UCDPA – John Lenehan

GitHub URL

https://github.com/jlenehan/UCDPA_JohnLenehan

Abstract

An analysis is conducted on eviction data in New York city from 2017 to 2020, compared to income data during that same period and population data from the 2020 census. This analysis shows that the Bronx is the most heavily affected by eviction trends, at a prevalence per capita over twice that of the next two boroughs. A correlation analysis indicates that eviction rates during this period are influenced strongly by average household income by borough, and that tackling eviction trends across the city would require income assistance measures.

Introduction

For this project, the use case is defined as a non-profit in New York city providing support for eviction cases across the municipality. Such a non-profit would naturally need to gather data to determine where best to concentrate their efforts in the city; i.e. where the most evictions are per capita and what drives these trends. The project here is therefore set up as an analysis that such a non-profit might perform.

Dataset

The dataset used for this analysis is a merging of data from three sources; a dataset of evictions in New York city from 2017 to 2022 [1], a dataset of household income in New York city from 2013 to 2020 [2], and census data of county population across New York state [3].

Implementation Process

Step 0: Import Libraries

To begin, the necessary libraries for analysis must be imported; these are laid out below:

- Import os
- Import pandas as pd
- From datetime import date
- Import matplotlib.pyplot as plt
- Import seaborn as sns

Step 1: Import Data

Step 1 is to import the data; the eviction and income data are imported from the web using the pd.read_json() method. The data for each can be found at the below addresses:

Data name	Data Source (URL)	JSON link
Eviction Data	https://data.cityofnewyork.us/City-Government/Evictions/6z8x-wfk4 [1]	https://data.cityofnewyork.us/resource/6z8x-wfk4.json?$limit=100000
Income Data	https://datausa.io/profile/geo/new-york-ny#economy [2]	Data downloaded and imported as a csv file
Population Data	https://data.ny.gov/Government-Finance/Annual-Population-Estimates-for-New-York-State-and/krt9-ym2k [3]	https://data.ny.gov/resource/krt9-ym2k.json?$limit=100000

Note that the query "?$limit=100000" is added to the json strings of both datasets, to increase the json row limit from 1,000 to 100,000.

```
##EVICTIONS DATA
##JSON METHOD
#Set limit to 100,000 to capture all entries in json file
NY_Evictions_gross = pd.read_json(r'https://data.cityofnewyork.us/resource/6z8x-wfk4.json?$limit=100000')

##POPULATION DATA
##JSON METHOD
NY_Pop_gross=pd.read_json(r'https://data.ny.gov/resource/krt9-ym2k.json?$limit=100000')
```

For the income data, the data is downloaded and stored as a csv file (included in the zip file for this assignment) and the data is read into the jupyter notebook using the .read_csv() method. This method is used as a review of the data showed it hadn't been updated since 2020, therefore the data is already static and wouldn't benefit from pulling the data live. The directory of the notebook is set to the assignment folder using the os.chdir() function:

```
#Setting directory

os.chdir(r'C:\Users\jlenehan\OneDrive\Documents\Intro to Data Analytics\UCDPA_JohnLenehan')
```

From there the file can be read into the read_csv() function locally:

```
##INCOME DATA
##CSV METHOD
#income-location data taken from datausa.io
NY_Income_gross = pd.read_csv('Income by Location.csv')
```

Step 2: Describe Data

For each dataset, the methods .columns, .info, .describe, and .shape are used to give an idea of what each dataset looks like.

```
#Describe Evictions dataset
print(NY_Evictions_gross.columns)
print(NY_Evictions_gross.info())
print(NY_Evictions_gross.describe())
print(NY_Evictions_gross.shape)
```

Additionally the unique values of each dataset column is printed using a for loop, along with a count of unique values for each column:

```
print('\nNY Evictions Data - Unique Values:')
for x in NY_Evictions_gross.columns:
    print(x+':')
    print(NY_Evictions_gross[x].unique())
    print(str(NY_Evictions_gross[x].nunique()) + str(' unique values'))
```

(i) Eviction Data

The eviction data has a number of columns which could be useful for analysis; the marshal first and last names are given, along with latitude and longitude data of each eviction, and the council district in which the eviction was carried out. The data shown is for the past 6 years (2017 to 2022), across all 5 boroughs of New York city. For this analysis the scope is limited to the number of evictions by borough and year – for these columns the .info() function shows no null values, so it won't be necessary to drop null values from this dataset. The year data is captured in the eviction_date column but not in a form useful for analysis, so this needs to be extracted.

