CMSC-6950 Project work

Muhammad Ifthkar, Prasun Thapa, Adnan Shafiq

06/22/2020

Github url: https://github.com/miaftab/finalproject

Introduction

The project is about the covid-19 Canada's data including all the provinces. Our main agenda of the project was to analyze, implement, append and plot the data in the visualized form. We are taking all the data associated with covid-19 from Canada's Health website:
https://health-infobase.canada.ca/src/data/covidLive/covid19.csv
The project was the perfect combo, to learn how to manage the source code using Github and python commands. We also learned to append, edit the csv files and create a plot diagrams based on the given data. Furthermore, it gave us a sense of testing the code, debugging and an environment to work on a team basis.

Source Code

```python
import matplotlib.pyplot as plt
from matplotlib.dates import date2num
import matplotlib.patches as mpatches
import datetime
import requests
import csv
import os
```

Importing Libraries

Function responsible for plotting the data over days

```python
def plotChartFunction(x_dates,setData,title):

    # plot chart on SetData
    x_dates = date2num(x_dates)
    ax = plt.subplot(111)
    legends = []

    # Colors Array
    colors = ['orange',
              'green',
              'red',
              'blue',
              'purple',
              'yellow',
              'grey',
              'magenta',
              'black',
              'cyan',
              'brown',
              'indigo',
              'olive',
              'navy',
              'orchid'
              ]

    count = 0
```

DataSet have key as each provence name and it’s data list as value

```
for key,value in setData.items():
        ax.plot(x_dates, value, color=colors[count])
        a=mpatches.Patch(color=colors[count],linestyle='--',label=key)
        legends.append(a)
        count = count + 1
    ax.xaxis_date()
    plt.xticks(rotation=70)
    plt.legend(handles=legends)
    plt.title(title,size=10,color='Green')
    plt.show()
```

Global plotting function created, which is responsible for plotting data over days. It contain three arguments xdates, setData, title. Xdates will be on the X-axis while title will be the title of graph plot. The most important parameter is setData which contains key value pairs. Keys are the name of provinces and value will be the corresponding data against particular key. An array of different colors used to differentiate between all provinces in sub plotting. At the end with the help of iteration loop all graphs plotted one by one.

Automating the download of latest Covid data

Listing 1: Question- 1 .

```python
try:
    os.remove("covid19.csv")
except:
    print("File not exist")


# Download new covid19 csv from HIC website
req = requests.get("https://health-infobase.canada.ca/src/data/covidLive/covid19.csv")
url_content = req.content
csv_file = open('covid19.csv', 'wb')
csv_file.write(url_content)
csv_file.close()

# open the file and read
with open('covid19.csv', 'r') as infile:
    # read the file as a dictionary for each row ({header : value})
    reader = csv.DictReader(infile)
    data = {}
    for row in reader:
        for header, value in row.items():
            try:
                data[header].append(value)
            except KeyError:
                data[header] = [value]

# extract the variables you want unique
provences = list(set(data['prname']))
totalCases = data['numtotal']

# Select only unique dates sorted
date = list(set(data['date']))
date = sorted(date, key=lambda x: datetime.datetime.strptime(x, '%d-%m-%Y'))

# Extracting dates and converting it to date time object
x_dates = []
eachProviceCases = {}

# append Unique dates in x_dates
for d in date:
    splitedDate = d.split('-')
    temp = datetime.date(int(splitedDate[2]),int(splitedDate[1]),int(splitedDate[0]))
    x_dates.append(temp)

```

```
![alt text](Q1_plot.png)
```

Question 1 is basically related to keep data updated. So every time at the start of program this function will check that either there is any existing file with the name “covid-19” in our system, if is exist it will be deleted automatically and then function will send a request for new file on internet. After downloading file, it will be open in writing mode. Needed data extracted from the file on the bases of keys. Dates are formatted as well as converted into the numbers. One important thing is that province names and dates extracted uniquely.

