables: cases_per_mill <dbl>, Population <dbl>
```

But we also want to see the totals for the US:

```
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) %>%
  mutate(cases_per_mill = cases *1000000 / Population) %>%
  select(Country_Region, date, cases, deaths, deaths_per_mill, cases_per_mill, Population) %>%
  ungroup()
```

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

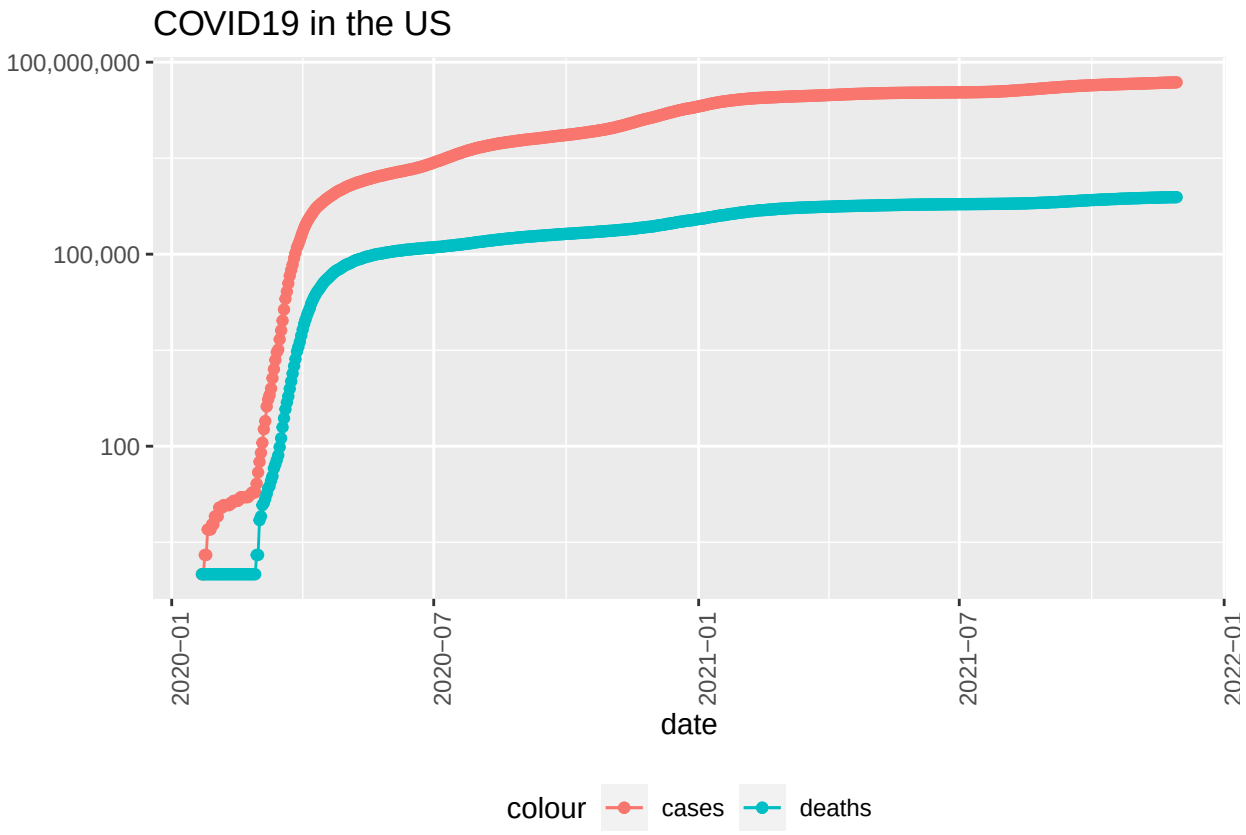
```
#Check the latest records
tail(US_totals)
```

```
## # A tibble: 6 x 7
##   Country_Region date          cases deaths deaths_per_mill cases_per_mill
##   <chr>          <date>        <dbl>  <dbl>           <dbl>          <dbl>
## 1 US             2021-11-24 48092823 775724           2330.        144477.
## 2 US             2021-11-25 48125425 776090           2331.        144575.
## 3 US             2021-11-26 48176523 776349           2332.        144729.
## 4 US             2021-11-27 48201079 776536           2333.        144802.
## 5 US             2021-11-28 48229210 776639           2333.        144887.
## 6 US             2021-11-29 48437955 778601           2339.        145514.
## # ... with 1 more variable: Population <dbl>
```

Timeseries

Now we want to visualize the development of cases and deaths for the US as a whole:

```
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(labels = scales::comma) +
  theme(legend.position="bottom",
        axis.text.x = element_text(angle = 90)) +
  labs(title = "COVID19 in the US", y= NULL)
```



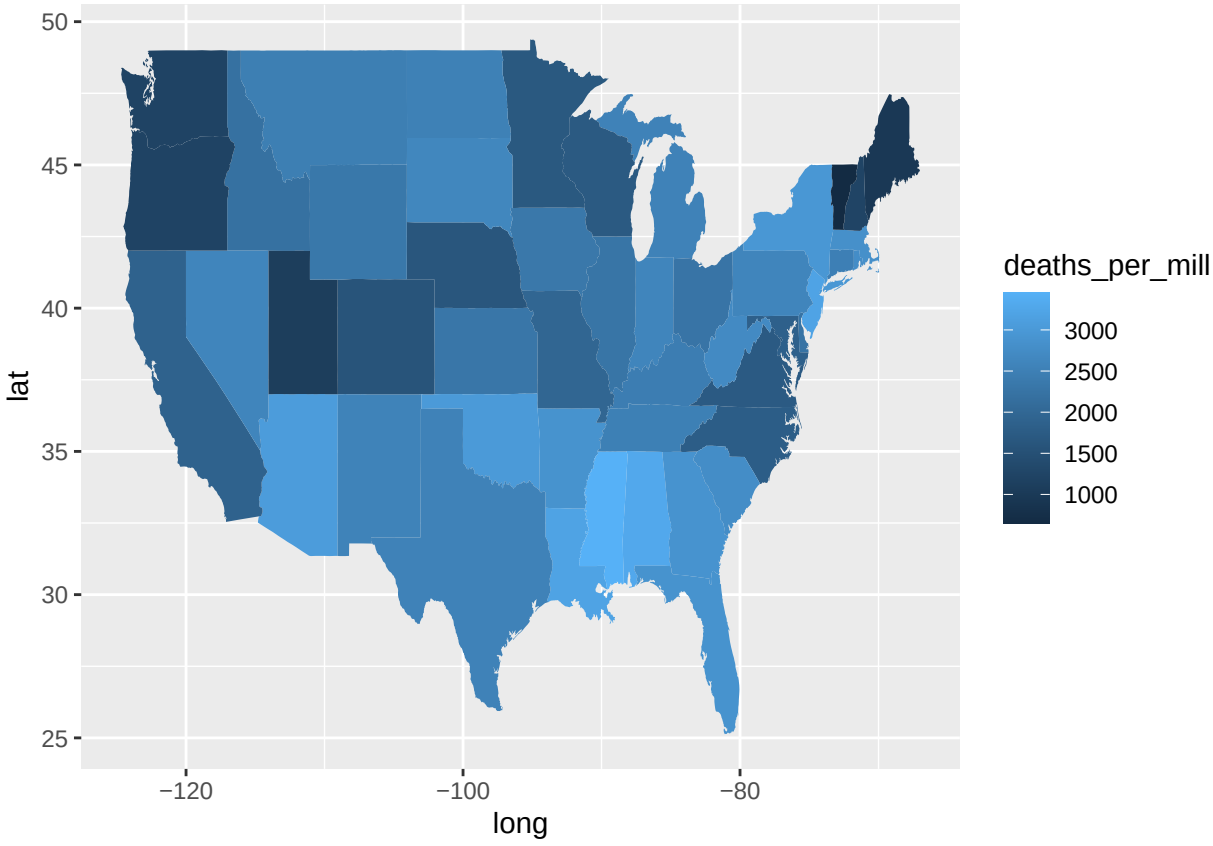
Choropleth map

To reflect differences by state, we use a choropleth map based on the latest date available and display the cases per million

```
US_states_latest <- US_by_state %>%
  filter(date == max(date)) %>%
  mutate(Province_State = tolower(Province_State))
##Get map data for US states
MainStates <- map_data("state")
MainStates <- as_tibble(MainStates)
MainStates <- MainStates %>%
  rename(Province_State = region)


US_Map_states_cases <- US_states_latest %>%
  inner_join(MainStates, by = "Province_State")

US_Map_states_cases %>%
  ggplot(data=US_Map_states_cases, mapping=aes(x=long, y=lat)) +
  geom_polygon(mapping=aes(x=long, y=lat,fill=deaths_per_mill, group=group))
```



Modeling deaths vs cases on data of the US

Linear regression

Now we want to develop a linear model to predict deaths depending on observed cases

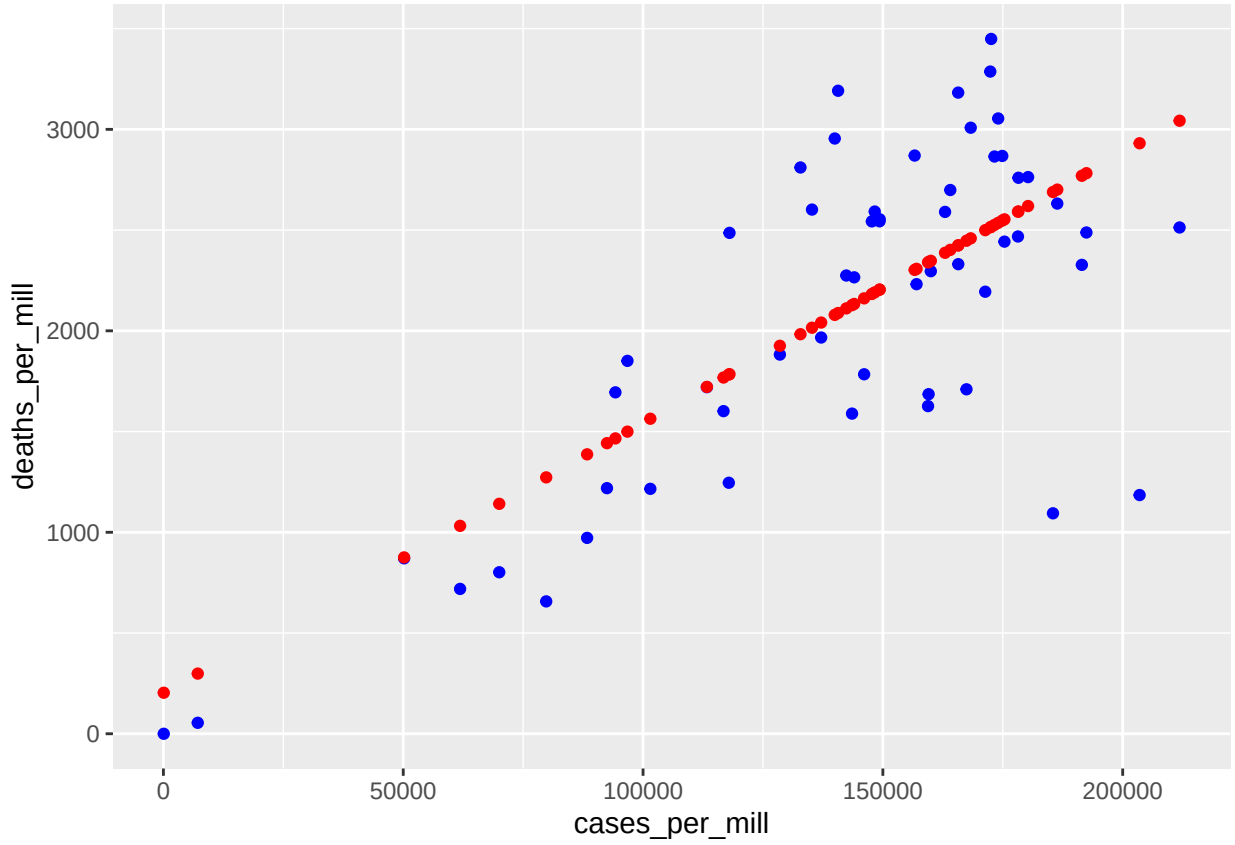
```
US_states_latest <- US_states_latest %>% filter(!(Population == 0))
mod <- lm(deaths_per_mill ~ cases_per_mill, data = US_states_latest)
print(summary(mod))
```

```
##
## Call:
## lm(formula = deaths_per_mill ~ cases_per_mill, data = US_states_latest)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1746.13  -319.64   -47.59   349.70  1103.10
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    202.70894  251.43003   0.806    0.424
## cases_per_mill   0.01341    0.00170   7.886 1.51e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 563.6 on 54 degrees of freedom
## Multiple R-squared:  0.5352, Adjusted R-squared:  0.5266
## F-statistic: 62.18 on 1 and 54 DF,  p-value: 1.509e-10
```

```
#US_states_latest %>% slice_max(cases_per_mill)

x_grid <- seq(1, 250000)
new_df <- tibble(cases_per_mill = x_grid)
US_states_pred <- US_states_latest %>% mutate(pred = predict(mod))
US_states_pred %>% ggplot() + geom_point(aes(x = cases_per_mill, y = deaths_per_mill),color="blue") + g
```



Multivariate regression

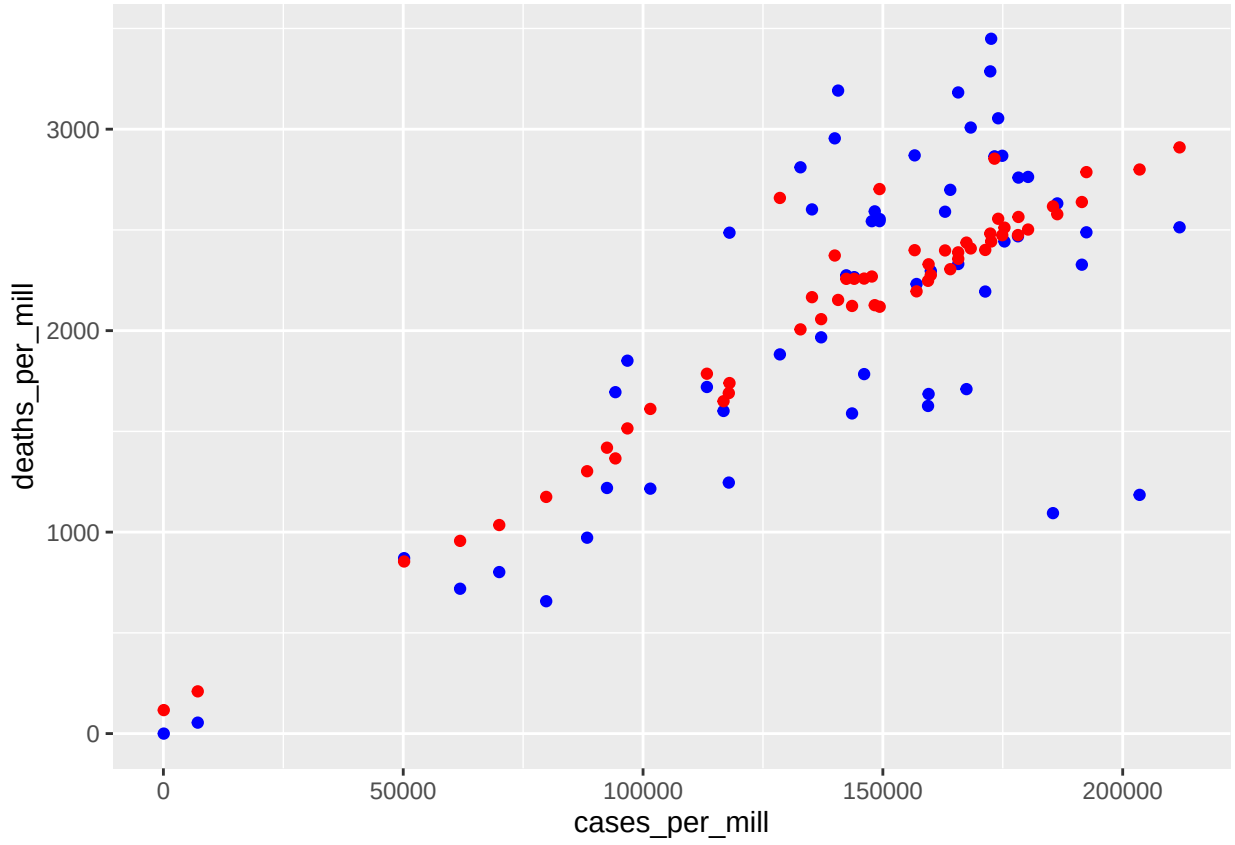
As a second model we want to test whether population size has an impact. We use now a multivariate regression.

```
mod2 <- lm(deaths_per_mill ~ cases_per_mill + Population, data = US_states_latest)
print(summary(mod2))
```

```
##
## Call:
## lm(formula = deaths_per_mill ~ cases_per_mill + Population, data = US_states_latest)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1614.98  -301.74     1.01   393.95  1039.14
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.145e+02  2.472e+02   0.463   0.6452
## cases_per_mill 1.312e-02  1.653e-03   7.934 1.43e-10 ***
## Population     2.173e-05  1.024e-05   2.121   0.0386 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 546.2 on 53 degrees of freedom
## Multiple R-squared:  0.5716, Adjusted R-squared:  0.5554
## F-statistic: 35.36 on 2 and 53 DF,  p-value: 1.755e-10
```

```
US_states_pred2 <- US_states_latest %>% mutate(pred = predict(mod2))
US_states_pred2 %>% ggplot() + geom_point(aes(x = cases_per_mill, y = deaths_per_mill),color="blue") + g
```



Conclusion

Deaths correlate with cases reported and the inclusion of the absolute population size improves the prediction slightly (higher adjusted R-squared). The reason therefore might be seen in the fact that locations with higher populations exhibit more contacts for each individual and hence a higher chance to get sick severely.

Bias

Be it for reported cases as well as for reported deaths one needs to be aware that there might be a bias on how data is collected and measured. How much tests per one million persons have been conducted, and how regular where those tests? Are deaths which where caused by COVID treated differently from deaths where the patient died from another disease while being COVID positive as well? I would expect that testing and reporting related to COVID differs from state to state, and also between cities and rural areas. Same is also true for the treatment of patients. How much medication was available and used? And what about the infrastructure? Availability of ICUs and their personal are definitely not equal across the entire US. Given that there are potentially many differences in reporting and also in the quality of treatment, the outcome of the regression model needs to be interpreted accordingly - meaning that we need to expect data points not to align perfectly to the regression line as we compare a little bit apples with oranges.

```
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Switzerland.1252  LC_CTYPE=English_Switzerland.1252
## [3] LC_MONETARY=English_Switzerland.1252 LC_NUMERIC=C
## [5] LC_TIME=English_Switzerland.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] maps_3.4.0      lubridate_1.8.0 forcats_0.5.1   dplyr_1.0.7
##  [5] purrr_0.3.4     readr_2.1.0     tidyr_1.1.4     tibble_3.1.6
##  [9] ggplot2_3.3.5   tidyverse_1.3.1 stringr_1.4.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.7       assertthat_0.2.1 digest_0.6.28    utf8_1.2.2
##  [5] R6_2.5.1         cellranger_1.1.0 backports_1.3.0  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_0.7.10     compiler_4.1.1   modelr_0.1.8     xfun_0.27
## [25] pkgconfig_2.0.3  htmltools_0.5.2  tidyselect_1.1.1 fansi_0.5.0
## [29] crayon_1.4.2     tzdb_0.2.0       dbplyr_2.1.1     withr_2.4.2
## [33] grid_4.1.1       jsonlite_1.7.2   gtable_0.3.0     lifecycle_1.0.1
## [37] DBI_1.1.1        magrittr_2.0.1   scales_1.1.1     cli_3.1.0
## [41] stringi_1.7.5    vroom_1.5.6      farver_2.1.0     fs_1.5.0
## [45] xml2_1.3.2       ellipsis_0.3.2   generics_0.1.1   vctrs_0.3.8
## [49] tools_4.1.1      bit64_4.0.5      glue_1.4.2       hms_1.1.1
## [53] parallel_4.1.1   fastmap_1.1.0    yaml_2.2.1       colorspace_2.0-2
## [57] rvest_1.0.2      knitr_1.36       haven_2.4.3
```