Exploratory Tasks

In [94]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Loading the exploratory dataset (NY arrests data)

In [95]:

```
arrests_df = pd.read_csv('nypd_arrests_data.csv').drop(columns=['ARREST_KEY', 'PD_CD', '
PD_DESC', 'KY_CD', 'LAW_CODE', 'LAW_CAT_CD', 'ARREST_BORO', 'ARREST_PRECINCT', 'JURISDIC
TION_CODE', 'X_COORD_CD', 'Y_COORD_CD', 'Latitude', 'Longitude', 'Lon_Lat'])
arrests_df.head()
```

Out[95]:

	ARREST_DATE	OFNS_DESC	AGE_GROUP	PERP_SEX	PERP_RACE
0	1/1/20	SEX CRIMES	25-44	M	WHITE HISPANIC
1	1/1/20	ASSAULT 3 & RELATED OFFENSES	18-24	F	BLACK
2	1/1/20	POSSESSION OF STOLEN PROPERTY	<18	M	BLACK
3	1/1/20	ASSAULT 3 & RELATED OFFENSES	45-64	M	BLACK
4	1/1/20	ASSAULT 3 & RELATED OFFENSES	18-24	M	BLACK

Loading the US-all confirmed cases dataset

In [96]:

```
confirmed_df = pd.read_csv('US_confirmed.csv')
confirmed_df.head()
```

Out[96]:

	State	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	2020-01-28	2020-01-29	2020-01-30	2020-01-31	2020-02-01	2020-02-02	2020-02-03	2020-02-04	2020-02-05	2020-02-06	2020-02-0
0	AK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	AL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	AR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	AZ	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	
4	CA	0	0	0	0	2	3	3	4	4	4	4	4	6	6	6	6	

5 rows × 439 columns

Loading the US-all deaths dataset

In [97]:

```
deaths_df = pd.read_csv('US_deaths.csv')
deaths_df.head()
```

Out[97]:

	State	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	2020-01-28	2020-01-29	2020-01-30	2020-01-31	2020-02-01	2020-02-02	2020-02-03	2020-02-04	2020-02-05	2020-02-06	2020-02-0

	State	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	2020-01-28	2020-01-29	2020-01-30	2020-01-31	2020-02-01	2020-02-02	2020-02-03	2020-02-04	2020-02-05	2020-02-06	2020-02-07
0	AK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	AL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	AR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	AZ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	CA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

5 rows × 439 columns

Parsing the confirmed data to get daily number of confirmed cases

In [98]:

```
def getPerDayData_confirmed(data):
  states=['Confirmed_Count']
  for m_state in states:
    data[m_state]=data[m_state].diff().fillna(data[m_state])
  return data

ny_confirmed_count_df = pd.DataFrame()
ny_confirmed_count_df['Date'] = pd.to_datetime(confirmed_df.columns[1:].to_list())
ny_confirmed_count_df['Confirmed_Count'] = np.array(confirmed_df.iloc[34].iloc[1:])
getPerDayData_confirmed(ny_confirmed_count_df)
len(ny_confirmed_count_df)
```

Out[98]:

```
438
```

Parsing the deaths data to get daily number of deaths

In [99]:

```
def getPerDayData_deaths(data):
  states=['Deaths_Count']
  for m_state in states:
    data[m_state]=data[m_state].diff().fillna(data[m_state])
  return data

ny_deaths_count_df = pd.DataFrame()
ny_deaths_count_df['Date'] = pd.to_datetime(deaths_df.columns[1:].to_list())
ny_deaths_count_df['Deaths_Count'] = np.array(deaths_df.iloc[34].iloc[1:])
getPerDayData_deaths(ny_deaths_count_df)
len(ny_deaths_count_df)
```

Out[99]:

```
438
```

Parsing the arrests data to get daily count of all types of arrests

In [100]:

```
arrests_df['ARREST_DATE'] = pd.to_datetime(arrests_df["ARREST_DATE"])
arrests_df.sort_values(by='ARREST_DATE')
counts = arrests_df.groupby('ARREST_DATE')['OFNS_DESC'].count()
arrests_count_df = pd.DataFrame()
arrests_count_df['ARREST_DATE'] = arrests_df['ARREST_DATE'].drop_duplicates()
arrests_count_df['Counts'] = np.array(counts)
len(arrests_count_df)
```

Out[100]:

```
456
```

Parsing the arrests data to get daily count of burglary and serious crimes

In [101]:

