HarvardX Data Science - Capstone - CYO Project

Mruthyum J Thatipamala

2021-02-03

Project Title: Germany COVID-19 - Regression Analysis

1. Introduction/Overview/Executive summary:

For DYO project, COVID-19 information of Germany is chosen to simulate the spread of the pandemic, analyze various regression models to fit its progress and predict the future of the disease adopting some machine learning algorithms.

For data purposes, two .csv files, namely "covid_de.csv", and "demographics_de.csv" are downloaded from Kaggle website.

The same can be found at - https://www.kaggle.com/headsortails/covid19-tracking-germany

The dataset "covid_de.csv" provides daily updated number of reported cases & deaths in Germany at state and county level. COVID-19 cases and deaths are provided on daily basis for a period of 225 days starting from March 23, 2019. The data is further fine tuned are at gender and agegroup levels also.

The dataser "demographics_de.csv" provides a general demographic data of Germany at state level.

Statical analyses conducted, regression models used and predictions algorithms implemented consists starting from linear models to progressing to non-linear models using the train sets and test sets. Results are visualized with appropriate graphs for better understanding of the data trends and outcomes of the analyses. Precision of every model is measured in terms of Residual Mean Square Error (RMSE) or Accuracy, as the case applicable, of actual cases and predicted numbers.

A comparision table is provided towards the end tabulating the precision number of each model. Also, some observations made during the analysis and recommendations for future work also documented.

NOTE: If you closely observe the data the cumulative daily cases are as high as the order 10 to the power of 5. Hence,to keep the calculated accuracy number less than 1.0 for better reading and understanding, RMSE/Accuracy is calculated after reducing the actual cases and predicted numbers per 10000 people respectively.

Step 1: Clean up heap memory and plots - Optimizing memory of environment

```
# Clear environment
rm(list = ls())
# Clear console
cat("\014")
```

```
# Clear plots
if(!is.null(dev.list())) dev.off()
```

```
## null device
##           1
```

```
#Supressing unharmful warnings
warning = FALSE
#Avoiding sceintific notation of scales in graphs and plots
options(scipen=10000)
```

Step 2: Installing packages and loading libraries

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
#if(!require(XQuartz)) install.packages("XQuartz", repos = "http://xquartz.macosforge.org")
if(!require(summarytools)) install.packages("summarytools", repos = "http://cran.us.r-project.org")

library(caret)
library(dplyr)
library(tidyr)
library(ggplot2)
library(lubridate)
library(randomForest)
library(rpart)
library(summarytools)
```

Step 3: Reading the data from CSV files and saving them to dataframes

```
#import data from csv file to a dataframe in Global Environment
coviddf <- read.csv(file = "covid_de.csv", head = TRUE, sep=",", stringsAsFactors = TRUE)
demographicsdf <- read.csv(file = "demographics_de.csv", head = TRUE, sep=",", stringsAsFactors = TRUE)
```

Step 4: First Look at data

```
#Describe data
nrow(coviddf)
```

```
## [1] 138724
```

```
names(coviddf)
```

```
## [1] "state"     "county"    "age_group" "gender"    "date"      "cases"
## [7] "deaths"    "recovered"
```

```
head(coviddf)
```

```
##                state             county age_group gender       date cases
## 1 Baden-Wuerttemberg LK Alb-Donau-Kreis     00-04      F 2020-03-27     1
## 2 Baden-Wuerttemberg LK Alb-Donau-Kreis     00-04      F 2020-03-28     1
## 3 Baden-Wuerttemberg LK Alb-Donau-Kreis     00-04      F 2020-04-03     1
## 4 Baden-Wuerttemberg LK Alb-Donau-Kreis     00-04      M 2020-04-05     1
## 5 Baden-Wuerttemberg LK Alb-Donau-Kreis     00-04      M 2020-05-18     1
## 6 Baden-Wuerttemberg LK Alb-Donau-Kreis     00-04      M 2020-07-27     1
##   deaths recovered
## 1      0         1
## 2      0         1
## 3      0         1
## 4      0         1
## 5      0         1
## 6      0         1
```

```
str(coviddf)
```

```
## 'data.frame':    138724 obs. of  8 variables:
##  $ state    : Factor w/ 16 levels "Baden-Wuerttemberg",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ county   : Factor w/ 413 levels "LK Ahrweiler",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ age_group: Factor w/ 6 levels "00-04","05-14",..: 1 1 1 1 1 1 1 1 1 2 ...
##  $ gender   : Factor w/ 2 levels "F","M": 1 1 1 2 2 2 2 2 2 1 ...
##  $ date     : Factor w/ 225 levels "2020-01-28","2020-01-29",..: 43 44 50 52 95 165 181 192 220 33 .
##  $ cases    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ deaths   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ recovered: int  1 1 1 1 1 1 1 1 0 1 ...
```

```
head(demographicsdf)
```

```
##                state gender age_group population
## 1 Baden-Wuerttemberg female     00-04     261674
## 2 Baden-Wuerttemberg female     05-14     490822
## 3 Baden-Wuerttemberg female     15-34    1293488
## 4 Baden-Wuerttemberg female     35-59    1919649
## 5 Baden-Wuerttemberg female     60-79    1182736
## 6 Baden-Wuerttemberg female     80-99     419471
```

```
names(demographicsdf)
```

```
## [1] "state"      "gender"     "age_group"  "population"
```

```
str(demographicsdf)
```

```
## 'data.frame':    192 obs. of  4 variables:
##  $ state     : Factor w/ 16 levels "Baden-Wuerttemberg",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ gender    : Factor w/ 2 levels "female","male": 1 1 1 1 1 1 2 2 2 2 ...
##  $ age_group : Factor w/ 6 levels "00-04","05-14",..: 1 2 3 4 5 6 1 2 3 4 ...
##  $ population: int  261674 490822 1293488 1919649 1182736 419471 274882 517387 1423367 1955828 ...
```

Column Descriptions:

1. COVID-19 dataset covid_de.csv: There are a total of 138724 observations of 8 variables. This data can be considered as a *Time Series* data as the measurements are made at evenly spaced time, i.e. daily.

- **state:** Name of the German federal state. Germany has 16 federal states.
- **county:** The name of the German Landkreis (LK) or Stadtkreis (SK), which correspond roughly to US counties.
- **age_group:** Data is reported at 6 age groups: 0-4, 5-14, 15-34, 35-59, 60-79, and "80-99" gender: Reported as male (M) or female (F).
- **date:** The calendar date of when a case, death and recovery were reported. It is in string format.
- **cases:** COVID-19 cases that have been confirmed counts per day, not cumulative counts.
- **deaths:** COVID-19 related deaths, not cumulative counts
- **recovered:** Recovered cases. Again, not cumulative counts

2. ***Demographic dataset demographics_de.csv:*** There are 192 observations of 4 variables

- **state, gender, age_group:** same as above.
- **population:** Population counts for the respective categories.

Predictive and Responsive variables:

A calculated intermediary variable **CUSUM_cases** (a cumulative sum of **cases** at daily level) is an effective number for better performance in modeling, regression, and forecasting using different models. Hence, this number is calculated where and when required and used as predictive variable.

The features, i.e. responsive variables, we will use to predict the outcome are ndays, age group and gender.

Step 5: Data Cleaning

Below code finds out the presence of NA values in any of the columns and deletes the corresponding rows.

```
#Check for na and delete rows with na values. Since the values are not cumulative, the overall numbers
any(is.na(coviddf))
```

```
## [1] TRUE
```

```
sum(is.na(coviddf))
```

```
## [1] 841
```

```
coviddf <- coviddf %>% na.omit()
```

Step 6: Data Wrangling and reshaping of the data - aka Data Munging

```
#Converting data from string to date format
coviddf$date <- as.Date(coviddf$date)
#Calculating time period in number of days
#The "Date" class means dates are stored as the number of days since January 1, 1970. The as.numeric fu

coviddf <- coviddf %>% mutate(ndays1970 = as.integer(as.ts(date)))
coviddf <- coviddf %>% mutate(ndays = (ndays1970 - min(ndays1970)))

#Summing the population to a single number at country level
demographicsdf_state_pop <- demographicsdf %>% group_by(state) %>% summarize(total_pop = sum(population
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

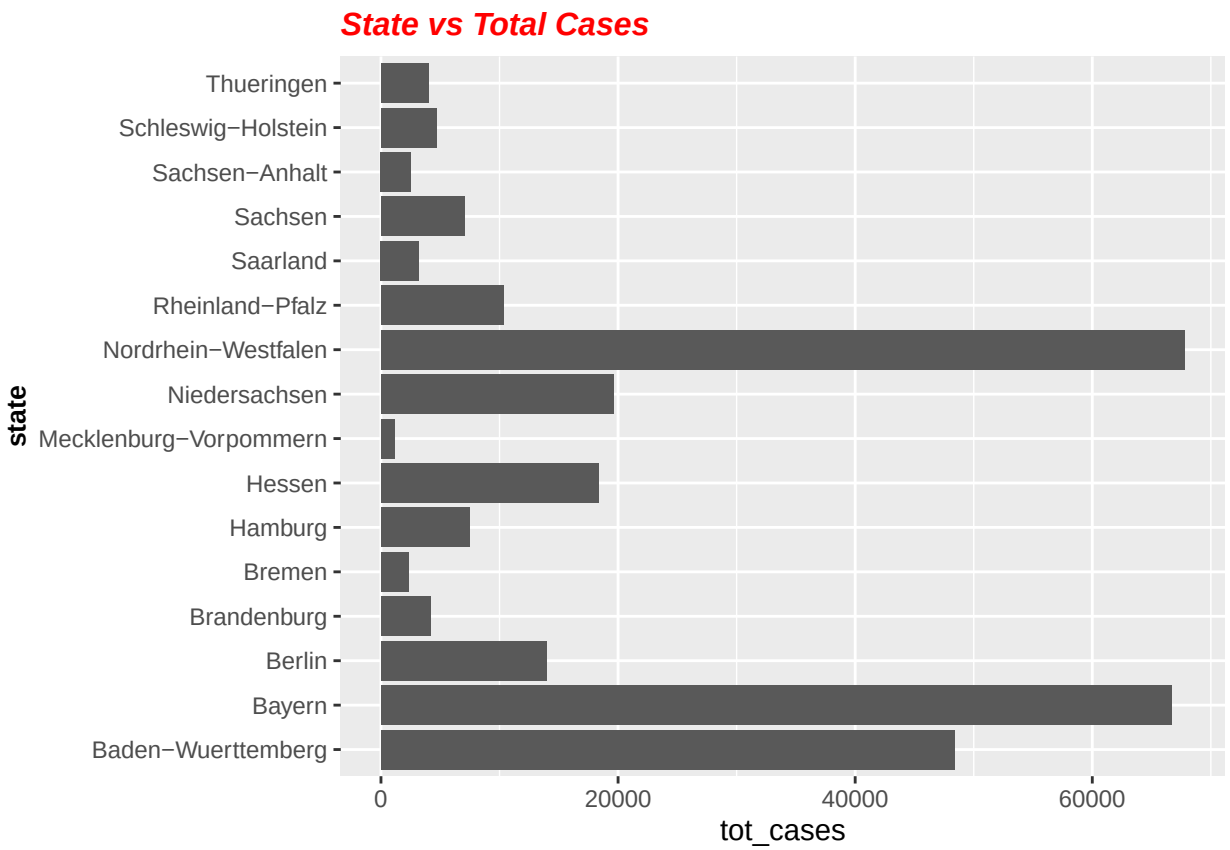
```
country_pop <- sum(demographicsdf_state_pop$total_pop)
```

