# 1_data_cleaning

May 14, 2021

[ ]:
```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import datetime
from datetime import datetime as dt
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
%cd '/content/drive/MyDrive/CSE544_PROJECT'
# %ls -l
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
/content/drive/.shortcut-targets-by-
id/1YQyVsZWGB7sACOZzGllQA0QwFc_E5Nb1/CSE544_PROJECT
```

[ ]:
```
df = pd.read_csv('14.csv')
df.head()
```

[ ]:

| | Date | MT confirmed | NC confirmed | MT deaths | NC deaths |
|---|---|---|---|---|---|
| 0 | 2020-01-22 | 0 | 0 | 0 | 0 |
| 1 | 2020-01-23 | 0 | 0 | 0 | 0 |
| 2 | 2020-01-24 | 0 | 0 | 0 | 0 |
| 3 | 2020-01-25 | 0 | 0 | 0 | 0 |
| 4 | 2020-01-26 | 0 | 0 | 0 | 0 |

[ ]:
```
MT_cases = df['MT confirmed'].tolist()
NC_cases = df['NC confirmed'].tolist()
MT_death = df['MT deaths'].tolist()
NC_death = df['NC deaths'].tolist()
```

[ ]:
```
# Get daily data

def get_daily(sample_A):
  data_daily = [0] * len(sample_A)
  data_daily[0] = sample_A[0]
  for x in range(1, len(sample_A)):
```

```
    data_daily[x] = int(sample_A[x] - sample_A[x-1])
  return data_daily
```

[ ]:
```
MT_cases_daily = get_daily(MT_cases)
NC_cases_daily = get_daily(NC_cases)
MT_death_daily = get_daily(MT_death)
NC_death_daily = get_daily(NC_death)
```

[ ]:
```
# Remove negative values (negative values are incorrect information which may␣
 ↪not be removed by Tukey's rule if IQR is high)

def remove_negative(sample_A):
  count = 0
  for i in range(1, len(sample_A)):
    if(sample_A[i]<0 and sample_A[i-1]>=0 and sample_A[i+1]>=0):
      sample_A[i] = (sample_A[i-1]+sample_A[i+1])/2
      count += 1
  print(count)
  return sample_A
```

[ ]:
```
MT_cases_daily = remove_negative(MT_cases_daily)
NC_cases_daily = remove_negative(NC_cases_daily)
MT_death_daily = remove_negative(MT_death_daily)
NC_death_daily = remove_negative(NC_death_daily)
```

```
2
1
3
4
```

[ ]:
```
# Substitute missing values (It can be seen that for some of the days the value␣
 ↪of new cases or deaths is zero in times they should infact be non-zero.␣
 ↪These are the
# days for which data is missing and is marked as 0 in the dataset.)

def substitute_missing_data(sample_A):
  non_zero = 0
  count = 0
  for i in range(len(sample_A)):
    if sample_A[i] != 0:
      non_zero = i
      break
  print(non_zero)
  for i in range(non_zero, len(df)):
    if sample_A[i] == 0:
        sample_A[i] = sample_A[i-1]
        count +=1
```

```
  print(count)
  return sample_A
```

[ ]:
```
MT_cases_daily = substitute_missing_data(MT_cases_daily)
NC_cases_daily = substitute_missing_data(NC_cases_daily)
MT_death_daily = substitute_missing_data(MT_death_daily)
NC_death_daily = substitute_missing_data(NC_death_daily)
```

```
51
26
41
26
66
130
63
51
```

[ ]:
```
# apply tukeys rule for removing outliers

import statistics

def tukey(price_list):
  month_price_list = []
  # lst1 = price_list[i:i+30] for i in range(0,len(df)-30+1,30)]
  for i in range(0,len(price_list)-30+1,30):
    month_price_list.append(price_list[i:i+30])
  month_price_list.append(price_list[(math.floor(len(price_list)/30))*30:])
  price_list_tukey = []
  print("Here are the outliers.")
  total_outliers = 0
  for month in month_price_list:
    median = statistics.median(month)
    month_sorted = np.sort(month)
    q25 = month_sorted[math.ceil((25/100)*len(month))-1]
    q75 = month_sorted[math.ceil((75/100)*len(month))-1]
    iqr = q75 - q25
    cut_off = iqr * 1.5
    lower, upper = q25 - cut_off, q75 + cut_off
    numchanges = 0
    for i, x in enumerate(month):
        if x < lower or x > upper:
            month[i] = median
            numchanges += 1
    total_outliers += numchanges

    # print("outliers = ", numchanges)
    price_list_tukey.extend(month)
    # plt.plot(price_list_tukey)
```

```
  print(total_outliers)
  return price_list_tukey
```

[ ]:
```
MT_cases_daily = tukey(MT_cases_daily)
NC_cases_daily = tukey(NC_cases_daily)
MT_death_daily = tukey(MT_death_daily)
NC_death_daily = tukey(NC_death_daily)
```

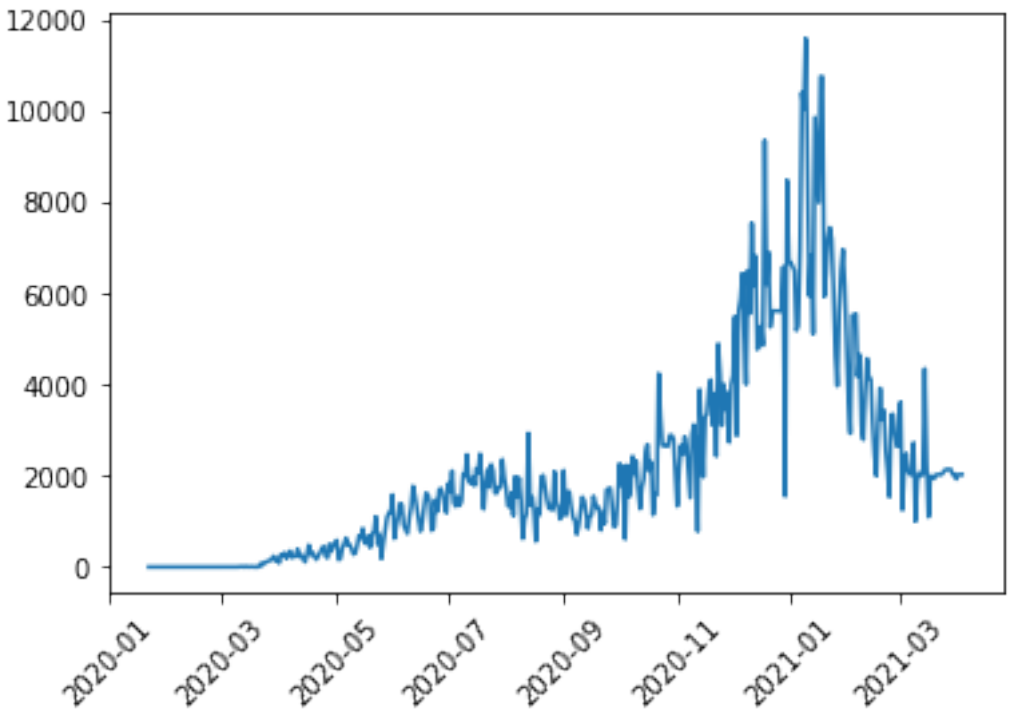
```
Here are the outliers.
12
Here are the outliers.
20
Here are the outliers.
28
Here are the outliers.
12
```

[ ]:
```
# Add up the daily cases to get the total cases till date

def cumulative_cases(sample_A):
  cum_cases = [0] * len(sample_A)
  total = 0
  for i in range(len(sample_A)):
    total += sample_A[i]
    cum_cases[i] = total
  return cum_cases
```

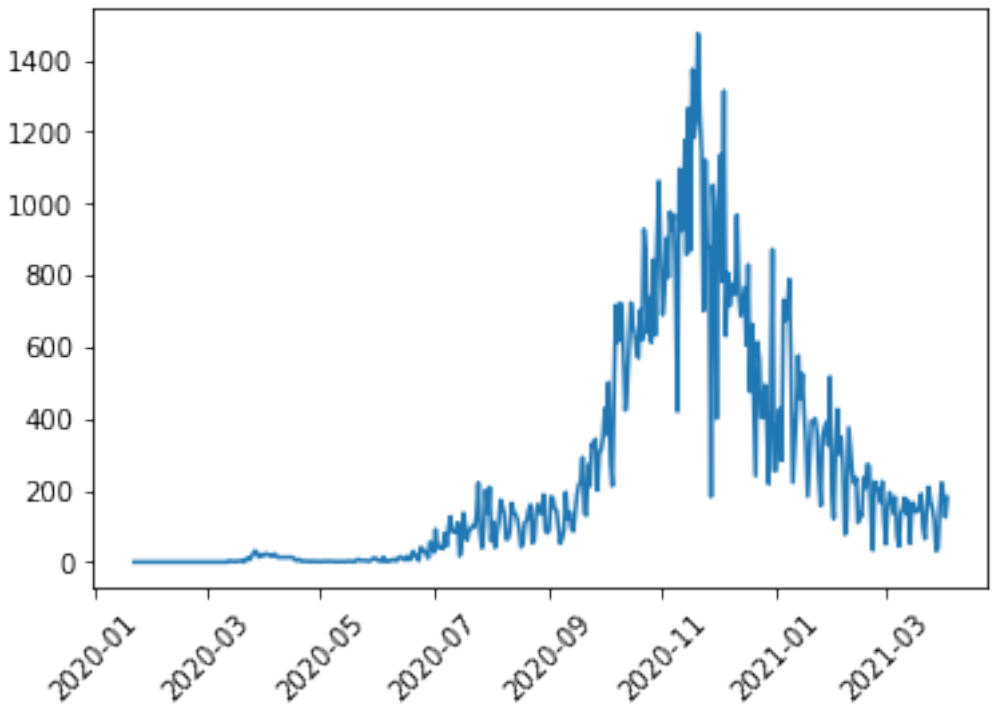
[ ]:
```
plt.plot(pd.to_datetime(df['Date']), NC_cases_daily)
plt.xticks(rotation=45)
```

[ ]:
```
(array([737425., 737485., 737546., 737607., 737669., 737730., 737791.,
        737850.]), <a list of 8 Text major ticklabel objects>)
```




[ ]:
```
plt.plot(pd.to_datetime(df['Date']), MT_cases_daily)
plt.xticks(rotation=45)
```

[ ]:
```
(array([737425., 737485., 737546., 737607., 737669., 737730., 737791.,
        737850.]), <a list of 8 Text major ticklabel objects>)
```




[ ]:
```
cumulative_MT_cases_daily = cumulative_cases(MT_cases_daily)
cumulative_NC_cases_daily = cumulative_cases(NC_cases_daily)
cumulative_MT_death_daily = cumulative_cases(MT_death_daily)
cumulative_NC_death_daily = cumulative_cases(NC_death_daily)
```

[ ]:
```
# convert date format to mm/dd/yyyy

def date_convert(sample_A):
  for i in range(len(sample_A)):
    d = datetime.datetime.strptime(sample_A[i], '%Y-%m-%d')
    sample_A[i] = d.strftime('%m/%d/%Y')
  return sample_A

date = df['Date'].tolist()
date = date_convert(date)
```

[ ]:
```
dict = {'Date': date, 'MT daily cases': MT_cases_daily, 'NC daily cases':␣
 ↪NC_cases_daily, 'MT daily death': MT_death_daily, 'NC daily death':␣
 ↪NC_death_daily, 'MT cumalative cases': cumulative_MT_cases_daily, 'NC␣
 ↪cumalative cases': cumulative_NC_cases_daily, 'MT cumalative death':␣
 ↪cumulative_MT_death_daily, 'NC cumalative death': cumulative_NC_death_daily}
df1 = pd.DataFrame(dict)
```

[ ]:
```
df1.to_csv('clean_organised.csv')
```

[ ]:
```

```