data_preprocess

May 17, 2022

[1]:
```python
import pandas as pd
import numpy as np
import datetime
```

1 Data Cleaning Utilities

Tukey rule for removing outliers

[2]:
```python
def tukey_range(values):

    alpha = 1.5
    sorted_values = sorted(values)

    # Q1
    q1_index = int(np.ceil(0.25 * len(values)))
    q1 = sorted_values[q1_index]

    # Q3
    q3_index = int(np.ceil(0.75 * len(values)))
    q3 = sorted_values[q3_index]

    # IQR
    iqr = q3 - q1

    return q1 - (alpha * iqr), q3 + (alpha * iqr)
```

Remove missing, invalid entries and outliers

[3]:
```python
def clean_data(series):

    #Check if any NA values
    print("Missing Values:{}\n".format(len(series[series.isna()])))

    invalid_entries = series[series < 0].values
    print("Invalid entries: {}\n".format(invalid_entries))

    series = series[series >=0]
    lower_limit, upper_limit = tukey_range(series.values)
```

```
    print("Lower Range: {}, Upper Range: {}\n".format(lower_limit, upper_limit))

    outliers = list(series[series < lower_limit].values) + list(series[series >␣
↪upper_limit].values)
    print("Total Outliers: {}\n".format(len(outliers)))
    print("Outliers: \n{}\n".format(outliers))

    outliers_index = list(series[series < lower_limit].index) +␣
↪list(series[series > upper_limit].index)
    return series.drop(index = outliers_index)
```

2 Cases

[4]:
```
cases = pd.read_csv("dataset/
↪United_States_COVID-19_Cases_and_Deaths_by_State_over_Time.csv")
```

[5]:
```
cases.head()
```

[5]:
```
  submission_date state  tot_cases  conf_cases  prob_cases  new_case  \
0      12/01/2021    ND     163565    135705.0     27860.0       589
1      08/17/2020    MD     100715         NaN         NaN       503
2      03/28/2022    VT     107785         NaN         NaN       467
3      03/18/2020    ME         44        44.0         0.0        12
4      02/06/2020    NE          0         NaN         NaN         0

   pnew_case  tot_death  conf_death  prob_death  new_death  pnew_death  \
0      220.0       1907         NaN         NaN          9         0.0
1        0.0       3765      3616.0       149.0          3         0.0
2       35.0        585         NaN         NaN          0         0.0
3        0.0          0         0.0         0.0          0         0.0
4        NaN          0         NaN         NaN          0         NaN

               created_at consent_cases consent_deaths
0  12/02/2021 02:35:20 PM         Agree      Not agree
1  08/19/2020 12:00:00 AM           NaN          Agree
2  03/29/2022 01:30:11 PM     Not agree      Not agree
3  03/20/2020 12:00:00 AM         Agree          Agree
4  03/26/2020 04:22:39 PM         Agree          Agree
```

[6]:
```
cases["submission_date"] = pd.to_datetime(cases["submission_date"])
```

3 Georgia

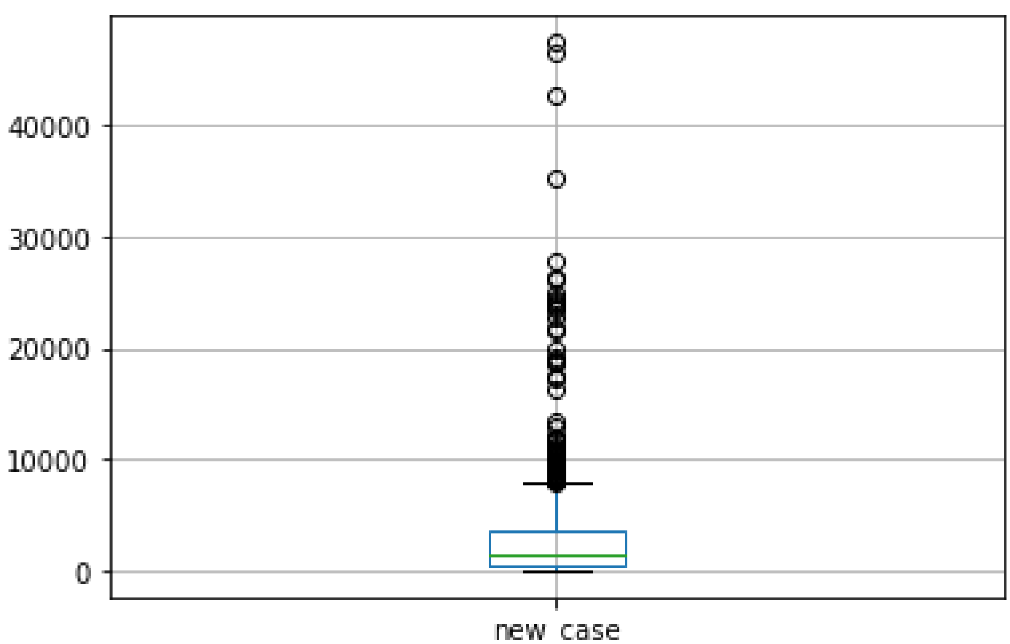
[7]:
```
georgia_df = cases[cases["state"].isin(['GA'])].sort_values(by =␣
↪"submission_date").set_index("submission_date")
georgia_case_death = georgia_df[["new_case", "new_death"]]
```

[8]:
```
georgia_case_death
```

[8]:
```
                 new_case  new_death
submission_date
2020-01-22              0          0
2020-01-23              0          0
2020-01-24              0          0
2020-01-25              0          0
2020-01-26              0          0
…                     …        …
2022-05-02              0          0
2022-05-03              0          0
2022-05-04           6525        163
2022-05-05              0          0
2022-05-06              0          0

[836 rows x 2 columns]
```

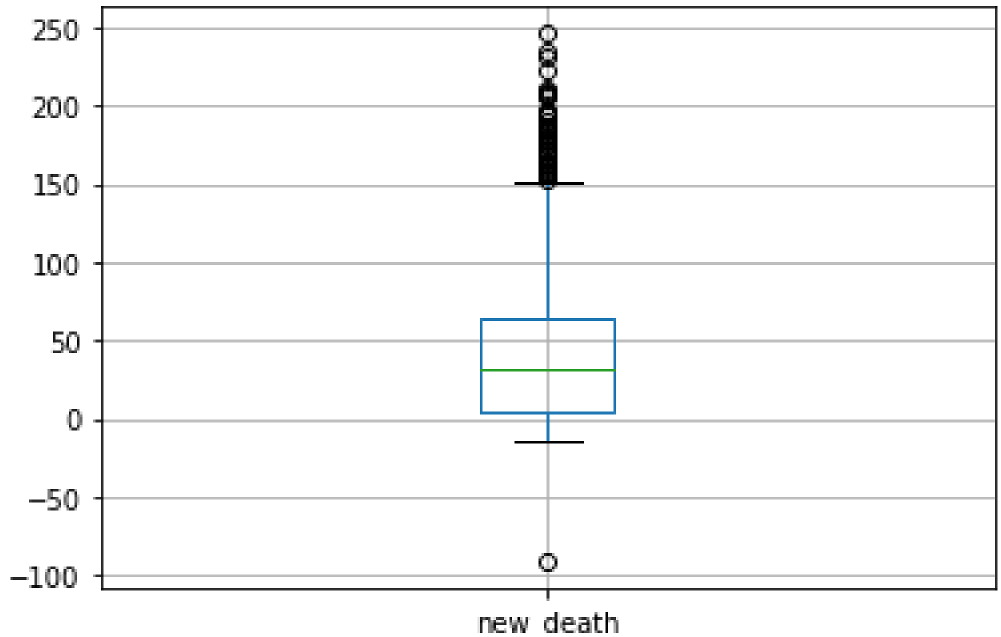
[15]:
```
georgia_case_death.boxplot("new_case")
```

[15]: <AxesSubplot:>



[16]:
```
georgia_case_death.boxplot("new_death")
```

[16]: <AxesSubplot:>



4 Indiana

[9]:
```
indiana_df = cases[cases["state"].isin(['IN'])].sort_values(by =␣
 ↪"submission_date").set_index("submission_date")
indiana_case_death = indiana_df[["new_case", "new_death"]]
```

[10]:
```
indiana_case_death
```

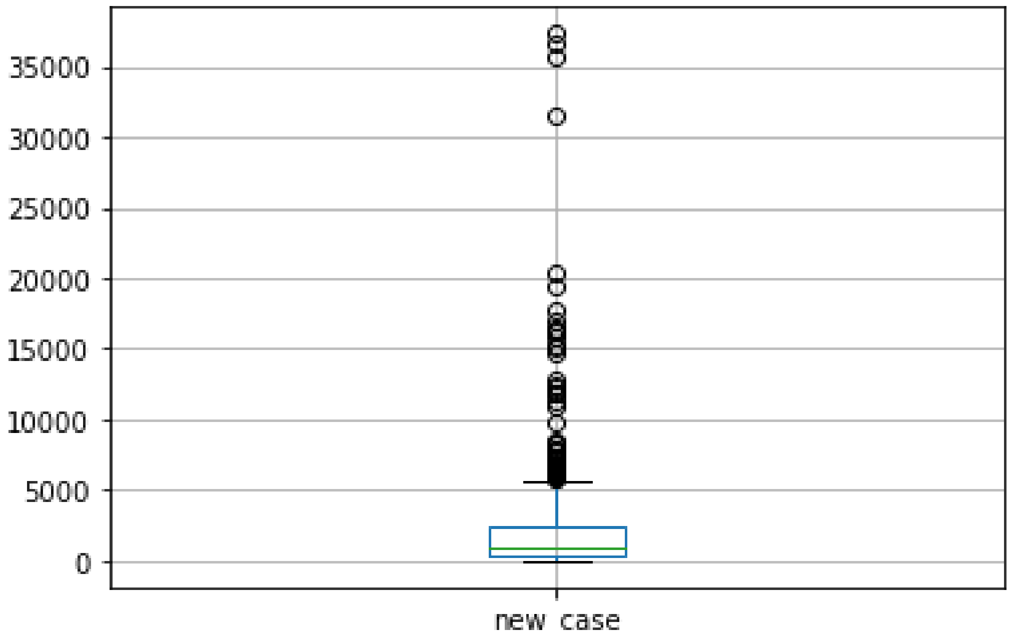
[10]:

	new_case	new_death
submission_date		
2020-01-22	0	0
2020-01-23	0	0
2020-01-24	0	0
2020-01-25	0	0
2020-01-26	0	0
…	…	…
2022-05-02	1287	4
2022-05-03	0	0
2022-05-04	1815	0

```
2022-05-05              0          0
2022-05-06           1691         23

[836 rows x 2 columns]
```

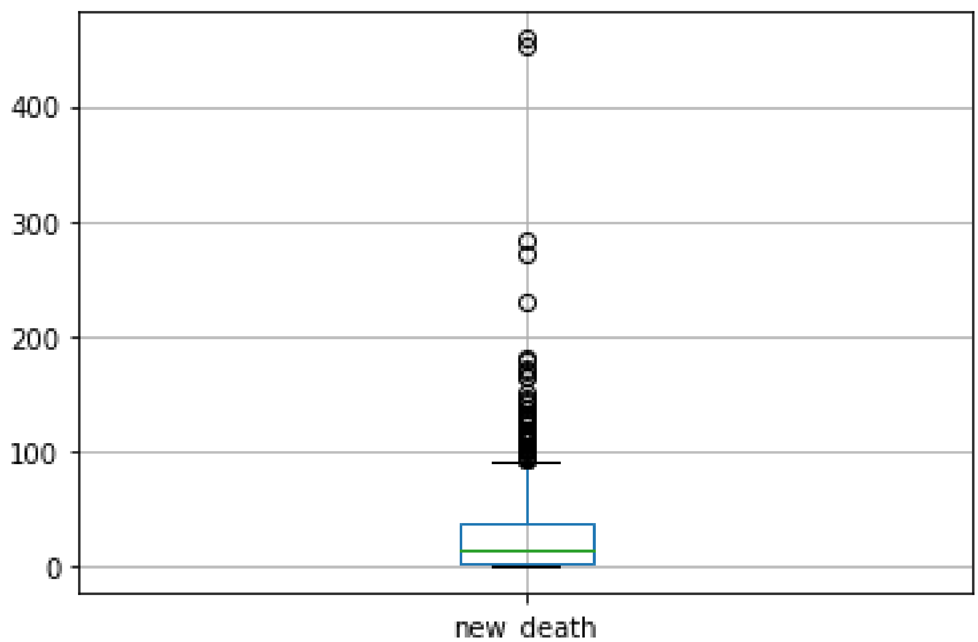
[13]:
```python
indiana_case_death.boxplot("new_case")
```

[13]: <AxesSubplot:>



[14]:
```python
indiana_case_death.boxplot("new_death")
```

[14]: <AxesSubplot:>



[143]:
```
dfs = {"Georgia":georgia_case_death, "Indiana":indiana_case_death}
res = []
for state, df in dfs.items():
    print("State:{}".format(state))
    print("-------------------------------------------------")
    print("-------------------------------------------------")
    s = []
    for col in df.columns:
        print("Feature: {}".format(col))
        print("----------------------------------")
        clean_series = clean_data(df[col])
        # clean_series.index = df.index
        s.append(clean_series)
    res.append(pd.concat(s, axis = 1))
georgia, indiana = [r.dropna(subset = ["new_case", "new_death"]) for r in res]
```

```
State:Georgia
-------------------------------------------------
-------------------------------------------------
Feature: new_case
----------------------------------
Missing Values:0

Invalid entries: [-5]

Lower Range: -3912.5, Upper Range: 8019.5
```

```
Total Outliers: 68

Outliers:
[8141, 9079, 10286, 9450, 9053, 11709, 11137, 10091, 9790, 13296, 11926, 8193,
9193, 8596, 9036, 9806, 8533, 8205, 8150, 8374, 8985, 8393, 9836, 9581, 9186,
10677, 10823, 11084, 10521, 9702, 8412, 12018, 8792, 8278, 9089, 10012, 9495,
26279, 13018, 19124, 24420, 23438, 47436, 17603, 24024, 23813, 26033, 46474,
24865, 18939, 24310, 19943, 35149, 12071, 27781, 22684, 18671, 42786, 16232,
17332, 21708, 18785, 21534, 10226, 8020, 8029, 8798, 11804]

Feature: new_death
----------------------------------
Missing Values:0

Invalid entries: [ -2  -2 -14 -91  -1  -1]

Lower Range: -81.0, Upper Range: 151.0

Total Outliers: 39

Outliers:
[157, 187, 172, 163, 153, 222, 196, 171, 186, 179, 161, 184, 159, 210, 208, 169,
232, 171, 195, 246, 207, 193, 156, 235, 197, 179, 182, 205, 176, 158, 180, 195,
191, 209, 158, 175, 164, 185, 163]

State:Indiana
-------------------------------------------------
-------------------------------------------------
Feature: new_case
----------------------------------
Missing Values:0

Invalid entries: []

Lower Range: -2842.0, Upper Range: 5750.0

Total Outliers: 64

Outliers:
[6591, 8322, 6710, 6015, 7282, 6809, 6873, 6175, 5984, 6374, 6597, 8460, 7899,
7690, 6598, 5754, 6518, 7200, 7401, 5898, 6220, 6352, 5961, 6495, 6208, 6480,
6300, 6158, 7282, 6107, 5978, 6728, 5996, 7922, 6004, 12384, 11385, 10894,
20371, 7914, 11935, 31431, 8305, 12890, 15208, 14735, 37331, 11808, 15012,
16447, 15856, 36579, 11833, 16207, 16370, 17661, 35590, 9733, 12198, 17067,
19367, 6913, 6614, 6438]

Feature: new_death
```

```
----------------------------------
Missing Values:0

Invalid entries: []

Lower Range: -49.5, Upper Range: 90.5

Total Outliers: 52

Outliers:
[113, 103, 141, 109, 128, 96, 96, 128, 132, 143, 164, 109, 103, 110, 147, 272,
179, 453, 283, 231, 459, 93, 95, 119, 93, 92, 96, 104, 119, 170, 172, 152, 166,
182, 112, 127, 103, 115, 137, 114, 132, 128, 109, 136, 131, 111, 100, 99, 117,
108, 126, 103]
```

[147]:
```
georgia.reset_index().to_csv("georgia_cases_deaths.csv", index = False)
indiana.reset_index().to_csv("indiana_cases_deaths.csv", index = False)
```

[180]:
```
# Calculating Outliers for all states
# all_states_df = cases.sort_values(by = "submission_date").
↪set_index("submission_date")
# all_states_case_death = all_states_df[["new_case", "new_death"]]
# s = []
# for col in all_states_case_death.columns:
#     print("Feature: {}".format(col))
#     print("----------------------------------")
#     clean_series = clean_data(all_states_case_death[col])
#     s.append(clean_series)
# pd.concat(s, axis = 1)
```

5 Vaccinations

[18]:
```
vaccinations = pd.read_csv("dataset/
↪COVID-19_Vaccinations_in_the_United_States_Jurisdiction.csv")
```

[19]:
```
vaccinations.head()
```

[19]:
```
         Date  MMWR_week Location  Distributed  Distributed_Janssen  \
0  05/09/2022         19       NV      5994810               259000
1  05/09/2022         19       LA      8296850               325200
2  05/09/2022         19       WI     11753545               452500
3  05/09/2022         19       NE      4007410               150500
4  05/09/2022         19       OH     23378055               986300

   Distributed_Moderna  Distributed_Pfizer  Distributed_Unk_Manuf  \
0              2056700             3679110                      0
```

```
1          3394180            4577470                        0
2          4409000            6892045                        0
3          1405480            2451430                        0
4          8800240           13591515                        0

   Dist_Per_100K  Distributed_Per_100k_12Plus  …  \
0         194627                       228652  …
1         178473                       211475  …
2         201867                       235081  …
3         207165                       247767  …
4         199999                       233764  …

   Administered_Dose1_Recip_5PlusPop_Pct  Series_Complete_5Plus  \
0                                   80.2              1876120.0
1                                   65.3              2488841.0
2                                   76.2              3822098.0
3                                   75.5              1231426.0
4                                   67.6              6833069.0

   Series_Complete_5PlusPop_Pct  Administered_5Plus  Admin_Per_100k_5Plus  \
0                          64.8           4920847.0              170002.0
1                          57.2           6261798.0              144038.0
2                          69.6          10334215.0              188171.0
3                          68.3           3223525.0              178734.0
4                          62.1          17935233.0              163073.0

   Distributed_Per_100k_5Plus  Series_Complete_Moderna_5Plus  \
0                    207105.0                       605948.0
1                    190850.0                       962252.0
2                    214015.0                      1338600.0
3                    222198.0                       416495.0
4                    212561.0                      2396695.0

   Series_Complete_Pfizer_5Plus  Series_Complete_Janssen_5Plus  \
0                     1097337.0                       172768.0
1                     1344351.0                       181102.0
2                     2181902.0                       300650.0
3                      724421.0                        88778.0
4                     3877700.0                       553167.0

   Series_Complete_Unk_Manuf_5Plus
0                             67.0
1                           1136.0
2                            946.0
3                           1732.0
4                           5507.0
```

```
[5 rows x 82 columns]
```

[20]:
```
vaccinations["Date"] = pd.to_datetime(vaccinations["Date"])
```

6 Georgia

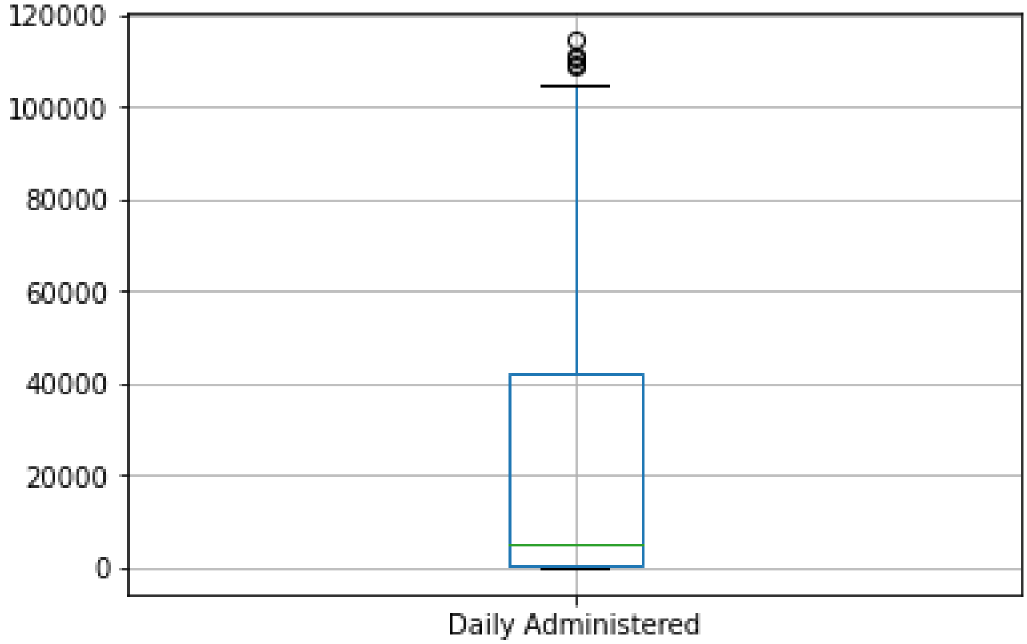
[21]:
```
georgia_vaccines = vaccinations[vaccinations["Location"].isin(['GA'])].
 ↪sort_values(by = "Date").set_index("Date")
```

[23]:
```
georgia_vaccines['Lag'] = georgia_vaccines.Administered.shift(1).fillna(0)
georgia_vaccines['Daily Administered'] = georgia_vaccines.Administered -␣
 ↪georgia_vaccines.Lag
georgia_vaccines['Daily Administered'] = clean_data(georgia_vaccines['Daily␣
 ↪Administered'])
```

```
Missing Values:0

Invalid entries: [-22526.  -6270. -81248.]

Lower Range: -69345.5, Upper Range: 116506.5

Total Outliers: 25

Outliers:
[149624.0, 151573.0, 202499.0, 181800.0, 118438.0, 415861.0, 128303.0, 147257.0,
171034.0, 158050.0, 119924.0, 182634.0, 127427.0, 137336.0, 121393.0, 118749.0,
163945.0, 163643.0, 163554.0, 120121.0, 118763.0, 139521.0, 135771.0, 156133.0,
160800.0]
```

[25]:
```
georgia_vaccines.boxplot('Daily Administered')
```

[25]:
```
<AxesSubplot:>
```



Outlier dates

[209]:
```
georgia_vaccines['Daily Administered'][georgia_vaccines['Daily Administered'].
↪isna()].index
```

[209]:
```
DatetimeIndex(['2021-01-22', '2021-02-19', '2021-03-20', '2021-04-07',
               '2021-04-09', '2021-04-13', '2021-04-14', '2021-05-07',
               '2021-05-31', '2021-06-04', '2021-06-17', '2021-06-22',
               '2021-06-29', '2021-07-20', '2021-07-28', '2021-08-05',
               '2021-08-19', '2021-08-25', '2021-09-01', '2021-09-08',
               '2021-09-15', '2021-10-30', '2021-11-03', '2021-11-16',
               '2021-11-30', '2021-12-04', '2021-12-28', '2022-01-12'],
              dtype='datetime64[ns]', name='Date', freq=None)
```

[26]:
```
georgia_vaccines = georgia_vaccines['Daily Administered']
georgia_vaccines.dropna(inplace = True)
```

[ ]:
```
georgia_vaccines.reset_index().to_csv("georgia_vaccinations.csv", index = False)
```

7 Indiana

[30]:
```
indiana_vaccines = vaccinations[vaccinations["Location"].isin(['IN'])].
↪sort_values(by = "Date").set_index("Date")
```

[31]:
```
indiana_vaccines['Lag'] = indiana_vaccines.Administered.shift(1).fillna(0)
indiana_vaccines['Daily Administered'] = indiana_vaccines.Administered -␣
↪indiana_vaccines.Lag
indiana_vaccines['Daily Administered'] = clean_data(indiana_vaccines['Daily␣
↪Administered'])
```

```
Missing Values:0

Invalid entries: []

Lower Range: -24250.0, Upper Range: 57734.0

Total Outliers: 17

Outliers:
[145215.0, 68118.0, 62176.0, 98243.0, 96848.0, 57954.0, 63433.0, 68471.0,
58356.0, 71096.0, 71831.0, 60735.0, 66382.0, 66406.0, 58781.0, 60419.0,
223259.0]
```

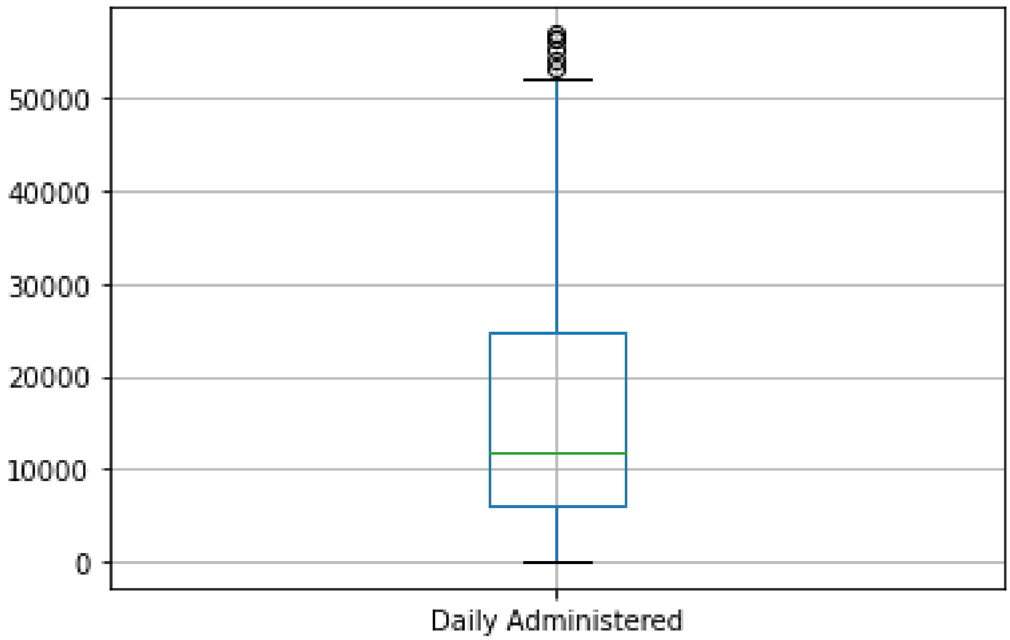
Outlier dates

[212]:
```
indiana_vaccines['Daily Administered'][indiana_vaccines['Daily Administered'].
↪isna()].index
```

[212]:
```
DatetimeIndex(['2021-02-13', '2021-02-19', '2021-02-24', '2021-02-25',
               '2021-03-13', '2021-03-26', '2021-04-02', '2021-04-03',
               '2021-04-08', '2021-04-09', '2021-04-10', '2021-04-11',
               '2021-04-15', '2021-04-16', '2021-04-17', '2021-05-07',
               '2021-07-03'],
              dtype='datetime64[ns]', name='Date', freq=None)
```

[32]:
```
indiana_vaccines.boxplot('Daily Administered')
```

[32]:
```
<AxesSubplot:>
```



[ ]:
```
indiana_vaccines = indiana_vaccines['Daily Administered']
indiana_vaccines.dropna(inplace = True)
indiana_vaccines.reset_index().to_csv("indiana_vaccinations.csv", index = False)
```

8 Remarks

For cases dataset, we found 68 and 39 total outliers for new_cases and new_deaths features for Georgia. Similarly, we found 64 and 52 outliers for Indiana. There were in total 6 negative entries in Georgia dataset,which were removed as pert of data cleaning.

For Vaccines dataset, we found 25 and 17 outliers respectively for Georgia and Indiana. In total, there were 3 negative entries in Georgia dataset, which were removed.

task_a

May 17, 2022

[9]:
```python
#import necessary libraries
import numpy as np
import pandas as pd
```

[10]:
```python
# Load the dataset

georgia = pd.read_csv("georgia_cases_deaths.csv")
indiana = pd.read_csv("indiana_cases_deaths.csv")

georgia_cases= georgia['new_case'].to_numpy()
georgia_deaths= georgia['new_death'].to_numpy()

indiana_cases= indiana['new_case'].astype(float).to_numpy()
indiana_deaths= indiana['new_death'].to_numpy()
```

[11]:
```python
#Data for deaths and cases for each of the states for Feb21 and March21

georgia_deaths_feb = georgia_deaths[343:363]
georgia_deaths_march = georgia_deaths[364:393]

georgia_cases_feb = georgia_cases[343:363]
georgia_cases_march = georgia_cases[364:393]


indiana_deaths_feb = indiana_deaths[331:357]
indiana_deaths_march = indiana_deaths[358:388]

indiana_cases_feb = indiana_cases[331:357]
indiana_cases_march = indiana_cases[358:388]
```

[12]:
```python
# Two sided Walds Test for single sample

def walds_single_sample(sample,prediction,hypothesis):

    # Using MLE of Poission which is the sample mean
    theta_estimate = np.mean(sample)

    #Computing the se(theta_estimate)
```

```python
    se = np.sqrt(theta_estimate/len(sample))

    # Computing walds statistic
    W = (theta_estimate - prediction)/se

    print("Walds Statistic",W)

    # Using alpha = 0.05 and Z_alpha_by_2 = 1.96 to accept or reject

    Z_alpha_by_2 = 1.96

    if(abs(W) > Z_alpha_by_2):
        print("Reject Ho:",hypothesis)
    else:
        print("Accept Ho:",hypothesis)

print("For Georgia:")
walds_single_sample(georgia_cases_march,np.mean(georgia_cases_feb),"Mean of the␣
 ↪number of COVID19 cases are same for Feb'21 and March'21 in Georgia")
print("\n")
walds_single_sample(georgia_deaths_march,np.mean(georgia_deaths_feb),"Mean of␣
 ↪the number of COVID19 deaths are same for Feb'21 and March'21 in Georgia")
print("\n")
print("For Indiana:")
walds_single_sample(indiana_cases_march,np.mean(indiana_cases_feb),"Mean of the␣
 ↪number of COVID19 cases are same for Feb'21 and March'21 in Indiana")
print("\n")
walds_single_sample(indiana_deaths_march,np.mean(indiana_deaths_feb),"Mean of␣
 ↪the number of COVID19 deaths are same for Feb'21 and March'21 in Indiana")
```

```
For Georgia:
Walds Statistic -197.83594450021198
Reject Ho: Mean of the number of COVID19 cases are same for Feb'21 and March'21
in Georgia


Walds Statistic -17.634729199836478
Reject Ho: Mean of the number of COVID19 deaths are same for Feb'21 and March'21
in Georgia


For Indiana:
Walds Statistic -89.04326703459927
Reject Ho: Mean of the number of COVID19 cases are same for Feb'21 and March'21
in Indiana
```

```
Walds Statistic -32.04809414405769
Reject Ho: Mean of the number of COVID19 deaths are same for Feb’21 and March’21
in Indiana
```

[13]:

```
# Two sided Z test for single sample

def z_test_single_sample(sample,prediction,entire_sample,hypothesis):

    n = len(sample)

    #Computing the mean of the sample
    X_bar = np.mean(sample)

    # Using the std deviation of the entire sample
    sigma = np.std(entire_sample)

    # Computing the Z statistic
    Z = (X_bar - prediction)/(sigma/np.sqrt(n))

    print("Z statistic",Z)

    # Using alpha = 0.05 and Z_alpha_by_2 = 1.96 to accept or reject

    Z_alpha_by_2 = 1.96

    if(abs(Z) > Z_alpha_by_2):
        print("Reject Ho:",hypothesis)
    else:
        print("Accept Ho:",hypothesis)


print("For Georgia:")
z_test_single_sample(georgia_cases_march,np.
 ↪mean(georgia_cases_feb),georgia_cases,"Mean of the number of COVID19 cases␣
 ↪are same for Feb’21 and March’21 in Georgia")
print("\n")
z_test_single_sample(georgia_deaths_march,np.
 ↪mean(georgia_deaths_feb),georgia_deaths,"Mean of the number of COVID19␣
 ↪deaths are same for Feb’21 and March’21 in Georgia")
print("\n")
print("For Indiana:")
z_test_single_sample(indiana_cases_march,np.
 ↪mean(indiana_cases_feb),indiana_cases,"Mean of the number of COVID19 cases␣
 ↪are same for Feb’21 and March’21 in Indiana")
print("\n")
```

```
z_test_single_sample(indiana_deaths_march,np.
 ↪mean(indiana_deaths_feb),indiana_deaths,"Mean of the number of COVID19␣
 ↪deaths are same for Feb'21 and March'21 in Indiana")
```

```
For Georgia:
Z statistic -4.5129098617929255
Reject Ho: Mean of the number of COVID19 cases are same for Feb'21 and March'21
in Georgia


Z statistic -3.813521351104936
Reject Ho: Mean of the number of COVID19 deaths are same for Feb'21 and March'21
in Georgia


For Indiana:
Z statistic -1.8692838881167924
Accept Ho: Mean of the number of COVID19 cases are same for Feb'21 and March'21
in Indiana


Z statistic -6.343576828280494
Reject Ho: Mean of the number of COVID19 deaths are same for Feb'21 and March'21
in Indiana
```

[14]:

```
#Two sided T test for single sample

def t_test_single_sample(sample,prediction,tvalue,hypothesis):

    n = len(sample)

    #Computing the mean of the sample
    X_bar = np.mean(sample)

    #Computing the corrected std deviation of the sample
    s = np.std(sample,ddof=1)

    # Computing the T statistic
    T = (X_bar - prediction)/(s/np.sqrt(n))
    print("T statistic",T)

    # Using the value of tn-1,alpha/2 to accept or reject

    if(abs(T) > tvalue):
        print("Reject Ho:",hypothesis)
    else:
        print("Accept Ho:",hypothesis)
```

```
print("For Georgia:")
t_test_single_sample(georgia_cases_march,np.mean(georgia_cases_feb),2.
 ↪048407,"Mean of the number of COVID19 cases are same for Feb’21 and March’21␣
 ↪in Georgia")
print("\n")
t_test_single_sample(georgia_deaths_march,np.mean(georgia_deaths_feb),2.
 ↪048407,"Mean of the number of COVID19 deaths are same for Feb’21 and␣
 ↪March’21 in Georgia")
print("\n")
print("For Indiana:")
t_test_single_sample(indiana_cases_march,np.mean(indiana_cases_feb),2.
 ↪04523,"Mean of the number of COVID19 cases are same for Feb’21 and March’21␣
 ↪in Indiana")
print("\n")
t_test_single_sample(indiana_deaths_march,np.mean(indiana_deaths_feb),2.
 ↪04523,"Mean of the number of COVID19 deaths are same for Feb’21 and March’21␣
 ↪in Indiana")
```

```
For Georgia:
T statistic -14.554543830491523
Reject Ho: Mean of the number of COVID19 cases are same for Feb’21 and March’21
in Georgia


T statistic -3.7751383617947973
Reject Ho: Mean of the number of COVID19 deaths are same for Feb’21 and March’21
in Georgia


For Indiana:
T statistic -12.81434557977271
Reject Ho: Mean of the number of COVID19 cases are same for Feb’21 and March’21
in Indiana


T statistic -14.885660164497072
Reject Ho: Mean of the number of COVID19 deaths are same for Feb’21 and March’21
in Indiana
```

[15]:
```
# Two sided Two sample walds test

def walds_two_sample_test(sample1,sample2,hypothesis):
    n = len(sample1)

    m = len(sample2)
```

```
    # Using MLE of Poission which is the sample mean
    theta_1  = np.mean(sample1)
    theta_2  = np.mean(sample2)

    #Calculating the se(theta_1 - theta_2)
    se = np.sqrt((theta_1/n)+(theta_2/m))

    #Compute the Wald's Statistic
    W = (theta_1 - theta_2)/se
    print("Walds Statistic: ",W)


    # Using alpha = 0.05 and Z_alpha_by_2 = 1.96 to accept or reject

    Z_alpha_by_2 = 1.96

    if(abs(W) > Z_alpha_by_2):
        print("Reject Ho:",hypothesis)
    else:
        print("Accept Ho:",hypothesis)


print("For Georgia:")
walds_two_sample_test(georgia_cases_march,georgia_cases_feb,"Mean of the number␣
 ↪of COVID19 cases are same for Feb'21 and March'21 in Georgia")
print("\n")
walds_two_sample_test(georgia_deaths_march,georgia_deaths_feb,"Mean of the␣
 ↪number of COVID19 deaths are same for Feb'21 and March'21 in Georgia")
print("\n")
print("For Indiana:")
walds_two_sample_test(indiana_cases_march,indiana_cases_feb,"Mean of the number␣
 ↪of COVID19 cases are same for Feb'21 and March'21 in Indiana")
print("\n")
walds_two_sample_test(indiana_deaths_march,indiana_deaths_feb,"Mean of the␣
 ↪number of COVID19 deaths are same for Feb'21 and March'21 in Indiana")
```

```
For Georgia:
Walds Statistic:  -102.46616574287205
Reject Ho: Mean of the number of COVID19 cases are same for Feb'21 and March'21
in Georgia


Walds Statistic:  -10.06229058540575
Reject Ho: Mean of the number of COVID19 deaths are same for Feb'21 and March'21
in Georgia
```

```
For Indiana:
Walds Statistic:  -53.08793994967661
Reject Ho: Mean of the number of COVID19 cases are same for Feb'21 and March'21
in Indiana


Walds Statistic:  -16.324483293500613
Reject Ho: Mean of the number of COVID19 deaths are same for Feb'21 and March'21
in Indiana
```

[16]:

```python
# Two sided unpaired T test

def t_test_unpaired(sample1,sample2,tvalue,hypothesis):
    n = len(sample1)

    m = len(sample2)

    # Computing the means of sample1 and sample2
    X_bar  = np.mean(sample1)
    Y_bar = np.mean(sample2)

    # Computing the corrected std deviation of sample1 and sample2
    s1 = np.std(sample1,ddof=1)
    s2 = np.std(sample2,ddof=1)

    # Compute the T statistic
    T = (X_bar - Y_bar)/(np.sqrt(((s1**2)/n)+((s2**2)/m)))
    print("T statistic",T)

    # Using the value of tn+m-2,0.025 where n and m is the number of datapoints␣
 ↪in sample1 and sample2
    # to accept or reject

    if(abs(T) > tvalue):
        print("Reject Ho:",hypothesis)
    else:
        print("Accept Ho:",hypothesis)


print("For Georgia:")
t_test_unpaired(georgia_cases_march,georgia_cases_feb,2.048407,"Mean of the␣
 ↪number of COVID19 cases are same for Feb'21 and March'21 in Georgia")
print("\n")
t_test_unpaired(georgia_deaths_march,georgia_deaths_feb,2.048407,"Mean of the␣
 ↪number of COVID19 deaths are same for Feb'21 and March'21 in Georgia")
print("\n")
```

```
print("For Indiana:")
t_test_unpaired(indiana_cases_march,indiana_cases_feb,2.04523,"Mean of the␣
 ↪number of COVID19 cases are same for Feb'21 and March'21 in Indiana")
print("\n")
t_test_unpaired(indiana_deaths_march,indiana_deaths_feb,2.04523,"Mean of the␣
 ↪number of COVID19 deaths are same for Feb'21 and March'21 in Indiana")
```

```
For Georgia:
T statistic -6.670734804444858
Reject Ho: Mean of the number of COVID19 cases are same for Feb'21 and March'21
in Georgia


T statistic -1.9788875675958115
Accept Ho: Mean of the number of COVID19 deaths are same for Feb'21 and March'21
in Georgia


For Indiana:
T statistic -4.414811766678156
Reject Ho: Mean of the number of COVID19 cases are same for Feb'21 and March'21
in Indiana


T statistic -5.776913802549136
Reject Ho: Mean of the number of COVID19 deaths are same for Feb'21 and March'21
in Indiana
```

0.1 Usage of the Tests

The walds test is applicable for both the states as well as the cases and deaths as the only requirement of the wald's test is an asymtotically normal estimator and since we have used the MLE estimator for poisson the test is valid.

The Z test is applicable where the size of the dataset is greater than 30. In our case georgia has around 29 datapoints and indiana has 30 datapoints. The Z test also requires the standard deviation of the entire dataset to be known which we can can calculate from the dataset. However since the dataset is constantly updated this standard deviation keeps on changing making Z test an impractical test for a real world scenario such as this.

The T test does not have any assumptions and is thus a valid test for the purposes of hypothesis testing on our dataset.

task_b

May 17, 2022

[39]:
```
#import necessary libraries
import numpy as np
from random import random
from random import randint
from random import seed
import pandas as pd
import bisect
from scipy.stats import poisson
from scipy.stats import geom
from scipy.stats import binom
```

[40]:
```
georgia = pd.read_csv("georgia_cases_deaths.csv")
indiana = pd.read_csv("indiana_cases_deaths.csv")

georgia_cases= georgia['new_case'].to_numpy()
georgia_deaths= georgia['new_death'].to_numpy()

indiana_cases= indiana['new_case'].astype(float).to_numpy()
indiana_deaths= indiana['new_death'].to_numpy()
print("######## BEGIN Task B #################")
```

```
######## BEGIN Task B #################
```

[41]:
```
#Data for deaths and cases for each of the states for Oct21 and Nov21 and Dec21

# georgia_deaths_oct = georgia_deaths[557:580]
# georgia_deaths_nov = georgia_deaths[581:609]
# georgia_deaths_dec = georgia_deaths[609:632]
# georgia_deaths_oct_dec = georgia_deaths[557:632]

# georgia_cases_oct = georgia_cases[557:580]
# georgia_cases_nov = georgia_cases[581:609]
# georgia_cases_dec = georgia_cases[609:632]
# georgia_cases_oct_dec = georgia_cases[557:632]

# indiana_deaths_oct = indiana_deaths[570:600]
# indiana_deaths_nov = indiana_deaths[601:626]
# indiana_deaths_dec = indiana_deaths[627:639]
```

```
indiana_deaths_oct_dec = indiana_deaths[570:639]

# indiana_cases_oct = indiana_cases[570:600]
# indiana_cases_nov = indiana_cases[601:626]
# indiana_cases_dec = indiana_cases[627:639]
indiana_cases_oct_dec = indiana_cases[570:639]
```

[42]:
```
def MME_poisson(data):
  # Computing the means of sample1 and sample2
  avg = np.mean(data)
  mu = avg
  print("For poisson")
  print("mu = " + str(mu))
  return mu

def MME_geometric(data):
  avg = np.mean(data)
  avg_1 = 1/avg
  p = avg_1
  print("For geometric")
  print("p = " + str(p))
  return p

def MME_binomial(data):
  avg = np.mean(data)
  Sx = np.var(data, ddof=1)
  n = (avg/(1-(Sx/avg)))
  p = avg/n
  print("For binomial")
  print("p = " + str(p))
  print("n = " + str(n))
  return [n, p]

# print("#######Georgia Deaths################")
# MME_poisson(georgia_deaths_oct_dec)
# MME_geometric(georgia_deaths_oct_dec)
# MME_binomial(georgia_deaths_oct_dec)

# print("#############Gerogia cases Deaths"#############)
# MME_poisson(georgia_cases_oct_dec)
# MME_geometric(georgia_cases_oct_dec)
# MME_binomial(georgia_cases_oct_dec)
```

[43]:
```
def KS_test_1sample(D , D1, distribution, datatype, state):
  print("KS 1 sample test for " + datatype)
  D = D[D!=0]
  D1 = D1[D1!=0]
```

```
min_x = min(D1)
max_x = max(D1)
total_elements = len(D1)
D1.sort()
mu = None
p = None
n = None
D1dict = dict()
max_diff = 0
for ele in D1:
  x = int(ele)
  if x not in D1dict:
    D1dict[x] = 1
  else:
    D1dict[x] += 1
# print(D1dict)
prev_ecdf = 0
curr_ecdf = 0
curr_sum = 0
for key in D1dict:
  curr_sum += D1dict[key]
  curr_ecdf = curr_sum/total_elements
  if distribution == "poisson":
    if mu is None:
      mu = MME_poisson(D)
    cdf = poisson.cdf(key, mu)
  elif distribution == "geometric":
    if p is None:
      p = MME_geometric(D)
    cdf = poisson.cdf(key, p)
  elif distribution == "binomial":
    if n is None or p is None:
      ret = MME_binomial(D)
      n = ret[0]
      p = ret[1]
    cdf = binom.cdf(key, n, p)
  else:
    print("Invalid distribution")
    return None
  left = abs(curr_ecdf - cdf)
  right = abs(prev_ecdf - cdf)
  prev_ecdf = curr_ecdf
  max_diff= max(max_diff, max(left, right))
  # print("ecdf = " + str(curr_ecdf) + ":: cdf = " + str(cdf))
print("p value = " + str(max_diff))
if max_diff > 0.05:
```

```python
        print("Rejecting Null hypothesis i.e :: Distribution of covid " + datatype␣
 ↪+ " for the state " + state + " is not " + distribution)
    else:
        print("Accepting Null hypothesis i.e :: Distribution of covid " + datatype␣
 ↪+ " for the state " + state + " is indeed " + distribution)


# KS_test_1sample(georgia_deaths_oct_dec, georgia_deaths_oct_dec, "poisson",␣
 ↪"deaths", "georgia")
# KS_test_1sample(georgia_deaths_oct_dec, georgia_deaths_oct_dec, "geometric",␣
 ↪"deaths", "georgia")
# KS_test_1sample(georgia_deaths_oct_dec, georgia_deaths_oct_dec, "binomial",␣
 ↪"deaths", "georgia")

# KS_test_1sample(georgia_deaths_oct_dec, georgia_cases_oct_dec, "poisson",␣
 ↪"cases", "georgia")
# KS_test_1sample(georgia_deaths_oct_dec, georgia_cases_oct_dec, "geometric",␣
 ↪"cases", "georgia")
# KS_test_1sample(georgia_deaths_oct_dec, georgia_cases_oct_dec, "binomial",␣
 ↪"cases", "georgia")

# KS_test_1sample(indiana_deaths_oct_dec, indiana_deaths_oct_dec, "poisson",␣
 ↪"deaths", "indiana")
# KS_test_1sample(indiana_deaths_oct_dec, indiana_deaths_oct_dec, "geometric",␣
 ↪"deaths", "indiana")
# KS_test_1sample(indiana_deaths_oct_dec, indiana_deaths_oct_dec, "binomial",␣
 ↪"deaths", "indiana")

# KS_test_1sample(indiana_deaths_oct_dec, indiana_cases_oct_dec, "poisson",␣
 ↪"cases", "indiana")
# KS_test_1sample(indiana_deaths_oct_dec, indiana_cases_oct_dec, "geometric",␣
 ↪"cases", "indiana")
# KS_test_1sample(indiana_deaths_oct_dec, indiana_cases_oct_dec, "binomial",␣
 ↪"cases", "indiana")

KS_test_1sample(georgia_deaths_oct_dec, indiana_deaths_oct_dec, "poisson",␣
 ↪"deaths", "indiana")
KS_test_1sample(georgia_deaths_oct_dec, indiana_deaths_oct_dec, "geometric",␣
 ↪"deaths", "indiana")
KS_test_1sample(georgia_deaths_oct_dec, indiana_deaths_oct_dec, "binomial",␣
 ↪"deaths", "indiana")

KS_test_1sample(georgia_deaths_oct_dec, indiana_cases_oct_dec, "poisson",␣
 ↪"cases", "indiana")
KS_test_1sample(georgia_deaths_oct_dec, indiana_cases_oct_dec, "geometric",␣
 ↪"cases", "indiana")
```

```
KS_test_1sample(georgia_deaths_oct_dec, indiana_cases_oct_dec, "binomial",␣
↪"cases", "indiana")
```

```
KS 1 sample test for deaths
For poisson
mu = 68.57692307692308
p value = 0.6184007206616742
Rejecting Null hypothesis i.e :: Distribution of covid deaths for the state
indiana is not poisson
KS 1 sample test for deaths
For geometric
p = 0.014582164890633763
p value = 1.0
Rejecting Null hypothesis i.e :: Distribution of covid deaths for the state
indiana is not geometric
KS 1 sample test for deaths
For binomial
p = -16.67721289300914
n = -4.112013411165938
p value = 1.0
Rejecting Null hypothesis i.e :: Distribution of covid deaths for the state
indiana is not binomial
KS 1 sample test for cases
For poisson
mu = 68.57692307692308
p value = 1.0
Rejecting Null hypothesis i.e :: Distribution of covid cases for the state
indiana is not poisson
KS 1 sample test for cases
For geometric
p = 0.014582164890633763
p value = 1.0
Rejecting Null hypothesis i.e :: Distribution of covid cases for the state
indiana is not geometric
KS 1 sample test for cases
For binomial
p = -16.67721289300914
n = -4.112013411165938
p value = 1.0
Rejecting Null hypothesis i.e :: Distribution of covid cases for the state
indiana is not binomial
```

[44]:
```
def KS_test_2sample(D1 , D2, datatype):
  print("KS 2 sample test for " + datatype)
  min_x_D1 = min(D1)
  max_x_D1 = max(D1)
```

```
total_elements_D1 = len(D1)
D1.sort()
D1dict = dict()
max_diff = 0
for ele in D1:
  x = int(ele)
  if x not in D1dict:
    D1dict[x] = 1
  else:
    D1dict[x] += 1
# print(D1dict)
min_x_D2 = min(D2)
max_x_D2 = max(D2)
total_elements_D2 = len(D2)
D2.sort()
D2dict = dict()
for ele in D2:
  x = int(ele)
  if x not in D2dict:
    D2dict[x] = 1
  else:
    D2dict[x] += 1
# print(D2dict)
curr_sum = 0
for key in D1dict:
  curr_sum += D1dict[key]
  D1dict[key] = curr_sum/total_elements_D1

curr_sum = 0
for key in D2dict:
  curr_sum += D2dict[key]
  D2dict[key] = curr_sum/total_elements_D2

list1 = list(D1dict.keys())
rev_list1 = list1[::-1]
list2 = list(D2dict.keys())
rev_list2 = list2[::-1]
# print(list1)
# print(list2)
for key1 in D1dict:
  left_key2 = bisect.bisect_left(list2, key1)
  right_key2 = bisect.bisect_left(list2, key1)
  left_key1 = bisect.bisect_left(list1, key1)
  right_key1 = bisect.bisect_left(list1, key1)
  left = abs(D2dict[list2[left_key2 - 1]] - D1dict[list1[left_key1 - 1]])
  right = abs(D2dict[list2[right_key2]] - D1dict[list1[right_key1]])
  max_diff = max(max_diff, max(left, right))
```

```
  print("p value = " + str(max_diff))
  if max_diff > 0.05:
    print("Rejecting Null hypothesis i.e :: Distribution of covid " + datatype␣
↪+ " is not same for both states")
  else:
    print("Accepting Null hypothesis i.e :: Distribution of covid " + datatype␣
↪+ "is indeed same for both states")

KS_test_2sample(indiana_cases_oct_dec , georgia_cases_oct_dec, "cases")
KS_test_2sample(indiana_deaths_oct_dec , georgia_deaths_oct_dec, "deaths")
```

```
KS 2 sample test for cases
p value = 0.35768115942028983
Rejecting Null hypothesis i.e :: Distribution of covid cases is not same for
both states
KS 2 sample test for deaths
p value = 0.24637681159420288
Rejecting Null hypothesis i.e :: Distribution of covid deaths is not same for
both states
```

[45]:
```
#Permustation test
def generate_perm(D1, D2):
  len1 = len(D1)
  len2 = len(D2)

  D = D1
  D = np.append(D1, D2)
  # print(D)

  length = len(D)
  last = length - 1
  for i in range(length):
    rand = randint(0, length - i - 1)
    # print(rand)
    D[[rand, length - i - 1]] = D[[length - i - 1, rand]]
  # print(D)
  D = np.split(D, 2)
  D1 = D[0]
  D2 = D[1]
  return [D1, D2]


def perm_test(D1, D2, datatype):
  print("Permutation test for " + datatype)
  len1 = len(D1)
  len2 = len(D2)
```

```
  og_mean1 = np.mean(D1)
  og_mean2 = np.mean(D2)
  og_mean_diff = abs(og_mean1 - og_mean2)
  perm_D1 = []
  perm_D2 = []
  count = 0
  for i in range(1000):
    D = generate_perm(D1, D2)
    perm_D1 = D[0]
    perm_D2 = D[1]
    # print(perm_D1)
    # print(perm_D2)
    mean1 = np.mean(perm_D1)
    mean2 = np.mean(perm_D2)
    mean_diff = abs(mean1 - mean2)
    if mean_diff > og_mean_diff:
      count+=1

  p_value = (count/1000)
  print("p_value = " + str(p_value))
  if p_value < 0.05:
    print("Rejeting Null Hypothesis i.e Distribution of " + datatype + " of two␣
↪states is not same")
  else:
    print("Failed to Reject Null Hypothesis i.e Distribution of " + datatype +␣
↪" of two states is indeed same")


perm_test(indiana_deaths_oct_dec, georgia_deaths_oct_dec, "deaths")
perm_test(indiana_cases_oct_dec, georgia_cases_oct_dec, "cases")
```

```
Permutation test for deaths
p_value = 0.003
Rejeting Null Hypothesis i.e Distribution of deaths of two states is not same
Permutation test for cases
p_value = 0.001
Rejeting Null Hypothesis i.e Distribution of cases of two states is not same
```

[ ]:

```
```

Poisson $(n\ ;\ \lambda)$

Task C Exponential Prior $\lambda \sim f(\lambda) = \left(\frac{1}{\beta}\right) e^{-\left(\frac{1}{\beta}\right)\lambda}$

$E[\lambda] = \beta = \frac{1}{\text{parameter}}$

$\therefore$ parameter $= \frac{1}{\beta}$

$\lambda_{mme} = \frac{1}{n}\sum x_i \qquad \therefore \frac{1}{\beta} = \frac{1}{\lambda_{mme}}$

Posterior = likelihood x prior

$f(\lambda|D) = P(D|\lambda)\, f(\lambda)$

1.) $P(D|\lambda) = \prod_{i=1}^{n} P(x_i|\lambda)$ $\qquad f(\lambda) = \frac{1}{\beta} e^{-\frac{1}{\beta}\lambda}$

$= \prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}$ $\qquad \propto e^{-\frac{1}{\beta}\lambda}$

$\propto \lambda^{n\bar{x}} e^{-n\lambda}$

$\therefore f(\lambda|D) \propto \cancel{\frac{1}{\beta}}\ \cancel{\lambda^{n\bar{x}} e^{-n\lambda}}\ \overbrace{\lambda^{n\bar{x}}}^{\alpha-1}\ e^{-\overbrace{\left(n+\frac{1}{\beta}\right)}^{\beta}\lambda}$

$= \text{Gamma}(1+n\bar{x},\ n+1/\beta)$

2.) $\cancel{P(D|\lambda)} = f(D|\lambda) \propto \lambda^{m\bar{y}} e^{-(m\lambda)}\ \lambda^{n\bar{x}} e^{-}$

$= \lambda^{m\bar{y}+n\bar{x}}\ e^{-m\lambda-(n+1/\beta)\lambda}$

task_c

May 17, 2022

1 Task C

Solve the required inferences below for your COVID19 datasets (Cases dataset consists of the cumulative #cases and #deaths while the Vaccinations dataset consists of #vaccines administered information). The datasets provided contain cumulative data and hence you should first calculate daily stats for each relevant column. Unless otherwise stated, always use daily stats for the purpose of reporting any inference/observation. Only use tools/tests learned in class. Show your work clearly and comment on results as appropriate. This will be 60% of the project grade, with 12% for each of the five tasks below. Use the Cases dataset for tasks a, b and c, and the Vaccinations dataset for tasks d and e.

For this task, sum up the daily stats (cases and deaths) from the two states assigned to you. Assume day 1 is June 1st 2020. Assume the combined daily deaths are Poisson distributed with parameter . Assume an Exponential prior (with mean) on . Assume = MME where the MME is found using the first four weeks data (so the first 28 days of June 2020) as the sample data. Now, use the fifth week's data (June 29 to July 5) to obtain the posterior for via Bayesian inference. Then, use the sixth week's data to obtain the new posterior, using prior as posterior after week 5. Repeat till the end of week 8 (that is, repeat till you have posterior after using 8th week's data). Plot all posterior distributions on one graph. Report the MAP for all posteriors.

[1]:
```
import math
import pandas as pd

import numpy as np
from scipy.stats import gamma, expon
from matplotlib import pyplot as plt
```

2 Indiana Cases/Deaths

[2]:
```
cases_deaths = pd.read_csv('./indiana_cases_deaths.csv')
cases_deaths.head()
```

[2]:

	submission_date	new_case	new_death
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

[3]:
```
cases_deaths.dtypes
```

[3]:
```
submission_date     object
new_case           float64
new_death          float64
dtype: object
```

[4]:
```
cases_deaths['submission_date'] = pd.
↪to_datetime(cases_deaths['submission_date'])
```

Combine cases and deaths

[5]:
```
cases_deaths['combined_stats'] = cases_deaths['new_case'] +␣
↪cases_deaths['new_death']
```

Peek into data from June

[6]:
```
mask_first_four_weeks = (cases_deaths['submission_date'] >= '2020-06-01') & \
(cases_deaths['submission_date'] <= '2020-06-28')
cases_deaths[mask_first_four_weeks]
```

[6]:

	submission_date	new_case	new_death	combined_stats
130	2020-06-01	256.0	8.0	264.0
131	2020-06-02	407.0	54.0	461.0
132	2020-06-03	475.0	10.0	485.0
133	2020-06-04	384.0	23.0	407.0
134	2020-06-05	482.0	28.0	510.0
135	2020-06-06	419.0	34.0	453.0
136	2020-06-07	400.0	11.0	411.0
137	2020-06-08	226.0	14.0	240.0
138	2020-06-09	410.0	24.0	434.0
139	2020-06-10	304.0	15.0	319.0
140	2020-06-11	411.0	25.0	436.0
141	2020-06-12	398.0	16.0	414.0
142	2020-06-13	397.0	17.0	414.0
143	2020-06-14	366.0	9.0	375.0
144	2020-06-15	521.0	12.0	533.0
145	2020-06-16	356.0	13.0	369.0
146	2020-06-17	227.0	28.0	255.0
147	2020-06-18	425.0	16.0	441.0
148	2020-06-19	308.0	25.0	333.0
149	2020-06-20	315.0	19.0	334.0
150	2020-06-21	362.0	5.0	367.0
151	2020-06-22	210.0	12.0	222.0
152	2020-06-23	238.0	16.0	254.0
153	2020-06-24	269.0	9.0	278.0
154	2020-06-25	515.0	9.0	524.0
155	2020-06-26	485.0	9.0	494.0

156	2020-06-27	435.0	20.0	455.0
157	2020-06-28	355.0	3.0	358.0

2.1 Compute lambda_mme and Prior

[8]:
```
def compute_lambda_mme(data):
    return data.mean()

lambda_mme =␣
↪int(compute_lambda_mme(cases_deaths[mask_first_four_weeks]['combined_stats']))
print(lambda_mme)
```

```
387
```

2.1.1 Compute parameters of posterior given data

[9]:
```
def compute_posterior_params(data, prior_alpha, prior_beta):
    x_bar = data.mean()
    n = len(data)
    alpha = n*x_bar + prior_alpha
    beta = n+prior_beta
    return alpha, beta
```

2.1.2 Compute posterior for first 4 weeks

[10]:
```
alpha_post, beta_post =␣
↪compute_posterior_params(cases_deaths[mask_first_four_weeks]['combined_stats'],
                                                        1, 1/lambda_mme)
alpha_post, beta_post
```

[10]:
```
(10841.0, 28.002583979328165)
```

2.1.3 Computer posterior from 5th until 8th week

[11]:
```
initial_index = cases_deaths[mask_first_four_weeks].index[-1] + 1
index = initial_index
params = [(alpha_post, beta_post)]

for i in range(4):
    data = cases_deaths.iloc[index: index + 7]
    alpha, beta = compute_posterior_params(data['combined_stats'],␣
↪params[-1][0], params[-1][1])
    params.append((alpha, beta))
    # increment index
    index = index + 7
```

[16]:
```
params
```

[16]: [(10841.0, 28.002583979328165),
 (13991.0, 35.002583979328165),
 (17662.0, 42.002583979328165),
 (22683.0, 49.002583979328165),
 (28562.0, 56.002583979328165)]

2.2 Validate using the whole data at once?

[17]:
```
mask_all_data = (cases_deaths['submission_date'] >= '2020-06-29') & \
(cases_deaths['submission_date'] <= '2020-07-26')
# cases_deaths[mask_all_data]
```

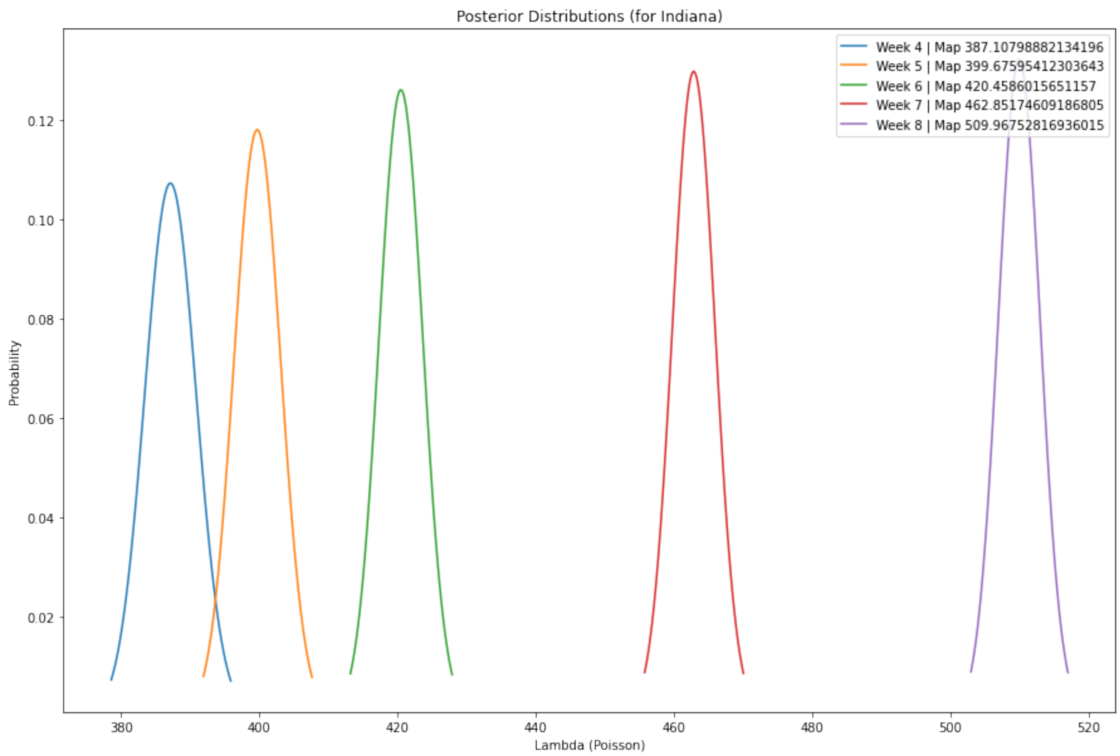
[18]:
```
compute_posterior_params(cases_deaths[mask_all_data]['combined_stats'], 10841.
↪0, 28.002583979328165)
```

[18]: (28562.0, 56.002583979328165)

2.2.1 Plot posteriors

[38]:
```
def gamma_pdf(x, alpha, beta):
    return gamma.pdf(x, alpha, scale=1/beta)

def gamma_inverse_cdf(prob, alpha, beta):
    return gamma.ppf(prob, alpha, scale=1/beta)
```

[62]:
```
fig = plt.gcf()
fig.set_size_inches((15, 10))
map_ = []

for i in range(0, 5):
    x = np.linspace(gamma_inverse_cdf(0.01, params[i][0], params[i][1]), \
                    gamma_inverse_cdf(0.99, params[i][0], params[i][1]),␣
↪num=100)
    y = gamma_pdf(x, params[i][0], params[i][1])
    map_.append(x[np.argmax(y)])
    plt.title('Posterior Distributions (Indiana State)')
    plt.xlabel('Lambda (Poisson)')
    plt.ylabel('Probability')
    plt.plot(x, y, label=f'Week {i+4} | Map {map_[-1]}')
plt.legend(loc='upper right')
plt.show()
```



3 Georgia Cases/Deaths

[63]:
```
cases_deaths = pd.read_csv('./georgia_cases_deaths.csv')
cases_deaths.head()
```

[63]:
```
   index submission_date  new_case  new_death
0      0      2020-01-22       0.0        0.0
1      1      2020-01-23       0.0        0.0
2      2      2020-01-24       0.0        0.0
3      3      2020-01-25       0.0        0.0
4      4      2020-01-26       0.0        0.0
```

[64]:
```
cases_deaths.dtypes
```

[64]:
```
index                int64
submission_date     object
new_case           float64
new_death          float64
dtype: object
```

[65]:
```
cases_deaths['submission_date'] = pd.
 ↪to_datetime(cases_deaths['submission_date'])
```

Combine cases and deaths

[66]:
```
cases_deaths['combined_stats'] = cases_deaths['new_case'] +␣
↪cases_deaths['new_death']
```

Peek into data from June

[67]:
```
mask_first_four_weeks = (cases_deaths['submission_date'] >= '2020-06-01') & \
(cases_deaths['submission_date'] <= '2020-06-28')
cases_deaths[mask_first_four_weeks]
```

[67]:

	index	submission_date	new_case	new_death	combined_stats
128	128	2020-06-01	889.0	22.0	911.0
129	129	2020-06-02	578.0	28.0	606.0
130	130	2020-06-03	723.0	21.0	744.0
131	131	2020-06-04	942.0	24.0	966.0
132	132	2020-06-05	799.0	27.0	826.0
133	133	2020-06-07	524.0	20.0	544.0
134	134	2020-06-08	648.0	28.0	676.0
135	135	2020-06-09	808.0	77.0	885.0
136	136	2020-06-10	804.0	44.0	848.0
137	137	2020-06-11	1101.0	46.0	1147.0
138	138	2020-06-12	963.0	43.0	1006.0
139	139	2020-06-13	1043.0	28.0	1071.0
140	140	2020-06-14	852.0	5.0	857.0
141	141	2020-06-15	851.0	43.0	894.0
142	142	2020-06-16	870.0	35.0	905.0
143	143	2020-06-17	1094.0	46.0	1140.0
144	144	2020-06-18	1023.0	30.0	1053.0
145	145	2020-06-19	1267.0	31.0	1298.0
146	146	2020-06-20	1814.0	6.0	1820.0
147	147	2020-06-21	885.0	1.0	886.0
148	148	2020-06-22	1271.0	5.0	1276.0
149	149	2020-06-23	1860.0	39.0	1899.0
150	150	2020-06-24	1835.0	11.0	1846.0
151	151	2020-06-25	1743.0	47.0	1790.0
152	152	2020-06-26	2076.0	25.0	2101.0
153	153	2020-06-27	1952.0	6.0	1958.0
154	154	2020-06-28	2186.0	2.0	2188.0

3.0.1 Compute prior parameters

[71]:
```
lambda_mme =␣
↪int(compute_lambda_mme(cases_deaths[mask_first_four_weeks]['combined_stats']))
print(lambda_mme)
```

```
1190
```

3.0.2 Compute posterior for first 4 weeks

[72]:
```
alpha_post, beta_post =␣
↪compute_posterior_params(cases_deaths[mask_first_four_weeks]['combined_stats'],
                                                            1, 1/lambda_mme)
alpha_post, beta_post
```

[72]: (32142.0, 27.000840336134456)

3.0.3 Computer posterior from 5th until 8th week

[73]:
```
initial_index = cases_deaths[mask_first_four_weeks].index[-1] + 1
index = initial_index
params = [(alpha_post, beta_post)]

for i in range(4):
    data = cases_deaths.iloc[index: index + 7]
    alpha, beta = compute_posterior_params(data['combined_stats'],␣
↪params[-1][0], params[-1][1])
    params.append((alpha, beta))
    # increment index
    index = index + 7
```

[74]:
```
params
```

[74]:
```
[(32142.0, 27.000840336134456),
 (50520.0, 34.00084033613446),
 (72497.0, 41.00084033613446),
 (99979.0, 48.00084033613446),
 (126466.0, 55.00084033613446)]
```

3.1 Validate using the whole data at once?

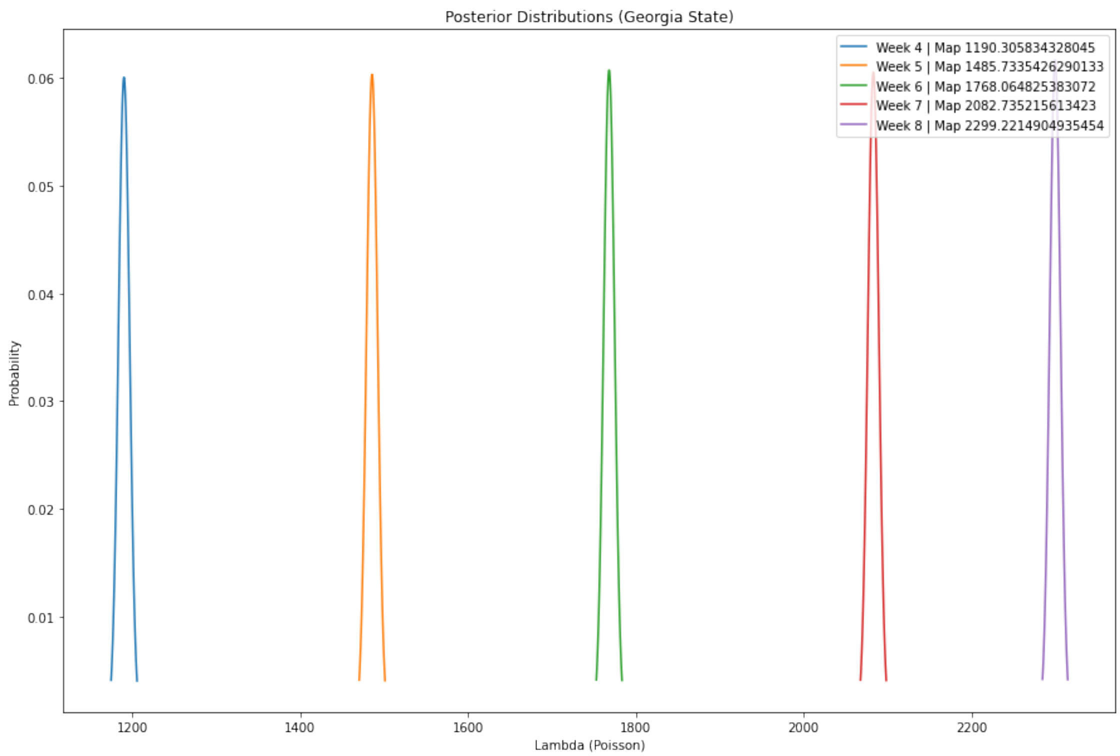
[75]:
```
mask_all_data = (cases_deaths['submission_date'] >= '2020-06-29') & \
(cases_deaths['submission_date'] <= '2020-07-26')
# cases_deaths[mask_all_data]
```

[77]:
```
compute_posterior_params(cases_deaths[mask_all_data]['combined_stats'], 32142.
↪0, 27.000840336134456)
```

[77]: (126466.0, 55.00084033613446)

[78]:
```
fig = plt.gcf()
fig.set_size_inches((15, 10))
map_ = []

for i in range(0, 5):
    x = np.linspace(gamma_inverse_cdf(0.01, params[i][0], params[i][1]), \
```

```
                    gamma_inverse_cdf(0.99, params[i][0], params[i][1]),␣
↪num=100)
    y = gamma_pdf(x, params[i][0], params[i][1])
    # map is nothing but
    # lambda = argmax P(lambda | data)
    map_.append(x[np.argmax(y)])
    plt.title('Posterior Distributions (Georgia State)')
    plt.xlabel('Lambda (Poisson)')
    plt.ylabel('Probability')
    plt.plot(x, y, label=f'Week {i+4} | Map {map_[-1]}')
plt.legend(loc='upper right')
plt.show()
```



[ ]:

task_d

May 17, 2022

[1]:
```python
import pandas as pd
import numpy as np
import datetime
import matplotlib.pyplot as plt
plt.style.use('bmh')
plt.rcParams["figure.figsize"] = (10,5)
```

[2]:
```python
start_date, end_date = datetime.date(2021,5,1), datetime.date(2021,5,28)
train_start_date, train_end_date = datetime.date(2021,5,1), datetime.
↪date(2021,5,21)
test_start_date, test_end_date = datetime.date(2021,5,22), datetime.
↪date(2021,5,28)
```

[3]:
```python
def MLR(trainX, trainY, testX, testY):
    '''
    Performs Multiple Linear Regression
    '''
    Y = trainY.to_numpy()
    X = trainX.to_numpy()
    X_intercept = np.ones((X.shape[0], 1))
    X = np.append(X_intercept, X, axis = 1)

    beta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, Y))

    Y_test = testY.to_numpy()
    X_test = testX.to_numpy()
    X_test_intercept = np.ones((X_test.shape[0], 1))
    X_test = np.append(X_test_intercept, X_test, axis = 1)

    Y_hat = np.dot(X_test, beta)

    mse = np.mean((Y_test - Y_hat)**2)
    # only consider non-zero test data points for MAPE
    mape = np.mean(np.abs((Y_test[Y_test!=0] - Y_hat[Y_test!=0])/Y_test[Y_test!
↪=0])*100)

    return Y_hat, mse, mape
```

[4]:
```
def AR(ts_data, p = 3):
    '''
    Performs Auto Regression on given Time Series dataframe
    '''
    col = ts_data.columns[0]
    for i in range(p):
        if i:
            ts_data["T-{}".format(i+1)] = ts_data["T-{}".format(i)].shift()
        else:
            ts_data["T-{}".format(i+1)] = ts_data[col].shift()
    ts_data.dropna(inplace = True)

    train_ts = ts_data.loc[:train_end_date]
    test_ts = ts_data.loc[test_start_date:]

    train_ts.reset_index(drop = True, inplace = True)
    test_ts.reset_index(drop = True, inplace = True)

    trainY = train_ts.iloc[:, 0] #.values.reshape(-1,1)
    trainX = train_ts.iloc[:, 1:] #.values

    testY = test_ts.iloc[:, 0] #.values.reshape(-1,1)
    testX = test_ts.iloc[:, 1:] #.values

    return MLR(trainX, trainY, testX, testY)
```

[5]:
```
def EWMA(ts_data, alpha=0.5):
    '''
    Performs EWMA technique on given Time Series dataframe
    '''
    preds = [ts_data[0]]
    for i in range(1,len(ts_data)):
        pred = alpha * ts_data[i-1] + (1 - alpha) * preds[i-1]
        preds.append(pred)

    Y_test = ts_data
    Y_hat = np.array(preds)

    mse = np.mean((Y_test - Y_hat)**2)
    # only consider non-zero test data points for MAPE
    mape = np.mean(np.abs((Y_test[Y_test!=0] - Y_hat[Y_test!=0])/Y_test[Y_test!
 ↪=0]) * 100)

    return preds, mse, mape
```

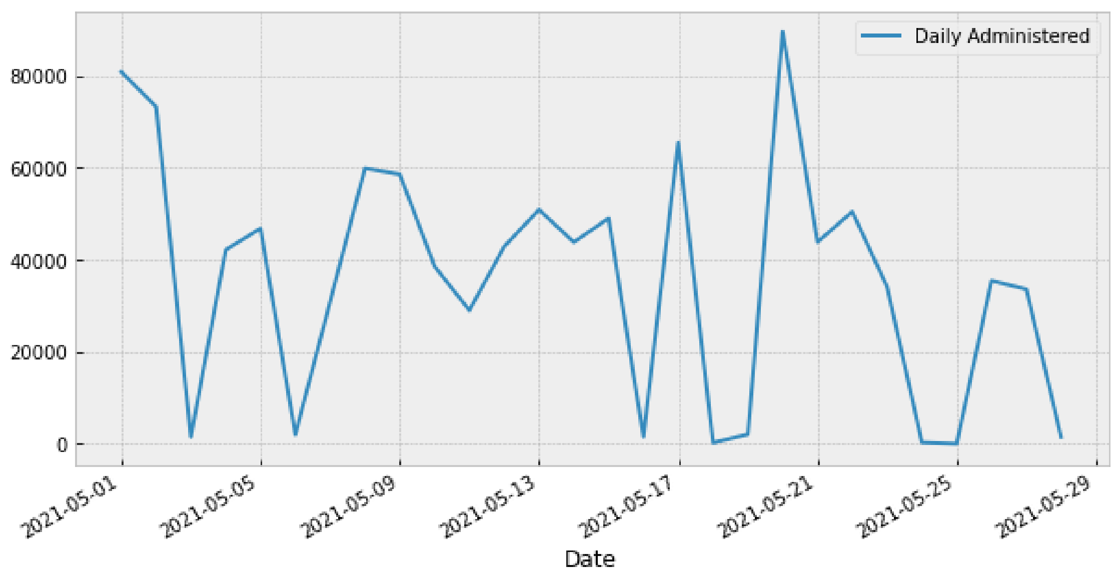
1 Georgia

[6]:
```
georgia_vaccines = pd.read_csv("georgia_vaccinations.csv")
georgia_vaccines["Date"] = pd.to_datetime(georgia_vaccines["Date"])
georgia_vaccines = georgia_vaccines.sort_values(by = "Date").set_index("Date")
```

[7]:
```
georgia_may = georgia_vaccines.loc[start_date:end_date][['Daily Administered']]
```

[8]:
```
georgia_may.plot()
```

[8]:
```
<AxesSubplot:xlabel='Date'>
```



2 AR(3)

[9]:
```
preds_3, mse, mape = AR(georgia_may.copy(), 3)
print("MSE:{}\nMAPE:{}".format(mse, mape))
```

```
MSE:1273507395.7608464
MAPE:2469.2428372465824
```

3 AR(5)

[10]:
```
preds_5, mse, mape = AR(georgia_may.copy(), 5)
print("MSE:{}\nMAPE:{}".format(mse, mape))
```

```
MSE:1217949112.4454513
MAPE:2110.1685310524813
```

[11]:
```
test_data = georgia_may.loc[test_start_date:]
test_data["Predicted:AR(3)"] = preds_3
test_data["Predicted:AR(5)"] = preds_5
test_data
```

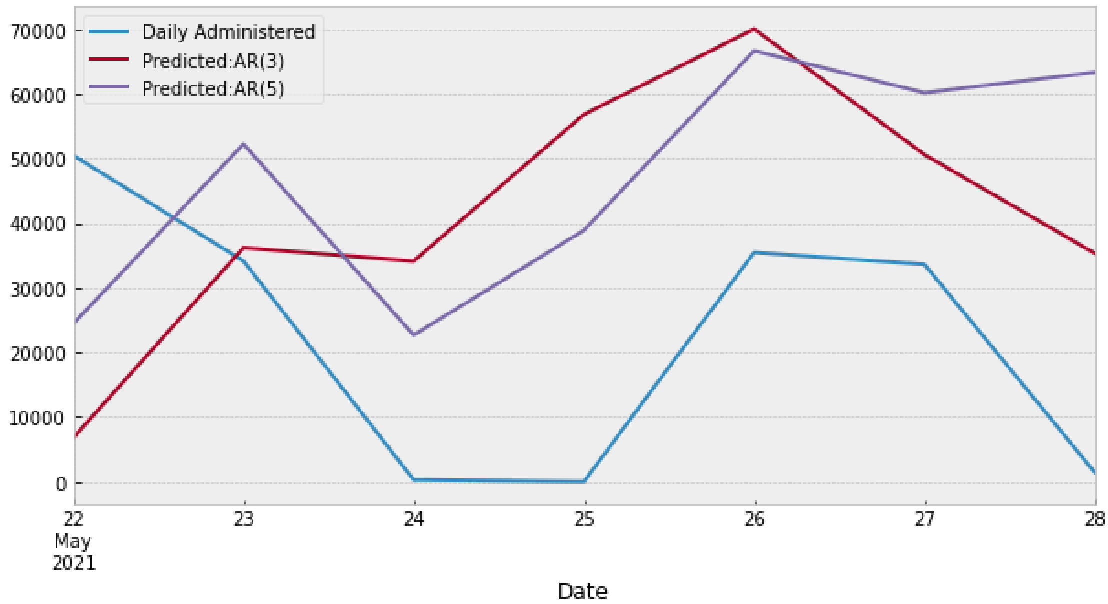
```
<ipython-input-11-ad5d2e6f18e9>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data["Predicted:AR(3)"] = preds_3
<ipython-input-11-ad5d2e6f18e9>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data["Predicted:AR(5)"] = preds_5
```

[11]:

	Daily Administered	Predicted:AR(3)	Predicted:AR(5)
Date			
2021-05-22	50463.0	6748.680788	24393.722503
2021-05-23	34122.0	36174.959561	52204.140957
2021-05-24	277.0	34104.000860	22643.618767
2021-05-25	0.0	56778.018657	38875.489667
2021-05-26	35444.0	70019.623264	66651.335541
2021-05-27	33609.0	50574.097076	60167.756507
2021-05-28	1434.0	35317.591760	63306.757055

[12]:
```
test_data.plot()
```

[12]: <AxesSubplot:xlabel='Date'>



4 EWMA

5 $\alpha = 0.5$

[13]:
```
preds_p5, mse, mape = EWMA(georgia_may.copy()["Daily Administered"], 0.5)
print("MSE:{}\nMAPE:{}".format(mse, mape))
```

```
MSE:956346481.6539252
MAPE:1896.146608075354
```

6 $\alpha = 0.8$

[14]:
```
preds_p8, mse, mape = EWMA(georgia_may.copy()["Daily Administered"], 0.8)
print("MSE:{}\nMAPE:{}".format(mse, mape))
```

```
MSE:1169488990.656236
MAPE:2008.2237034717157
```

[15]:
```
test_data = georgia_may.loc[test_start_date:]
test_data["Predicted:EWMA(0.5)"] = preds_p5[-7:]
test_data["Predicted:EWMA(0.8)"] = preds_p8[-7:]
test_data
```

```
<ipython-input-15-50d7695815ed>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data["Predicted:EWMA(0.5)"] = preds_p5[-7:]
<ipython-input-15-50d7695815ed>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data["Predicted:EWMA(0.8)"] = preds_p8[-7:]
```

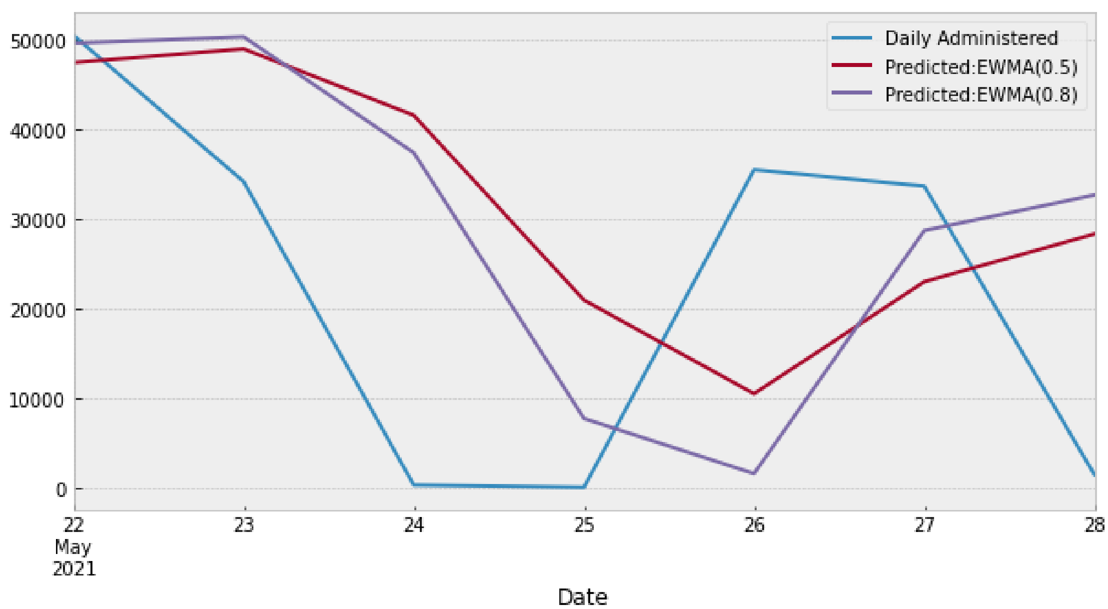
[15]:

Date	Daily Administered	Predicted:EWMA(0.5)	Predicted:EWMA(0.8)
2021-05-22	50463.0	47399.468256	49570.872674
2021-05-23	34122.0	48931.234128	50284.574535
2021-05-24	277.0	41526.617064	37354.514907
2021-05-25	0.0	20901.808532	7692.502981
2021-05-26	35444.0	10450.904266	1538.500596
2021-05-27	33609.0	22947.452133	28662.900119
2021-05-28	1434.0	28278.226066	32619.780024

[16]:

```
test_data.plot()
```

[16]: <AxesSubplot:xlabel='Date'>



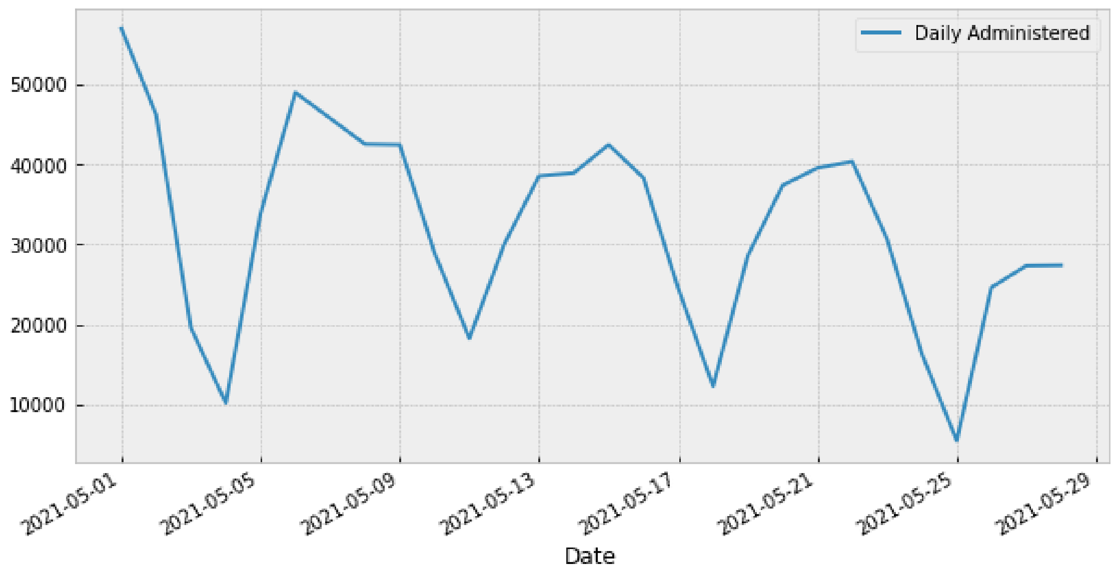
7 Indiana

[17]:
```
indiana_vaccines = pd.read_csv("indiana_vaccinations.csv")
indiana_vaccines["Date"] = pd.to_datetime(indiana_vaccines["Date"])
indiana_vaccines = indiana_vaccines.sort_values(by = "Date").set_index("Date")
```

[18]:
```
indiana_may = indiana_vaccines.loc[start_date:end_date][['Daily Administered']]
```

[19]:
```
indiana_may.plot()
```

[19]: <AxesSubplot:xlabel='Date'>



8 AR(3)

[20]:
```
preds_3, mse, mape = AR(indiana_may.copy(), 3)
print("MSE:{}\nMAPE:{}".format(mse, mape))
```

```
MSE:138610049.2708631
MAPE:77.20496686304068
```

9 AR(5)

[21]:
```
preds_5, mse, mape = AR(indiana_may.copy(), 5)
print("MSE:{}\nMAPE:{}".format(mse, mape))
```

```
MSE:171446604.5410115
MAPE:80.5978322366637
```

[22]:
```python
test_data = indiana_may.loc[test_start_date:]
test_data["Predicted:AR(3)"] = preds_3
test_data["Predicted:AR(5)"] = preds_5
test_data
```

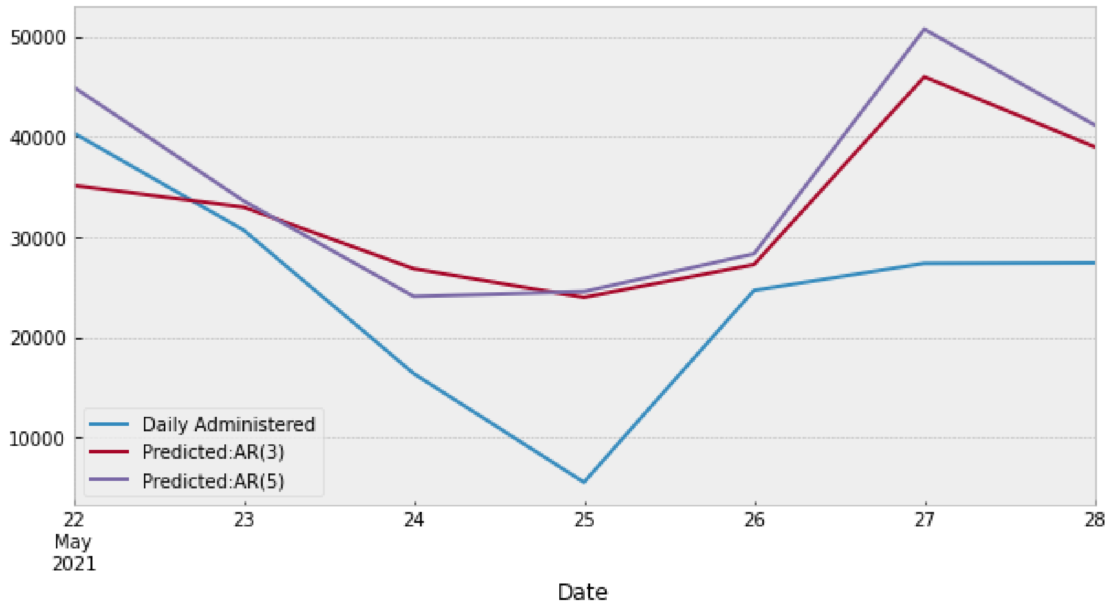
```
<ipython-input-22-a11ec8827679>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data["Predicted:AR(3)"] = preds_3
<ipython-input-22-a11ec8827679>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data["Predicted:AR(5)"] = preds_5
```

[22]:

Date	Daily Administered	Predicted:AR(3)	Predicted:AR(5)
2021-05-22	40370.0	35122.711302	44961.861234
2021-05-23	30675.0	32962.308384	33584.023201
2021-05-24	16346.0	26818.083516	24059.305337
2021-05-25	5506.0	23959.265482	24539.020670
2021-05-26	24657.0	27232.342761	28305.421742
2021-05-27	27350.0	45967.779235	50718.376818
2021-05-28	27405.0	38983.553124	41168.562579

[23]:
```python
test_data.plot()
```

[23]: <AxesSubplot:xlabel='Date'>



10 EWMA

11 $\alpha = 0.5$

[24]:
```
preds_p5, mse, mape = EWMA(indiana_may.copy()["Daily Administered"], 0.5)
print("MSE:{}\nMAPE:{}".format(mse, mape))
```

```
MSE:176321374.69042543
MAPE:59.39813552165008
```

12 $\alpha = 0.8$

[25]:
```
preds_p8, mse, mape = EWMA(indiana_may.copy()["Daily Administered"], 0.8)
print("MSE:{}\nMAPE:{}".format(mse, mape))
```

```
MSE:155019486.29836497
MAPE:51.13497840349774
```

[26]:
```
test_data = indiana_may.loc[test_start_date:]
test_data["Predicted:EWMA(0.5)"] = preds_p5[-7:]
test_data["Predicted:EWMA(0.8)"] = preds_p8[-7:]
test_data
```

```
<ipython-input-26-cc85fedd5278>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

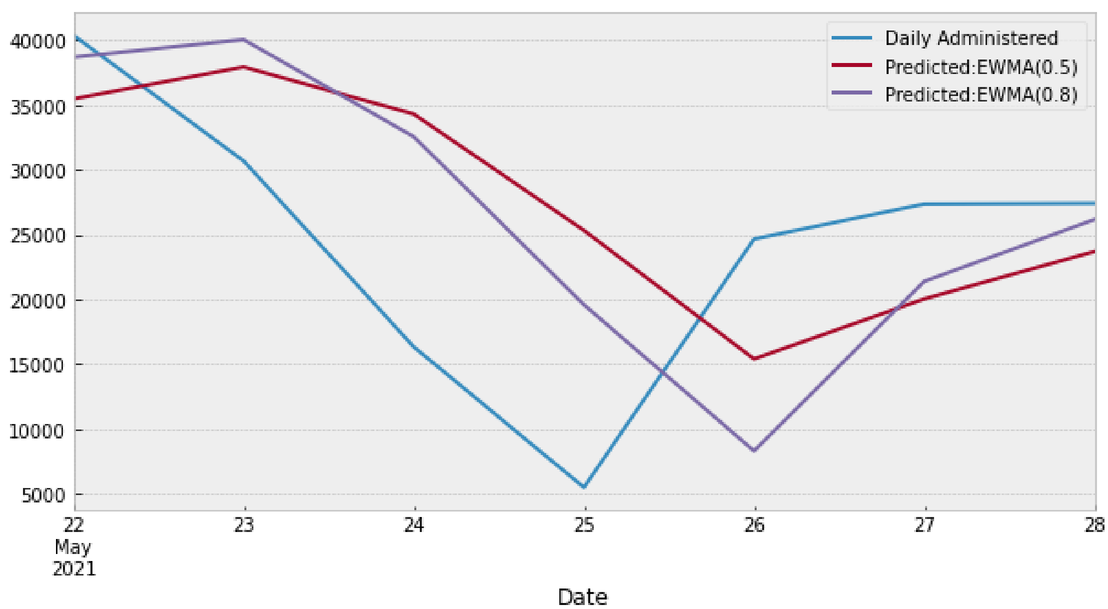
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data["Predicted:EWMA(0.5)"] = preds_p5[-7:]
<ipython-input-26-cc85fedd5278>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data["Predicted:EWMA(0.8)"] = preds_p8[-7:]
```

[26]:

Date	Daily Administered	Predicted:EWMA(0.5)	Predicted:EWMA(0.8)
2021-05-22	40370.0	35455.498657	38672.624945
2021-05-23	30675.0	37912.749329	40030.524989
2021-05-24	16346.0	34293.874664	32546.104998
2021-05-25	5506.0	25319.937332	19586.021000
2021-05-26	24657.0	15412.968666	8322.004200
2021-05-27	27350.0	20034.984333	21390.000840
2021-05-28	27405.0	23692.492167	26158.000168

[27]:
```
test_data.plot()
```

[27]: <AxesSubplot:xlabel='Date'>



[ ]:

task_e

May 17, 2022

[24]:
```
#import the necessary libraries

import numpy as np
import pandas as pd
```

[25]:
```
# Load the dataset

georgia = pd.read_csv("georgia_vaccinations.csv")
indiana= pd.read_csv("indiana_vaccinations.csv")

georgia_vaccines=georgia['Daily Administered'].to_numpy()
indiana_vaccines=indiana['Daily Administered'].to_numpy()
```

[26]:
```
georgia_vaccines_sept = georgia_vaccines[243:269]
georgia_vaccines_nov = georgia_vaccines[300:326]

indiana_vaccines_sept = indiana_vaccines[244:273]
indiana_vaccines_nov = indiana_vaccines[305:334]
```

[27]:
```
# Two sided Paired T test

def paired_t_test(sample1,sample2,tvalue,hypothesis):

    # New Sample D = Xi - Yi
    D = np.subtract(sample1,sample2)

    n = len(D)

    # Mean of the new sample
    D_bar = np.mean(D)

    # Uncorrected std deviation of the new sample
    s_d = np.std(D)

    # Compute the T statistic
    T = (D_bar/s_d)*np.sqrt(n)
    print("T statistic",T)
```

```
    # Using the value of tn-1,0.025 where n is the number of datapoints in the␣
↪sample to accept or reject

    if(abs(T) > tvalue):
        print("Reject Ho:",hypothesis)
    else:
        print("Accept Ho:",hypothesis)


print("For Georgia:")
paired_t_test(georgia_vaccines_sept,georgia_vaccines_nov,2.059539,"Mean of the␣
↪number vaccinations are different for Feb'21 and March'21 in Georgia")
print("\n")
print("For Indiana:")
paired_t_test(indiana_vaccines_sept,indiana_vaccines_nov,2.048407,"Mean of the␣
↪number vaccinations are different for Feb'21 and March'21 in Indiana")
```

```
For Georgia:
T statistic -0.030400115657766028
Accept Ho: Mean of the number vaccinations are different for Feb'21 and March'21
in Georgia


For Indiana:
T statistic -2.481255364562329
Reject Ho: Mean of the number vaccinations are different for Feb'21 and March'21
in Indiana
```

0.1 Inference:

In the month of September, Georgia saw a decrease in the covid cases as well as an increase in the vaccinations. Wheras, in the month of November Georgia saw an increase in the covid cases which may have caused the vaccination rate to grow at a much slower rate. This is in accordance with our result from the hypothesis test that the means of the two months are different as their vaccination growth rates are also different.

In the month of September and November it can be observed that there was a plateau in the number of covid cases with almost the same amount of cases. The vaccination rate is also increasing at the same pace in both of these months. This is in accordance with our result from the hypothesis test that the means of the two months are same.

inference 1

May 17, 2022

[58]:
```python
# import the necessary libraries

import numpy as np
import pandas as pd
import datetime
```

[59]:
```python
########################################## DATA CLEANING␣
↪############################################
```

[60]:
```python
df = pd.read_csv("dataset/GA/NIBRS_VICTIM.csv")
df = df.dropna(subset=["RACE_ID"])
df
```

[60]:
```
        DATA_YEAR  VICTIM_ID  INCIDENT_ID  VICTIM_SEQ_NUM  VICTIM_TYPE_ID  \
0            2020  140089911    127549239               1               4
1            2020  140090335    127544730               1               4
3            2020  140090735    127545052               1               4
4            2020  140094441    127545061               1               4
5            2020  140094498    127545102               2               4
…             …          …          …               …             …
344929       2020  147700653    134360105               1               4
344930       2020  147703114    134360114               1               4
344932       2020  147703158    134364293               1               4
344935       2020  147703256    134360196               1               4
344936       2020  147700748    134360205               1               4

        ASSIGNMENT_TYPE_ID  ACTIVITY_TYPE_ID  OUTSIDE_AGENCY_ID  AGE_ID  \
0                      NaN               NaN                NaN     5.0
1                      NaN               NaN                NaN     5.0
3                      NaN               NaN                NaN     5.0
4                      NaN               NaN                NaN     5.0
5                      NaN               NaN                NaN     5.0
…                   …                 …                  …      …
344929                 NaN               NaN                NaN     5.0
344930                 NaN               NaN                NaN     5.0
344932                 NaN               NaN                NaN     5.0
344935                 NaN               NaN                NaN     5.0
344936                 NaN               NaN                NaN     5.0
```

```
        AGE_NUM SEX_CODE  RACE_ID  ETHNICITY_ID RESIDENT_STATUS_CODE  \
0          37.0        M      1.0           2.0                    R
1          46.0        F      2.0           2.0                    N
3          35.0        M      8.0           1.0                    R
4          84.0        F      1.0           2.0                    R
5          33.0        M      1.0           2.0                    R
…            …       …       …             …                    …
344929     22.0        F      2.0           2.0                    R
344930     43.0        M      1.0           2.0                    R
344932     35.0        M      1.0           2.0                    R
344935     35.0        M      2.0           2.0                    R
344936     18.0        M      1.0           2.0                    R

        AGE_RANGE_LOW_NUM  AGE_RANGE_HIGH_NUM
0                    37.0                 0.0
1                    46.0                 0.0
3                    35.0                 0.0
4                    84.0                 0.0
5                    33.0                 0.0
…                     …                   …
344929               22.0                 0.0
344930               43.0                 0.0
344932               35.0                 0.0
344935               35.0                 0.0
344936               18.0                 0.0

[263099 rows x 16 columns]
```

[61]:

```python
incidents = pd.read_csv("dataset/GA/NIBRS_incident.csv")
incidents
```

[61]:

```
        DATA_YEAR  AGENCY_ID  INCIDENT_ID  NIBRS_MONTH_ID CARGO_THEFT_FLAG  \
0            2020       3627    127542279        15667489              NaN
1            2020       3627    127542321        15667490              NaN
2            2020       3627    127542429        15667491                N
3            2020       3627    127547123        15667489              NaN
4            2020       3627    127542291        15667489              NaN
…             …         …           …              …               …
305995       2020       3488    135933655        17898831              NaN
305996       2020       3488    135930672        17898831                N
305997       2020       3488    135933940        17090516                N
305998       2020       3488    135934042        18280306                N
305999       2020       3488    135930906        18280306              NaN

       SUBMISSION_DATE INCIDENT_DATE REPORT_DATE_FLAG  INCIDENT_HOUR  \
0            19-AUG-20     24-JAN-20              NaN           13.0
```

```
1              19-AUG-20     10-FEB-20                NaN           13.0
2              19-AUG-20     03-MAR-20                NaN           16.0
3              19-AUG-20     13-JAN-20                NaN            9.0
4              19-AUG-20     28-JAN-20                NaN           13.0
…                  …             …               …            …
305995         15-MAR-21     05-OCT-20                NaN            8.0
305996         15-MAR-21     24-OCT-20                NaN           16.0
305997         15-MAR-21     10-NOV-20                NaN            6.0
305998         15-MAR-21     18-DEC-20                NaN           19.0
305999         15-MAR-21     28-DEC-20                NaN           13.0

        CLEARED_EXCEPT_ID CLEARED_EXCEPT_DATE  INCIDENT_STATUS DATA_HOME  \
0                       6                 NaN                0         C
1                       2           10-FEB-20                0         C
2                       6                 NaN                0         C
3                       6                 NaN                0         C
4                       2           28-JAN-20                0         C
…                     …                   …               …        …
305995                  6                 NaN                0         C
305996                  6                 NaN                0         C
305997                  6                 NaN                0         C
305998                  6                 NaN                0         C
305999                  6                 NaN                0         C

       ORIG_FORMAT        DID
0                F   80052715
1                F   80052761
2                F   80050483
3                F   80050299
4                F   80052721
…            …          …
305995           F   97873731
305996           F   97869947
305997           F   97870048
305998           F   97870169
305999           F   97874291

[306000 rows x 15 columns]
```

[62]:
```
georgia_victims = pd.merge(incidents,df,on="INCIDENT_ID")
georgia_victims["INCIDENT_DATE"] = pd.
 ↪to_datetime(georgia_victims["INCIDENT_DATE"])
georgia_victims
```

[62]:
```
        DATA_YEAR_x  AGENCY_ID  INCIDENT_ID  NIBRS_MONTH_ID CARGO_THEFT_FLAG  \
0              2020       3627    127542279        15667489              NaN
1              2020       3627    127542429        15667491                N
```

```
2               2020        3627    127542291        15667489              NaN
3               2020        3627    127542281        15667489                N
4               2020        3627    127542351        15667490              NaN
…                  …           …            …               …              …
263094          2020        3488    135933655        17898831              NaN
263095          2020        3488    135930672        17898831                N
263096          2020        3488    135933940        17090516                N
263097          2020        3488    135934042        18280306                N
263098          2020        3488    135930906        18280306              NaN

       SUBMISSION_DATE INCIDENT_DATE REPORT_DATE_FLAG  INCIDENT_HOUR  \
0            19-AUG-20    2020-01-24              NaN           13.0
1            19-AUG-20    2020-03-03              NaN           16.0
2            19-AUG-20    2020-01-28              NaN           13.0
3            19-AUG-20    2020-01-17              NaN           12.0
4            19-AUG-20    2020-02-13              NaN           13.0
…                    …             …                …              …
263094       15-MAR-21    2020-10-05              NaN            8.0
263095       15-MAR-21    2020-10-24              NaN           16.0
263096       15-MAR-21    2020-11-10              NaN            6.0
263097       15-MAR-21    2020-12-18              NaN           19.0
263098       15-MAR-21    2020-12-28              NaN           13.0

        CLEARED_EXCEPT_ID  … ACTIVITY_TYPE_ID  OUTSIDE_AGENCY_ID AGE_ID  \
0                       6  …              NaN                NaN    5.0
1                       6  …              NaN                NaN    5.0
2                       2  …              NaN                NaN    5.0
3                       4  …              NaN                NaN    5.0
4                       6  …              NaN                NaN    5.0
…                       …  …                …                  …      …
263094                  6  …              NaN                NaN    5.0
263095                  6  …              NaN                NaN    5.0
263096                  6  …              NaN                NaN    5.0
263097                  6  …              NaN                NaN    5.0
263098                  6  …              NaN                NaN    5.0

       AGE_NUM  SEX_CODE  RACE_ID  ETHNICITY_ID  RESIDENT_STATUS_CODE  \
0         15.0         M      1.0           2.0                     R
1         16.0         M      1.0           1.0                     R
2         13.0         F      2.0           2.0                     R
3         55.0         F      2.0           2.0                     R
4         15.0         F      1.0           2.0                     R
…            …         …        …             …                     …
263094    43.0         F      2.0           NaN                   NaN
263095    60.0         M      1.0           NaN                   NaN
263096    27.0         F      1.0           NaN                   NaN
263097    38.0         F      1.0           NaN                   NaN
```

```
263098    23.0          M      1.0          NaN                     NaN

        AGE_RANGE_LOW_NUM  AGE_RANGE_HIGH_NUM
0                    15.0                 0.0
1                    16.0                 0.0
2                    13.0                 0.0
3                    55.0                 0.0
4                    15.0                 0.0
…                     …                  …
263094               43.0                 0.0
263095               60.0                 0.0
263096               27.0                 0.0
263097               38.0                 0.0
263098               23.0                 0.0

[263099 rows x 30 columns]
```

[63]:
```
victims = georgia_victims[["INCIDENT_DATE","RACE_ID"]].sort_values(by =
 ↪"INCIDENT_DATE")
victims
```

[63]:
```
       INCIDENT_DATE  RACE_ID
68995     2020-01-01      2.0
143864    2020-01-01      2.0
197615    2020-01-01      1.0
107678    2020-01-01      1.0
107595    2020-01-01      2.0
…              …       …
188259    2020-12-31      1.0
212903    2020-12-31      1.0
142341    2020-12-31      2.0
69862     2020-12-31      2.0
54554     2020-12-31      1.0

[263099 rows x 2 columns]
```

[64]:
```
asian_victims_georgia = victims[victims.RACE_ID==4]
asian_victims_georgia
```

[64]:
```
       INCIDENT_DATE  RACE_ID
94367     2020-01-01      4.0
160424    2020-01-01      4.0
86724     2020-01-01      4.0
112922    2020-01-01      4.0
225268    2020-01-02      4.0
…              …       …
72131     2020-12-31      4.0
```

```
142285    2020-12-31      4.0
115810    2020-12-31      4.0
218547    2020-12-31      4.0
203371    2020-12-31      4.0

[3021 rows x 2 columns]
```

[65]:
```python
asian_victims = pd.DataFrame(asian_victims_georgia["INCIDENT_DATE"].
 ↪value_counts())
asian_victims= asian_victims.reset_index()
asian_victims.columns=["Date","Number of asian victims"]
asian_victims = asian_victims.sort_values(by = "Date").set_index(['Date'])
asian_victims
```

[65]:
```
            Number of asian victims
Date
2020-01-01                        4
2020-01-02                        9
2020-01-03                        2
2020-01-04                       10
2020-01-05                        4
…                               …
2020-12-27                       21
2020-12-28                       13
2020-12-29                       10
2020-12-30                       10
2020-12-31                        5

[366 rows x 1 columns]
```

[66]:
```python
# Perform Tukey's Rule to Remove outliers

def tukey_range(values):

    alpha = 1.5
    sorted_values = sorted(values)

    # Q1
    q1_index = int(np.ceil(0.25 * len(values)))
    q1 = sorted_values[q1_index]

    # Q3
    q3_index = int(np.ceil(0.75 * len(values)))
    q3 = sorted_values[q3_index]

    # IQR
    iqr = q3 - q1
```

```
    return q1 - (alpha * iqr), q3 + (alpha * iqr)

def clean_data(series):

    series = series[series >=0]
    lower_limit, upper_limit = tukey_range(series.values)
    print("Lower Range: {}, Upper Range: {}\n".format(lower_limit, upper_limit))

    outliers = list(series[series < lower_limit].values) + list(series[series >␣
 ↪upper_limit].values)
    print("Total Outliers: {}\n".format(len(outliers)))
    print("Outliers: \n{}\n".format(outliers))

    outliers_index = list(series[series < lower_limit].index) +␣
 ↪list(series[series > upper_limit].index)
    return series.drop(index = outliers_index)

asian_victims= clean_data(asian_victims["Number of asian victims"])
asian_victims = pd.DataFrame(asian_victims)
```

```
Lower Range: -1.5, Upper Range: 18.5

Total Outliers: 5

Outliers:
[19, 19, 20, 19, 21]
```

[67]:
```
# Georgia cases dataset
georgia = pd.read_csv("georgia_cases_deaths.csv")
georgia["submission_date"] = pd.to_datetime(georgia["submission_date"])
georgia_cases = georgia[["submission_date","new_case"]]
georgia_cases = georgia_cases.set_index(["submission_date"])
georgia_cases
```

[67]:

	new_case
submission_date	
2020-01-22	0.0
2020-01-23	0.0
2020-01-24	0.0
2020-01-25	0.0
2020-01-26	0.0
…	…
2022-05-01	0.0
2022-05-02	0.0
2022-05-03	0.0

```
2022-05-05           0.0
2022-05-06           0.0

[731 rows x 1 columns]
```

[68]:
```
##################################### HYPOTHESIS TESTING␣
↪#########################################
```

Inference : As the number of Covid cases increases the crimes against Asians also increases.

[69]:
```
# Get Data for September and November

asian_victims_sept = asian_victims.loc[datetime.date(2020,9,1):datetime.
↪date(2020,9,30)][['Number of asian victims']]
asian_victims_sept= asian_victims_sept.to_numpy()
asian_victims_nov = asian_victims.loc[datetime.date(2020,11,1):datetime.
↪date(2020,11,30)][['Number of asian victims']]
asian_victims_nov = asian_victims_nov.to_numpy()

cases_sept = georgia_cases.loc[datetime.date(2020,9,1):datetime.
↪date(2020,9,30)][['new_case']]
cases_sept = cases_sept.to_numpy()
cases_nov = georgia_cases.loc[datetime.date(2020,11,1):datetime.
↪date(2020,11,30)][['new_case']]
cases_nov = cases_nov.to_numpy()
```

[70]:
```
# Performing one sided T test for two sample

def t_test_unpaired(sample1,sample2,tvalue,hypothesis):
    n = len(sample1)
    m = len(sample2)

    # Computing the means of sample1 and sample2
    X_bar  = np.mean(sample1)
    Y_bar = np.mean(sample2)

    # Computing the corrected std deviation of sample1 and sample2
    s1 = np.std(sample1,ddof=1)
    s2 = np.std(sample2,ddof=1)

    # Compute the T statistic
    T = (X_bar - Y_bar)/(np.sqrt(((s1**2)/n)+((s2**2)/m)))
    print("T statistic",T)

    # Using the value of tn+m-2,0.05 where n and m is the number of datapoints␣
↪in sample1 and sample2
    # to accept or reject
```

```
    if(T < tvalue):
        print("Reject Ho:",hypothesis)
    else:
        print("Accept Ho:",hypothesis)

print("Hypothesis Test on Crime against Asians")
t_test_unpaired(asian_victims_sept,asian_victims_nov,-1.672522,
"Mean of the crimes against Asians in September is greater than the mean in the␣
 ↪month of November")
print("\n")

print("Hypothesis Test on Covid Cases in Georgia")
t_test_unpaired(cases_sept,cases_nov,-1.672522,
"Mean of the number of COVID cases in September is greater than the mean in the␣
 ↪month of November")
```

```
Hypothesis Test on Crime against Asians
T statistic -6.001271101758944
Reject Ho: Mean of the crimes against Asians in September is greater than the
mean in the month of November


Hypothesis Test on Covid Cases in Georgia
T statistic -4.00719930559382
Reject Ho: Mean of the number of COVID cases in September is greater than the
mean in the month of November
```

0.1 Inference

In the month of September there is a decrease in the number of covid cases. However, it can be observed that in the month of November the number of covid cases begin to increase mainly due to the elections taking place in Georgia at that time. Similarly, we observe that the number of crimes against Asians decreases in September but begins to increase rapidly in the month of November. This is in accordance with our result from the hypothesis test that the means of the crimes against Asians in the month of November is more than the mean in September and the number of covid cases in November is more than in September. Thus we can also imply that as the covid cases increases the crimes against Asians also increases.

The reason we chose to analyze the crime data against Asians in Georgia for the purposes of this inference is that it helps us to to understand the trends in the crime so as to make people aware about the situation in Georgia. This is especially needed to avoid future crimes against Asians such as the one observed in the March of 2021 in Atlanta,Georgia. This analysis can thus be used by law officials to enforce more safety protocols so as to make everyone in the community feel safe.

inference_2

May 17, 2022

[1]:
```python
import numpy as np
import pandas as pd
import datetime
```

1 Pearson Correlation Inference

[2]:
```python
def pearsonCorr(x, y):
    assert len(x) == len(y)

    x_bar = np.mean(x)
    y_bar = np.mean(y)

    corr = np.dot(x - x_bar, y - y_bar) / (np.sqrt(np.sum((x - x_bar)**2)) * np.
 ↪sqrt(np.sum((y - y_bar)**2)))

    return corr
```

[3]:
```python
def runHypothesis(corr, hypothesis):
    print(corr)
    if corr < -.5:
        print("Negatively Correlated")
    elif corr > .5:
        print("Positively Correlated")
    else:
        print("Not Correlated")

    if abs(corr) > 0.5:
        print("Reject Ho:", hypothesis)
    else:
        print("Accept Ho:", hypothesis)
```

[4]:
```python
# Perform Tukey's Rule to Remove outliers

def tukey_range(values):

    alpha = 1.5
    sorted_values = sorted(values)
```

```
    # Q1
    q1_index = int(np.ceil(0.25 * len(values)))
    q1 = sorted_values[q1_index]

    # Q3
    q3_index = int(np.ceil(0.75 * len(values)))
    q3 = sorted_values[q3_index]

    # IQR
    iqr = q3 - q1

    return q1 - (alpha * iqr), q3 + (alpha * iqr)

def clean_data(series):

    series = series[series >=0]
    lower_limit, upper_limit = tukey_range(series.values)
    print("Lower Range: {}, Upper Range: {}\n".format(lower_limit, upper_limit))

    outliers = list(series[series < lower_limit].values) + list(series[series >␣
 ↪upper_limit].values)
    print("Total Outliers: {}\n".format(len(outliers)))
    print("Outliers: \n{}\n".format(outliers))

    outliers_index = list(series[series < lower_limit].index) +␣
 ↪list(series[series > upper_limit].index)
    return series.drop(index = outliers_index)
```

2 Null Hypothesis Ho:

3 COVID-19 cases are not correlated with the number of arrests.

Time Range of First Outbreak - Lockdown in both states

[5]:
```
start_date = datetime.date(2020,3,1)
end_date = datetime.date(2020,3,31)
```

Time Range of one month once the lockdown is lifted

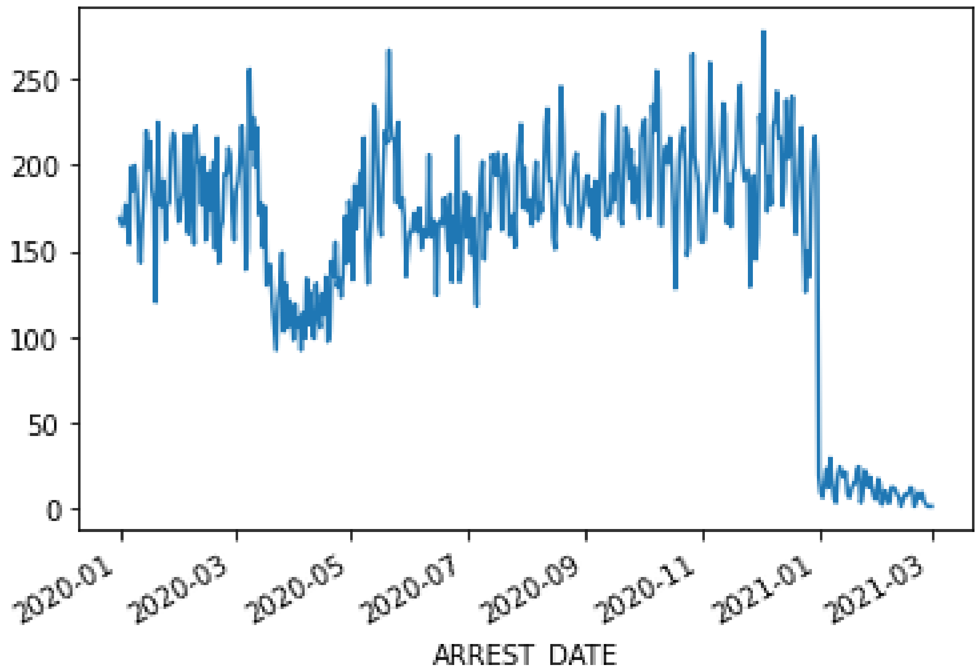
[44]:
```
start_date_1 = datetime.date(2020,7,1)
end_date_1 = datetime.date(2020,7,31)
```

4 Gerogia

Check the trends of arrests and crime incidents during and after lockdown

```
[14]: ga_df = pd.read_csv("dataset/GA/NIBRS_ARRESTEE.csv")
ga_df["ARREST_DATE"] = pd.to_datetime(ga_df["ARREST_DATE"])
ga_df["ARREST_COUNT"] = 1
ga_df.groupby(by = ["ARREST_DATE"]).agg('sum')["ARREST_COUNT"].plot()
```

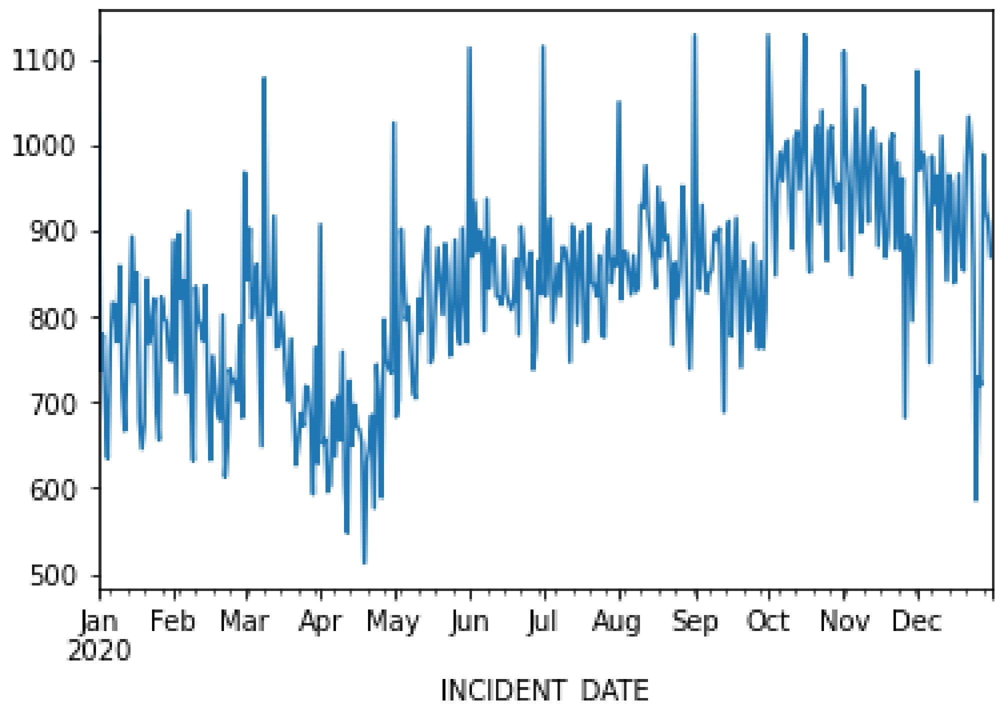
```
[14]: <AxesSubplot:xlabel='ARREST_DATE'>
```



We see that there is a sharp decline in arrests during March 2020.

```
[64]: ga_incident = pd.read_csv("dataset/GA/NIBRS_incident.csv")
ga_incident["INCIDENT_DATE"] = pd.to_datetime(ga_incident["INCIDENT_DATE"])
ga_incident["INCIDENT_COUNT"] = 1
ga_incident = ga_incident.groupby(by = ["INCIDENT_DATE"]).
 ↪agg('sum')["INCIDENT_COUNT"]
ga_incident.plot()
```

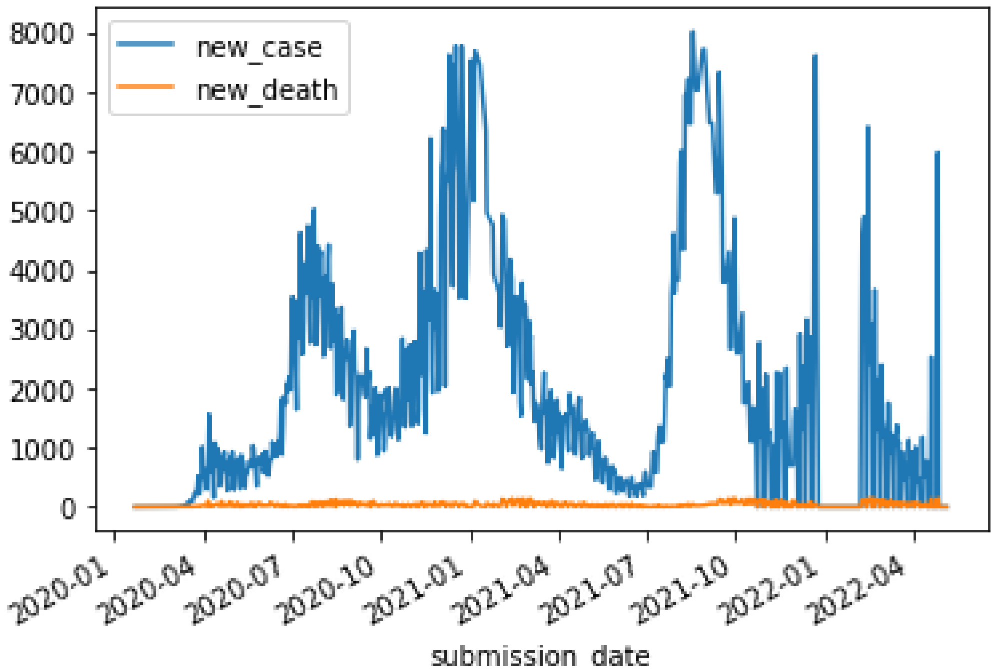
```
[64]: <AxesSubplot:xlabel='INCIDENT_DATE'>
```



There is noticable decline in crime incidents as well during March-April 2020.

[15]:
```
ga_cases = pd.read_csv("georgia_cases_deaths.csv")
ga_cases["submission_date"] = pd.to_datetime(ga_cases["submission_date"])
ga_cases.groupby(by = ["submission_date"]).agg('sum')[["new_case",␣
 ↪"new_death"]].plot()
```

[15]: <AxesSubplot:xlabel='submission_date'>



5 During Lockdown

[16]:
```
s1 = ga_cases.groupby(by = ["submission_date"]).agg('sum')["new_case"].
↪loc[start_date:end_date]
s2 = ga_df.groupby(by = ["ARREST_DATE"]).agg('sum')["ARREST_COUNT"].
↪loc[start_date:end_date]
ga_corr = pd.concat([s1,s2], axis = 1).dropna()

#Remove Outliers using Tukey's rule
ga_corr['new_case'] = clean_data(ga_corr['new_case'])
ga_corr['ARREST_COUNT'] = clean_data(ga_corr['ARREST_COUNT'])

ga_corr.dropna(inplace = True)
```

```
Lower Range: -305.0, Upper Range: 543.0

Total Outliers: 1

Outliers:
[1012.0]

Lower Range: 26.5, Upper Range: 302.5

Total Outliers: 0
```

```
Outliers:
[]
```

[17]:
```
corr = pearsonCorr(ga_corr['new_case'].to_numpy(), ga_corr['ARREST_COUNT'].
 ↪to_numpy())
runHypothesis(corr, "COVID-19 Cases in March 2021 are not correlated with March␣
 ↪2021 Arrests in GA state")
```

```
-0.6712694907495717
Negatively Correlated
Reject Ho: COVID-19 Cases in March 2021 are not correlated with March 2021
Arrests in GA state
```

6 After Lockdown is lifted

[45]:
```
s1 = ga_cases.groupby(by = ["submission_date"]).agg('sum')["new_case"].
 ↪loc[start_date_1:end_date_1]
s2 = ga_df.groupby(by = ["ARREST_DATE"]).agg('sum')["ARREST_COUNT"].
 ↪loc[start_date_1:end_date_1]
ga_corr = pd.concat([s1,s2], axis = 1).dropna()

#Remove Outliers using Tukey's rule
ga_corr['new_case'] = clean_data(ga_corr['new_case'])
ga_corr['ARREST_COUNT'] = clean_data(ga_corr['ARREST_COUNT'])

ga_corr.dropna(inplace = True)
```

```
Lower Range: 1178.0, Upper Range: 6066.0

Total Outliers: 0

Outliers:
[]

Lower Range: 106.0, Upper Range: 258.0

Total Outliers: 0

Outliers:
[]
```

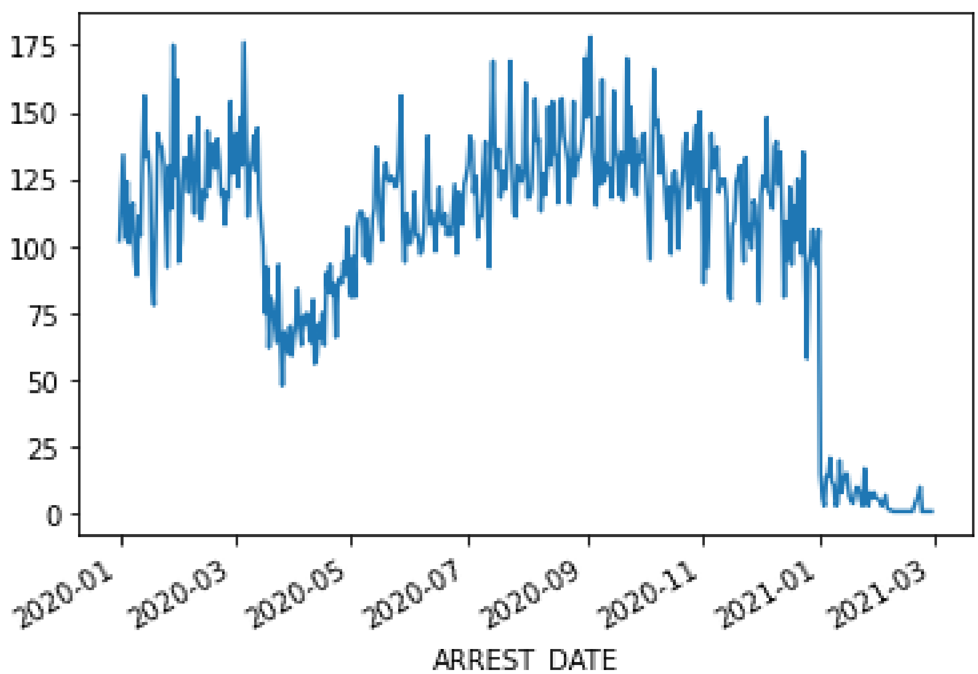
[46]:
```
corr = pearsonCorr(ga_corr['new_case'].to_numpy(), ga_corr['ARREST_COUNT'].
 ↪to_numpy())
runHypothesis(corr, "COVID-19 Cases in March 2021 are not correlated with March␣
 ↪2021 Arrests in GA state")
```

```
0.4252553588572149
Not Correlated
Accept Ho: COVID-19 Cases in March 2021 are not correlated with March 2021
Arrests in GA state
```

7 Indiana

[33]:
```
in_df = pd.read_csv("dataset/IN/NIBRS_ARRESTEE.csv")
in_df["ARREST_DATE"] = pd.to_datetime(in_df["ARREST_DATE"])
in_df["ARREST_COUNT"] = 1
in_df.groupby(by = ["ARREST_DATE"]).agg('sum')["ARREST_COUNT"].plot()
```

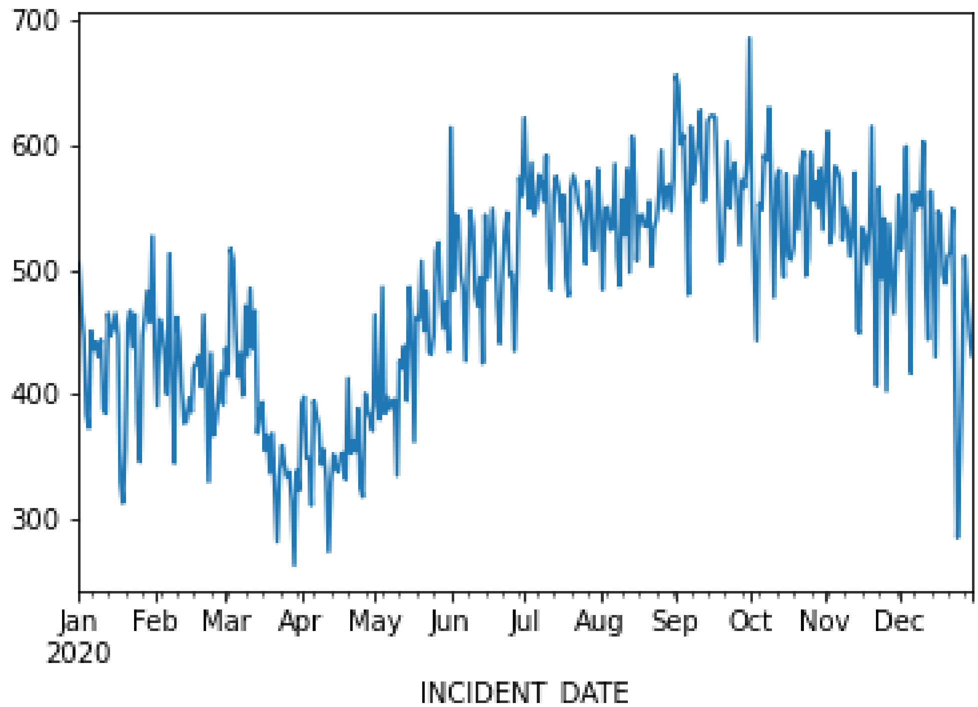
[33]:
```
<AxesSubplot:xlabel='ARREST_DATE'>
```



There is a sharp decline in arrests during March 2020.

[56]:
```
in_incident = pd.read_csv("dataset/IN/NIBRS_incident.csv")
in_incident["INCIDENT_DATE"] = pd.to_datetime(in_incident["INCIDENT_DATE"])
in_incident["INCIDENT_COUNT"] = 1
in_incident.groupby(by = ["INCIDENT_DATE"]).agg('sum')["INCIDENT_COUNT"].plot()
```

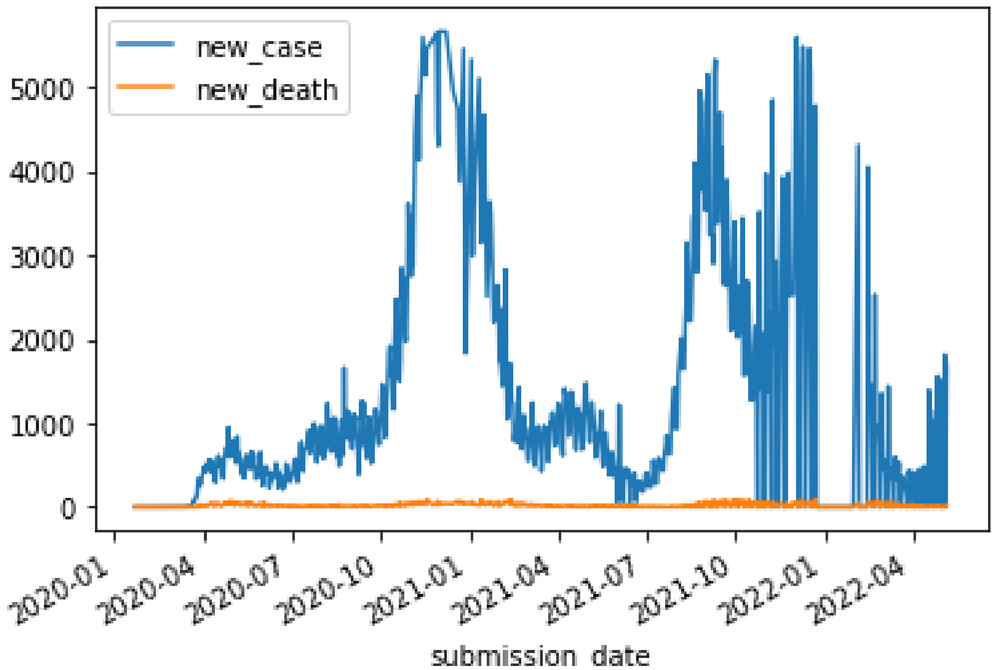
[56]:
```
<AxesSubplot:xlabel='INCIDENT_DATE'>
```



There is a noticable decline in crime incidents during March-April 2020.

[34]:
```
in_cases = pd.read_csv("indiana_cases_deaths.csv")
in_cases["submission_date"] = pd.to_datetime(in_cases["submission_date"])
in_cases.groupby(by = ["submission_date"]).agg('sum')[["new_case",␣
↪"new_death"]].plot()
```

[34]: <AxesSubplot:xlabel='submission_date'>



8 During Lockdown

[35]:
```
s1 = in_cases.groupby(by = ["submission_date"]).agg('sum')["new_case"].
 ↪loc[start_date:end_date]
s2 = in_df.groupby(by = ["ARREST_DATE"]).agg('sum')["ARREST_COUNT"].
 ↪loc[start_date:end_date]
in_corr = pd.concat([s1,s2], axis = 1).dropna()

#Remove Outliers using Tukey's rule
in_corr['new_case'] = clean_data(in_corr['new_case'])
in_corr['ARREST_COUNT'] = clean_data(in_corr['ARREST_COUNT'])

in_corr.dropna(inplace = True)
```

```
Lower Range: -165.5, Upper Range: 278.5

Total Outliers: 3

Outliers:
[336.0, 282.0, 373.0]

Lower Range: -23.0, Upper Range: 225.0

Total Outliers: 0
```

```
Outliers:
[]
```

[36]:
```python
corr = pearsonCorr(in_corr['new_case'].to_numpy(), in_corr['ARREST_COUNT'].
 ↪to_numpy())
runHypothesis(corr, "COVID-19 Cases in March 2021 are not correlated with March␣
 ↪2021 Arrests in IN state")
```

```
-0.6310159282421549
Negatively Correlated
Reject Ho: COVID-19 Cases in March 2021 are not correlated with March 2021
Arrests in IN state
```

9 After Lockdown is lifted

[47]:
```python
s1 = in_cases.groupby(by = ["submission_date"]).agg('sum')["new_case"].
 ↪loc[start_date_1:end_date_1]
s2 = in_df.groupby(by = ["ARREST_DATE"]).agg('sum')["ARREST_COUNT"].
 ↪loc[start_date_1:end_date_1]
in_corr = pd.concat([s1,s2], axis = 1).dropna()

#Remove Outliers using Tukey's rule
in_corr['new_case'] = clean_data(in_corr['new_case'])
in_corr['ARREST_COUNT'] = clean_data(in_corr['ARREST_COUNT'])

in_corr.dropna(inplace = True)
```

```
Lower Range: 42.0, Upper Range: 1338.0

Total Outliers: 0

Outliers:
[]

Lower Range: 91.5, Upper Range: 167.5

Total Outliers: 2

Outliers:
[169, 169]
```

[48]:
```python
corr = pearsonCorr(in_corr['new_case'].to_numpy(), in_corr['ARREST_COUNT'].
 ↪to_numpy())
runHypothesis(corr, "COVID-19 Cases in March 2021 are not correlated with March␣
 ↪2021 Arrests in IN state")
```

```
0.2232995139862886
Not Correlated
Accept Ho: COVID-19 Cases in March 2021 are not correlated with March 2021
Arrests in IN state
```

10 Inference

For both the states, in the month of March, there is a sharp increase in the COVID-19 cases, whereas huge decline in the number of arrests. After running the pearson correlation test, we find that they are indeed negatively correlated and rejects the null hyopthesis. This could be due to inforced lockdown after the first outbreak of COVID-19 cases. Once the lockdown is lifted, we can see that hypothesis is no longer rejected and there is a very weak correlation between cases and arrests.

Why is this inference useful? After the initial decrease in March 2020, jail populations rebounded but stabilized in July-August 2020. During this time, there was no substantial increase in overall crime. We confirm that by looking at the noticable decline in overall incidents for both the states during March-April 2020. There are challenges ahead in keeping jail populations low by maintaining lower arrests. The response to COVID-19 has shown that such reforms are possible and can safely reduce the number of persons held in jail but sustaining lower jail populations will require maintaining these reforms in some manner. (https://safetyandjusticechallenge.org/wp-content/uploads/2021/07/The-Impact-of-COVID-19-on-Crime-Arrests-and-Jail-Populations-JFA-Institute.pdf)

[ ]:

```

```

inference_3

May 17, 2022

1 The Question

In this piece, we make an effort to study if the introduction of vaccines cause any change in crimes? The reasons for why this maybe a useful inference: 1. Vaccines could've been a very limited resource initially and people would want to make sure they get it as soon no matter what. 2. Anti-vaxxers!!!

2 Null Hypothesis Ho

2.0.1 H0: The start of COVID vaccinations didn't cause any "change" in number of crime incidents.

To study this, we compute average crime incidents reported per day in the ~15 days interval before and after vaccinations went live. We use paired Wald's test as the statistic to measure the significance.

[43]:
```
import datetime

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

from scipy.stats import norm
```

3 Exploratory Data Analysis

But first let's exlore the data a bit.

[2]:
```
states = ['indiana', 'georgia']
dfs = []
for state in states:
    df = pd.read_csv(f'./{state}_vaccinations.csv')
    df['Date'] = pd.to_datetime(df['Date'])
    dfs.append(df)
in_vacc, georgia_vacc = dfs[:]
```

[3]:
```
in_vacc.head()
```

[3]:
```
        Date  Daily Administered
0 2020-12-14                 0.0
1 2020-12-15                 0.0
2 2020-12-16                 0.0
3 2020-12-17               686.0
4 2020-12-18              2092.0
```

[4]:
```
in_vacc.tail()
```

[4]:
```
          Date  Daily Administered
490 2022-05-05              6426.0
491 2022-05-06              6021.0
492 2022-05-07              6047.0
493 2022-05-08              6831.0
494 2022-05-09              3613.0
```

[5]:
```
georgia_vacc.head()
```

[5]:
```
        Date  Daily Administered
0 2020-12-14                 0.0
1 2020-12-15                 0.0
2 2020-12-16                 0.0
3 2020-12-17                 0.0
4 2020-12-18               198.0
```

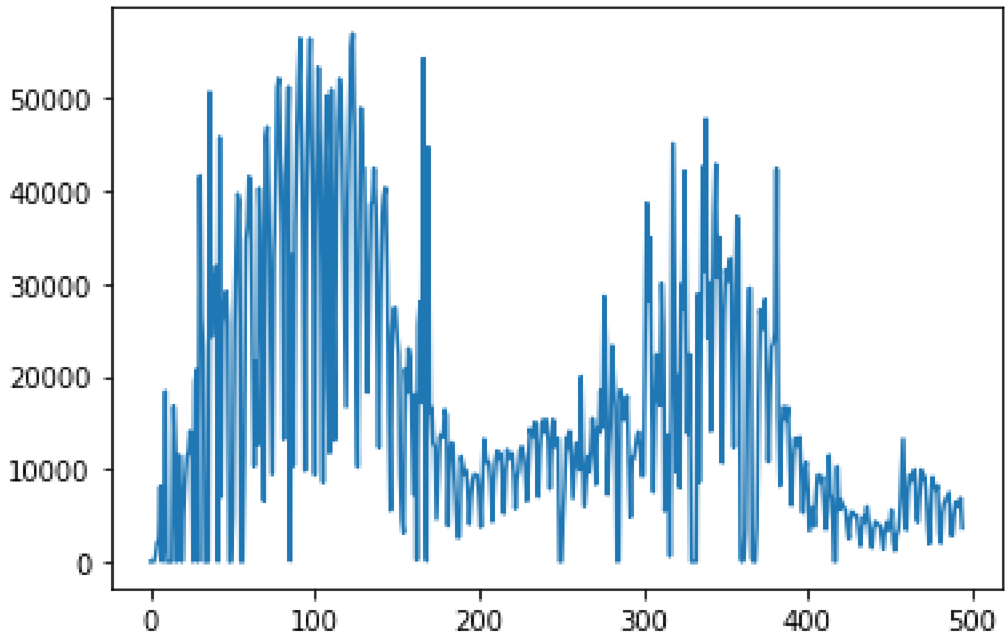
[6]:
```
georgia_vacc.tail()
```

[6]:
```
          Date  Daily Administered
479 2022-05-05              8753.0
480 2022-05-06             11189.0
481 2022-05-07              9993.0
482 2022-05-08               171.0
483 2022-05-09                 1.0
```

For both datasets, the record starts from '2020-12-14' and end on '2022-05-09'

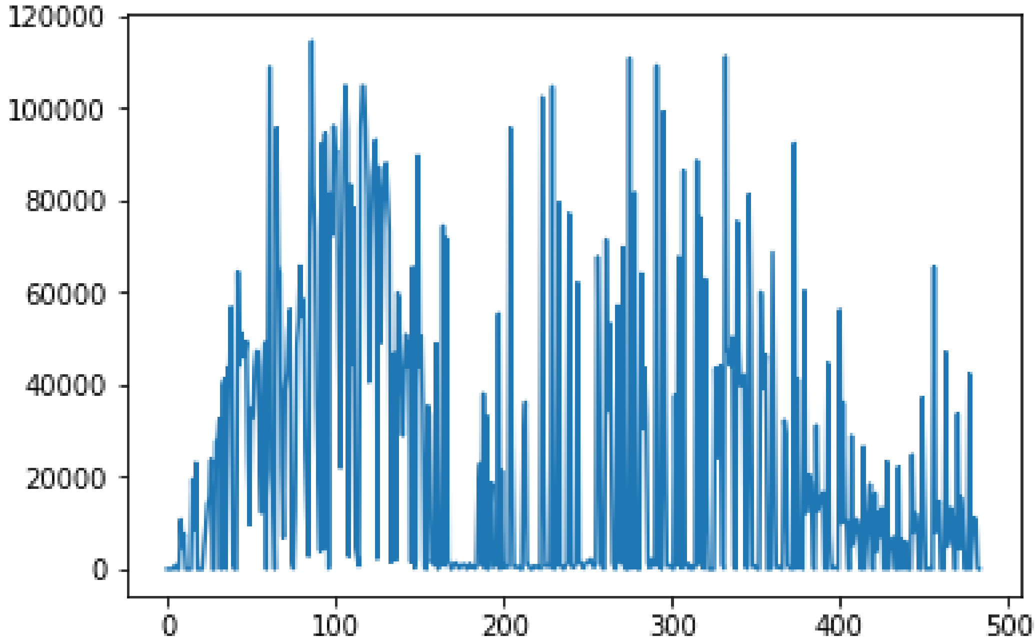
[7]:
```
in_vacc['Daily Administered'].plot()
```

[7]:
```
<AxesSubplot:>
```



[8]:
```
georgia_vacc['Daily Administered'].plot()
```

[8]: <AxesSubplot:>



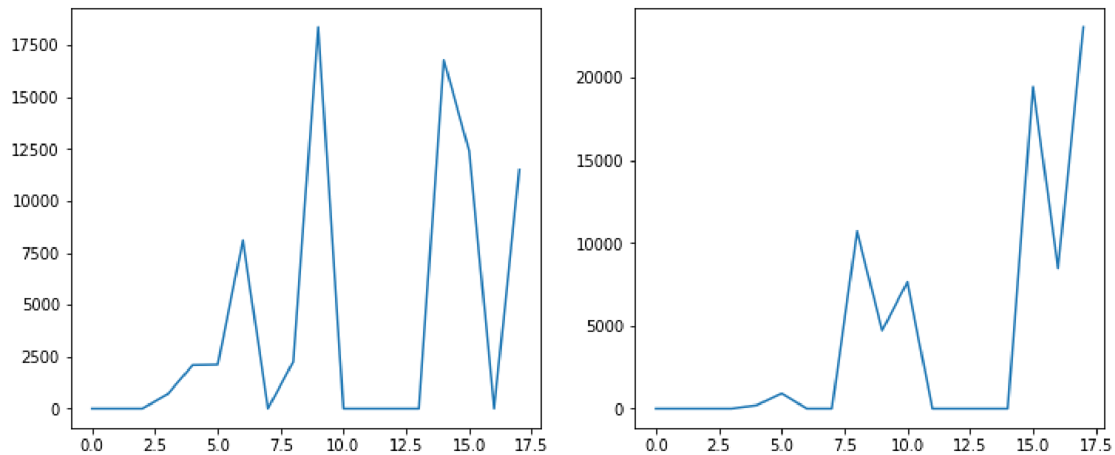
3.1 Slice vaccination data

Our crime incident dataset has information only until Dec 2020 so we slice the vaccine data until that date.

[9]:
```
in_vacc = in_vacc[in_vacc['Date'] < '2021-01-01']
georgia_vacc = georgia_vacc[georgia_vacc['Date'] < '2021-01-01']
```

[10]:
```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))

ax[0].plot(in_vacc['Date'].index, in_vacc['Daily Administered'])
ax[1].plot(georgia_vacc['Date'].index, georgia_vacc['Daily Administered'])
```

[10]:
```
[<matplotlib.lines.Line2D at 0x7f27ba8a3f70>]
```



We would like to see if people started having crazy thoughts about vaccines. And if their crazy thoughts made them commit petty crimes.

4 Indiana: Crime Incidents Before and After Vaccinations

Since the crime "incident" dataset is only available between 15th Dec 2020 and 31st Dec 2020, and vaccinations start on 14th Dec 2020, we analyze the difference in crime 15 days before and 15 days after vaccinations started.

[12]:
```
before_vacc_start = '2020-12-01'
before_vacc_end = '2020-12-13'

after_vacc_start = '2020-12-14'
after_vacc_end = '2020-12-31'
```

[13]:
```
# get incident count
indiana_crime_incidents = "dataset/IN/NIBRS_incident.csv"
```

```
ic_df = pd.read_csv(indiana_crime_incidents)
ic_df['INCIDENT_DATE'] = pd.to_datetime(ic_df['INCIDENT_DATE'])
ic_df = ic_df.sort_values(by = ['INCIDENT_DATE'])

ic_df["INCIDENT_COUNT"] = 1
```

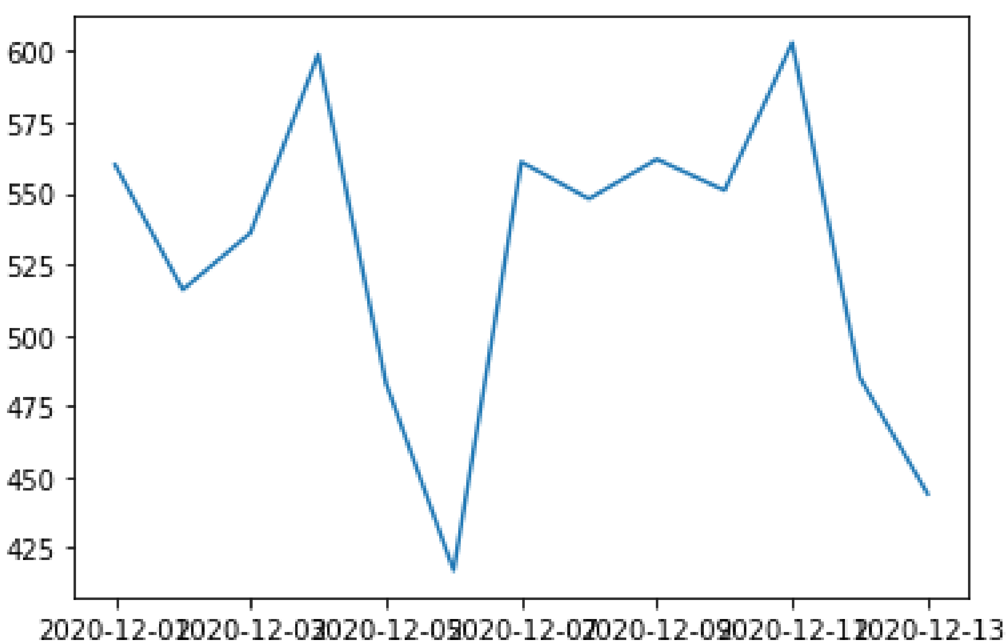
4.0.1 Compute before stats

[24]:
```
ic_before = ic_df[(ic_df['INCIDENT_DATE'] >= before_vacc_start) &␣
 ↪(ic_df['INCIDENT_DATE'] <= before_vacc_end)]

ic_before_count = len(ic_before.groupby(by = ["INCIDENT_DATE"]).
 ↪agg('sum')["INCIDENT_COUNT"])
ic_before_mean_incidents = ic_before.groupby(by = ["INCIDENT_DATE"]).
 ↪agg('sum')["INCIDENT_COUNT"].mean()
ic_before_var_incidents = ic_before.groupby(by = ["INCIDENT_DATE"]).
 ↪agg('sum')["INCIDENT_COUNT"].var()

print(f'Before: Count={ic_before_count}, Mean={ic_before_mean_incidents},␣
 ↪Var={ic_before_var_incidents}')
```

```
Before: Count=13, Mean=528.0769230769231, Var=3193.5769230769233
```

[38]:
```
plt.plot(ic_before.groupby(by = ["INCIDENT_DATE"]).agg('sum')["INCIDENT_COUNT"])
```

[38]: `[<matplotlib.lines.Line2D at 0x7f27b9bbd2b0>]`



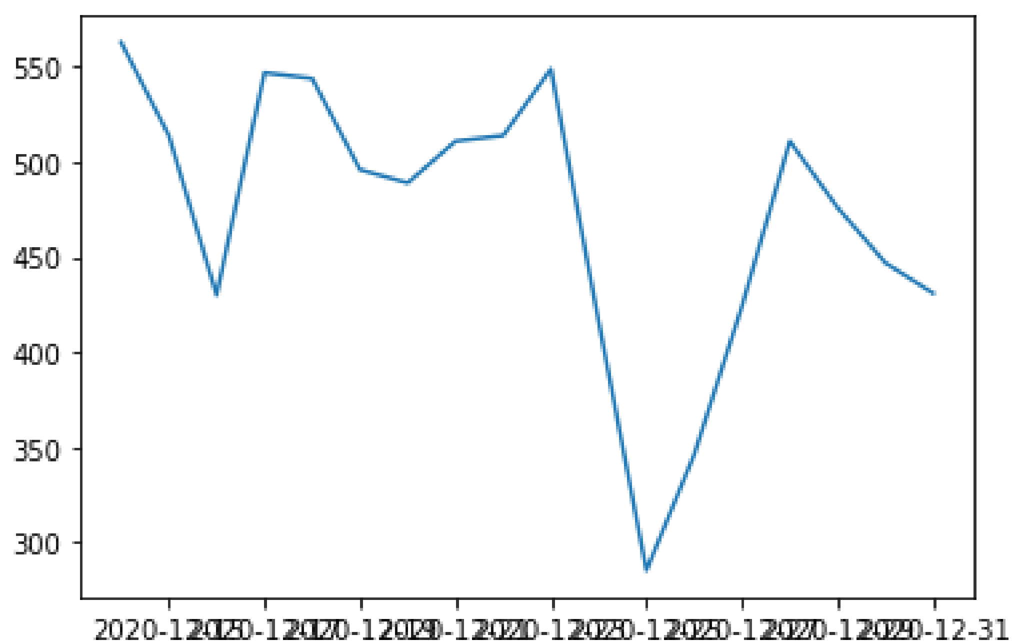
4.0.2 Compute after stats

[25]:
```python
ic_after = ic_df[(ic_df['INCIDENT_DATE'] >= after_vacc_start) & (ic_df['INCIDENT_DATE'] <= after_vacc_end)]

ic_after_count = len(ic_after.groupby(by = ["INCIDENT_DATE"]).agg('sum')["INCIDENT_COUNT"])
ic_after_mean_incidents = ic_after.groupby(by = ["INCIDENT_DATE"]).agg('sum')["INCIDENT_COUNT"].mean()
ic_after_var_incidents = ic_after.groupby(by = ["INCIDENT_DATE"]).agg('sum')["INCIDENT_COUNT"].var()

print(f'After: Count={ic_after_count}, Mean={ic_after_mean_incidents}, Var={ic_after_var_incidents}')
```

```
After: Count=18, Mean=471.8333333333333, Var=5460.264705882352
```

[40]:
```python
plt.plot(ic_after.groupby(by = ["INCIDENT_DATE"]).agg('sum')["INCIDENT_COUNT"])
```

[40]:
```
[<matplotlib.lines.Line2D at 0x7f27b95e21c0>]
```



4.0.3 Compute Wald's Statistic

This is a two tailed Wald's test. We computed sample means of count of incidents and check it against alpha=0.05

[49]:
```
delta = ic_after_mean_incidents - ic_before_mean_incidents
W = delta / math.sqrt(ic_after_var_incidents/ic_after_count +␣
↪ic_before_var_incidents/ic_before_count)
print(f'W = {W}')
p_val = (1 - norm.cdf(abs(W), 0, 1)) * 2
print(f'p-value = {p_val} < 0.05? | Reject Null: {p_val < 0.05}')
```

```
W = -2.400400858168235
p-value = 0.016377126422557176 < 0.05? | Reject Null: True
```

5 Georgia: Crime Incidents Before and After Vaccinations

Since the crime "incident" dataset is only available between 15th Dec 2020 and 31st Dec 2020, and vaccinations start on 14th Dec 2020, we analyze the difference in crime 15 days before and 15 days after vaccinations started.

[29]:
```
# get incident count
georgia_crime_incidents = "dataset/GA/NIBRS_incident.csv"
ga_df = pd.read_csv(georgia_crime_incidents)
ga_df['INCIDENT_DATE'] = pd.to_datetime(ga_df['INCIDENT_DATE'])
ga_df = ga_df.sort_values(by = ['INCIDENT_DATE'])

ga_df["INCIDENT_COUNT"] = 1
```

5.0.1 Compute before stats

[31]:
```
ga_before = ga_df[(ga_df['INCIDENT_DATE'] >= before_vacc_start) &␣
↪(ga_df['INCIDENT_DATE'] <= before_vacc_end)]

ga_before_count = len(ga_before.groupby(by = ["INCIDENT_DATE"]).
↪agg('sum')["INCIDENT_COUNT"])
ga_before_mean_incidents = ga_before.groupby(by = ["INCIDENT_DATE"]).
↪agg('sum')["INCIDENT_COUNT"].mean()
ga_before_var_incidents = ga_before.groupby(by = ["INCIDENT_DATE"]).
↪agg('sum')["INCIDENT_COUNT"].var()

print(f'Before: Count={ga_before_count}, Mean={ga_before_mean_incidents},␣
↪Var={ga_before_var_incidents}')
```

```
Before: Count=13, Mean=943.6153846153846, Var=6963.25641025641
```

5.0.2 Compute after stats

[32]:
```
ga_after = ga_df[(ga_df['INCIDENT_DATE'] >= after_vacc_start) &␣
↪(ga_df['INCIDENT_DATE'] <= after_vacc_end)]
```

```
ga_after_count = len(ga_after.groupby(by = ["INCIDENT_DATE"]).
 ↪agg('sum')["INCIDENT_COUNT"])
ga_after_mean_incidents = ga_after.groupby(by = ["INCIDENT_DATE"]).
 ↪agg('sum')["INCIDENT_COUNT"].mean()
ga_after_var_incidents = ga_after.groupby(by = ["INCIDENT_DATE"]).
 ↪agg('sum')["INCIDENT_COUNT"].var()

print(f'After: Count={ga_after_count}, Mean={ga_after_mean_incidents},␣
 ↪Var={ga_after_var_incidents}')
```

```
After: Count=18, Mean=885.2222222222222, Var=12899.830065359478
```

5.0.3 Compute Wald's Statistic

This is a two tailed Wald's test. We computed sample means of count of incidents and check it against alpha=0.05

[50]:
```
delta = ga_after_mean_incidents - ga_before_mean_incidents
W = delta / math.sqrt(ga_after_var_incidents/ga_after_count +␣
 ↪ga_before_var_incidents/ga_before_count)
print(f'W = {W}')
p_val = (1 - norm.cdf(abs(W), 0, 1)) * 2
print(f'p-value = {p_val} < 0.05? | Reject Null: {p_val < 0.05}')
```

```
W = -1.6500957087439831
p-value = 0.0989233623179373 < 0.05? | Reject Null: False
```

5.1 Applicability of Wald's test

Since we are modeling the number of occurrences of a random event, that is the number of crime incidents each day, we think of the data following a Poisson distribution. The MLE estimate of Poisson is the sample mean. Since MLE estimates are Asymptotically Normal, we can apply Wald's test here.

6 Inference

In this piece, we tried evaluating a probably unheard of hypothesis i.e. trying to find if vaccinations and crimes are correlated. This information might be useful to detect early signs of coordinated crimes for a certain scarce resource. But we do agree that this could also come from a latent factor not seen in the study (such as an ongoing social issue).

For the state of Indiana, Wald's statistic rejects null hypotheses whereas for the state of Georgia, it fails to reject the null hypothesis. This may not be easily understood in this example just by looking at the plots which to some extent proves usefulness of statistical tests we learned in the class.

[ ]:
```
```