Step 7: Data Visualization - To better understand the data in visual form, following graphs and plots are drwan

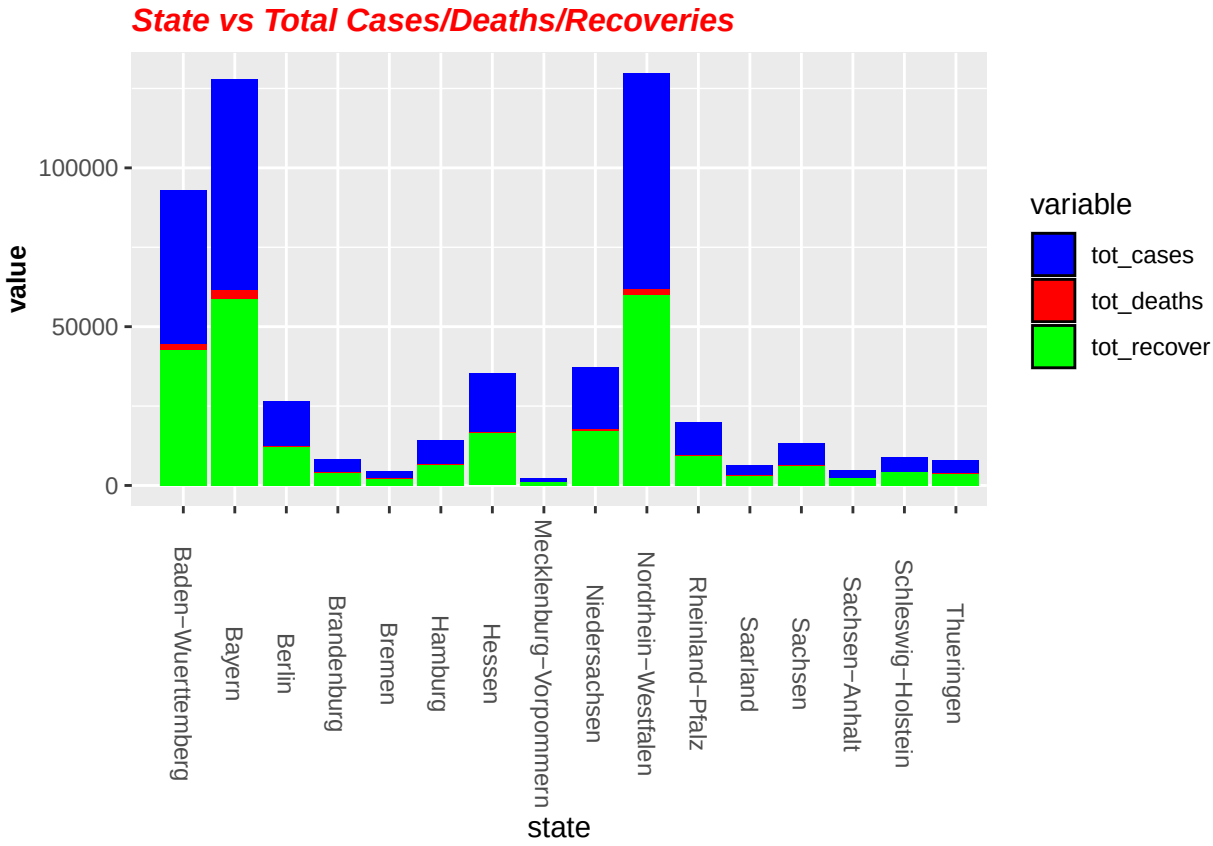
```
# State Level Stats and visualization
coviddf_lab_state_totals <- coviddf %>% group_by(state) %>% summarize(tot_cases = sum(cases), tot_deaths
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
coviddf_lab_state_cumtotals <- coviddf_lab_state_totals %>% mutate(CUSUM_cases=cumsum(tot_cases), cum_t

ggplot(coviddf_lab_state_totals, aes(x= tot_cases, y=state)) +
  geom_bar(position="stack", stat="identity")  + ggtitle("State vs Total Cases")+ theme(
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
```




```
melt_coviddf <- melt(coviddf_lab_state_totals, id.var="state")
ggplot(melt_coviddf, aes(x = state, y = value, fill = variable)) +
  geom_bar(stat = "identity") + scale_fill_manual(values=c('blue','red','green')) + geom_density(alpha=
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
```

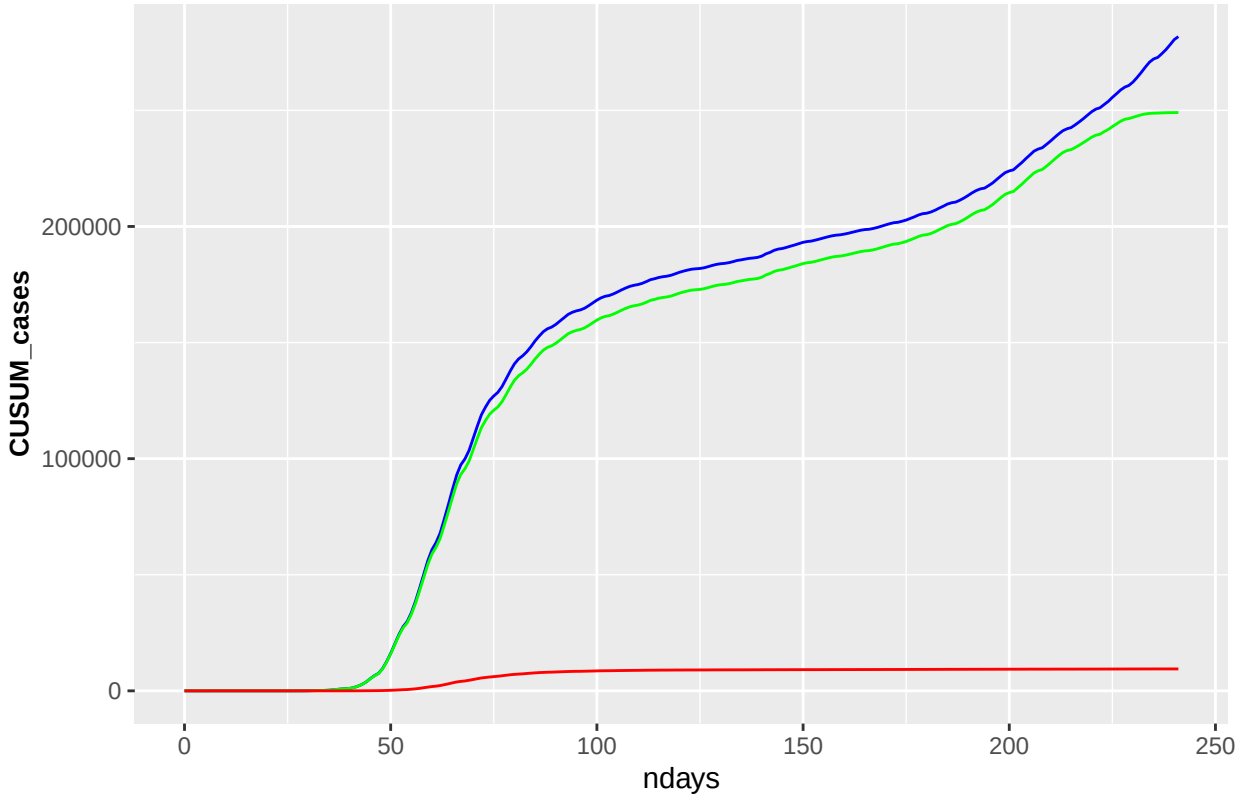



```
#Country level stats
coviddf_date_totals <- coviddf %>% group_by(ndays) %>% summarize(tot_cases=sum(cases), tot_deaths=sum(d
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
coviddf_date_totals <- coviddf_date_totals %>% mutate(CUSUM_cases=cumsum(tot_cases), cum_t_d=cumsum(tot_

plot <- ggplot(coviddf_date_totals, aes(x=ndays))
plot <- plot + geom_line(aes(y=CUSUM_cases), color="blue", alpha = 1)
plot <- plot + geom_line(aes(y=cum_t_r), color="green", alpha = 1)
plot <- plot + geom_line(aes(y=cum_t_d), color="red", alpha = 1)
plot <- plot + ggtitle("Country Level Cumulative Cases") + ylab("CUSUM_cases")
plot <- plot + theme(
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
plot
```

Country Level Cumulative Cases

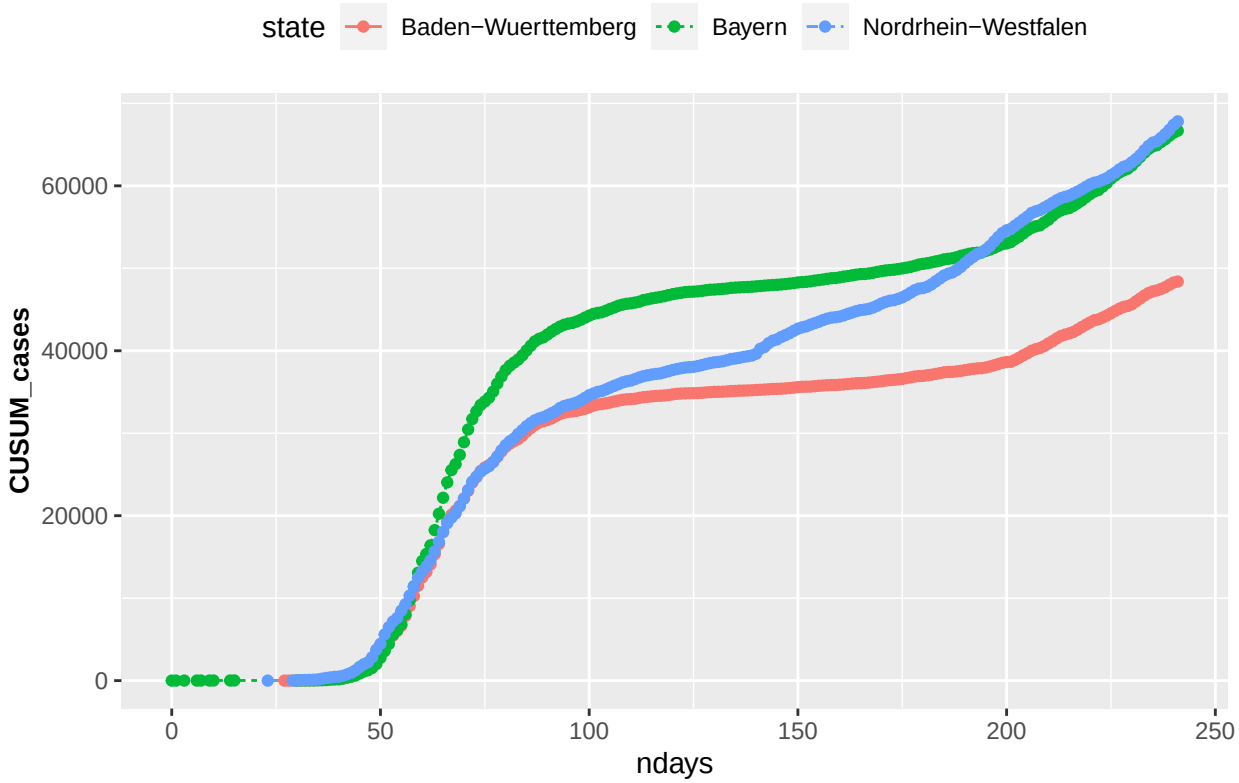



```
#Top 3 states - Case Stats Visualization
coviddf_lab_3states_days <- coviddf %>% filter(state %in% c('Nordrhein-Westfalen', 'Bayern', 'Baden-Wue
```

```
## `summarise()` regrouping output by 'state' (override with `.groups` argument)
```

```
coviddf_lab_3states_cum <- coviddf_lab_3states_days %>% mutate(CUSUM_cases=cumsum(tot_cases), cum_t_d=cu

coviddf_lab_3states_cum %>% ggplot(aes(ndays, CUSUM_cases, group = state)) +
  geom_line(aes(linetype = state, color = state)) + ggtitle("Top 3 states - Cumulative cases") + theme(
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
) + geom_point(aes(color = state)) +
  theme(legend.position = "top")
```

