Title: Modelling and Forecasting for Coronavirus Disease 2019 (COVID-19) spread in Miami-Dade County, Florida, USA, 2020.

Codes for Prophet's Logistic Growth Model

Data Management

```
#setting the library
setwd("C:/Users/Sumedh Kaul/Desktop/Miami-Dade County(Predictions Project)/July, 2020/12th
august")

#Loading the required libraries
library(dplyr)
library(prophet)
library(lubridate)
library(ggplot2)
library(htmlwidgets)
library(webshot)

#opening the dataset
county <- read.csv("us-counties.csv")

#Subsetting the data to Miami-Dade county
miami_county <- county %>%
  filter(county == 'Miami-Dade')

#Looking at the structure of the dataset
str(miami_county)

#changing the date to date format
miami_county$date <- ymd(miami_county$date)

#changing the formats of required variables
miami_county$cases <- as.numeric(miami_county$cases)
miami_county$deaths <- as.numeric(miami_county$deaths)

#calculating the daily cases and deaths
daily = miami_county %>%
  mutate(daily_cases = cases - lag(cases)) %>%
  mutate(daily_deaths = deaths - lag(deaths))
```

str(daily)

#replace the 1st row value of cases and deaths with 1 and 0 respectively as the above algorithm did not fill them but it's confirmed that there were 1 cases and 0 deaths on 1st day of data collection.
daily$daily_cases[1] <- 1
daily$daily_deaths[1] <- 0

#Look at the structure again of the dataset
str(daily)

Mapping the graphs

#plot (daily number of cases)
ggplot(aes(x=date, y= daily_cases), data = daily) +
  geom_bar(colour = "black", fill = "#DD8888", stat = "identity") +
  ggtitle('Figure 1: Daily no. of new COVID-19 cases (August 11th, 2020 - 1507 cases) in \nMiami-Dade County, Florida, USA') +
  labs(y = "Daily cases", x = "Months")

#save the above plot as an image
ggsave("Fig 1 Daily New Cases.png")

#plot (confirmed cumulative numbers of cases)
qplot(date, cases, data = daily,
      main = 'Figure 2: Cumulative no. of COVID-19 cases from March 11th, 2020 to \nAugust 11th, 2020 (135129 cases) in Miami-Dade County, Florida, USA', xlab = 'Months', ylab = 'Cases')

#save the above plot as an image
ggsave("Fig 2 Cumultive cases.png")

#plot (daily number of deaths)
ggplot(aes(x=date, y= daily_deaths), data = daily) +
  geom_bar(colour = "black", fill = "#DD8888", stat = "identity") +
  ggtitle('Figure 3: Daily no. of new COVID-19 deaths (August 11th, 2020 - 35 deaths) in \nMiami-Dade County, Florida, USA') +
  labs(y = "Daily deaths", x = "Months")

#save the above plot as an image
ggsave("Fig 3 Daily new deaths.png")

```
#plot (confirmed cumulative numbers of cases)
qplot(date, deaths, data = daily,
    main = 'Figure 4: Cumulative no. of COVID-19 deaths from March 11th, 2020 to \nAugust 11th, 2020(1909 deaths) in Miami-Dade County, Florida, USA', xlab = 'Months', ylab = 'Deaths')

#save the above plot as an image
ggsave("Fig 4 Cumultive deaths.png")
```