Total No of cases in each province over time

Listing 2: Question- 2 .

```
for prov in provences:

    # Exclude Canada
    if prov != 'Canada':
        # add province name as key and empty list as value
        eachProviceCases[prov] = []

        # append list of zeros equal to no of dates against province
        for d in date:
            eachProviceCases[prov].append(0)

        for index,d in enumerate(date): # iterate on sorted date list
            for idx,da in enumerate(data["date"]): # match it within date column of data
                if da == d and prov == data['prname'][idx]: # if date match and province name is equal
                    eachProviceCases[prov][index] = int(data['numtotal'][idx]) # add data against province for given date
                    break

# plot graph Total No of cases in each provence over time
plotChartFunction(x_dates,eachProviceCases,'Total No of cases in each provence over time')
```

A unique name of provinces is created, and the names are extracted from those created province name. The Canada is excluded here, province named has been passed to an empty key value pair creating an empty list as a value. The number of zeros is being appended in the above empty list, which is equal to the number of dates. Sorted date list are being enumerated and each date is iterated through date column. The province name and the date are checked if they are equal. Then the data is converted into integer and is saved in the particular province name.

A common function called plotchartFunction is used, which takes x-dates as the parameter, representing the X-axis, each province cases representing the Y-axis and the last parameter is the title of the chart.

Question 2 Graph

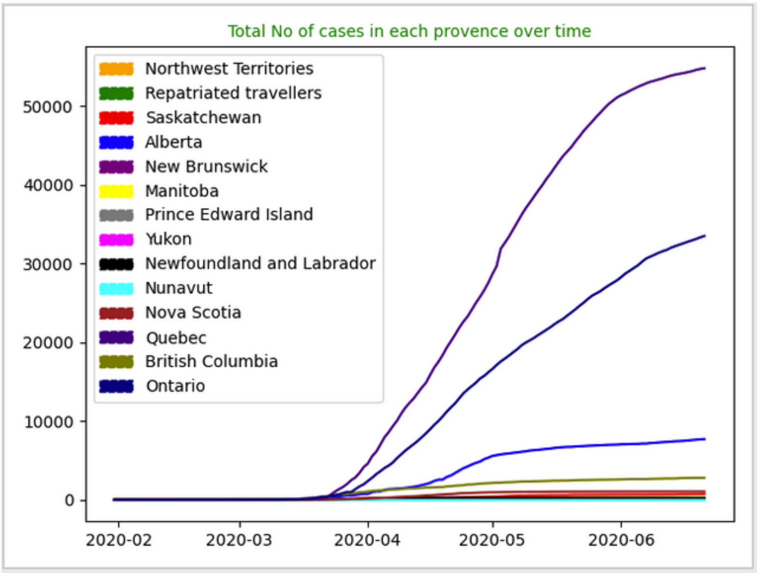


Figure 1: Total No of cases in each provence over time

Passing those data, we came up with this plot, you can see these purple lines indicates the number of cases in Ontario, which is more than 50,000 cases and so on. These datas are mainly rising from the month of march.

Total Number of individuals tested in each province overtime

Listing 3: Question- 3 .

```
eachProviceTestedCases = {}

# iterate through each provence
for prov in provences:

    # Exclude Canada
    if prov != 'Canada':

        # add province name as key and empty list as value
        eachProviceTestedCases[prov] = []

        # append list of zeros equal to no of dates against province
        for d in date:
            eachProviceTestedCases[prov].append(0)

        for index,d in enumerate(date):  # iterate on sorted date list
            for idx,da in enumerate(data["date"]):  # match it within date column of data
                if da == d and prov == data['prname'][idx]: # if date match and province name is equal
                    if data['numtested'][idx] != '':
                        eachProviceTestedCases[prov][index] = int(data['numtested'][idx])  # add data against province for given date
                    else:
                        eachProviceTestedCases[prov][index] = 0
                    break
# plot graph Total No of cases tested each provence over time
plotChartFunction(x_dates,eachProviceTestedCases,'Total No of cases tested each provence over time')
```

The source used to solve question number 2 is very similar to the one that has been used to solve question 3. It, basically follows the same basic principle. The only major difference was numtotal column name was replaced with numtested. Where numtested column represents the total number of individuals tested in each province. Again, the plotchartFunction is used in order to plot and name the title of the chart below.