Top 3 states − Cumulative cases

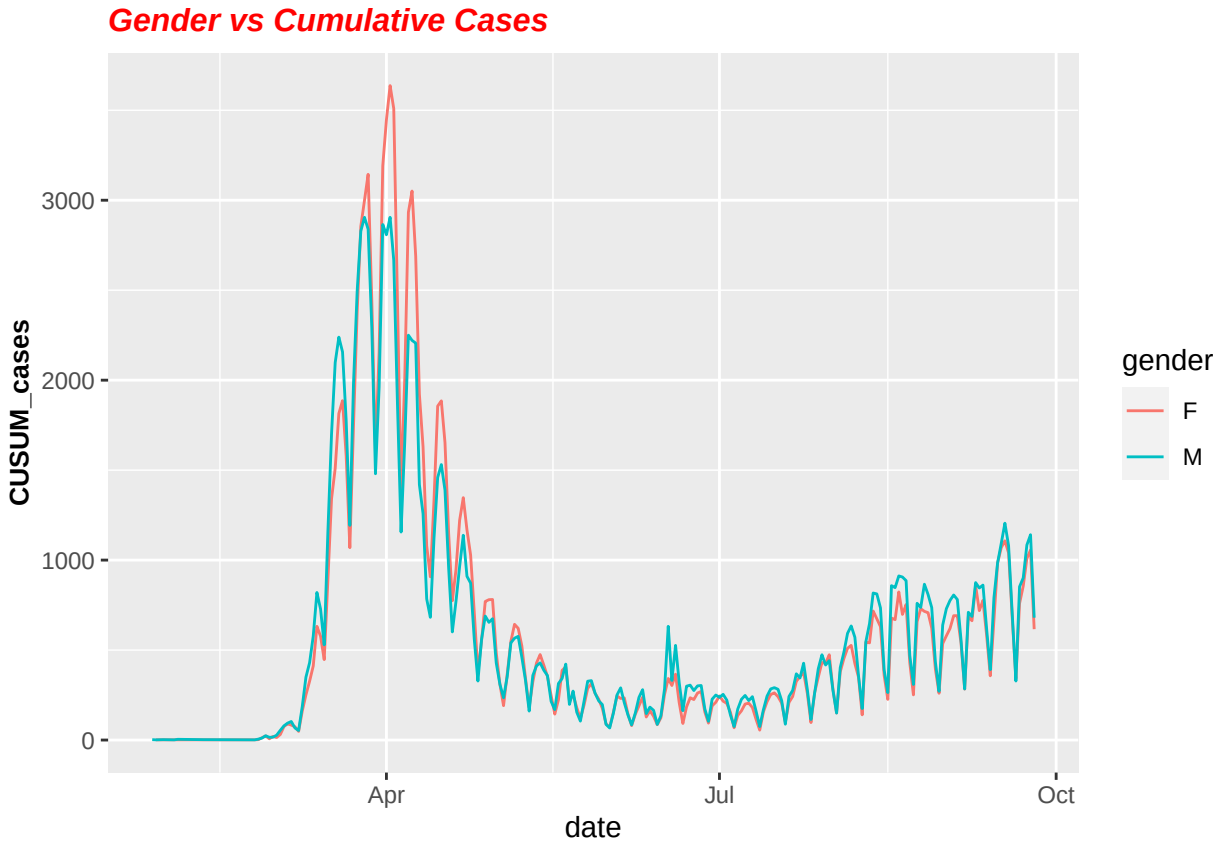



```
#Gender Stats and Visualizations
coviddf_date_gender_totals <- coviddf %>% group_by(date, gender) %>% summarize(tot_cases=sum(cases), to
```

```
## `summarise()` regrouping output by 'date' (override with `.groups` argument)
```

```
coviddf_date_gender_totals <- coviddf_date_gender_totals %>% group_by(date, gender) %>% mutate(CUSUM_ca
```

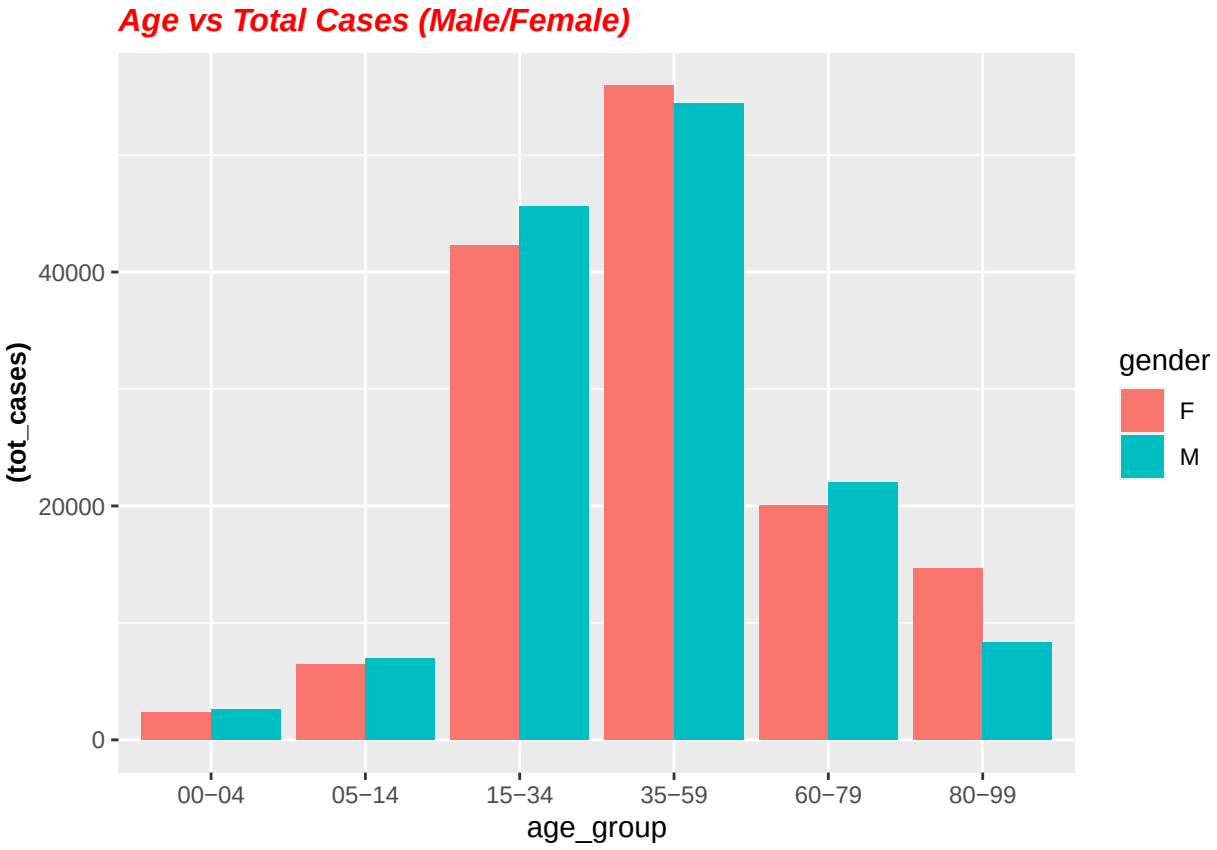
```
coviddf_date_gender_totals %>%
  ggplot(aes(date, CUSUM_cases, color=gender)) +
  geom_line(alpha = 1)  + ggtitle("Gender vs Cumulative Cases") + theme(
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
```




```
#Gender and Age Stats and Visualizations
coviddf_age_gender_totals <- coviddf %>% group_by(age_group, gender) %>% summarize(tot_cases=sum(cases)
```

```
## `summarise()` regrouping output by 'age_group' (override with `.groups` argument)
```

```
coviddf_age_gender_cumtotals <- coviddf_age_gender_totals %>% mutate(CUSUM_cases=cumsum(tot_cases), cum

ggplot(coviddf_age_gender_totals, aes(x = age_group, y = (tot_cases), fill = gender)) +
  geom_bar(stat = "identity", position = 'dodge') + ggtitle("Age vs Total Cases (Male/Female)") + theme
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
```



Commentary on Visualization:

Followings **observations** are made at data level and their relationship

1.

- Three states dominate and follow the country level spread of COVID-19.
- The age group that mostly affected is 35-59.
- There is no clear indication that a particular gender is significantly affected compared to other.

2.

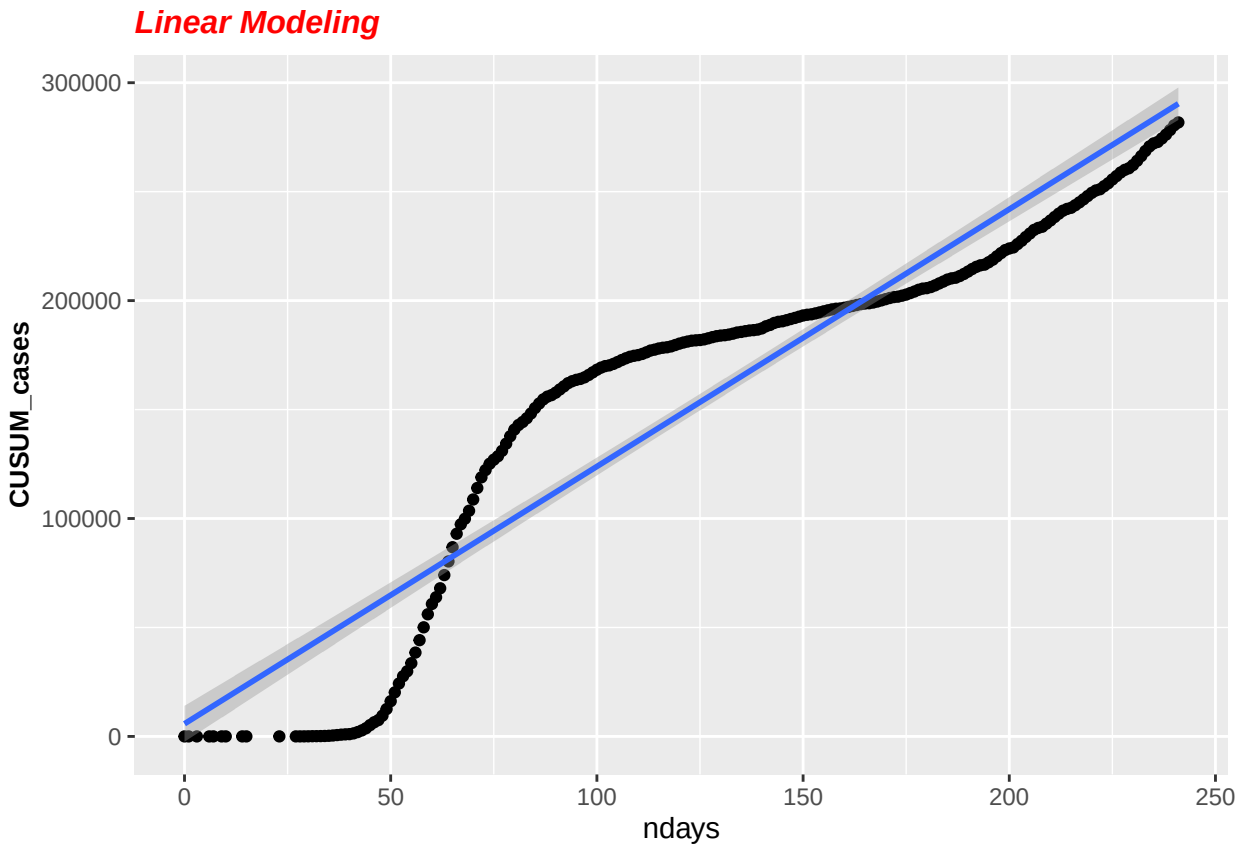
- It is clear that there is a relation between ndays and cumulative cases. Cases increase as the days passby.
- But, state and country level graphs represent that the relationship is not linear.
- Moreover, the curve also represents higher level order (exponential growth and sigmoidal growth) as applicable in case of infectious diseases.
- Hence, more complicated non-linear models are needed for regressing this behavior.