(ii) Income Data

Income data is given here for 8 years, from 2013 to 2020, and is also for all 5 boroughs of New York city. Similar to the eviction data, the income data has no null values at all across any of its columns. There is a column for race, but this only has 1 unique value – therefore the column should be filtered out as it has no use. Geography data is also given, but in longform instead of by county – this needs to be separated.

(iii) Population Data

The census population data contains no null values, and contains population data for all New York counties from 1970 to 2021. Some filtering is necessary to get only the most recent census data for the 5 counties (boroughs) under the New York municipality, and an aggregating function is

necessary to get the income data in a useful format, but otherwise there isn't as much cleaning required as the other datasets.

Step 3: Clean and Manipulate Data

(i) Eviction Data

From looking at the eviction data, the executed_date column is stored as an object datatype which makes datetime functions difficult – as such this needs to be converted to a datetime datatype. This is done using the pandas to_datetime() method. From there the .year method is used on the DatetimeIndex() function to extract the year of each eviction.

```
#Adding year info
NY_Evictions_gross['executed_date']=pd.to_datetime(NY_Evictions_gross['executed_date'])
NY_Evictions_gross['year']=pd.DatetimeIndex(NY_Evictions_gross['executed_date']).year
```

Following this a count of evictions via the eviction_zip column is executed, producing a pivot table grouped by year and borough using groupby(). For further analysis this is then ungrouped using reset_index(), and the column is renamed from eviction_zip to sum_evictions. This gives the final evictions dataset of evictions by year and borough.

```
#grouping evictions by year and borough
NY_Evictions = NY_Evictions_gross.groupby(['year','borough'], sort = [True,True])['eviction_zip'].count()

#resetting index on pivot table
NY_Evictions = NY_Evictions.reset_index()

#renaming column to sum_evictions
NY_Evictions.rename(columns = {'eviction_zip':'sum_evictions'}, inplace=True)
```

(ii) Income Data

The original income data has a geography column giving the census tract, county and state the data was collected from; this is separated out using the str.split() on the ',' delimiter into 3 columns. From there the new data is merged with the original data using .merge() and an inner join, with the indices used as the merge key (left_index = True, right_index = True).

```
#if statement to avoid duplicates on reruns
if 'Borough' not in NY_Income_gross.columns:

    #splitting geography column by ',' delimiter to get county column
    NY_Income_geodata = NY_Income_gross.Geography.str.split(',',expand=True)

    #merging new columns to income table by index
    NY_Income_gross = NY_Income_gross.merge(NY_Income_geodata,
                                            left_index=True,
                                            right_index=True,
                                            how='inner')
```

Columns are renamed using .rename, and the values in the Borough column are replaced to correspond with the terms for those boroughs given in the evictions dataset (i.e. Bronx, Brooklyn, Manhatten, Queens, Staten Island).

```
#Renaming columns
NY_Income_gross.rename(columns = {'Household Income by Race':'Household Income',
                                  'Household Income by Race Moe':'Household Income MOE',
                                  0:'Census Tract',1:'Borough',2:'State'},
                       inplace=True)

#Adjusting borough data to match evictions borough data format
NY_Income_gross['Borough']=NY_Income_gross['Borough'].replace([' Bronx County',
                                                               ' Kings County',
                                                               ' New York County',
                                                               ' Queens County',
                                                               ' Richmond County'],
                                                              ['BRONX','BROOKLYN','MANHATTAN','QUEENS','STATEN ISLAND'])
```

Finally, similar to the process used with the eviction data, the mean household income and household income margin of error (MOE) by year and borough is displayed using groupby(). The values of mean household income and household income MOE are rounded to 2 decimal places using .round(), and the index on the table is reset using .reset_index():

```
#filtering relevant columns for analysis
NY_Income_gross = NY_Income_gross[['Borough','Year','Household Income','Household Income MOE']]

#getting mean household income and margin of error by year and borough
NY_Income = NY_Income_gross.groupby(['Year','Borough'],
                                    sort = [True,True])['Household Income','Household Income MOE'].mean()

#rounding values to 2 decimal places
NY_Income = NY_Income.round(decimals=2)

#resetting index
NY_Income = NY_Income.reset_index()
```

(iii) Population Data

With the population data, the dataset gives 3 different population figures under the program_type column, 2 of which are population estimates:

```
program_type:
['Postcensal Population Estimate' 'Census Base Population'
 'Intercensal Population Estimate']
```

For this analysis and for convenience only the “Census Base Population” entries from 2020 are used; to filter for this the following line of code is run:

```
#filtering data to only see census data from 2020, and ignore population estimates
NY_Pop_gross = NY_Pop_gross[(NY_Pop_gross['program_type']=='Census Base Population') & (NY_Pop_gross['year']==2020)]
```

The original dataset showed census figures for all counties in New York state – however as this analysis is only interested in the counties for New York city, this is filtered down to those 5 counties (or boroughs) using .loc() and .isin(). Subsequently the names of the boroughs are replaced to match the values in the eviction dataset using .replace(), and the name of the column is changed from ‘geography’ to ‘borough’ using .rename():

```
#filtering to 5 boroughs (counties) of New York city
NY_Pop_gross = NY_Pop_gross.loc[NY_Pop_gross['geography'].isin(['Bronx County',
                                                                'Kings County',
                                                                'New York County',
                                                                'Queens County',
                                                                'Richmond County'])]

#Adjusting borough data to match evictions borough data format
NY_Pop_gross['geography']=NY_Pop_gross['geography'].replace(['Bronx County',
                                                             'Kings County',
                                                             'New York County',
                                                             'Queens County',
                                                             'Richmond County'],
                                ['BRONX','BROOKLYN','MANHATTAN','QUEENS','STATEN ISLAND'])

#renaming geography column to borough
NY_Pop_gross.rename(columns = {'geography':'borough'}, inplace=True)
```

Finally the dataset is filtered down to borough, year and population data, and the borough is set as the dataframe index:

```
NY_Pop = NY_Pop_gross[['borough','year','population']]

NY_Pop.set_index('borough',inplace=True)
```

For subsequent analysis, the 2020 census data for these boroughs will be assumed as constant across all years under review.

Step 4: Merge Data

With all individual dataframes now cleaned and aggregated to show the desired data, these data must now be merged to produce the final amalgamated dataframe for analysis. Both joins used for this process are left joins, as the primary goal for this analysis is evictions; any data from the income or population datasets that doesn't correspond with the can be dropped using the dropna() function.

Firstly the income dataframe is merged to the population data using a left join, on the borough column; this has the effect of attaching the 2020 population data to all rows in the income dataframe, irrespective of year:

```
NY_Income_Pop = NY_Income.merge(NY_Pop['population'], how = 'left',
                                left_on = 'Borough', right_on = 'borough')
```

Next the evictions dataframe is merged to the previously merged income-population dataframe, also via a left join. However for this join it's necessary to join on both the year and borough columns for each dataframe, to ensure income data matches the year in the evictions table as well as the borough:

```
NY_Merged = NY_Evictions.merge(NY_Income_Pop, how = 'left',
                               left_on = ['year','borough'],
                               right_on = ['Year','Borough'])
```

This produces a dataframe of evictions across the 5 boroughs from 2017 to 2020, with income data accurate to each year and population data by borough from the 2020 census, assumed to be constant for this analysis. Some further cleaning of the new "NY_Merged" dataframe is required;

rows with null values are dropped using dropna(), redundant columns from the joins are filtered out, columns are renamed, and the year column is changed from a number entry to a string (a string entry is preferrable for the plotting step):

```
#dropping missing values
NY_Merged.dropna(inplace=True)

#filtering out redundant columns
NY_Merged = NY_Merged[['year','borough','sum_evictions','Household Income','Household Income MOE','population']]

#Renaming columns
NY_Merged.rename(columns = {'year':'Year',
                            'borough':'Borough',
                            'sum_evictions':'Sum Evictions',
                            'population':'Population'},
                 inplace=True)

#Changing Year column to string
NY_Merged['Year']=NY_Merged['Year'].apply(str)
```

Subsequently the evictions data can be normalized to the population data by dividing the ‘Sum Evictions’ column by the ‘Population’ column. Furthermore the upper and lower margins of the household income can be calculated by either adding or subtracting the ‘Household Income MOE’ column from the ‘Household Income’ column.

```
#Evictions per 1,000 people
NY_Merged['Evictions per 1,000'] = 1000*(NY_Merged['Sum Evictions']/NY_Merged['Population'])

#Upper margin of Household Income
NY_Merged['Household Income - Upper Margin'] = NY_Merged['Household Income']+NY_Merged['Household Income MOE']

#Lower margin of Household Income
NY_Merged['Household Income - Lower Margin'] = NY_Merged['Household Income']-NY_Merged['Household Income MOE']
```