Question 3 Graph

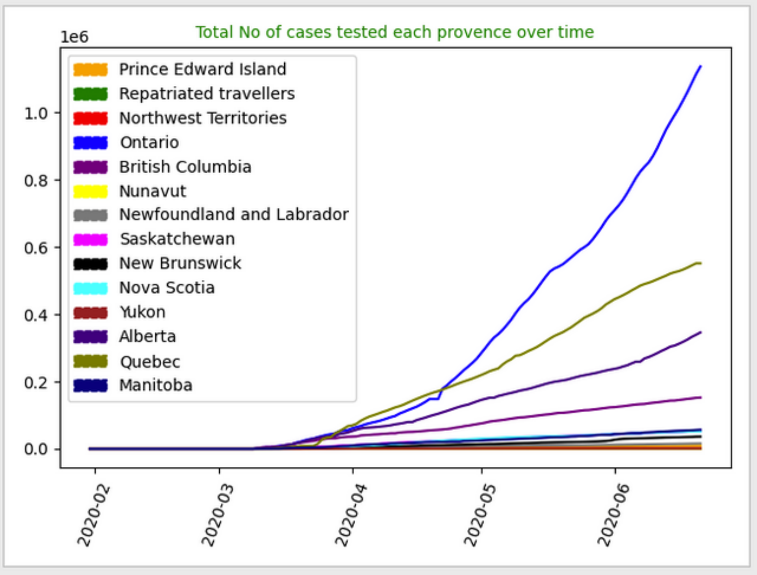


Figure 2: Total Number of individuals tested in each province overtime

From the above graph, we can clearly depict as Ontario leading the plots with the most number of individual tests. Quebec takes the second lead on the plot, followed by Alberta.

Total No of new cases per day in each province over time

Listing 4: Question- 4 .

```
newCasesPerDay = {}
dates = set()
# iterate through each provence
for prov in provences:

    # Exclude Canada
    if prov != 'Canada':

        # add province name as key and empty list as value
        newCasesPerDay[prov] = []

        for index,d in enumerate(date): # iterate on sorted date list

            proceed = False
            # check if all provinces have zero data for a specific date
            for idx2,checkDate in enumerate(data["date"]):
                if checkDate == d and int(data['numtoday'][idx2]) > 50:
                    proceed = True
                    break

            if proceed:
                dates.add(d)
                ignore = False
                for idx,da in enumerate(data["date"]):  # match it within date column of data
                    if da == d and prov == data['prname'][idx]: # if date match and province name is equal
                        if data['numtoday'][idx] != '':
                            newCasesPerDay[prov].append(int(data['numtoday'][idx])) # add data against province for given date
                        else:
                            newCasesPerDay[prov].append(0)
                        ignore=True
                        break
                if not ignore:
                    newCasesPerDay[prov].append(0)

date = list(dates)
date = sorted(date, key=lambda x: datetime.datetime.strptime(x, '%d-%m-%Y'))
x_dates = []
```

```
for d in date:
    splitedDate = d.split('-')
    temp = datetime.date(int(splitedDate[2]),int(splitedDate[1]),int(splitedDate[0]))
    x_dates.append(temp)


# plot graph Total No of new cases per day each provence over time
plotChartFunction(x_dates,newCasesPerDay,'No of new cases per day each provence over time')
```

Again, the key value pairs is created and canada is being exculded from the list. For this question, a minimum cut-off score was determined in order to figure out to extract the value form the particular date. The data are being verified if they are equal, going through the data sets. If any of the province for a particular date is going to get a value greater than 50, the proceed check turns true. If the proceed is true, for that particular date the data is stored for the plot. Rest of the source code is similar to 2 and 3, but here equal number of dates is given excluding some dates above. Not including all the dates was our main concern here. Finally, the same plot chart Function is used to plot a graph against new cases per day over time.

Question 4 Graph

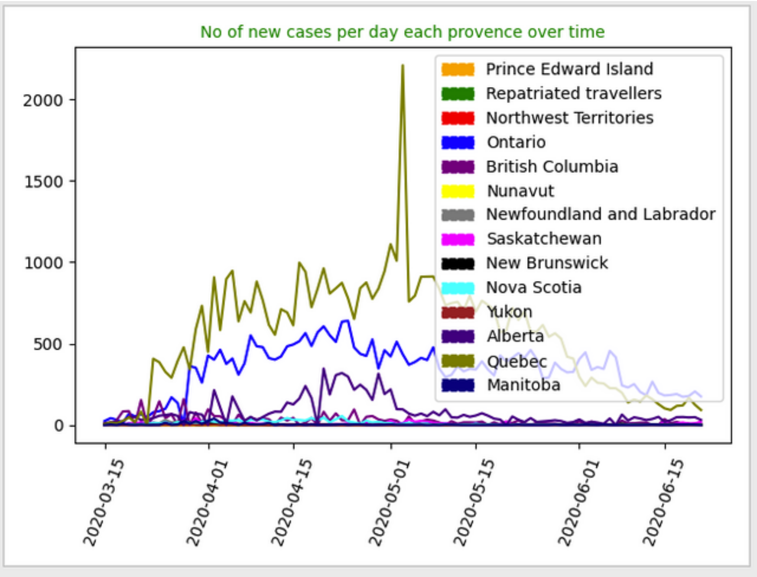


Figure 3: Total No of new cases per day in each province over time

Here we are excluding all the zeros starting from 50. Quebec are more cases compared to any other followed by Ontario.

Doubling Rate of the Number of Cases

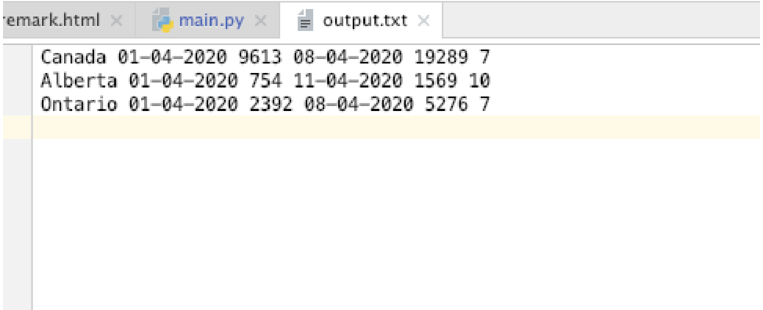
Listing 5: Doubling Rate.

```
try:
    os.remove("output.txt")
except:
    print("File not exist")

os.system("bash doubling.sh  Canada 'number of cases' 01-04-2020")
os.system("bash doubling.sh  Alberta 'number of cases' 01-04-2020")
os.system("bash doubling.sh  Ontario 'number of cases' 01-04-2020")

f = open("output.txt", "r")
prvinceName = []
NoOfDays=[]
legends=[]
colors = ['r','g','b']
index=0
for x in f:
    temp = x.split(" ")
    a=mpatches.Patch(color=colors[index],linestyle='--',label=temp[0]+'('+↩
       temp[2]+') --> '+'('+temp[4]+')')
    legends.append(a)
    prvinceName.append(temp[0]+'('+temp[1]+')')
    NoOfDays.append(int(temp[5].replace('\n', '')))
    index=index+1
print(prvinceName)
print(NoOfDays)

barlist=plt.bar(prvinceName,NoOfDays)
barlist[0].set_color('r')
barlist[1].set_color('g')
barlist[2].set_color('b')

plt.legend(handles=legends)
plt.title('Doubling Rate From Given Date')
plt.xlabel('Provinces From Date')
plt.ylabel('No Of Days')

plt.show()
\begin{lstlisting}[label={list:fifth},caption= Question- 5 .]
```