Step 6: Defining RMSE function - A function that computes the RMSE for daily cumulative cases and the ir corresponding predictors, as a measure of accuracy for linear models

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Step 7: Linear Modeling: Though above graphs visually represent that the time series data is close to non-linear relation, we deduce the same by some linear methods

```
#Trying a grid plot with geom_smooth feature adding a smoothing line in order to see what the
#trends look like
ggplot(coviddf_date_totals, aes(x = ndays, y = CUSUM_cases)) +
  geom_point() +
  geom_smooth(method = "glm", formula = y~x) + ggtitle("Linear Modeling") + theme(
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
```

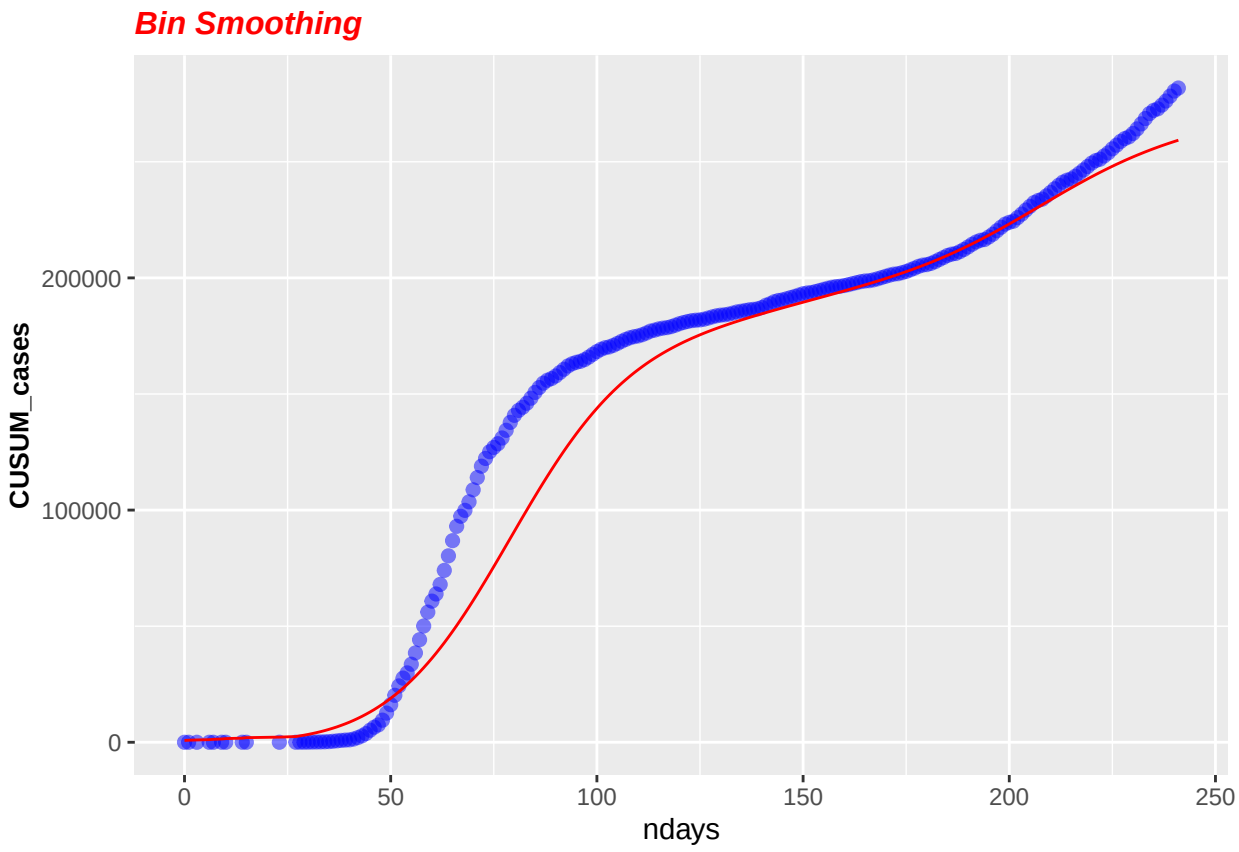



```
fit_lm <- lm(CUSUM_cases ~ ndays, data = coviddf_date_totals)
y_hat <- predict(fit_lm, coviddf_date_totals)

#Calculate RMSE and insert into a tibble
rmse_lm <- RMSE((y_hat)/country_pop*10000, (coviddf_date_totals$CUSUM_cases/country_pop)*10000)
rmse_lm
```

```
## [1] 3.456049
```

```
rmse_table <- tibble(Method = "Linear Model", RMSE = rmse_lm)

#We have seen that linear regression was not flexible enough to capture the non-linear nature of
#CUSUM_cases. Hence, adopting smoothing approaches may provide an improvement.
#Bin smoothing to detect trends in the presence of noisy data. A 50-day span is used to
#compute the average of the values within that span.
```

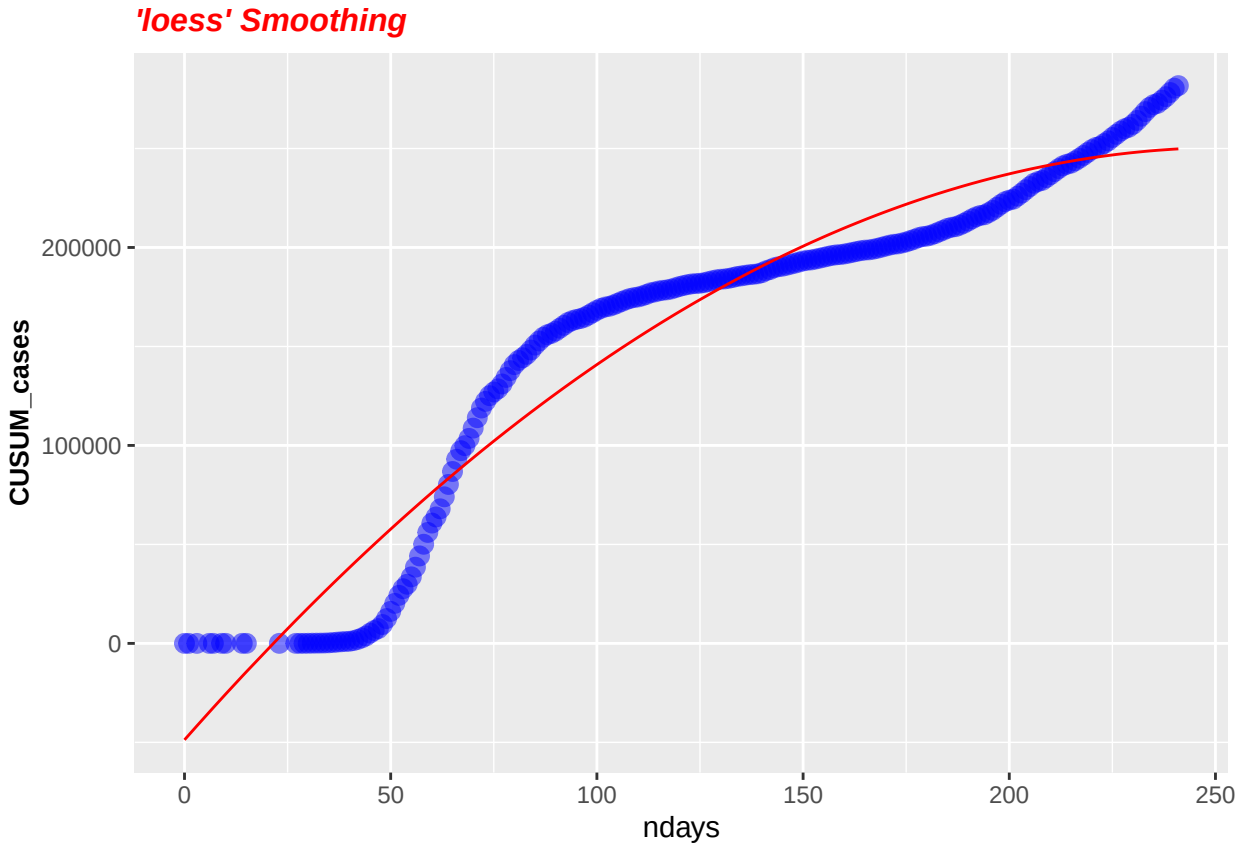
```
span <- 50
fit <- with(coviddf_date_totals,
            ksmooth(ndays, CUSUM_cases, kernel = "normal", bandwidth = span))
coviddf_date_totals %>% mutate(smooth = fit$y) %>% ggplot(aes(ndays, CUSUM_cases)) +
  geom_point(size = 2, alpha = .5, color = "blue") + geom_line(aes(ndays, smooth), color="red") + ggtitl
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
```




```
rmse1 <- RMSE((fit$y)/country_pop*10000, (coviddf_date_totals$CUSUM_cases/country_pop)*10000)
rmse1
```

```
## [1] 2.300176
```

```
rmse_table <- rmse_table %>% add_row(Method = "ksmooth", RMSE = rmse1)

#Local Weighted Regression (loess), another smoothing approach, assumes that data is locally
#linear. For this purpose a 25-day span used

span <- 25
fit <- loess(CUSUM_cases~ndays, degree=2, span = span, data=coviddf_date_totals)
coviddf_date_totals %>% mutate(smooth = fit$fitted) %>% ggplot(aes(ndays, CUSUM_cases)) +
  geom_point(size =3, alpha = .5, color = "blue") + geom_line(aes(ndays, smooth), color="red") + ggtitl
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
```




```
rmse2 <- RMSE((fit$fitted)/country_pop*10000, (coviddf_date_totals$CUSUM_cases/country_pop)*10000)
rmse2
```

```
## [1] 2.600655
```

```
rmse_table <- rmse_table %>% add_row(Method = "loess", RMSE = rmse2)
```

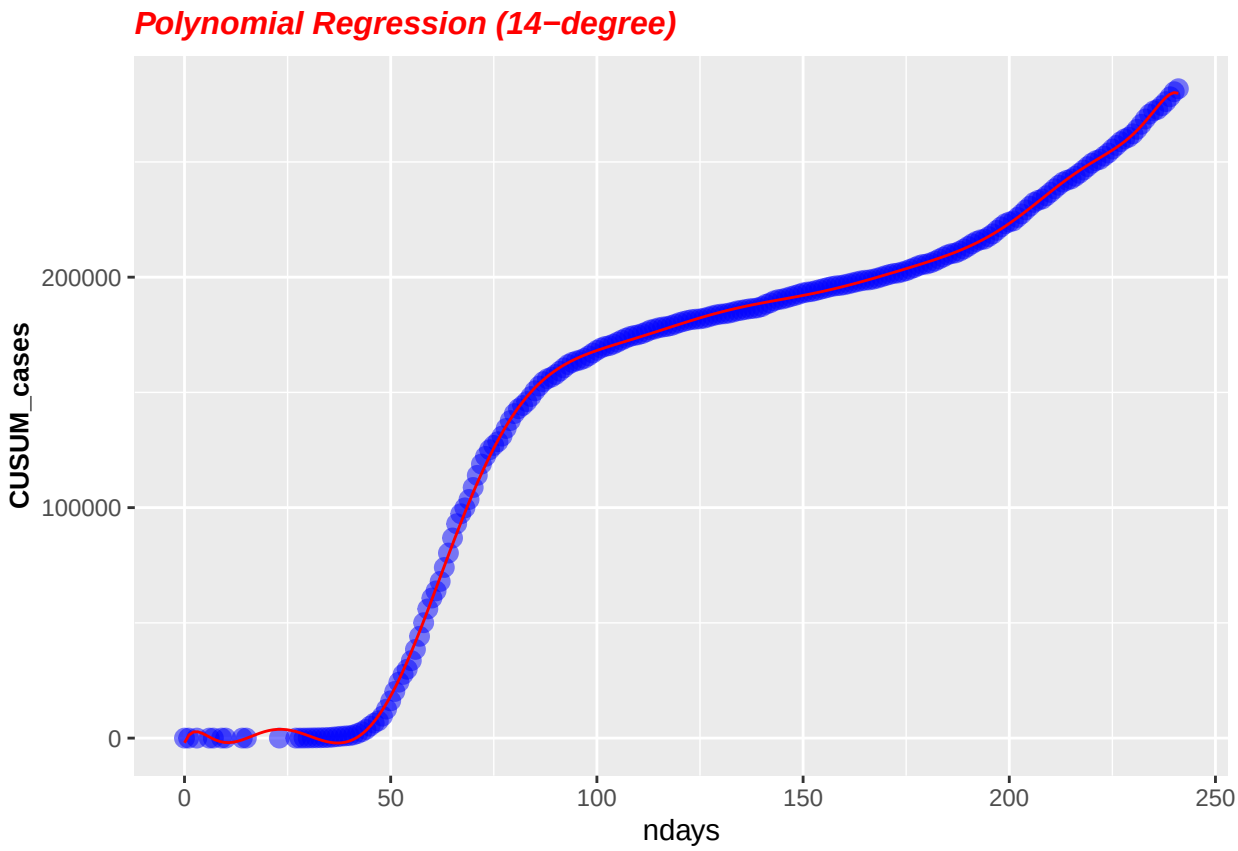
Commentary on Linear Regression Models:

The features used above, geom_smooth, 'lm' function, bin smoothing, loess method in linear modeling and RMSE values represent that the trend is not linear and the data can't be fit and modeled realistically using parametric (linear) methods and hence, statistical predictions can not be performed with higher accuracy. Hence, to progress further, we move to non-linear modeling to decrease the bias. These approaches may improve the predictive power.

Step 8: We start with Non-Linear approaches available in linear methods like polynomial regression and knn algorithim. These methods are known to provide some flexibility for non-linear regression

```
#Polynomial Regression - Extending linear regression to model the curve for capturing the
#nonlinear effects using higher-ordered values of the predictor. After several trails, polynomial
#term 14 was generating a smooth curve tracing the closely the curve generated by data.

polyfit2 <- lm(CUSUM_cases ~ poly(ndays, 14), data=coviddf_date_totals)

ntime <- seq(min(coviddf_date_totals$ndays), max(coviddf_date_totals$ndays), length=225)
```

```
preds <- predict (polyfit2, newdata = list(ndays = ntime), se=TRUE)

ggplot(coviddf_date_totals, aes(ndays, CUSUM_cases)) +
  geom_point(size =3, alpha = .5, color = "blue") + geom_line(aes(ntime, preds$fit), color="red", alpha
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
```




```
rmse3 <- RMSE((preds$fit)/country_pop*10000, (coviddf_date_totals$CUSUM_cases/country_pop)*10000)
rmse3
```

```
## [1] 2.485243
```

```
rmse_table <- rmse_table %>% add_row(Method = "poly-14degree", RMSE = rmse3)

#Knn algorithm is similar to bin and loess smoothing, but it maximizes accuracy, or minimizes the
#expected MSE
#Splitting the data into training and testing sets using caret library

set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = coviddf_date_totals$CUSUM_cases, times = 1, p = 0.3, list = FALSE)
testset <- coviddf_date_totals[test_index,]
trainset <- coviddf_date_totals[-test_index,]

#Cross Validaton - Performing 10-fold cross validation on the data using knn classifier for a
#sequence of k values

train_knn <- train(CUSUM_cases ~ ., method = "knn", data = trainset)
```

```
y_hat_knn <- predict(train_knn, testset, type = "raw", na.action = na.pass)
rmse5 <- RMSE((y_hat_knn)/country_pop*10000, (testset$CUSUM_cases/country_pop)*10000)
rmse5
```

```
## [1] 0.2765765
```

```
rmse_table <- rmse_table %>% add_row(Method = "knn(k=5)", RMSE = rmse5)

#We want to pick the k that maximizes accuracy, or minimizes the expected MSE

control <- trainControl(method = "cv", number = 10, p = .9)
train_knn_cv <- train(CUSUM_cases ~ ., method = "knn",
                      data = trainset,
                      tuneGrid = data.frame(k = seq(9, 71, 2)),
                      trControl = control)
train_knn_cv$bestTune
```

```
##   k
## 1 9
```

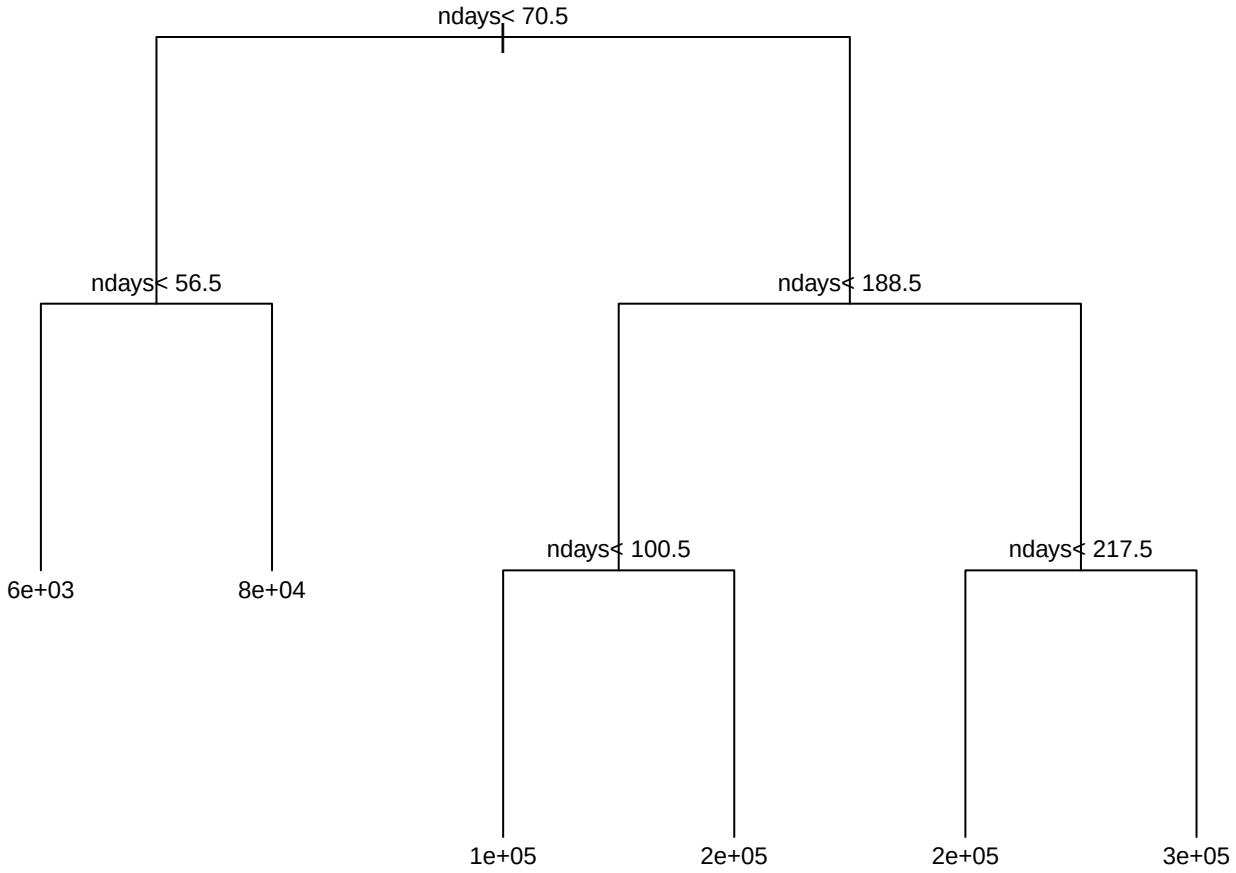
```
#The function 'predict' will use this best performing model. Here is the accuracy of the best #model wh

y_hat_knn <- predict(train_knn_cv, testset, type = "raw", na.action = na.pass)
rmse6 <- RMSE((y_hat_knn)/country_pop*10000, (testset$CUSUM_cases/country_pop)*10000)
rmse6
```

```
## [1] 0.3297879
```

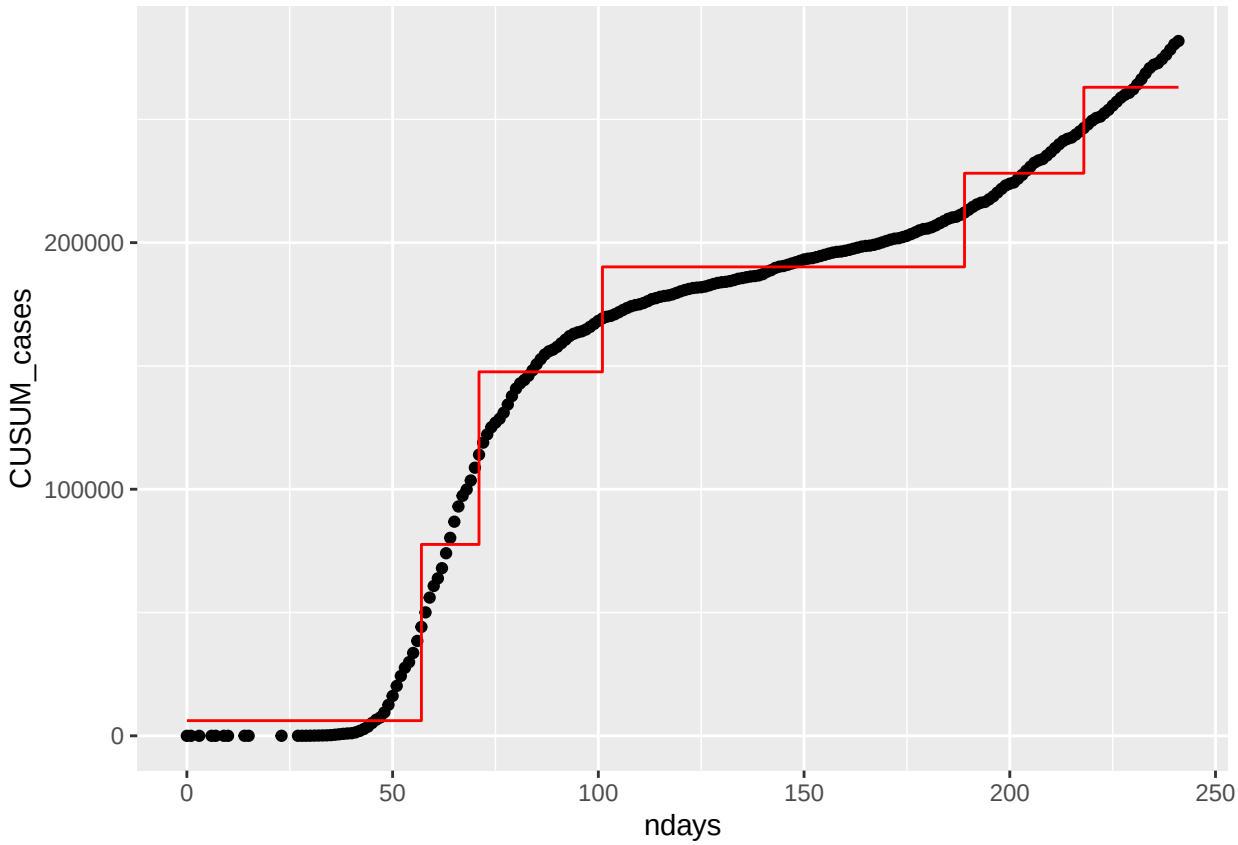
```
rmse_table <- rmse_table %>% add_row(Method = "knn-cross_valid", RMSE = rmse6)
```

Step 9: Non-Linear Models - Regression Trees

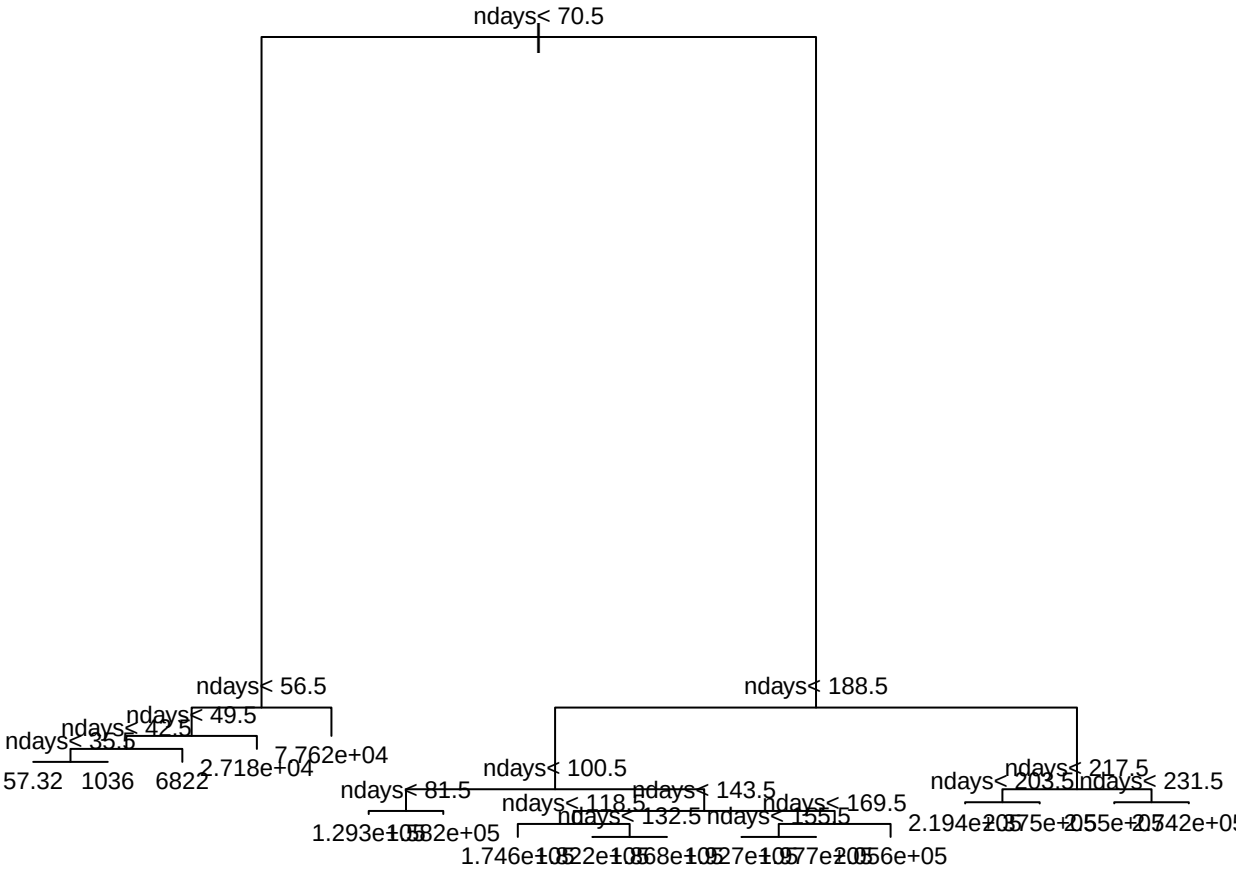
```
#When the outcome is continuous, we can also use a method called 'regression tree'. Regression
#trees create partitions recursively.
par(mar=c(0,0,0,0)) #setting the margins for dynamic graph
fit4 <- rpart(CUSUM_cases ~ ndays, data = coviddf_date_totals)
plot(fit4, CUSUM_cases = 10, compress = TRUE, uniform = TRUE, digits = -2)
text(fit4, pretty=0, use.n = FALSE, cex = 0.75, n = TRUE, xpd = TRUE, digits = -2)
```




```
coviddf_date_totals %>%
  mutate(y_hat = predict(fit4)) %>% ggplot() +
  geom_point(aes(ndays, CUSUM_cases)) + geom_step(aes(ndays, y_hat), col="red")
```



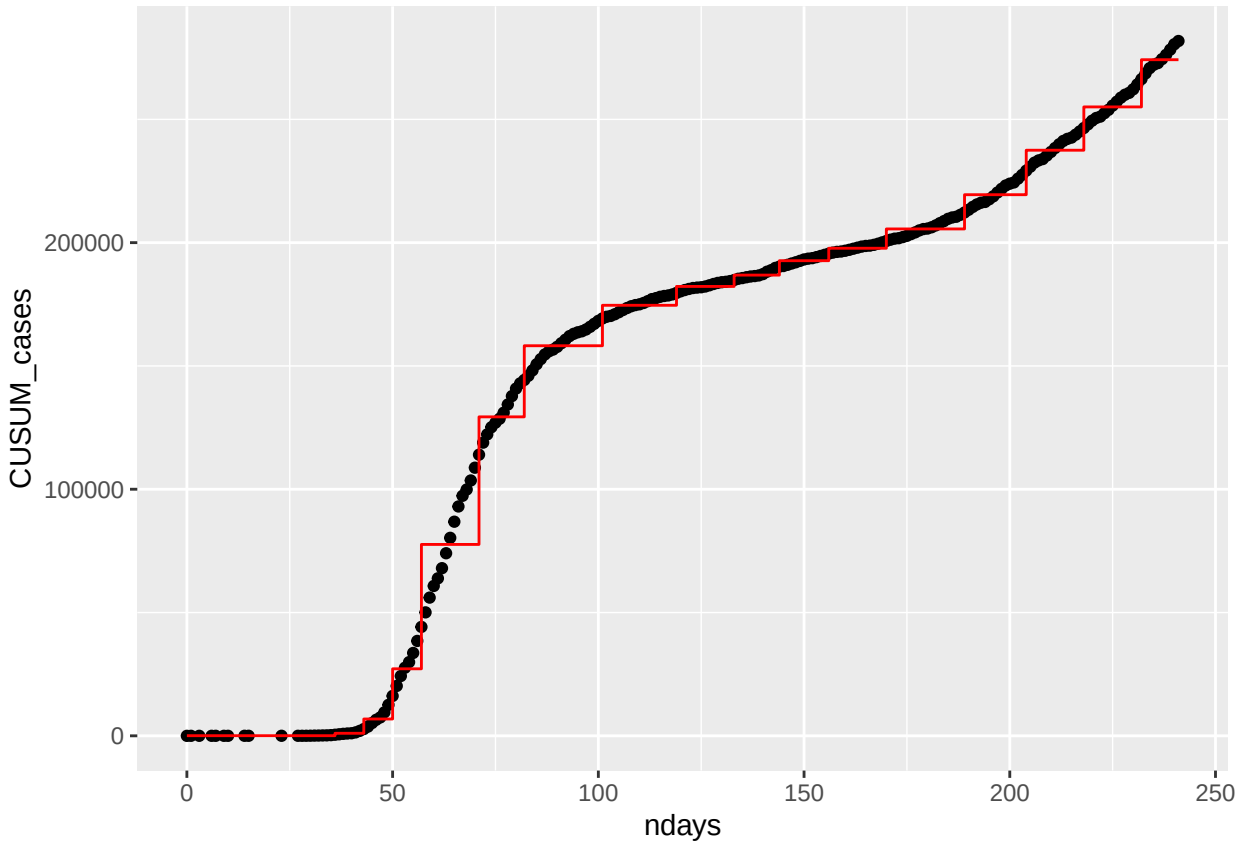

```
train_rpart <- train(CUSUM_cases ~ ndays,
                     method = "rpart",
                     tuneGrid = data.frame(cp = seq(0, 0.05, len = 25)), data = coviddf_date_totals)
par(mar=c(0,0,0,0)) #setting the margins for dynamic graph
plot(train_rpart$finalModel, CUSUM_cases = 10, branch = 1, margin = 0, minbranch = 0.1, compress = TRUE
text(train_rpart$finalModel, pretty=1, use.n = FALSE, cex = 0.75, n = TRUE, xpd = TRUE)
```




```
labels(train_rpart$finalModel, minlength=0)
```

```
##  [1] "root"          "ndays< 70.5"  "ndays< 56.5"  "ndays< 49.5"  "ndays< 42.5"
##  [6] "ndays< 35.5"   "ndays>=35.5"  "ndays>=42.5"  "ndays>=49.5"  "ndays>=56.5"
## [11] "ndays>=70.5"   "ndays< 188.5" "ndays< 100.5" "ndays< 81.5"  "ndays>=81.5"
## [16] "ndays>=100.5"  "ndays< 143.5" "ndays< 118.5" "ndays>=118.5" "ndays< 132.5"
## [21] "ndays>=132.5"  "ndays>=143.5" "ndays< 169.5" "ndays< 155.5" "ndays>=155.5"
## [26] "ndays>=169.5"  "ndays>=188.5" "ndays< 217.5" "ndays< 203.5" "ndays>=203.5"
## [31] "ndays>=217.5"  "ndays< 231.5" "ndays>=231.5"
```

```
coviddf_date_totals %>%
  mutate(y_hat = predict(train_rpart)) %>% ggplot() +
  geom_point(aes(ndays, CUSUM_cases)) + geom_step(aes(ndays, y_hat), col="red")
```




```
#Gender, age_group estimates using 'rpart' function and drawing decision trees
coviddf_ndays_age_gender_totals <- coviddf %>% group_by(ndays, age_group, gender) %>% summarize(tot_cas
```

```
## `summarise()` regrouping output by 'ndays', 'age_group' (override with `.groups` argument)
```

```
coviddf_ndays_age_gender_cumtotals <- coviddf_ndays_age_gender_totals %>% mutate(CUSUM_cases=cumsum(tot
coviddf_ndays_age_gender_cumtotals <- coviddf_ndays_age_gender_cumtotals[, c(-4,-5,-6)]

index <- sample(1:nrow(coviddf_ndays_age_gender_cumtotals), nrow(coviddf_ndays_age_gender_cumtotals) *0
train <- coviddf_ndays_age_gender_cumtotals[index,]
test <- coviddf_ndays_age_gender_cumtotals[-index,]

fit_tree <- rpart(gender~ ., data = train)
test$pred <- predict(fit_tree, newdata = test, type="class")

#Confusion Matrix and Accuracy
cm <- confusionMatrix(test$pred, test$gender)
cm$table
```

```
##           Reference
## Prediction   F   M
##          F 155  74
##          M  92 185
```

```
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.6719368
```

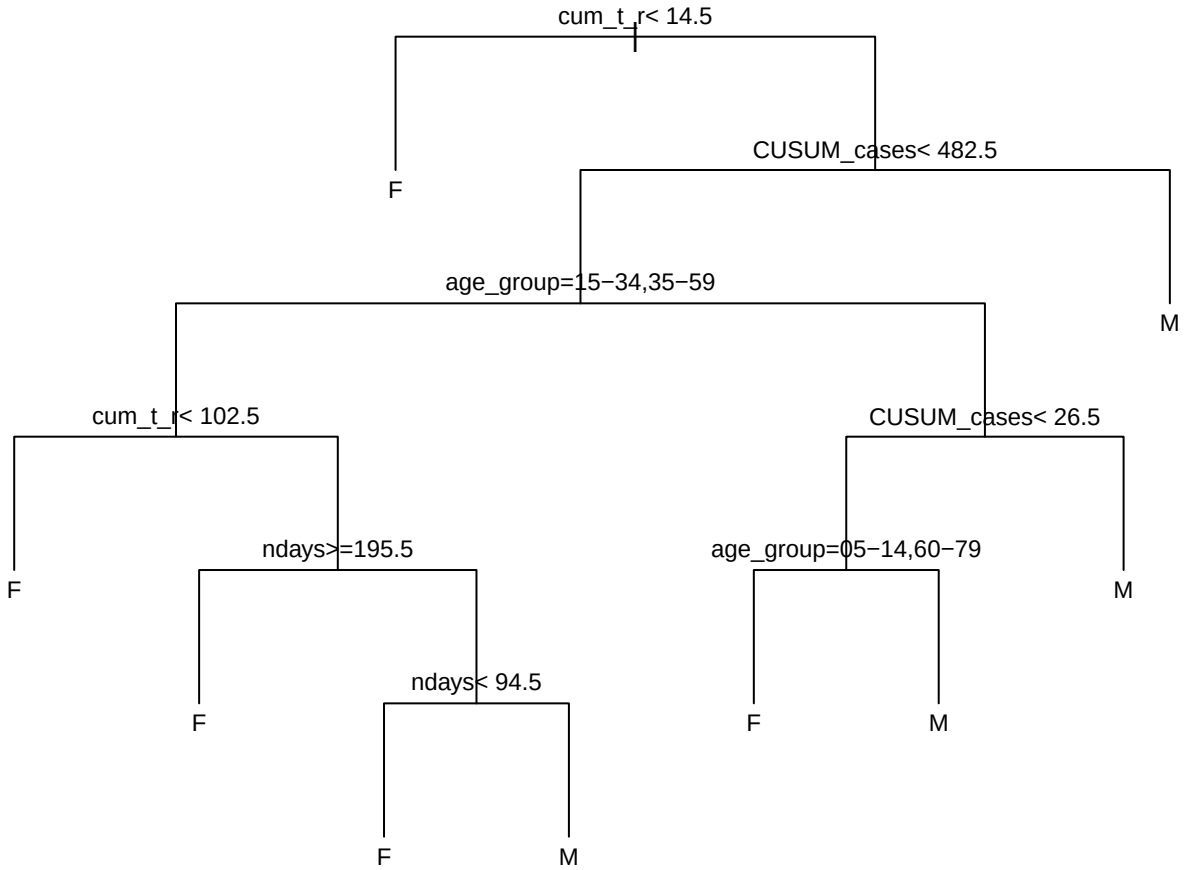
```
accuracy_table <- tibble(Method = "DecisionTree - Gender", Accuracy = cm$overall["Accuracy"])

#It is also important to examine sensitivity and specificity and not just accuracy
cm$byClass[c("Sensitivity","Specificity", "Prevalence")] #> Sensitivity Specificity Prevalence
```

```
## Sensitivity Specificity  Prevalence
##   0.6275304   0.7142857   0.4881423
```

```
table(test$gender)
```

```
##
##   F   M
## 247 259
```

```
par(mar=c(0,0,0,0)) #setting the margins for dynamic graph
plot(fit_tree, compress = TRUE, uniform = TRUE, digits = -2)
text(fit_tree, pretty=0, use.n = FALSE, cex = 0.75, n = TRUE, xpd = TRUE, digits = -2)
```




```
fit_tree <- rpart(age_group~ ., data = train)
test$pred <- predict(fit_tree, newdata = test, type="class")

#Confusion Matrix and Accuracy
cm <- confusionMatrix(test$pred, test$age_group)
cm$table
```

```
##           Reference
## Prediction 00-04 05-14 15-34 35-59 60-79 80-99
##      00-04    77    34     6     9    12    15
```

```
##       05-14    10    38    13     7     9     0
##       15-34     0     7    63    46     3     0
##       35-59     0     0     1    19     2     0
##       60-79     1     0     0     6    33    11
##       80-99     1     0     0     0    26    57
```

```
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.5671937
```

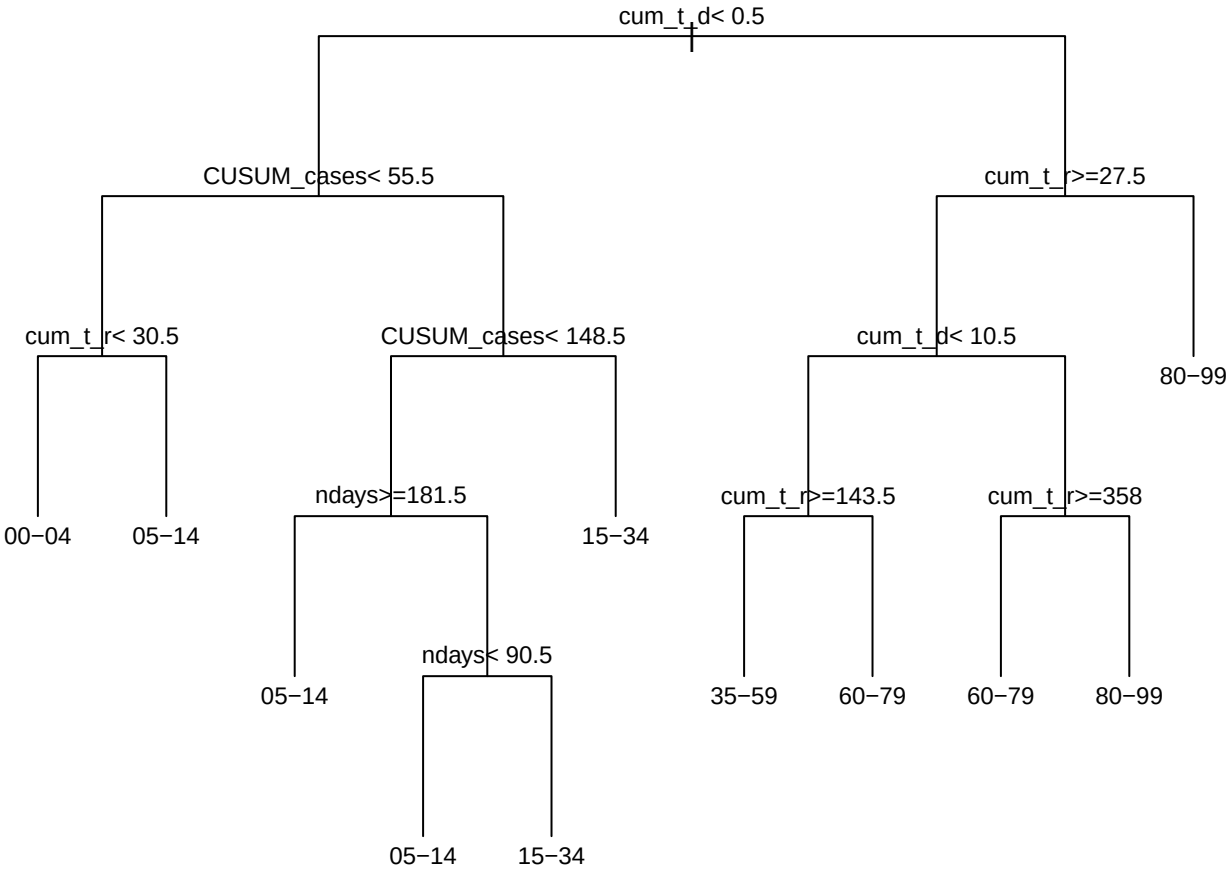
```
accuracy_table <- accuracy_table %>% add_row(Method = "DecisionTree - Age", Accuracy = cm$overall["Accu

#It is also important to examine sensitivity and specificity and not just accuracy
cm$byClass[c("Sensitivity","Specificity", "Prevalence")] #> Sensitivity Specificity Prevalence
```

```
## [1] NA NA NA
```

```
table(test$age_group)
```

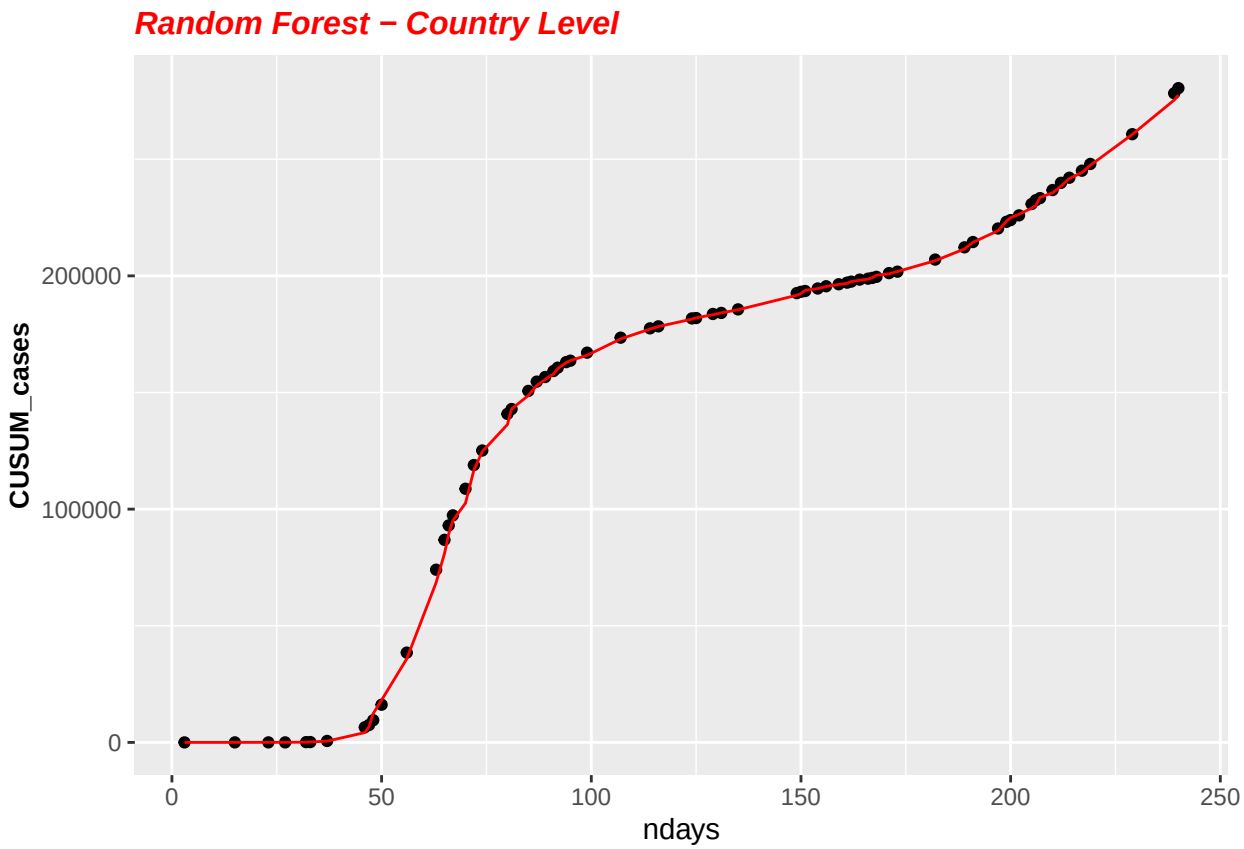
```
##
## 00-04 05-14 15-34 35-59 60-79 80-99
##    89    79    83    87    85    83
```

```
par(mar=c(0,0,0,0)) #setting the margins for dynamic graph
plot(fit_tree, compress = TRUE, uniform = TRUE, digits = -2)
text(fit_tree, pretty=0, use.n = FALSE, cex = 0.75, n = TRUE, xpd = TRUE, digits = -2)
```



Step 10: Non-Linear Models - Random forests

```
#'Random forest' is a very popular machine learning approach. It addresses the
#short-comings of decision trees. Its goal is to improve prediction performance and reduce
#instability by averaging multiple decision trees (a forest of trees constructed with randomness).

fit <- randomForest(CUSUM_cases~ndays, data = trainset, na.action = na.omit)
y_hat <- predict(fit, testset)
testset %>% ggplot() +
  geom_point(aes(ndays, CUSUM_cases)) +
  geom_line(aes(ndays, y_hat), col="red") + ggtitle("Random Forest - Country Level") + theme(
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
```




```
rmse7 <- RMSE((y_hat/country_pop)*10000, (testset$CUSUM_cases/country_pop)*10000)
rmse7
```

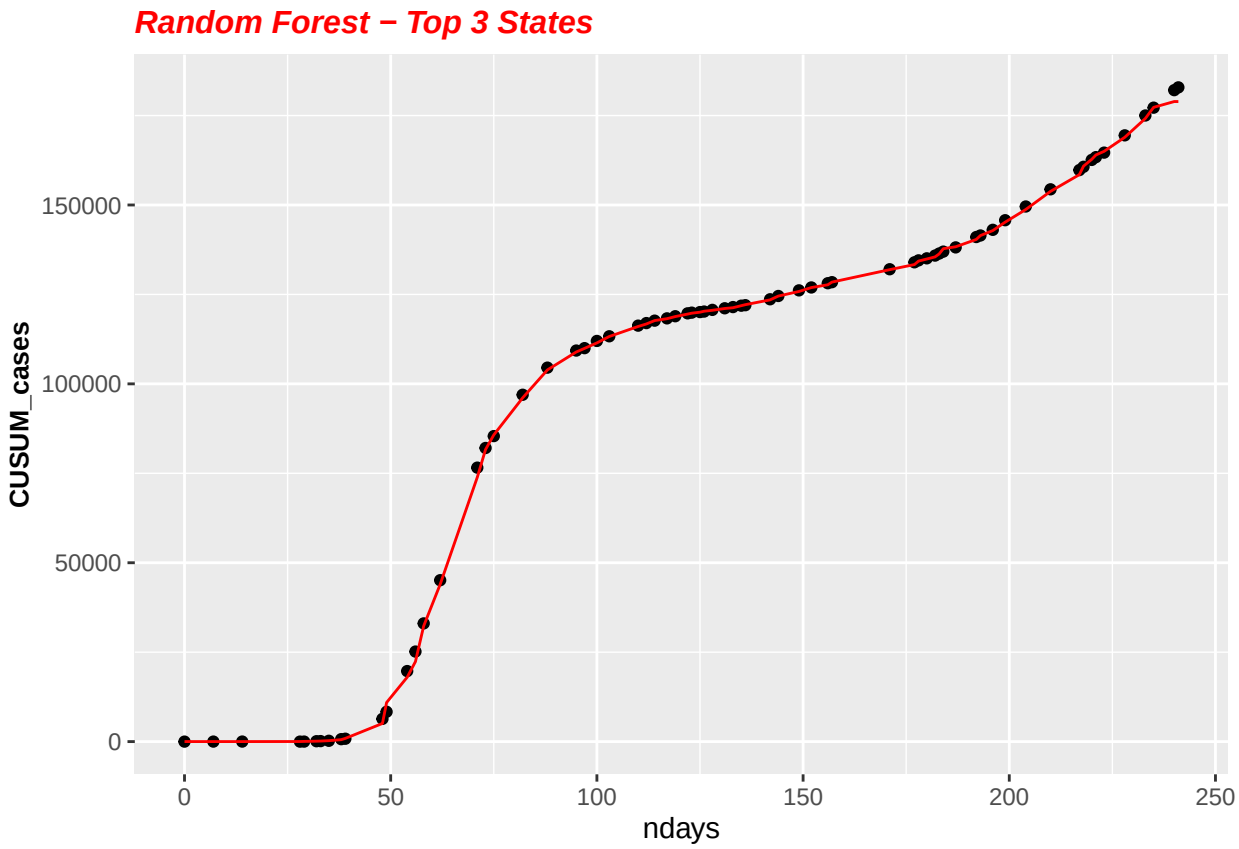
```
## [1] 0.2147167
```

```
rmse_table <- rmse_table %>% add_row(Method = "rforest-country", RMSE = rmse7)

#Random Forest for sum of top 3 states
coviddf_lab_3states_cum_ndays <- coviddf_lab_3states_cum %>% group_by(ndays) %>% summarize(tot_cases = s
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
test_index <- createDataPartition(y = coviddf_lab_3states_cum_ndays$CUSUM_cases, times = 1, p = 0.3, lis
testset <- coviddf_lab_3states_cum_ndays[test_index,]
```

```
trainset <- coviddf_lab_3states_cum_ndays[-test_index,]

fit3 <- randomForest(CUSUM_cases~ndays, data = trainset)
y_hat = predict(fit3, newdata = testset)
testset %>% ggplot() +
  geom_point(aes(ndays, CUSUM_cases)) +
  geom_line(aes(ndays, y_hat), col="red") + ggtitle("Random Forest - Top 3 States") + theme(
  plot.title = element_text(color="red", size=12, face="bold.italic"),
  axis.title.y = element_text(size=10, face="bold")
)
```




```
rmse8 <- RMSE((y_hat/country_pop)*10000, (testset$CUSUM_cases/country_pop)*10000)
rmse8
```

```
## [1] 0.1150955
```

```
rmse_table <- rmse_table %>% add_row(Method = "rforest-top3States", RMSE = rmse8)
```

Step 11: Models Performance

```
kable(rmse_table, caption = "Linear RMSE Performance")
```

Table 1: Linear RMSE Performance

Method	RMSE
Linear Model	3.4560486
ksmooth	2.3001760
loess	2.6006553
poly-14degree	2.4852434
knn(k=5)	0.2765765
knn-cross_valid	0.3297879
rforest-country	0.2147167
rforest-top3States	0.1150955

```
kable(accuracy_table, caption = "Non-Linear Accuracy Performace" )
```

Table 2: Non-Linear Accuracy Performace

Method	Accuracy
DecisionTree - Gender	0.6719368
DecisionTree - Age	0.5671937

Step 12: Summary & Conclusions:

We observe the above table and can easily deduce that RMSE value keeps decreasing steadily as the finess of regression improves from bare linear model (f(x) = ax + b) to adopting smoothing techniques to polynonial regression.

However, the RMSE values prove that COVID-19 data can not be modeled to a linear form. Still, there is a room for improvement for cancelling the noise.

There are some nonlinear models which are actually called intrinsically linear (like knn, k-nearest neighbor algorithm) because they can be made linear in the parameters by a simple transformation. When basic knn (with k = 5) and knn cross validation is applied, the RMSE value decreased drastically (from 2.5 to 0.28).

However, the nature and shape of cumulative cases vs ndays represent that it is not possible to transform COVID-19 model to a linear form. The above used regression approaches had no chance of capturing the non-linear nature of true COVID-19. We stil observe some “noisy” ups and downs for some regions.

Hence, the performance of the model is improved further by applying advanced non-linear techniques such as Decision Trees and Random Forest. These modelings are applied on train data first and the results are used in the context of forecasting using test data. Forecasting/prediction is the art of modeling patterns in time series plots.The forecasting accuarcy calculated on test data for these techniques is at the order of ~0.6 (60%) and RMSE to the lowest value of 0.215.

Step 13: Limitations and Future Work:

COVID-19 is first time of its kind. Still the end of it is not seen. Hence, it is very difficult to model this highly unpredictable infectious disease into any existing statistical model (linear or non-linear).

Maybe, using few other advanced models like PCA (principal component analysis), Hierarchial clustering, nls function, and further advanced supervised models can improve the calculation of accuracy.

The significant limitation of the data is its time period. Only, data for 225 days is avaiable in these data sets since exception of COVID-19. Still it has not reached the peak and has not started the downward trend. Hence, this analysis may not represent the trends of COVID-19 in a 360 degree view.

————————————————-End Report————————————————-