JHU CSSE COVID-19 Dataset analysis

cloud-erik

21 5 2021

Question of interest

I would like to show if the development of the Covid-19 cases over time in absolute as well as per population is different for different countries.

Also I would address the question how the cases are globally distributed.

Libraries

Load required libraries

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- 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
```

```
library(timetk)
library(ggmap)
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
library(ggplot2)
library(PBSmapping)
```

```
##
## ------------------------------------------------------------
## PBS Mapping 2.73.0 -- Copyright (C) 2003-2021 Fisheries and Oceans Canada
##
## PBS Mapping comes with ABSOLUTELY NO WARRANTY;
## for details see the file COPYING.
## This is free software, and you are welcome to redistribute
## it under certain conditions, as outlined in the above file.
##
## A complete user guide 'PBSmapping-UG.pdf' is located at
## C:/Users/e.schneider.KLARENMUEHLE/Documents/R/win-library/4.0/PBSmapping/doc/PBSmapping-UG.pdf
##
## Packaged on 2021-01-12
## Pacific Biological Station, Nanaimo
##
## All available PBS packages can be found at
## https://github.com/pbs-software
##
## To see demos, type '.PBSfigs()'.
## ------------------------------------------------------------
```

```
library(maps)
```

```
##
## Attaching package: 'maps'

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

```
library(mapproj)
```

Data Import

First importing the COVID-19 Dataset as CSV file from JHU Github https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data.

The data is provided by Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE): https://systems.jhu.edu/

See also Lancet Article: "An interactive web-based dashboard to track COVID-19 in real time" https://doi.org/10.1016/S1473-3099(20)30120-1

Because the dataset does not include information about population which is vital for comparison between countries I also imported the population and population density from UN https://population.un.org/wpp/Download/Standard/CSV/.

```r
url_confirmed <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/cs
url_death <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_
url_recovered <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/cs

url_population <- "https://population.un.org/wpp/Download/Files/1_Indicators%20(Standard)/CSV_FILES/WPP

#check.names has to be FALSE because otherwise "/" will be replaced in Header names and could later not
covid_confirmed <- read.csv(url_confirmed, header = TRUE, check.names = FALSE)
covid_death <- read.csv(url_death, header = TRUE, check.names = FALSE)
covid_recovered <- read.csv(url_recovered, header = TRUE, check.names = FALSE)
population <- read.csv(url_population, header = TRUE, check.names = FALSE)
```

Data cleaning

Clean the data and join everything into one table. "transpose" with pivot_loger the dates into rows, drop the lat and long columns, make date a date format, give country column a valid name and drop the Province by sum it by country. Take population data only from last year and current variant "2". To merge data together I have to harmonize the country name because they are slightly different between the sources and what ggmap needs to plot it later on a map. At the end we have one table with all Covid cases, recoveries and death per country and day in absolute values and per population.

```r
covid_confirmed <- covid_confirmed %>%
  pivot_longer(cols = -c("Province/State", "Country/Region", "Lat", "Long"),
               names_to = "date", values_to = "confirmed") %>%
  mutate(date = mdy(date)) %>%
  select(-c(Lat, Long)) %>%
  rename(Country = "Country/Region") %>%
  group_by(Country, date) %>% summarise(confirmed = sum(confirmed))
```

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

```r
covid_death <- covid_death %>%
  pivot_longer(cols = -c("Province/State", "Country/Region", "Lat", "Long"),
               names_to = "date", values_to = "death") %>%
  mutate(date = mdy(date)) %>%
  select(-c(Lat, Long)) %>%
  rename(Country = "Country/Region") %>%
  group_by(Country, date) %>% summarise(death = sum(death))
```

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

```r
covid_recovered <- covid_recovered %>%
  pivot_longer(cols = -c("Province/State", "Country/Region", "Lat", "Long"),
               names_to = "date", values_to = "recovered") %>%
  mutate(date = mdy(date)) %>%
  select(-c(Lat, Long)) %>%
  rename(Country = "Country/Region") %>%
  group_by(Country, date) %>% summarise(recovered = sum(recovered))
```