A shell script called 'doubling.sh' is used to compute the doubling rate of the cases. To compute the doubling rate through the shell script, three arguments are passed. For instance, the three arguments are province name, number of cases or number of deaths and date required to complete the doubling rate respectively. example
bash doubling.sh Canada'number of cases'01-04-2020

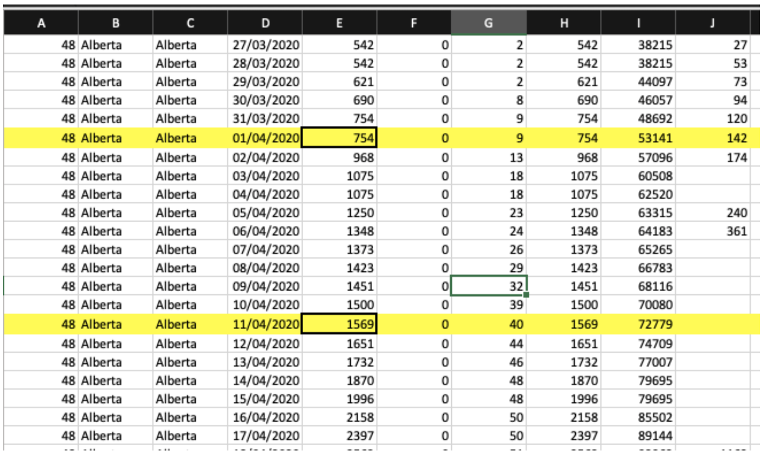
The output of doubling.sh is stored at output.txt. These data from the output.txt file are further used to plot the graph representing the doubling rate for minimum three provinces. To clarify, a screenshot is added below where the number of cases in Alberta was 754 and it took 10 days for it to exactly double the value and got close to 1569.

Question 5 Graph



```
Canada 01-04-2020 9613 08-04-2020 19289 7
Alberta 01-04-2020 754 11-04-2020 1569 10
Ontario 01-04-2020 2392 08-04-2020 5276 7
```

Figure 4: Doubling Rate of the Number of Cases



A	B	C	D	E	F	G	H	I	J
48	Alberta	Alberta	27/03/2020	542	0	2	542	38215	27
48	Alberta	Alberta	28/03/2020	542	0	2	542	38215	53
48	Alberta	Alberta	29/03/2020	621	0	2	621	44097	73
48	Alberta	Alberta	30/03/2020	690	0	8	690	46057	94
48	Alberta	Alberta	31/03/2020	754	0	9	754	48692	120
48	Alberta	Alberta	01/04/2020	754	0	9	754	53141	142
48	Alberta	Alberta	02/04/2020	968	0	13	968	57096	174
48	Alberta	Alberta	03/04/2020	1075	0	18	1075	60508	
48	Alberta	Alberta	04/04/2020	1075	0	18	1075	62520	
48	Alberta	Alberta	05/04/2020	1250	0	23	1250	63315	240
48	Alberta	Alberta	06/04/2020	1348	0	24	1348	64183	361
48	Alberta	Alberta	07/04/2020	1373	0	26	1373	65265	
48	Alberta	Alberta	08/04/2020	1423	0	29	1423	66783	
48	Alberta	Alberta	09/04/2020	1451	0	32	1451	68116	
48	Alberta	Alberta	10/04/2020	1500	0	39	1500	70080	
48	Alberta	Alberta	11/04/2020	1569	0	40	1569	72779	
48	Alberta	Alberta	12/04/2020	1651	0	44	1651	74709	
48	Alberta	Alberta	13/04/2020	1732	0	46	1732	77007	
48	Alberta	Alberta	14/04/2020	1870	0	48	1870	79695	
48	Alberta	Alberta	15/04/2020	1996	0	48	1996	79695	
48	Alberta	Alberta	16/04/2020	2158	0	50	2158	85502	
48	Alberta	Alberta	17/04/2020	2397	0	50	2397	89144	

Figure 5: Doubling Rate of the Number of Cases

Doubling Rate (.sh scripts)

Listing 6: .sh scripts .

```
numTotalindex=0;
dateindex=0;
provinceindex=0;
numDeathsindex=0;

var=$(cat covid19.csv | head -n 1 | tr ',' ' ')
echo $var
for colNames in $var
```

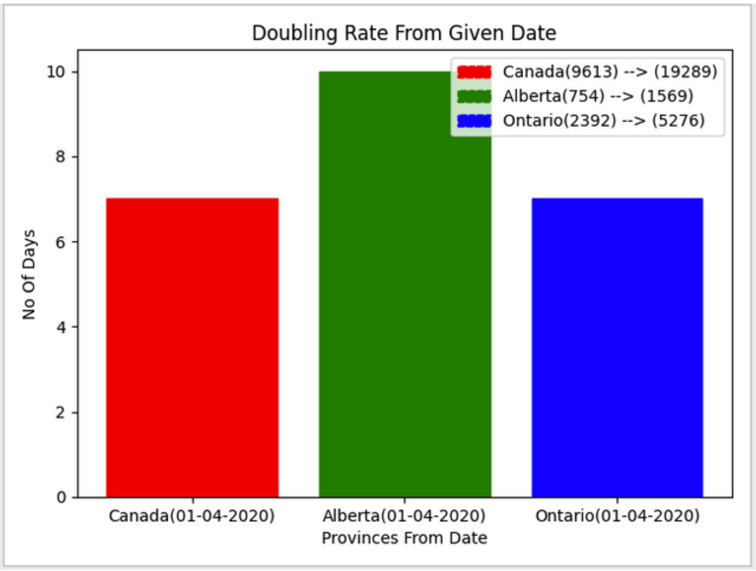


Figure 6: Doubling Rate Ploting

```
do
 if [ "numtotal" = $colNames ]
 then
   echo $colNames
   echo $numTotalindex
   break
 fi
 numTotalindex=$((numTotalindex+1))
done

for colNames in $var
do
 if [ "date" = $colNames ]
 then
   echo $colNames
   echo $dateindex
   break
 fi
 dateindex=$((dateindex+1))
done

for colNames in $var
do
 if [ "prname" = $colNames ]
 then
   echo $colNames
   echo $provinceindex
   break
 fi
 provinceindex=$((provinceindex+1))
```

```
done

for colNames in $var
do
 if [ "numdeaths" = $colNames ]
 then
   echo $colNames
   echo $numDeathsindex
   break
 fi
 numDeathsindex=$((numDeathsindex+1))
done

convertDate()
{
  var=$(echo $1 | tr "-" " " )
  arr=($var)
  d1=${arr[2]}"-"${arr[1]}"-"${arr[0]}
  echo "$d1"
}

# Find Doubling Rate For Number Of Cases
if [ "$2" = "number of cases" ]
then
   doublingRate=0
   noOfdays=0
   arr=’’
   echo "number of cases"
   numberOfCsvRows=$(cat covid19.csv  | wc -l)
   for((i=2;i<=numberOfCsvRows;i++))
   do
     var=$(cat covid19.csv | head -n "$i" | tail -n 1 | tr ’,’ ’ ’)
     arr=($var)
     d1=$(convertDate ${arr[dateindex]})
     d2=$(convertDate "$3")
     if [ "$1" = ${arr[provinceindex]} ] && [ "$d1" ">" "$d2" ]
     then
       #echo $d1
       #echo $d2
       echo $var
       if [ ${arr[numTotalindex]} -gt "$doublingRate" ]
       then
         noOfdays=$((noOfdays+1))
         break
       else
        noOfdays=$((noOfdays+1))
       fi
     elif [ "$1" = ${arr[provinceindex]} ] && [ "$d1" "=" "$d2" ]
     then
      doublingRate=$((${arr[numTotalindex]} * 2))
      echo $doublingRate
     fi
   done

   echo 'No Of Days = '$noOfdays
   echo $1 $3 $(($doublingRate / 2)) ${arr[dateindex]} ${arr[numTotalindex]} $noOfdays  >> output.txt

else
   doublingRate=0
   noOfdays=0
   arr=''
   echo "number of deaths"
   numberOfCsvRows=$(cat covid19.csv  | wc -l)
   for((i=2;i<=numberOfCsvRows;i++))
   do
     var=$(cat covid19.csv | head -n "$i" | tail -n 1 | tr ',' ' ')
     arr=($var)
     d1=$(convertDate ${arr[dateindex]})
     d2=$(convertDate "$3")
     if [ "$1" = ${arr[provinceindex]} ] && [ "$d1" ">" "$d2" ]
     then
       #echo $d1
       #echo $d2
       echo $var
       if [ ${arr[numDeathsindex]} -gt "$doublingRate" ]
       then
         noOfdays=$((noOfdays+1))
         break
       else
        noOfdays=$((noOfdays+1))
       fi
     elif [ "$1" = ${arr[provinceindex]} ] && [ "$d1" "=" "$d2" ]
     then
      doublingRate=$((${arr[numDeathsindex]} * 2))