Lastly the correlation function .corr() is run to show how each column correlates against the others:

	Sum Evictions	Household Income	Household Income MOE	Population	Evictions per 1,000	Income per 1,000	Household Income UCL	Household Income LCL	Household Income - Upper Margin	Household Income - Lower Margin
Sum Evictions	1.000000	-0.620283	-0.708677	0.441593	0.864418	-0.635183	-0.645123	-0.574023	-0.645123	-0.574023
Household Income	-0.620283	1.000000	0.926916	-0.155193	-0.648283	0.411063	0.997217	0.993048	0.997217	0.993048
Household Income MOE	-0.708677	0.926916	1.000000	-0.142458	-0.733238	0.354091	0.952313	0.876298	0.952313	0.876298
Population	0.441593	-0.155193	-0.142458	1.000000	-0.001535	-0.858652	-0.154484	-0.154551	-0.154484	-0.154551
Evictions per 1,000	0.864418	-0.648283	-0.733238	-0.001535	1.000000	-0.341622	-0.672769	-0.602266	-0.672769	-0.602266
Income per 1,000	-0.635183	0.411063	0.354091	-0.858652	-0.341622	1.000000	0.404569	0.416652	0.404569	0.416652
Household Income UCL	-0.645123	0.997217	0.952313	-0.154484	-0.672769	0.404569	1.000000	0.981508	1.000000	0.981508
Household Income LCL	-0.574023	0.993048	0.876298	-0.154551	-0.602266	0.416652	0.981508	1.000000	0.981508	1.000000
Household Income - Upper Margin	-0.645123	0.997217	0.952313	-0.154484	-0.672769	0.404569	1.000000	0.981508	1.000000	0.981508
Household Income - Lower Margin	-0.574023	0.993048	0.876298	-0.154551	-0.602266	0.416652	0.981508	1.000000	0.981508	1.000000

Step 5: Plot Data

Seaborn lineplots are first deployed showing the evictions and evictions per 1,000, with hue set to borough to show the distinction between boroughs. This is done using the .lineplot() function:

```
#Get evictions by borough, year
sns.lineplot(x=NY_Merged['Year'],y=NY_Merged['Sum Evictions'],hue=NY_Merged['Borough'])
plt.show()

#Show evictions per 1000 by borough, year
sns.lineplot(x=NY_Merged['Year'],y=NY_Merged['Evictions per 1,000'],hue=NY_Merged['Borough'])
plt.show()
```

Next a scatter plot is produced showing the evictions per 1,000 against household income, grouped by borough:

```
#Show evictions per 1,000 against household income
sns.relplot(x='Evictions per 1,000',
            y='Household Income',
            hue='Borough',
            data=NY_Merged)

plt.show()
```

A custom function is defined to produce a scatter plot with a trendline using lnplot(), along with a correlation of the variables using .corr(); this function is named plot_correlation():

```
#define function for getting scatter plot of 2 variables and printing correlation coefficient
def plot_correlation(df,var1,var2,group=None):
    sns.lmplot(x=var1,y=var2,hue=group,data=df,ci=0)
    plt.show()

    print(df[[var1,var2]].corr())

    return
```

Lastly this new function is used to show the correlation of evictions per 1,000 against 2 income variables in the dataframe; household income, and household income MOE:

```
#get scatter of evictions per 1,000 vs mean household income
plot_correlation(NY_Merged,'Evictions per 1,000','Household Income')

#get scatter of evictions per 1,000 vs mean household income MOE
plot_correlation(NY_Merged,'Evictions per 1,000','Household Income MOE')
```

Results

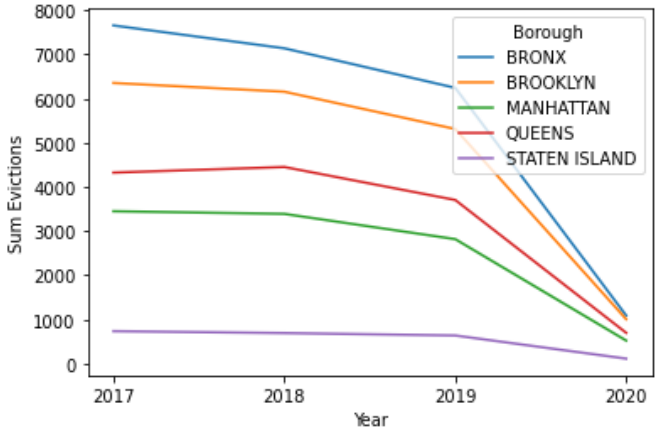


Figure 1: Graph showing total eviction trends by borough from 2017 to 2020

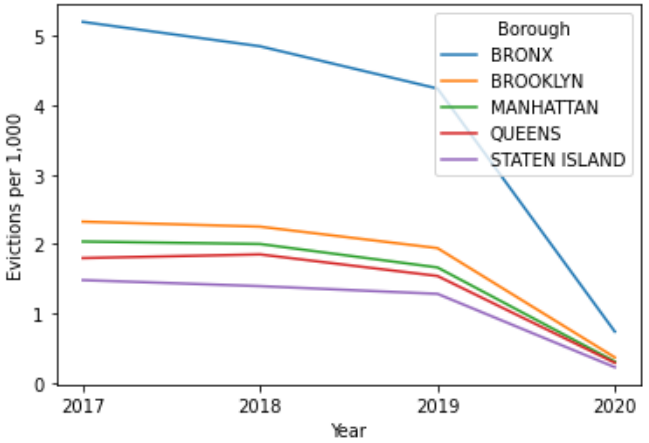


Figure 2: Graph showing eviction trends per 1,000 population by borough from 2017 to 2020.

	Year	Borough	Evictions per 1,000
0	2017	BRONX	5.200135
1	2017	BROOKLYN	2.322671
2	2017	MANHATTAN	2.036298
3	2017	QUEENS	1.797990
4	2017	STATEN ISLAND	1.480594
5	2018	BRONX	4.848389
6	2018	BROOKLYN	2.250305
7	2018	MANHATTAN	2.000884
8	2018	QUEENS	1.850786
9	2018	STATEN ISLAND	1.393856
10	2019	BRONX	4.239964
11	2019	BROOKLYN	1.941468
12	2019	MANHATTAN	1.663272
13	2019	QUEENS	1.540243
14	2019	STATEN ISLAND	1.282912
15	2020	BRONX	0.738802
16	2020	BROOKLYN	0.367315
17	2020	MANHATTAN	0.307511
18	2020	QUEENS	0.289341
19	2020	STATEN ISLAND	0.225922

Figure 3: Table of Evictions per 1,000 by borough and year, subsetted from the NY_Merged dataframe.

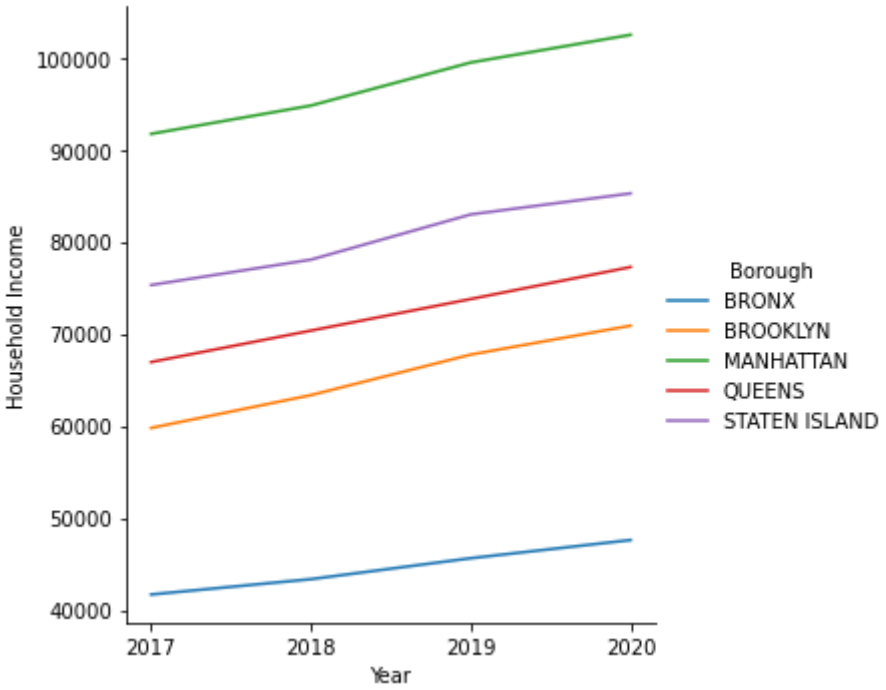


Figure 4: Line plot of household income by year, grouped by borough.

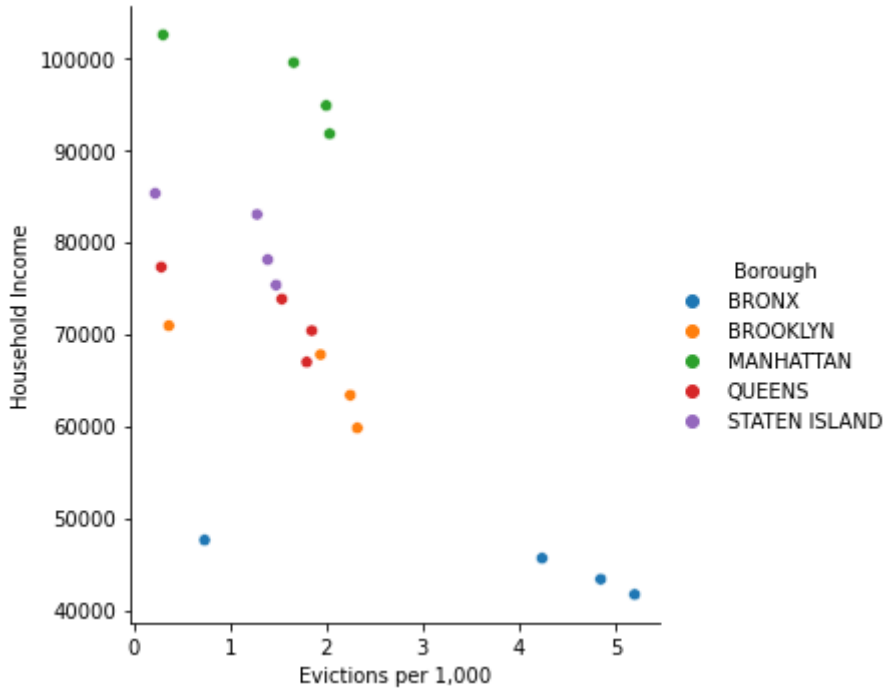
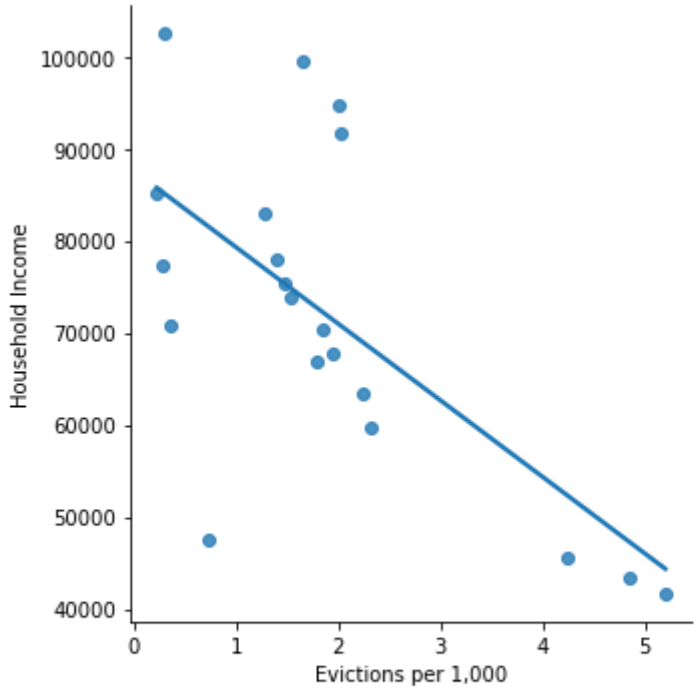
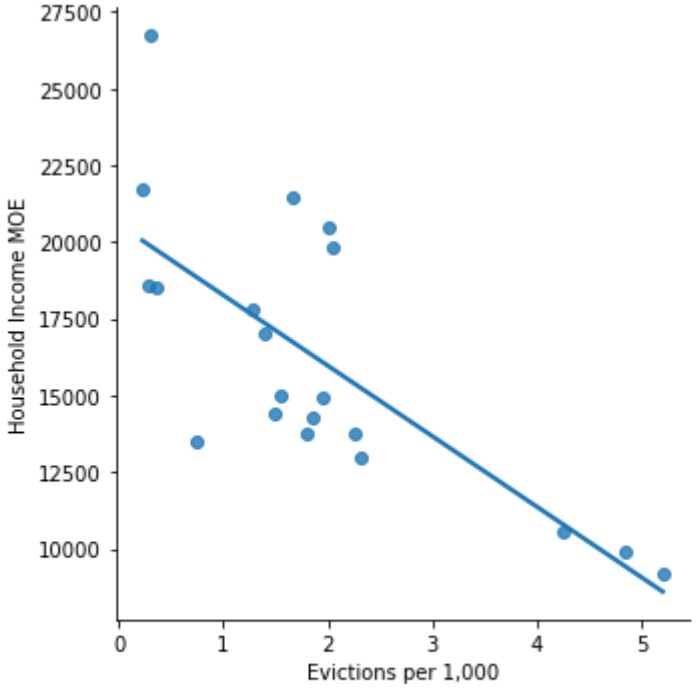


Figure 5: Scatter plot of household income against evictions per 1,000, grouped by borough.



	Evictions per 1,000	Household Income
Evictions per 1,000	1.000000	-0.648283
Household Income	-0.648283	1.000000

Figure 6: Correlation analysis of evictions per 1,000 against mean household income.



	Evictions per 1,000	Household Income MOE
Evictions per 1,000	1.000000	-0.733238
Household Income MOE	-0.733238	1.000000

Figure 7: Correlation analysis of evictions per 1,000 against mean household income margin of error (standard deviation).

Insights

- The scatter plot in figure 1 shows the Bronx has the highest eviction trends in the city, followed by Brooklyn and Queens. Adjusting for population in figure 2 the discrepancy is more stark – in 2017 the Bronx had 5.2 evictions per 1,000 people, compared to 2.3 and 2.04 for Brooklyn and Queens respectively. This indicates that the Bronx is most heavily affected by evictions for the city overall, and therefore this is where the most effort should be put by a hypothetical non-profit.
- Further to this, the Bronx has more evictions per capita than the next 2 boroughs combined, as indicated by the table in figure 3, for 2017 through 2019. Note that the dropoff in evictions for 2020 can be attributed to the Tenant Safe Harbor Act, passed during the pandemic to protect tenants from eviction [4]. From this, it's reasonable to assume that twice as much funding and resources should be spent in the Bronx compared to Brooklyn or Queens.
- The line plot shown in figure 4 of household income by year shows that the Bronx is the least affluent area of New York city; while the mean household income has improved over this period, it's still much lower than the other boroughs. This would suggest the key issue is low income leading to high eviction rate.
- The scatter plot of household income against evictions per 1,000 in figure 5 shows the Bronx has the lowest mean household income along with the highest evictions per capita across the city. This further supports the argument that the eviction problem is income driven.
- A correlation analysis of household income and household income MOE against evictions per 1,000 in figures 6 and 7 also supports this argument; both are negatively correlated, with a -0.648 and -0.733 correlation respectively. This indicates that income support is needed to tackle eviction rates across the city.

References

[1] (DOI), D.of I. (2022) *Evictions: NYC open data*, *Evictions | NYC Open Data*. Available at: https://data.cityofnewyork.us/City-Government/Evictions/6z8x-wfk4 (Accessed: October 24, 2022).

[2] *New York, NY* (no date) *Data USA*. Available at: https://datausa.io/profile/geo/new-york-ny#economy (Accessed: October 24, 2022).

[3] New York State Department of Labor (2022) *Annual population estimates for New York State and counties: Beginning 1970: State of New York, Annual Population Estimates for New York State and Counties: Beginning 1970 | State of New York*. Available at: https://data.ny.gov/Government-Finance/Annual-Population-Estimates-for-New-York-State-and/krt9-ym2k (Accessed: October 24, 2022).

[4] *Covid-19 eviction protections for tenants* (no date) *Homes and Community Renewal*. Available at: https://hcr.ny.gov/covid-19-eviction-protections-tenants#:~:text=The%20Tenant%20Safe%20Harbor%20Act,hardship%20due%20to%20COVID%2D19. (Accessed: October 24, 2022).