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

```r
# Join together in covid_data
covid_data <- merge.data.frame(covid_confirmed, covid_death)
covid_data <- merge.data.frame(covid_data, covid_recovered)

# rename Countries to be able to merge the data later together
# rename "US" in JHU Data with correct value "USA".
covid_data[["Country"]][covid_data[["Country"]]=="US"] = "USA"
#rename "Czechia" with "Czech Republic"
covid_data[["Country"]][covid_data[["Country"]]=="Czechia"] = "Czech Republic"
#rename "United Kingdom" with "UK"
covid_data[["Country"]][covid_data[["Country"]]=="United Kingdom"] = "UK"
covid_data[["Country"]][covid_data[["Country"]]=="Taiwan*"] = "Taiwan"
covid_data[["Country"]][covid_data[["Country"]]=="Korea*"] = "South Korea"

population <- population %>%
  filter(VarID == "2", Time == year(Sys.Date())-1) %>%
  select(-c(LocID, VarID, Variant, Time, MidPeriod, PopMale, PopFemale)) %>%
  rename(Country = "Location")

# rename Countries to be able to merge the data later together
#rename "United Kingdom" with "UK"
population[["Country"]][population[["Country"]]=="United Kingdom"] = "UK"
#rename "United States of America" with "USA"
population[["Country"]][population[["Country"]]=="United States of America"] = "USA"
#rename "Russian Federation" with "Russia"
population[["Country"]][population[["Country"]]=="Russian Federation"] = "Russia"
#rename "Iran (Islamic Republic of)" with "Iran"
population[["Country"]][population[["Country"]]=="Iran (Islamic Republic of)"] = "Iran"
population[["Country"]][population[["Country"]]=="Czechia"] = "Czech Republic"
#rename "Brunei Darussalam" with "Brunei"
population[["Country"]][population[["Country"]]=="Brunei Darussalam"] = "Brunei"
#rename "Bolivia (Plurinational State of)" with "Bolivia"
population[["Country"]][population[["Country"]]=="Bolivia (Plurinational State of)"] = "Bolivia"
#rename "CÃ´te d'Ivoire" with "Cote d'Ivoire"
population[["Country"]][population[["Country"]]=="CÃ´te d'Ivoire"] = "Cote d'Ivoire"
#rename "Syrian Arab Republic" with "Syria"
population[["Country"]][population[["Country"]]=="Syrian Arab Republic"] = "Syria"
#rename "Venezuela (Bolivarian Republic of)" with "Venezuela"
population[["Country"]][population[["Country"]]=="Venezuela (Bolivarian Republic of)"] = "Venezuela"
#rename "Viet Nam" with "Vietnam"
population[["Country"]][population[["Country"]]=="Viet Nam"] = "Vietnam"
#rename "Viet Nam" with "Vietnam"
population[["Country"]][population[["Country"]]=="Viet Nam"] = "Vietnam"
population[["Country"]][population[["Country"]]=="United Republic of Tanzania"] = "Tanzania"
population[["Country"]][population[["Country"]]=="China, Taiwan Province of China"] = "Taiwan"
population[["Country"]][population[["Country"]]=="Republic of Moldova"] = "Moldova"
population[["Country"]][population[["Country"]]=="Republic of Korea"] = "South Korea"
#population[["Country"]][population[["Country"]]=="Democratic Republic of the Congo"] = "Republic of Co



# Join together covid_data with population
covid_data <- merge.data.frame(covid_data, population, all.covid_data = TRUE)
```

```
# Add cases per population columns
covid_data <- covid_data %>%
  mutate(conf_per_pop = confirmed / PopTotal,
         death_per_pop = death / PopTotal,
         rec_per_pop = recovered / PopTotal)

summary(covid_data)
```

```
##    Country               date              confirmed            death
##  Length:90712       Min.   :2020-01-22   Min.   :       0   Min.   :     0.0
##  Class :character   1st Qu.:2020-05-24   1st Qu.:     262   1st Qu.:     3.0
##  Mode  :character   Median :2020-09-24   Median :    6352   Median :   108.5
##                     Mean   :2020-09-24   Mean   :  286254   Mean   :  6819.1
##                     3rd Qu.:2021-01-25   3rd Qu.:   87198   3rd Qu.:  1557.0
##                     Max.   :2021-05-28   Max.   :33239963   Max.   :593963.0
##    recovered           PopTotal           PopDensity        conf_per_pop
##  Min.   :       0   Min.   :      0.8   Min.   :    2.11   Min.   :  0.00000
##  1st Qu.:      61   1st Qu.:   2082.3   1st Qu.:   35.92   1st Qu.:  0.06695
##  Median :    3433   Median :   9227.9   Median :   88.59   Median :  1.04981
##  Mean   :  168001   Mean   :  40947.2   Mean   :  361.80   Mean   : 10.37830
##  3rd Qu.:   51739   3rd Qu.:  28611.2   3rd Qu.:  207.61   3rd Qu.:  9.57409
##  Max.   :25178011   Max.   :1439323.8   Max.   :26338.26   Max.   :177.22125
##  death_per_pop        rec_per_pop
##  Min.   :0.0000000   Min.   :  0.00000
##  1st Qu.:0.0004138   1st Qu.:  0.01825
##  Median :0.0182855   Median :  0.60251
##  Mean   :0.1988414   Mean   :  7.71271
##  3rd Qu.:0.1649606   3rd Qu.:  5.40096
##  Max.   :3.0725595   Max.   :173.63619
```

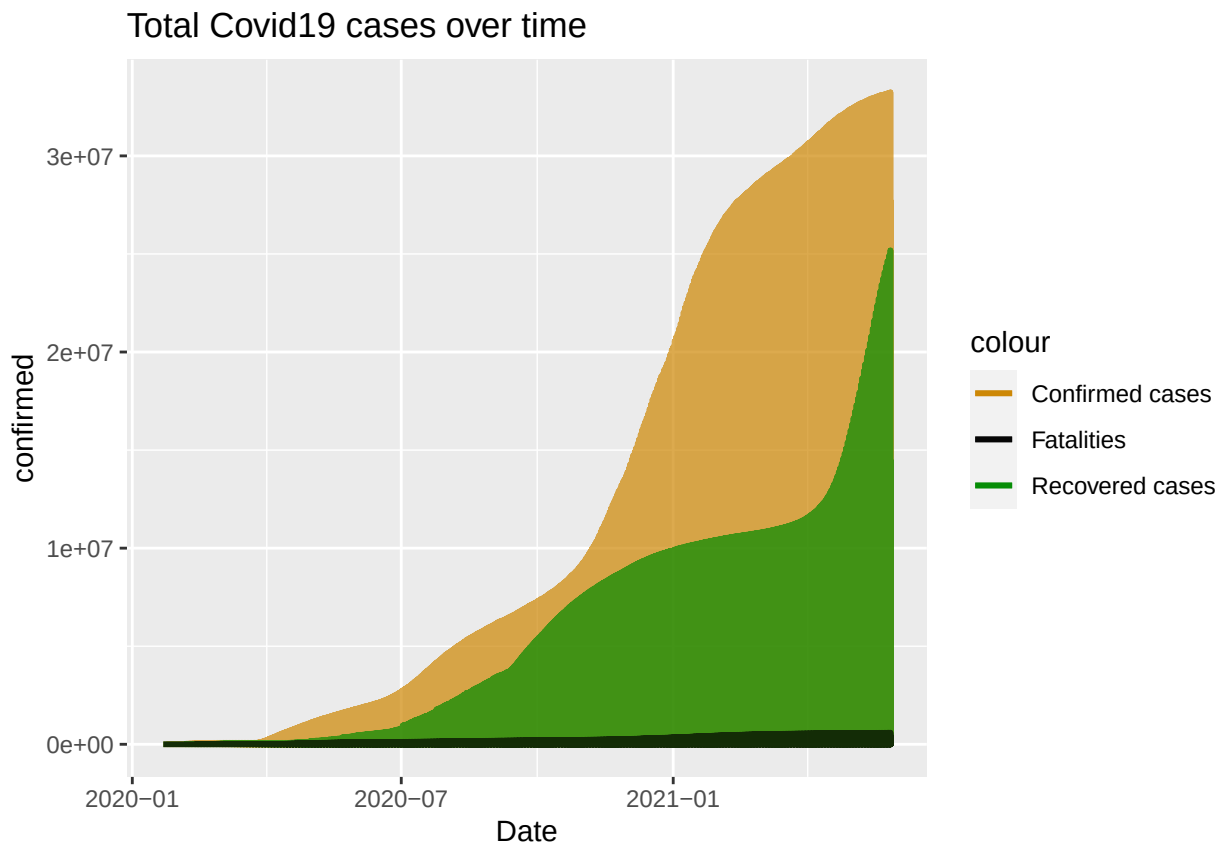
Development over time

Ee will have a look on the development over the time.