In [101]:

```
arrests_burglary_df = arrests_df[arrests_df['OFNS_DESC'].isin(['BURGLARY'])]
arrests_serious_crimes_df = arrests_df[arrests_df['OFNS_DESC'].isin(['MURDER & NON-NEGL.
MANSLAUGHTE', 'FELONY ASSAULT', 'KIDNAPPING & RELATED OFFENSES'])]
```

In [102]:

```
counts = arrests_burglary_df.groupby('ARREST_DATE')['OFNS_DESC'].count()
arrests_burglary_count_df = pd.DataFrame()
arrests_burglary_count_df['ARREST_DATE'] = arrests_burglary_df['ARREST_DATE'].drop_duplic
ates()
arrests_burglary_count_df['Counts'] = np.array(counts)
arrests_burglary_count_df.head()
```

Out[102]:

	ARREST_DATE	Counts
19	2020-01-01	14
491	2020-01-02	15
1147	2020-01-03	18
1620	2020-01-04	23
2178	2020-01-05	5

In [103]:

```
counts = arrests_serious_crimes_df.groupby('ARREST_DATE')['OFNS_DESC'].count()
arrests_serious_crimes_count_df = pd.DataFrame()
arrests_serious_crimes_count_df['ARREST_DATE'] = arrests_serious_crimes_df['ARREST_DATE']
.drop_duplicates()
arrests_serious_crimes_count_df['Counts'] = np.array(counts)
arrests_serious_crimes_count_df
```

Out[103]:

	ARREST_DATE	Counts
10	2020-01-01	60
492	2020-01-02	60
1132	2020-01-03	38
1604	2020-01-04	27
2005	2020-01-05	30
...	...	...
177941	2021-03-27	35
178238	2021-03-28	31
178502	2021-03-29	37
178803	2021-03-30	33
179228	2021-03-31	53

456 rows × 2 columns

These functions below are used to calculate Pearson Coefficients of two datasets. We use these to check if datasets related to our X dataset is independent of datasets in US_all_confirm and US_all_death

In [104]:

```
def calculate_sample_mean(X):
    return sum(X) / len(X)
```

In [105]:

```
p_value = 0.5
def calculate_pearsons_coefficient(X, Y):
    x_mean = calculate_sample_mean(X)
    print("X mean =", x_mean)
    y_mean = calculate_sample_mean(Y)
    print("Y mean =", y_mean)
    numer = sum([(x_i - x_mean) * (y_i - y_mean) for x_i, y_i in zip(X, Y)])
    print("numerator =", numer)
    denom = (sum([(x_i - x_mean)**2 for x_i in X]) * sum([(y_i - y_mean)**2 for y_i in Y
])) ** 0.5
    print("denominator =", denom)
    return numer / denom
```

Lockdown was imposed in NY to reduce the impact of covid. Below are the details

March 7, 2020 **State of emergency declared.**

May 15, 2020 **Phase 1 of reopening allowed for counties that met qualifications. Five counties met qualifications and began reopening on this date.** **May 15, 2020** **Drive-in theaters, landscaping/gardening businesses allowed to reopen state-wide (regardless of Phase 1 qualifications).**

source: https://en.wikipedia.org/wiki/COVID-19_pandemic_in_New_York_(state)#Government_response

We obtain count of confirmed cases during the lockdown period as below.

In [106]:

```
mask_1 = (ny_confirmed_count_df['Date'] >= '2020-3-7') & (ny_confirmed_count_df['Date']
< '2020-05-15')

ny_confirmed_count_df_lockdown = ny_confirmed_count_df.loc[mask_1]
ny_confirmed_count_df_lockdown.head()
```

Out[106]:

	Date	Confirmed_Count
45	2020-03-07	52
46	2020-03-08	30
47	2020-03-09	38
48	2020-03-10	28
49	2020-03-11	34

We obtain count of death cases during the lockdown period as below.

In [107]:

```
mask_1 = (ny_deaths_count_df['Date'] >= '2020-3-7') & (ny_deaths_count_df['Date'] < '202
0-05-15')

ny_deaths_count_df_lockdown = ny_deaths_count_df.loc[mask_1]
ny_deaths_count_df_lockdown.head()
```

Out[107]:

	Date	Deaths_Count
45	2020-03-07	0
46	2020-03-08	0
47	2020-03-09	0
48	2020-03-10	0
49	2020-03-11	0

We obtain count of arrests during the lockdown period as below

We obtain count of arrests during the lockdown period as below.