Forecasting

```
####Forecasting of cumulative cases
ds <- miami_county$date
y <- miami_county$cases
df <- data.frame(ds, y)

#Forecasting
k <- prophet(df)

#prediction
future <- make_future_dataframe(k, periods = 14)
forecast <- predict(k, future)
tail(forecast[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])

#Plot forecast
plot_cases <- dyplot.prophet(k, forecast, xlab = 'Months', ylab = 'Cases', main = 'Figure 5: Predicted no. of Cumulative cases from August 12th, 2020 to August 25th, 2020 (Average: 177914 cases) in Miami-Dade County, Florida, USA')
plot_cases

# save html to png
saveWidget(plot_cases, "cases.html", selfcontained = FALSE)
width<- 1080
height <- 610
webshot("cases.html", file = "Fig 5 Cases prediction from 12th August.png",
      cliprect = c(10,30,width+50,height+50)
      ,vwidth = width, vheight = height )


#Forecast components
prophet_plot_components(k, forecast)
```

```
#Checking Model performance
pred <- forecast$yhat[1:154]
actual <- k$history$y
par(mar=c(1,1,1,1))
plot(actual, pred)
abline(lm(pred~actual), col = 'red')
summary(lm(pred~actual))
#r-squared value: 0.99

####Forecasting of cumulative deaths
ds <- miami_county$date
y <- miami_county$deaths
df <- data.frame(ds, y)

#Forecasting
n <- prophet(df)

#prediction
future <- make_future_dataframe(n, periods = 14)

forecast <- predict(n, future)
tail(forecast[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])

#Plot forecast
plot_deaths <- dyplot.prophet(n, forecast, xlab = 'Months', ylab = 'Deaths', main = 'Figure 6:
Predicted no. of Cumulative deaths from August 12th, 2020 to August 25th, 2020 (Average :
2220 deaths) in Miami-Dade County, Florida, USA')
plot_deaths

# save html to png
saveWidget(plot_deaths, "deaths.html", selfcontained = FALSE)
width<- 1080
height <- 610
webshot("deaths.html", file = "Fig 6 deaths prediction from 12th August.png",
        cliprect = c(10,30,width+50,height+50)
        ,vwidth = width, vheight = height )

#Forecast components
prophet_plot_components(n, forecast)

#Model performance

pred <- forecast$yhat[1:154]
```

```
actual <- n$history$y
par(mar=c(1,1,1,1))
plot(actual, pred)
abline(lm(pred~actual), col = 'red')
summary(lm(pred~actual))

#r-squared value 0.99
```

Codes for ARIMA Model

Data Managment

```
#setting the library
setwd("C:/Users/Sumedh Kaul/Desktop/Miami-Dade County(Predictions Project)/July, 2020/12th august")

#Loading the required libraries
library(dplyr)
library(prophet)
library(lubridate)
library(ggplot2)
library(htmlwidgets)
library(webshot)

#open the dataset
county <- read.csv("us-counties.csv")

#Subset to Miami-Dade county
miami_county <- county %>%
  filter(county == 'Miami-Dade')

#Look at the structure of the dataset
str(miami_county)

#change the date to date format
miami_county$date <- ymd(miami_county$date)

#changing the format of the required variables
miami_county$cases <- as.numeric(miami_county$cases)
miami_county$deaths <- as.numeric(miami_county$deaths)

#calculating number of cases and deaths per day
daily = miami_county %>%
  mutate(daily_cases = cases - lag(cases)) %>%
```

```
  mutate(daily_deaths = deaths - lag(deaths))
str(daily)

#replace the 1st row value of cases and deaths with 1 and 0 respectively as the above algorithm
did not fill them but it's confirmed that there were 1 cases and 0 deaths on 1st day of data
collection.
daily$daily_cases[1] <- 1
daily$daily_deaths[1] <- 0

#Look at the structure of the dataset again
str(daily)
```

Stationarity

```
#check the stationary in the data of daily cases
plot(daily$daily_cases, xlab = 'Days', ylab = 'Daily cases')
title(main = 'Checking the Data stationary of daily cases')

#check the stationary in the data of daily deaths
plot(daily$daily_deaths, xlab = 'Days', ylab = 'Daily cases')
title(main = 'Checking the Data stationary')

#Daily cases does not look stationary

#As the cases do not look stationary, we take the difference of cases from the previous values

#take the difference of desired variables
d.cases <- diff(daily$daily_cases)
d.deaths <- diff(daily$daily_deaths)

#checking the data stationary for daily cases
plot(d.cases, xlab = 'Days', ylab = 'Difference of daily cases')
title(main = 'Checking the Data stationary for daily cases')

#checking the data stationary for daily deaths
plot(d.deaths, xlab = 'Days', ylab = 'Difference of daily deaths')
title(main = 'Checking the Data stationary for daily deaths')

#The difference values look stationary. Let's check the stats before we proceed

summary(daily$daily_cases)
summary(d.cases)

summary(daily$daily_deaths)
```

```r
summary(d.deaths)

#Although we have studied stationarity visually, we have to perform Dickey Fuller tests and augmented Dickey Fuller test to be statistically obvious and proceed.

#DF and ADF tests for stationarity in Y i.e. daily cases or daily deaths
#k is the number of lags, dickey fuller test for stationarity
#load the required library
library(tseries)

#adf for daily cases
adf.test(daily$daily_cases, "stationary", k=0)
adf.test(daily$daily_cases, "stationary")

#adf for diff(daily cases)
adf.test(d.cases, "stationary", k=0)
adf.test(d.cases, "stationary")

#adf for daily deaths
adf.test(daily$daily_deaths, "stationary", k=0)
adf.test(daily$daily_deaths, "stationary")

#adf for diff(daily deaths)
adf.test(d.deaths, "stationary", k=0)
adf.test(d.deaths, "stationary")

#We observe that the daily Cases is not stationary(p values > 0.05). When we take the differenced values of daily Cases, we achieve stationarity. (p values < 0.05). Similar is the case with daily deaths.

#The autocorrelation function (ACF) gives the autocorrelation at all possible lags. The autocorrelation at lag 0 is included by default which always takes the value 1 as it represents the correlation between the data and themselves. This function also helps in predicting which model to use under time series. 1. Autoregression (AR) model 2. Moving average (MA) model 3. ARMA (AR+MA) 4. ARIMA Autoregression Integrated Moving Average model.
As well as to get a rough estimate of the number of lags in the model.

#ACF and PACF graphs for visualising the differenced value of Positive cases
acf(d.cases, na.action = na.omit)
pacf(d.cases, na.action = na.omit)

acf(d.deaths, na.action = na.omit)
pacf(d.deaths, na.action = na.omit)
```