      echo $doublingRate
     fi
   done

   echo 'No Of Days = '$noOfdays
   echo $1 $3 $(($doublingRate / 2)) ${arr[dateindex]} ${arr[numDeathsindex]} $noOfdays  >> output.txt
fi
```

Doubling script is a shell script. It comprises of shell commands and with the help of pipelines and filters, doubling rate is being computed. Column index for required values like prname, numdeaths etc are identified to make the shell script dynamic. A loop is executed for number of rows and for each row, the code verifies whether the date is equal to the desired input date. The number of cases is computed on the basis of input date and it looks for the index where the value is being doubled. The number of days in parallel is counted and stored securely on Output.txt.

Predicted data for next 7 days

Listing 7: FREE CHOICE .

```
eachProviceCases={}
for prov in provences:

    # Exclude Canada
    if prov != 'Canada':
        # add province name as key and empty list as value
        eachProviceCases[prov] = []

        # append list of zeros equal to no of dates against province
        for d in date:
            eachProviceCases[prov].append(0)

        for index,d in enumerate(date): # iterate on sorted date list
            for idx,da in enumerate(data["date"]): # match it within date column of data
                if da == d and prov == data['prname'][idx]: # if date match and province name is equal
                    eachProviceCases[prov][index] = int(data['numtotal'][idx]) # add data against province for given date
                    break

x_dates=[]

for i in range((0),14):
    d=datetime.datetime.now() + datetime.timedelta(days=i)
    x_dates.append(d.date())

for prov in provences:
    # Exclude Canada
    if prov != 'Canada':
        eachProviceCasesUpdated={}
        Stats = eachProviceCases[prov]
        last7Values=[]
        nextTwoWeekData=[]

        for i in range((0), len(Stats)):
            if i >= (len(Stats) - 7):
                last7Values.append(Stats[i])
        print(prov)
        print(last7Values)

        #generate next 2 week data
        for i in range((0),14):
            count=0
            sum=0
            for j in range((i+1),len(last7Values)):
               sum = sum + (last7Values[j] - last7Values[j-1])
               count=count+1
            last7Values.append(last7Values[len(last7Values)-1] + (sum/count))
            nextTwoWeekData.append(last7Values[len(last7Values)-1] + (sum/count))
            print(last7Values[len(last7Values)-1] + (sum/count))

        eachProviceCasesUpdated[prov] = nextTwoWeekData
        print(x_dates)
        print(eachProviceCasesUpdated)
        plotChartFunction(x_dates,eachProviceCasesUpdated,'No of cases in '+prov+' forecast data')
```

Question Free Choice Graph

For the free choice task, the team decided to forecast the number of cases, data prediction for the upcoming next two weeks. A dictionary with sorted data for each province is ready and last seven days of data is excluded from the excel file. That data is used to take the average of difference in last 7 days values, adding it to the last value to predict the data for the next day. Moving are window across after adding the next value and repeating again until for next 14 days.

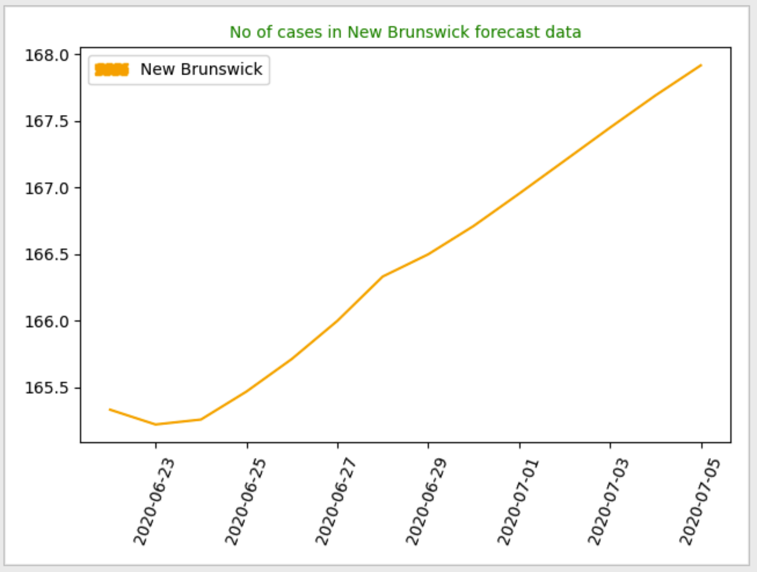


Figure 7: Forcast for burnswick

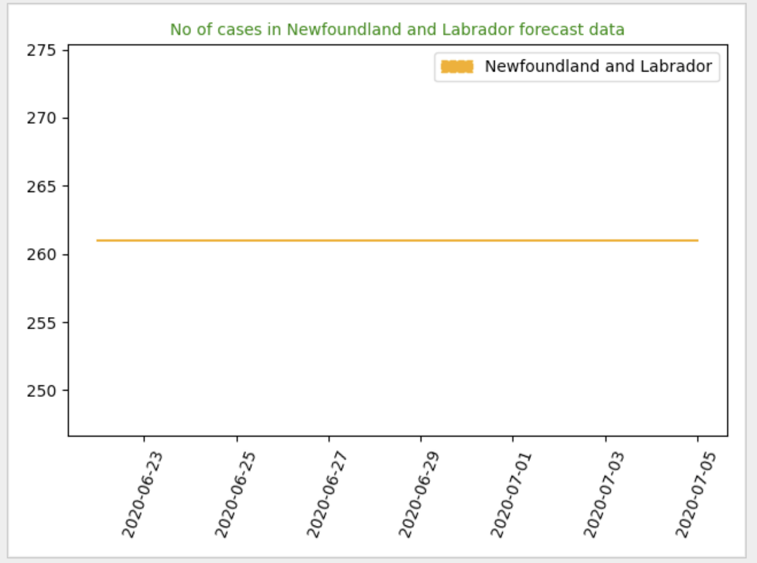


Figure 8: Forcast for NewFoundLand