In [108]:

```
mask_1 = (arrests_count_df['ARREST_DATE'] >= '2020-3-7') & (arrests_count_df['ARREST_DAT
E'] < '2020-05-15')

arrests_count_df_lockdown = arrests_count_df.loc[mask_1]
arrests_count_df_lockdown.head()
print(len(arrests_count_df_lockdown))
```

```
69
```

Inference 1
Relation of #arrests with #confimed cases during stay-at-home order
Relation of # burglary arrests with #confimed cases during stay-at-home order

We check if arrests during lockdown are correlated with confirmed cases during lockdown. We found a negative correlation between #arrests and #confirmed cases.

This implicates that as # of covid cases increased, the arrests numbers decreased. This sounds just.

In [109]:

```
corr_arrest_cases_lockdown = calculate_pearsons_coefficient(arrests_count_df_lockdown.loc
[:,'Counts'].values,ny_confirmed_count_df_lockdown.loc[:,'Confirmed_Count'].values)
print(corr_arrest_cases_lockdown)
print("Correlation" if (np.abs(corr_arrest_cases_lockdown)>p_value) else "No Correlation"
)
```

```
X mean = 344.84057971014494
Y mean = 4972.391304347826
numerator = -21907247.695652176
denominator = 31456636.806513343
-0.6964268885577783
Correlation
```

We obtain count of arrests pertaining to burglary during the lockdown period as below.

In [110]:

```
mask_1 = (arrests_burglary_count_df['ARREST_DATE'] >= '2020-3-7') & (arrests_burglary_co
unt_df['ARREST_DATE'] < '2020-05-15')

arrests_burglary_count_df_lockdown = arrests_burglary_count_df.loc[mask_1]
arrests_burglary_count_df_lockdown.head()
print(len(arrests_burglary_count_df_lockdown))
```

```
69
```

We check if arrests pertaining to burglary during lockdown are correlated with confirmed cases during lockdown. We found a low negative correlation between #arrests and #confirmed cases.

This implicates that as # of covid cases increased, the arrests numbers decreased. This sounds just. But the correlation coefficient is "<0.5" and thus we reject the dependence of these two sets.

In [111]:

```
corr_arrest_burg_cases_lockdown = calculate_pearsons_coefficient(arrests_burglary_count_d
f_lockdown.loc[:,'Counts'].values,ny_confirmed_count_df_lockdown.loc[:,'Confirmed_Count']
.values)
print(corr_arrest_burg_cases_lockdown)
print("Correlation" if (np.abs(corr_arrest_burg_cases_lockdown)>p_value) else "No Correla
```

```
tion")
```

```
X mean = 17.434782608695652
Y mean = 4972.391304347826
numerator = -461700.7391304347
denominator = 2293099.951807953
-0.2013434864740262
No Correlation
```

We obtain count of arrests pertaining to Serious Crimes like Murder, kidnapping, etc. during the lockdown period as below.

In [112]:

```
mask_1 = (arrests_serious_crimes_count_df['ARREST_DATE'] >= '2020-3-7') & (arrests_serio
us_crimes_count_df['ARREST_DATE'] < '2020-05-15')

arrests_serious_crimes_count_df_lockdown = arrests_serious_crimes_count_df.loc[mask_1]
arrests_serious_crimes_count_df_lockdown.head()
print(len(arrests_serious_crimes_count_df_lockdown))
```

```
69
```

Inference 2
Serious crimes during lockdown vs confirmed cases and deaths during lockdown

We check if arrests pertaining to serious crimes during lockdown are correlated with confirmed cases during lockdown. We found a significant negative correlation between #arrests and #confirmed cases.

This implicates that as # of covid cases increased, the arrests numbers for serious crimes decreased. This sounds just.

In [113]:

```
corr_arrest_serious_crimes_cases_lockdown = calculate_pearsons_coefficient(arrests_seriou
s_crimes_count_df_lockdown.loc[:,'Counts'].values,ny_confirmed_count_df_lockdown.loc[:,'C
onfirmed_Count'].values)
print("Correlation" if (np.abs(corr_arrest_serious_crimes_cases_lockdown)>p_value) else
"No Correlation")
print(corr_arrest_serious_crimes_cases_lockdown)
```

```
X mean = 35.43478260869565
Y mean = 4972.391304347826
numerator = -1849140.739130435
denominator = 2907190.2642081245
Correlation
-0.6360576952585913
```

Serious crimes during lockdown vs deaths during lockdowns We check if arrests pertaining to serious crimes during lockdown are correlated with death cases during lockdown. We found a fair negative correlation between #arrests and #death cases.