#As we can infer from the graph above, the autocorrelation continues to decrease as the lag increases, confirming that there is no linear association between observations separated by larger lags. Also, the autocorrelation is oscillating, meaning the coefficient of the dependent variable is negative.

ARIMA Model

```
library(forecast)
auto.arima(daily$daily_cases, trace=TRUE)
auto.arima(daily$daily_deaths, trace=TRUE)
```

#All the possible models are estimated here, Under ARIMA model (p,d,q) p = number of lags for autoregression (i.e. past values of Postive cases) d = number of times differenced (Integrated) q = number of lags of the residual value (i.e. past values of the unexplained error term)

#and it is observed that all estimated models have d = 1. As we already, saw earlier that out positive cases was not statonary. Hence, the auto.arima function made the series stationary by differencing it once. As tested earlier, which is stationary.

#The best model suggested is ARIMA(0,1,1) for daily cases and (1,1,1) for daily deaths with drift. We have to check the AIC/ BIC values for its minimal to choose the model. At the same time, the model should be parsimonious i.e. having lesser varaibles.

#ARIMA(0,1,1) and (1,1,1) is the best model as per lowest AIC value.

```
#daily cases
library(lmtest)
fitarima1 <- arima(daily$daily_cases, order = c(0,1,1))
coeftest(fitarima1)

#predict for next 14 days
fitarimadrift1 <- Arima(daily$daily_cases, order = c(0,1,1), include.drift = TRUE)
Next14days_drift1 <- forecast(fitarimadrift1, h=14, level = c(80, 95))

#plot(Next14days)
png(filename = "Figure 7 Daily Cases from 12th August.png")
plot(Next14days_drift1, xlab = 'Days starting from 11th March', ylab = 'Daily Cases', main =
'Figure 7: Daily no. of COVID-19 cases from August 12th, 2020 to \nAugust 25th, 2020 (Average
: 1550 cases) in Miami-Dade County, \nFlorida, USA')
dev.off()

#round of mean, lower bound and upper bound values
round(Next14days_drift1$mean)
```

```
round(Next14days_drift1$lower)
round(Next14days_drift1$upper)

#daily deaths
library(lmtest)
fitarima2 <- arima(daily$daily_deaths, order = c(1,1,1))
coeftest(fitarima2)

#predict for next 14 days
fitarimadrift2 <- Arima(daily$daily_deaths, order = c(1,1,1), include.drift = TRUE)
Next14days_drift2 <- forecast(fitarimadrift2, h=14, level = c(80, 95))

#Next14days <- forecast(fitarima, h=14, level = c(80, 95))
#plot(Next14days)
png(filename = "Figure 8 Daily deaths from 12th August.png")
plot(Next14days_drift2, xlab = 'Days starting from 11th March', ylab = 'Daily Deaths', main =
'Figure 8: Daily no. of COVID-19 deaths from August 12th, 2020 to \nAugust 25th, 2020
(Average : 31 deaths) in Miami-Dade County, \nFlorida, USA')
dev.off()

#round of mean, lower bound and upper bound values
round(Next14days_drift2$mean)
round(Next14days_drift2$lower)
round(Next14days_drift2$upper)
```