Absolut cases

First plot a time line of total cases, recovered and Fatalities.

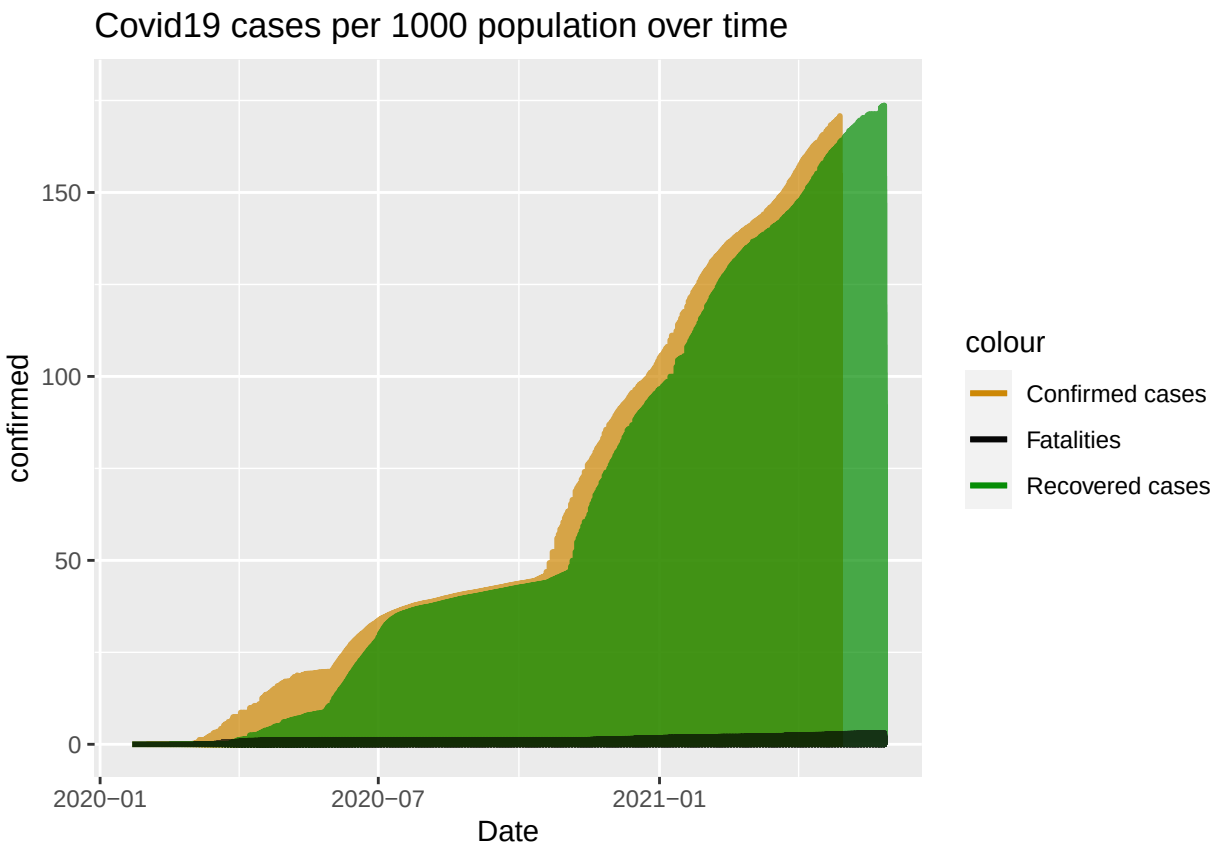
```
colors <- c("Confirmed cases" = "orange3",
            "Recovered cases" = "green4",
            "Fatalities" = "black",
            "Predicted Recovered cases" = "gray")
p <- ggplot(covid_data, aes(x = date, y = confirmed)) +
  geom_line(size=1, alpha=0.7, linetype=1, aes(y = confirmed, color="Confirmed cases")) +
  geom_line(size=1, alpha=0.7, linetype=1, aes(y = recovered, color="Recovered cases")) +
  geom_line(size=1, alpha=0.7, linetype=1, aes(y = death, color="Fatalities")) +
  labs(x="Date") +
  ggtitle("Total Covid19 cases over time")+
  scale_color_manual(values = colors)
p
```



Cases per population

Compare to the time line of cases per population

```
p <- ggplot(covid_data, aes(x = date, y = confirmed)) +
  geom_line(size=1, alpha=0.7, linetype=1, aes(y = conf_per_pop, color="Confirmed cases")) +
  geom_line(size=1, alpha=0.7, linetype=1, aes(y = rec_per_pop, color="Recovered cases")) +
  geom_line(size=1, alpha=0.7, linetype=1, aes(y = death_per_pop, color="Fatalities")) +
  labs(x="Date") +
  ggtitle("Covid19 cases per 1000 population over time")+
  scale_color_manual(values = colors)

p
```



Model to predict Cases per population

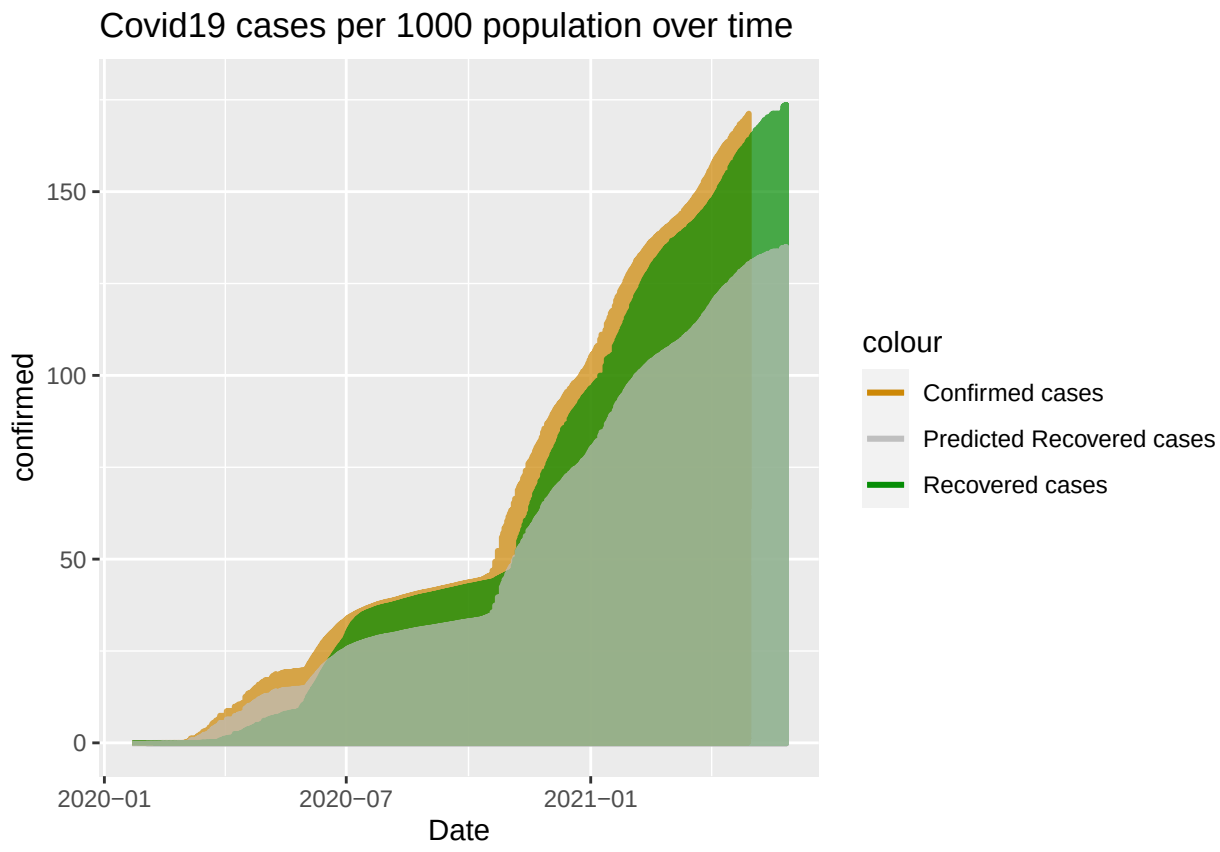
It looks like these is a strong relationship between the confirmed and recovered cases per population over time. Of lets check how good a linear model performs in predicting this.

```
# create model
cases_ppop.model <- lm(rec_per_pop~conf_per_pop, data=covid_data)

# Show summary
summary(cases_ppop.model)
```

```
##
## Call:
## lm(formula = rec_per_pop ~ conf_per_pop, data = covid_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -80.475   0.136   0.201   0.516  38.695
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.201390   0.027733  -7.262 3.85e-13 ***
## conf_per_pop  0.762562   0.001186 642.897  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 7.485 on 90710 degrees of freedom
## Multiple R-squared:   0.82,  Adjusted R-squared:   0.82
## F-statistic: 4.133e+05 on 1 and 90710 DF,  p-value: < 2.2e-16
```

```
# Get slope an dintercept of model
slope = as.numeric(cases_ppop.model$coefficients[2])
intercept = as.numeric(cases_ppop.model$coefficients[1])

# add predictaed data to the
covid_data <- covid_data %>%
  mutate(pred_rec_per_pop = intercept + ( slope * conf_per_pop ))

# Plot Data with the model
p <- ggplot(covid_data, aes(x = date, y = confirmed)) +
  geom_line(size=1, alpha=0.7, linetype=1, aes(y = conf_per_pop, color="Confirmed cases")) +
  geom_line(size=1, alpha=0.7, linetype=1, aes(y = rec_per_pop, color="Recovered cases")) +
  geom_line(size=1, alpha=0.7, linetype=1, aes(y = pred_rec_per_pop, color="Predicted Recovered cases")) +
  labs(x="Date") +
  ggtitle("Covid19 cases per 1000 population over time")+
  scale_color_manual(values = colors)

p
```

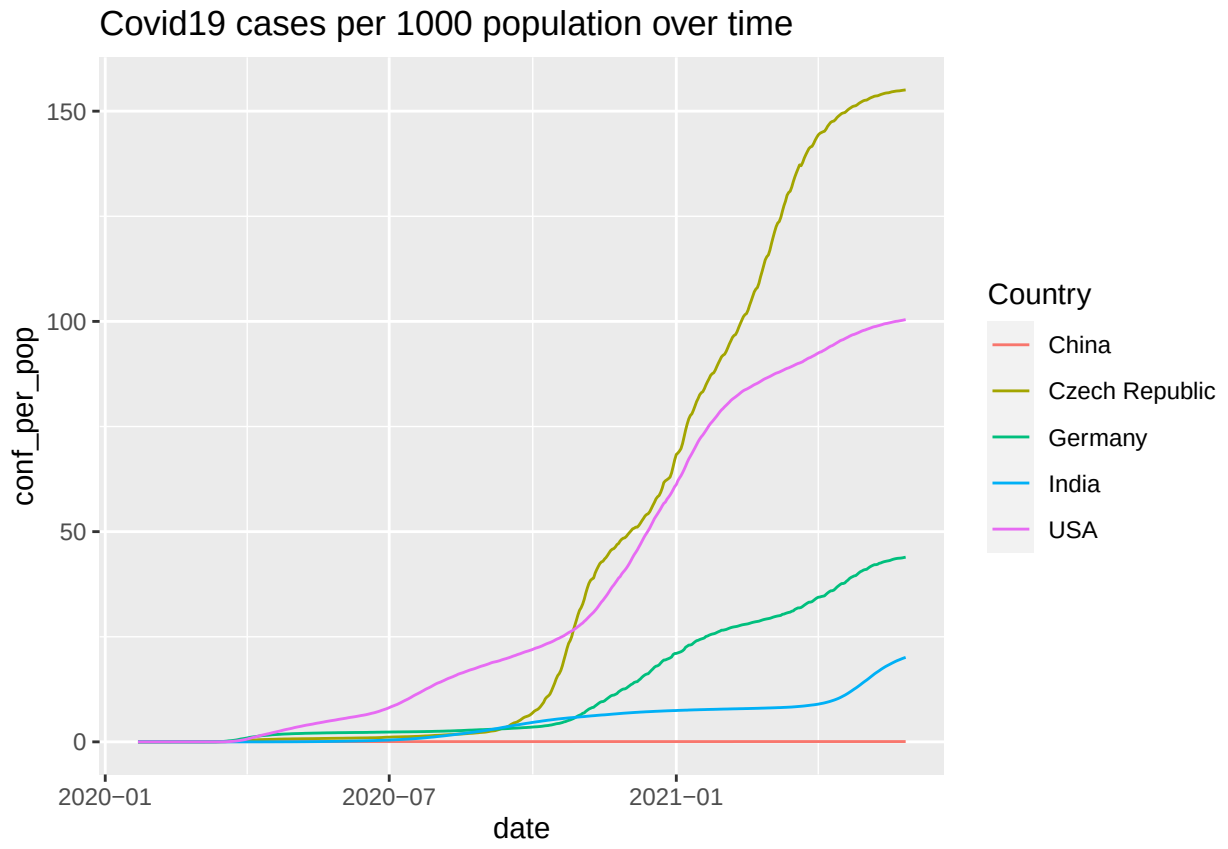


Prediction looks not to bad even if the for lower number of cases its predicting to high and for higher values to low values.

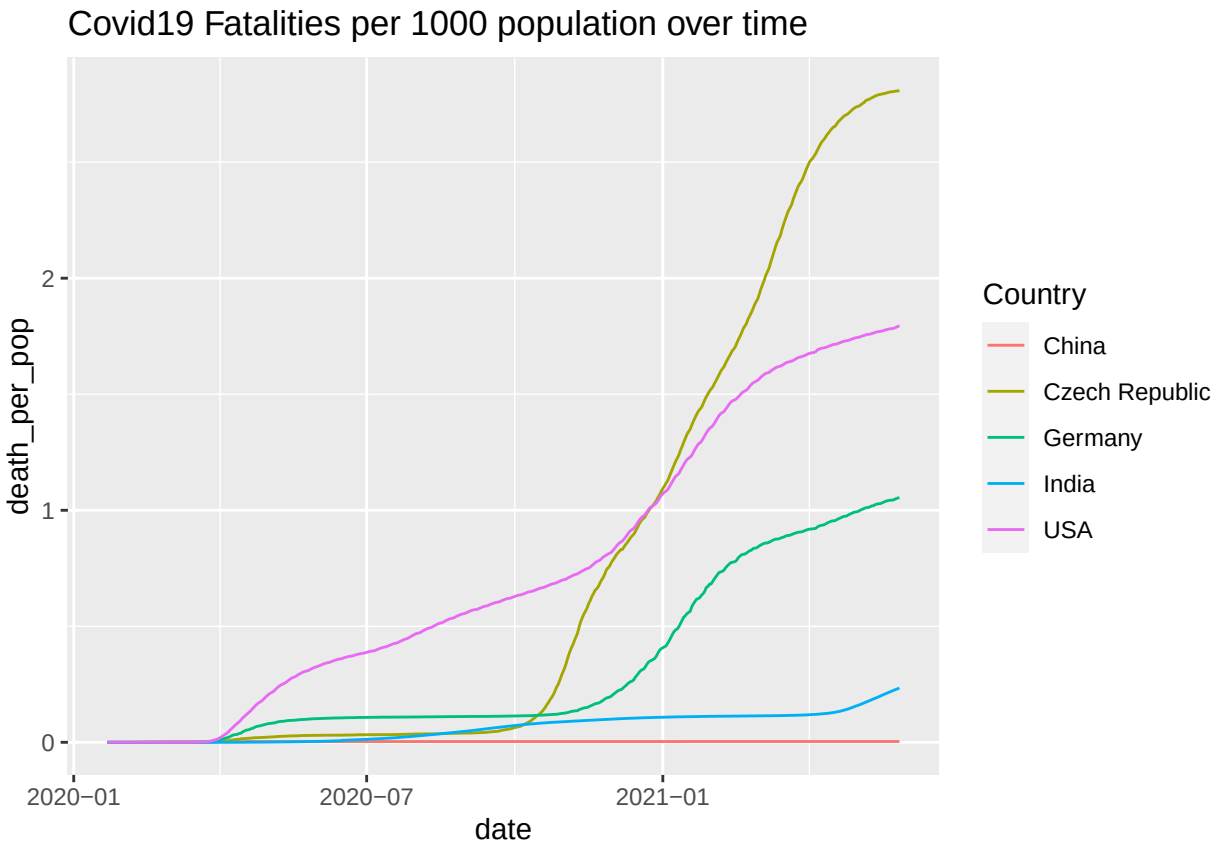
Cases per population of different countries

Lets compare the development of Covid 19 cases and fatalities per population for some countries.

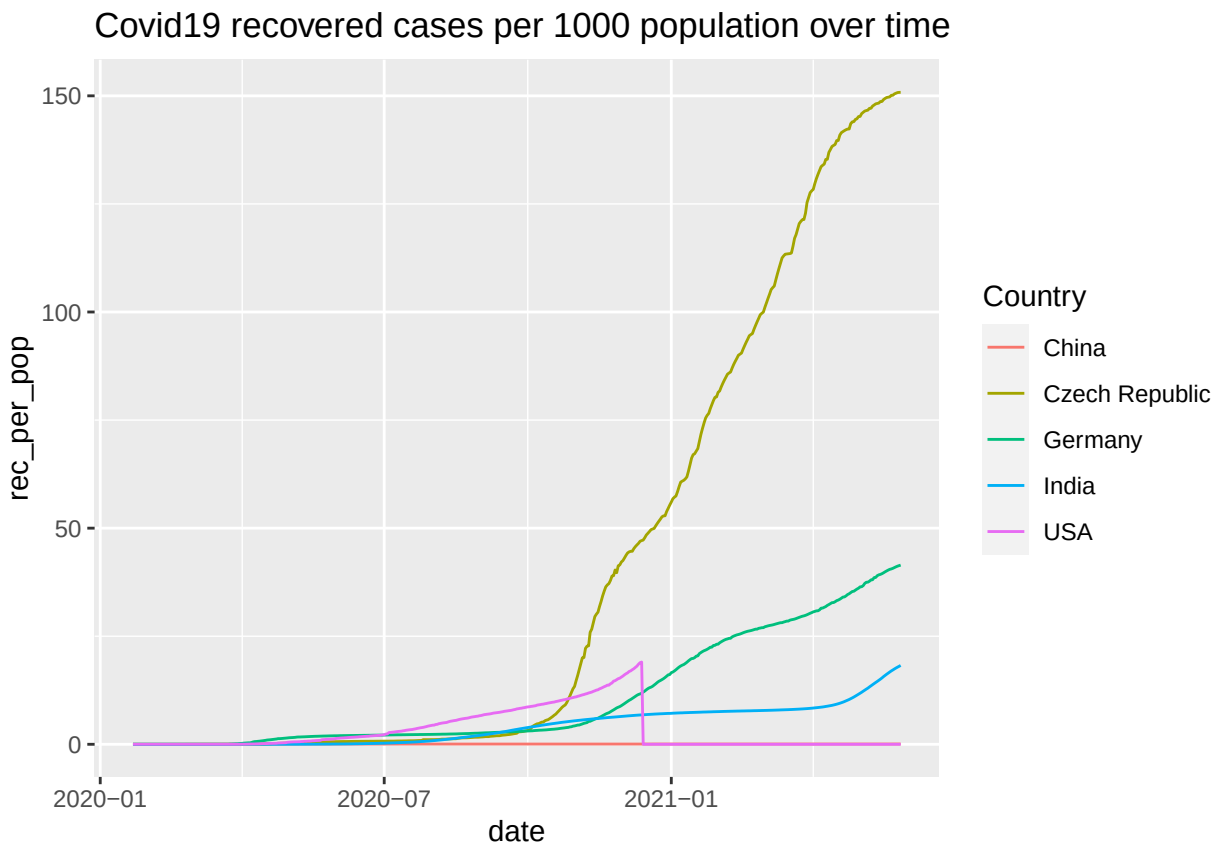
```
covid_US_CN <- covid_data %>%
  filter(Country =="USA" |
           Country == "China" |
           Country == "India" |
           Country == "Czech Republic" |
           Country=="Germany")

p1 <- ggplot(covid_US_CN, aes(x = date, y = conf_per_pop, colour = Country)) +
  geom_line() +
  ggtitle("Covid19 cases per 1000 population over time")

p1
```




```
p2 <- ggplot(covid_US_CN, aes(x = date, y = death_per_pop, colour = Country)) +
  geom_line() +
  ggtitle("Covid19 Fatalities per 1000 population over time")

p2
```




```
p3 <- ggplot(covid_US_CN, aes(x = date, y = rec_per_pop, colour = Country)) +
  geom_line() +
  ggtitle("Covid19 recovered cases per 1000 population over time")

p3
```



As we see, the ration between cases and fatalities are for the countries comparable, even on a different scale.

Global number of cases

Next part is to have an overview of the global geographical distribution of cases.

Cases per population

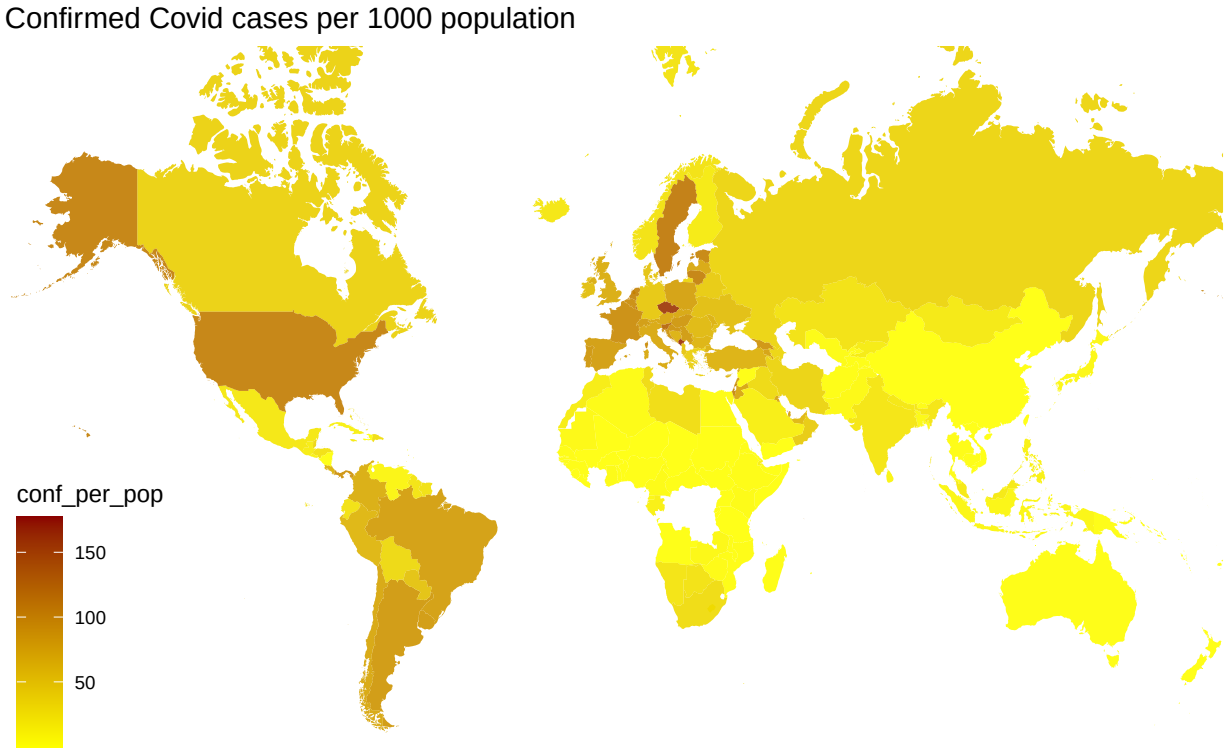
Plot a map with confirmed cases

```
# create new data frame with sum per region (Country)
covid_region <- covid_data %>%
  rename(region = "Country") %>%
  group_by(region) %>% summarise(confirmed = max(confirmed),
                                 death = max(death),
                                 recovered = max(recovered),
                                 PopTotal = max(PopTotal),
                                 PopDensity = max(PopDensity)) %>%
  mutate(conf_per_pop = confirmed / PopTotal,
         death_per_pop = death / PopTotal,
         rec_per_pop = recovered / PopTotal)

myLocation<-c(-180, -60, 179, 85) # World
nMap <- get_stamenmap(bbox=myLocation, maptype="toner-lite", zoom=2)
```

```
## Source : http://tile.stamen.com/toner-lite/2/0/0.png

## Source : http://tile.stamen.com/toner-lite/2/1/0.png

## Source : http://tile.stamen.com/toner-lite/2/2/0.png

## Source : http://tile.stamen.com/toner-lite/2/3/0.png

## Source : http://tile.stamen.com/toner-lite/2/0/1.png

## Source : http://tile.stamen.com/toner-lite/2/1/1.png

## Source : http://tile.stamen.com/toner-lite/2/2/1.png

## Source : http://tile.stamen.com/toner-lite/2/3/1.png

## Source : http://tile.stamen.com/toner-lite/2/0/2.png

## Source : http://tile.stamen.com/toner-lite/2/1/2.png

## Source : http://tile.stamen.com/toner-lite/2/2/2.png

## Source : http://tile.stamen.com/toner-lite/2/3/2.png
```

```
#get country polygon data
mapdata <- map_data("world")
mapdata <- left_join(mapdata, covid_region, by="region")

#get bounding box for map
bb<-attr(nMap, "bb");
ylim<-c(bb$ll.lat, bb$ur.lat)
xlim<-c(bb$ll.lon, bb$ur.lon)

#clip polygons to map
colnames(mapdata)[1:6] <- c("X","Y","PID","POS","region","subregion")
mapdata<-clipPolys(mapdata, xlim=xlim, ylim=ylim, keepExtra=TRUE)

#plot map Confirmed Covid cases per 1000 population
ggmap(nMap)+coord_map(xlim=xlim,ylim=ylim) +
    geom_polygon(data=mapdata, aes(x=X, y=Y, group=PID, fill=conf_per_pop), alpha=0.9) +
    ggthemes::theme_map() +
    ggtitle("Confirmed Covid cases per 1000 population") +
    scale_fill_gradient(low = "yellow", high = "red4", na.value = NA)
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```

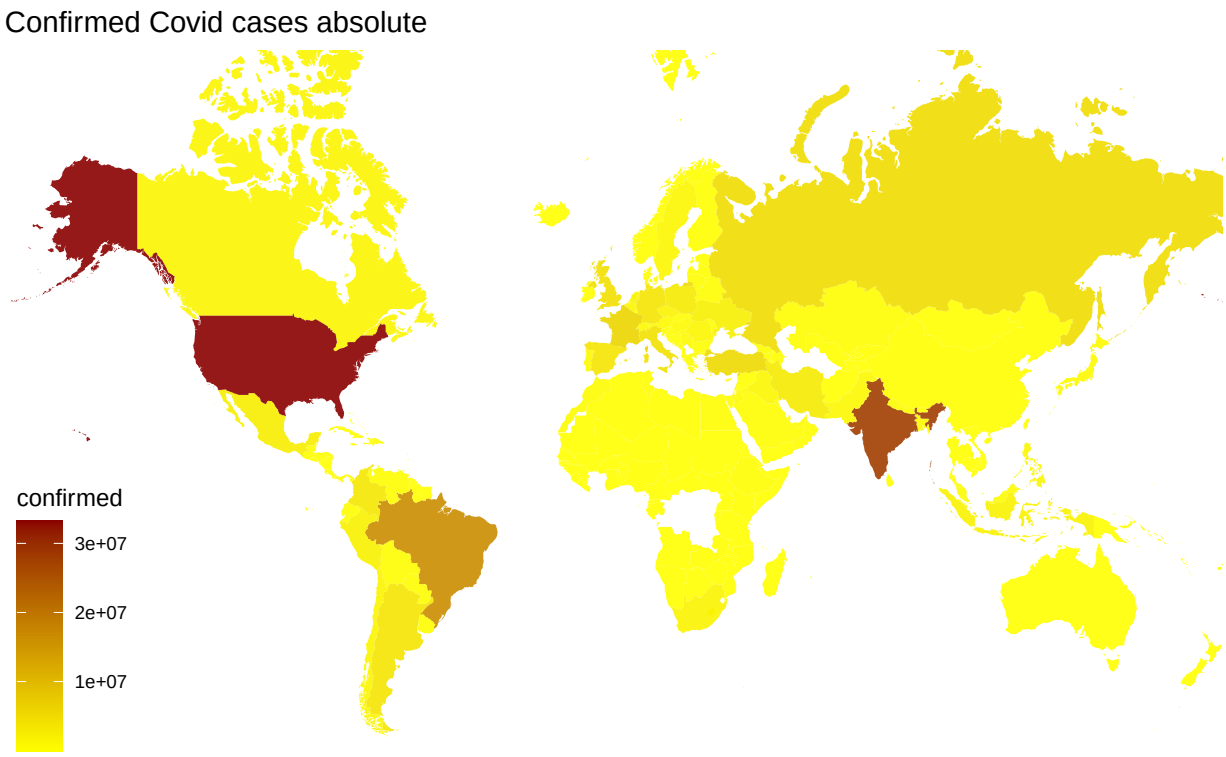


Absolut number of Cases

Just to show the difference let's plot the same map with absolute number of cases, what could lead to wrong or distorted conclusion. The difference is clearly visible for e.g. USA or India which have a lot of Covid cases because of there big population and Czech Republic or Sweden the opposite.

```
#plot map Confirmed Covid cases absolute
ggmap(nMap)+coord_map(xlim=xlim,ylim=ylim) +
    geom_polygon(data=mapdata, aes(x=X, y=Y, group=PID, fill=confirmed), alpha=0.9) +
    ggthemes::theme_map() +
    ggtitle("Confirmed Covid cases absolute") +
    scale_fill_gradient(low = "yellow", high = "red4", na.value = NA)
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```



Fatalities per population

Plot a map with fatalities

```
#plot map Death Covid cases per 1000 population
ggmap(nMap)+coord_map(xlim=xlim,ylim=ylim) +
    geom_polygon(data=mapdata, aes(x=X, y=Y, group=PID, fill=death_per_pop), alpha=0.9) +
    ggthemes::theme_map() +
    ggtitle("Covid Fatalities per 1000 population") +
    scale_fill_gradient(low = "grey", high = "black", na.value = NA)
```

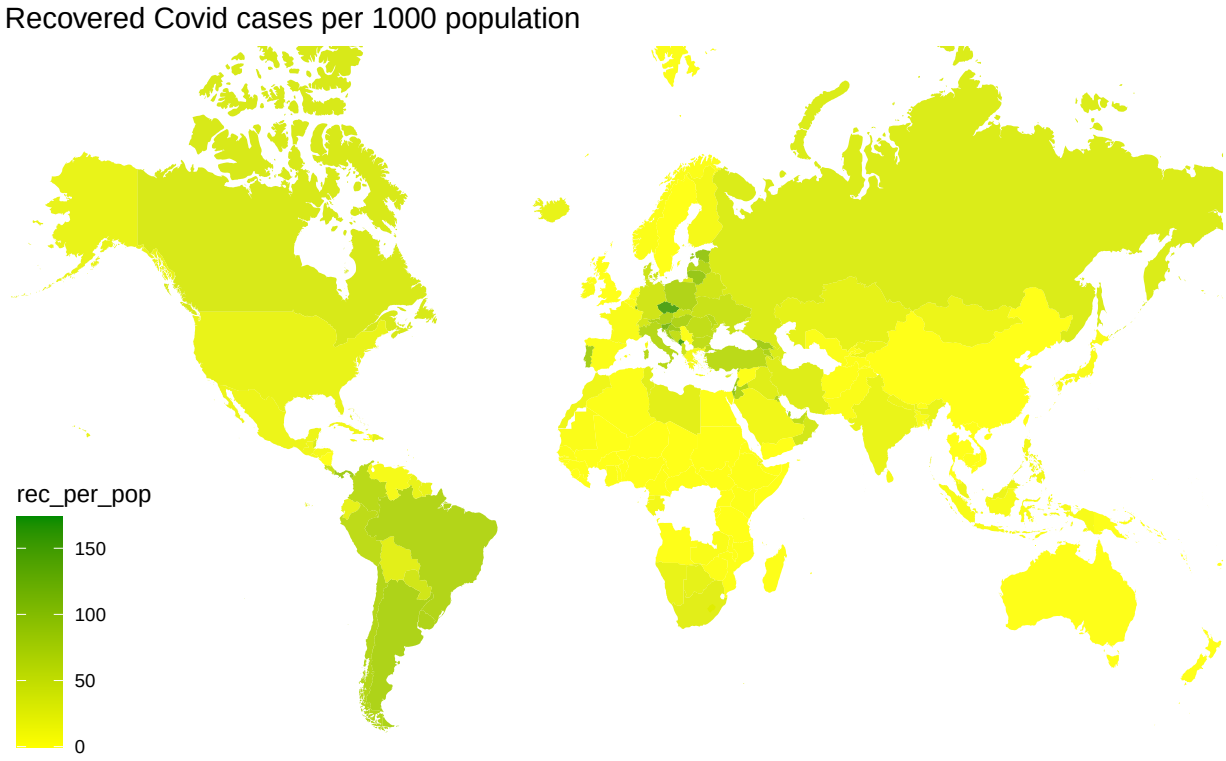
```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```

Recovered cases per population

Plot a map with recovered cases

```
#plot map Recovered Covid cases per 1000 population
ggmap(nMap)+coord_map(xlim=xlim,ylim=ylim) +
    geom_polygon(data=mapdata, aes(x=X, y=Y, group=PID, fill=rec_per_pop), alpha=0.9) +
    ggthemes::theme_map() +
    ggtitle("Recovered Covid cases per 1000 population") +
    scale_fill_gradient(low = "yellow", high = "green4", na.value = NA)
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```



Bias

There is plenty room of bias within the covid 19 data. First its collected all over the world from different organizations which could lead to offset and errors. Also the figures strongly depend on the infrastructure and the willing within a specific region to collect the data. e.g. there must be sufficient test capacity available and the people must be willing to be tested. Therefore its not reliable to compare e.g. figures from highly developed countries with huge testing capacity with countries where almost no health care system is in place. In some areas also political effects influence the data like wars, suppression or if no data is shared. And of course the values are absolute and have to be weighted or set in correlation of the population of the country. What was here done by merging population data from UN into the dataset.

Conclusion

The global distribution of Covid-19 cases is in absolute as well as in weighted cases per population very different. There are several reasons for that including different health care systems and testing capacity, political causes and bias in data collection. On the other hand the ration between Cases, recovered cases is comparable even on a different scale. On the time-line clearly the is a close relationship between the cases and recovered cases are visible. Also the created simple linear model performs quite good in predicting the recovered cases per population.

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
```

```
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=German_Germany.1252  LC_CTYPE=German_Germany.1252
## [3] LC_MONETARY=German_Germany.1252 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] mapproj_1.2.7     maps_3.3.0        PBSmapping_2.73.0 ggmap_3.0.0
##  [5] timetk_2.6.1      lubridate_1.7.10  forcats_0.5.1     stringr_1.4.0
##  [9] dplyr_1.0.5       purrr_0.3.4       readr_1.4.0       tidyr_1.1.3
## [13] tibble_3.0.5      ggplot2_3.3.3     tidyverse_1.3.0
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-6        fs_1.5.0            xts_0.12.1
##  [4] httr_1.4.2          tools_4.0.3         backports_1.2.1
##  [7] R6_2.5.0            rpart_4.1-15        DBI_1.1.1
## [10] colorspace_2.0-0    nnet_7.3-14         withr_2.4.1
## [13] sp_1.4-5            tidyselect_1.1.0    curl_4.3
## [16] compiler_4.0.3      cli_2.2.0           rvest_1.0.0
## [19] xml2_1.3.2          labeling_0.4.2      scales_1.1.1
## [22] digest_0.6.27       rmarkdown_2.7       jpeg_0.1-8.1
## [25] pkgconfig_2.0.3     htmltools_0.5.1.1   parallelly_1.25.0
## [28] dbplyr_2.1.0        ggthemes_4.2.4      rlang_0.4.10
## [31] readxl_1.3.1        rstudioapi_0.13     farver_2.1.0
## [34] generics_0.1.0      zoo_1.8-9           jsonlite_1.7.2
## [37] magrittr_2.0.1      Matrix_1.2-18       Rcpp_1.0.6
## [40] munsell_0.5.0       fansi_0.4.2         lifecycle_1.0.0
## [43] furrr_0.2.2         stringi_1.5.3       yaml_2.2.1
## [46] MASS_7.3-53         plyr_1.8.6          recipes_0.1.16
## [49] grid_4.0.3          parallel_4.0.3      listenv_0.8.0
## [52] crayon_1.3.4        lattice_0.20-41     haven_2.3.1
## [55] splines_4.0.3       hms_1.0.0           knitr_1.30
## [58] pillar_1.4.7        rjson_0.2.20        codetools_0.2-16
## [61] reprex_1.0.0        glue_1.4.2          evaluate_0.14
## [64] rsample_0.1.0       modelr_0.1.8        vctrs_0.3.6
## [67] png_0.1-7           RgoogleMaps_1.4.5.3 cellranger_1.1.0
## [70] gtable_0.3.0        future_1.21.0       assertthat_0.2.1
## [73] xfun_0.20           gower_0.2.2         prodlim_2019.11.13
## [76] broom_0.7.5         class_7.3-17        survival_3.2-7
## [79] timeDate_3043.102   lava_1.6.9          globals_0.14.0
## [82] ellipsis_0.3.1      ipred_0.9-11
```