This implicates that as # of covid deaths increased, the arrests for serious crimes numbers decreased. This sounds just. But the correlation coefficient is "<0.5" and thus we reject the dependence of these two sets.

In [114]:

```
corr_arrest_serious_crimes_deaths_lockdown = calculate_pearsons_coefficient(arrests_serio
us_crimes_count_df_lockdown.loc[:,'Counts'].values,ny_deaths_count_df_lockdown.loc[:,'Dea
ths_Count'].values)
print("Correlation" if (np.abs(corr_arrest_serious_crimes_deaths_lockdown)>p_value) else
"No Correlation")
```

```
print(corr_arrest_serious_crimes_deaths_lockdown)
```

```
X mean = 35.43478260869565
Y mean = 400.84057971014494
numerator = -134672.21739130438
denominator = 304067.00273626903
No Correlation
-0.44290309760481195
```

Below functions are defined to do 2 population KS test on #death in NY and #arrest in NY. These functions are also used in Question 2.c. in Main.ipynb

In [115]:

```
def get_ecdf(X):
    c = [None] * len(X)
    for i in reversed(range(len(X))):
        if i < len(X) - 1 and X[i] == X[i + 1]:
            c[i] = c[i + 1]
        else:
            c[i] = (i + 1) / len(X)

    return c
```

A threshold of 0.05 is chosen as discussed in class.

In [116]:

```
threshold = 0.05
```

In [117]:

```
def get_ecdf_val_minus(X, eCDF, x):
    for i in range(len(X)):
        if X[i] >= x:
            if i == 0:
                return 0
            else:
                return eCDF[i - 1]
    return 1


def get_ecdf_val_plus(X, eCDF, x):
    for i in range(len(X)):
        if X[i] == x:
            return eCDF[i]
        elif X[i] > x:
            if i == 0:
                return 0
            else:
                return eCDF[i - 1]
    return 1
```

In [118]:

```
def perform_2_pop_KS(X, Y, X_label, Y_label):
    print("2 population KS test for", X_label, "and", Y_label)

    X.sort()
    Y.sort()

    X_len = len(X)
    Y_len = len(Y)

    if X_len > Y_len:
        t = Y
        Y = X
        X = t
        X_len = len(X)
```

```
        Y_len = len(Y)
        t = X_label
        X_label = Y_label
        Y_label = t

    print("X len", X_len)
    print("Y len", Y_len)

    X_eCDF = get_ecdf(X)
    Y_eCDF = get_ecdf(Y)
    print("X_eCDF len", len(X_eCDF))
    print("Y_eCDF len", len(Y_eCDF))

    d = -1
    max_diff_idx = 0
    max_diff_vals = [None, None]
    for i in range(X_len):
        x_plus_y_plus_diff = abs(get_ecdf_val_plus(X, X_eCDF, X[i]) - get_ecdf_val_plus(
Y, Y_eCDF, X[i]))
        x_minus_y_minus_diff = abs(get_ecdf_val_minus(X, X_eCDF, X[i]) - get_ecdf_val_mi
nus(Y, Y_eCDF, X[i]))
        if d < x_plus_y_plus_diff:
            max_diff_idx = i
            max_diff_vals[0] = get_ecdf_val_plus(X, X_eCDF, X[i])
            max_diff_vals[1] = get_ecdf_val_plus(Y, Y_eCDF, X[i])
            d = x_plus_y_plus_diff
        if d < x_minus_y_minus_diff:
            max_diff_idx = i
            max_diff_vals[0] = get_ecdf_val_minus(X, X_eCDF, X[i])
            max_diff_vals[1] = get_ecdf_val_minus(Y, Y_eCDF, X[i])
            d = x_minus_y_minus_diff

    print("KS statistic =", d)

    print("Max value at x =", X[max_diff_idx], "with values", max_diff_vals)

    if d >= threshold:
        print("We Reject the Null Hypothesis:", X_label, "and", Y_label, "does NOT have
the same distribution")
    else:
        print("We Accept the Null Hypothesis:", X_label, "and", Y_label, "have the same
distribution")

    plt.xlabel("x")
    plt.ylabel('eCDF')

    X_len = len(X)
    Y_len = len(Y)

    plt.step(X, X_eCDF, label=X_label)
    plt.step(Y, Y_eCDF, label=Y_label)
    plt.scatter([X[max_diff_idx], X[max_diff_idx]], max_diff_vals, color='red', marker='
x', label='max difference')
    plt.legend()
    plt.show()
```

We extract two dataset from arrest data. One dataset contains #of arrests when death was 0 and another contains #of arrests when death was not 0.

In [119]:

```
mask_no_deaths = (ny_deaths_count_df['Deaths_Count'] == 0)

ny_no_deaths_count_df = ny_deaths_count_df.loc[mask_no_deaths]
ny_no_deaths_count_df.head()
```

Out[119]:

	Date	Deaths_Count
0	**2020-01-22**	**0**

	Date	Deaths_Count
1	2020-01-23	0
2	2020-01-24	0
3	2020-01-25	0
4	2020-01-26	0

In [120]:

```
mask_yes_deaths = (ny_deaths_count_df['Deaths_Count'] > 0)

ny_yes_deaths_count_df = ny_deaths_count_df.loc[mask_yes_deaths]
ny_yes_deaths_count_df.head()
```

Out[120]:

	Date	Deaths_Count
52	2020-03-14	3
54	2020-03-16	11
55	2020-03-17	8
56	2020-03-18	8
57	2020-03-19	20

In [121]:

```
arrests_count_df['Date'] = arrests_count_df['ARREST_DATE']
arrest_count_no_death = pd.merge(arrests_count_df, ny_no_deaths_count_df, on="Date", how
="inner")
arrest_count_no_death.head()
```

Out[121]:

	ARREST_DATE	Counts	Date	Deaths_Count
0	2020-01-22	671	2020-01-22	0
1	2020-01-23	634	2020-01-23	0
2	2020-01-24	525	2020-01-24	0
3	2020-01-25	389	2020-01-25	0
4	2020-01-26	374	2020-01-26	0

In [122]:

```
arrest_count_yes_death = pd.merge(arrests_count_df, ny_yes_deaths_count_df, on="Date", h
ow="inner")
arrest_count_yes_death.head()
```

Out[122]:

	ARREST_DATE	Counts	Date	Deaths_Count
0	2020-03-14	491	2020-03-14	3
1	2020-03-16	327	2020-03-16	11
2	2020-03-17	367	2020-03-17	8
3	2020-03-18	482	2020-03-18	8
4	2020-03-19	414	2020-03-19	20

Inference 3

Whether death/no death matter to number of arrests

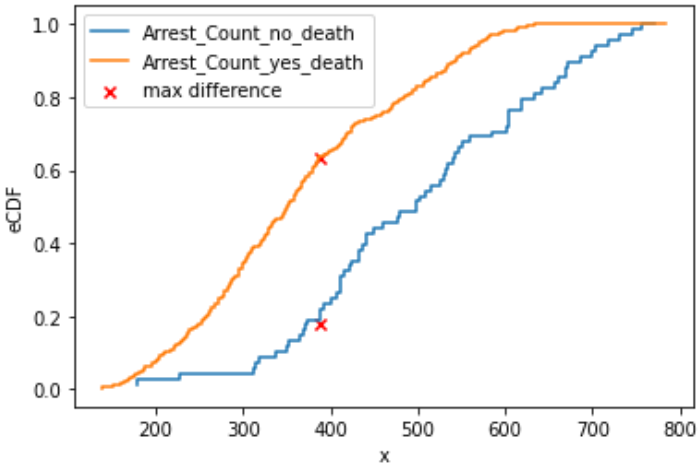
We perform 2 population KS test to check if the distrbution of number of arrests on zero death day and number

arrests on non zero death. The KS test rejects this.
This implicates that zero covid deaths or non-zero covid deaths does not effect the #arrests.

In [123]:

```
perform_2_pop_KS(arrest_count_no_death.loc[:,'Counts'].values,
                 arrest_count_yes_death.loc[:,'Counts'].values,
                 "Arrest_Count_no_death", "Arrest_Count_yes_death")
print("\n")
```

```
2 population KS test for Arrest_Count_no_death and Arrest_Count_yes_death
X len 68
Y len 363
X_eCDF len 68
Y_eCDF len 363
KS statistic = 0.454383406255064
Max value at x = 387 with values [0.17647058823529413, 0.6308539944903582]
We Reject the Null Hypothesis: Arrest_Count_no_death and Arrest_Count_yes_death does NOT
have the same distribution
```



The End

In [ ]: