An Analysis of Toronto Neighbourhood Crime Rates

Author: Bryan Lau

Neighbourhood crime rates vs income in Toronto

Introduction

A link between crime and social circumstances are often highlighted in the media, whether in the news, TV shows or movies, or other reports. But using Python and publically available datasets anyone can explore the facts for themselves and discover any relationships that exist. This is just such an exploration that I did a little while ago for the city of Toronto.

I sourced data from the City of Toronto's Open Data Portal, which is part of the city's Open Data Master Plan to release datasets that help solve civic issues and improve data accessibility for public benefit.

Methodology

Toronto has 140 distinctly named neighbourhoods which were also present in the datasets, so this seemed like a logical sequence of steps:

1. Download and process the neighbourhood crime and income data.
2. Link the two data sets by neighbourhood.
3. Visualize and analyze the data.

By linking the datasets I could determine whether there was any correlation between the two.

Libraries Used

Most of the libraries I used for this are standard fare for Python data analysis exercises:

Folium
For plotting the interactive map.

GeoPandas
For plotting the neighbourhood boundaries on the map of Toronto.

Matplotlib
Used for plotting graphs and charts.

Plotly
Used for rendering data tables.

In [ ]:
```
import requests
import json
import pandas as pd
import re
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.io as pio
import geopandas as gpd
import folium
from shapely.geometry import shape
```

About The Data

Source

As I mentioned previously, I sourced the datasets from the City of Toronto's Open Data Portal, specifically the following datasets:

- **Neighbourhood Profiles**, produced by *Social Development Finance & Administration, Research and Information Management Unit*
- **Neighbourhood Crime Rates**, published by *Toronto Police Services*

License

The data license used on the Open Data Portal is derived from the Open Government License, modified to refer to the City of Toronto (instead of the UK). My obligations were that I acknowledge the source of the information and provide a link to the license.

- Open Data License - Toronto

Alternatives

I did consider using data from Statistics Canada which is a national agency, but I wanted to use data at a neighbourhood level and Statistics Canada only seems to release that to the public for an annual subscription fee of $5000, which naturally was out of the question for me.

Caveats

The income data originated from the 2016 national census, and I was a bit concerned at first that the results might be skewed if there was a low response rate, but as it turned out there seemed to have been a relatively high rate of participation with the Toronto non-response rate measured at 3.1% and 4.2% for the short- and long- form census, respectively. That seemed like an acceptable margin to me.

It would also have been nice to obtain more granular crime data, but maybe for privacy reasons the Toronto Police Services don't make this available to the general public.

Data Retrieval

First, I defined a handy helper function to download the datasets. This also meant that I could turn off the file downloads globally if I wanted to re-use the files that I'd already downloaded to save time or for repeatability.

The `downloadFiles` variable could be set to False to prevent files from being downloaded again.

In [ ]:
```
# Set this to False to prevent downloading of files
downloadFiles = True
```

In [ ]:
```
# Function to download a file from the Toronto Open Data Portal given the package ID
def download_file(package_id: str, explicit_filename=None, format="CSV", download=True):

        # Check for global download flag
        if download == False:
                if explicit_filename != None:
                        print("Skipping download of file " + explicit_filename)
                        return explicit_filename
                return None

        # Construct the dataset URL using the package ID
        request_params = {
                        "id": package_id,
                        "format": format
                        }

        base_url = "https://ckan0.cf.opendata.inter.prod-toronto.ca"
        source_url = base_url + "/api/3/action/package_show"

        # Request the metadata for this file from the data portal
        metadata_response = requests.get(source_url, params = request_params)
        if(metadata_response.status_code != 200):
                print("Couldn't fetch the file metadata!")
                print("Server replied with: [" + str(metadata_response.status_code) + "] " + metadata_response.reason)
                return None
        metadata_response = metadata_response.json()

        # Match the datastore corresponding to the requested format
        data_url = None
        resource_dump_data = None
        for idx, resource in enumerate(metadata_response["result"]["resources"]):
                if resource["datastore_active"] and resource["format"] == format:
                        data_url = base_url + "/datastore/dump/" + resource["id"]
                        resource_dump_data = requests.get(data_url).text
                        source_filename = resource["name"]
                        break

        # No matching datastore
        if data_url == None:
                print("*** There was no matching datastore for the given parameters")
                return None

        # Retrieve the data
        print("Requesting: %s (%s)" % (source_filename, format))
        data_response = requests.get(data_url)
        if(data_response.status_code != 200):
                print("Couldn't download data!")
                print("Server replied with: [" + str(data_response.status_code) + "] " + data_response.reason)
                return None

        # Extract the file data
        file_data = data_response.content
        #file_data = resource_dump_data

        # Set explicit filename
        if explicit_filename != None:
                filename = explicit_filename
```

```
        else:
                # Use the name provided in the metadata
                metadata_name = resource["name"]
                if metadata_name == None:
                        metadata_name = "datafile"
                filename = metadata_name

        # Save the file
        data_file = open(filename, "wb")
        data_file.write(file_data)
        data_file.close()
        print("Downloaded file as:", filename)

        # Return the filename
        return filename

#download_file("neighbourhood-profiles", "income_data.csv", "CSV", downloadFiles)
```

Some of the filenames that I used for the datasets:

In [ ]:
```
# Neighbourhood income dataset filename
nh_income_filename = "income_data.csv"

# Neighbourhood crime dataset filename
nh_crime_filename = "crime_data.csv"

# Neighbourhood GeoJSON data filename
nh_geodata_filename = "geo_data.geojson"

# Neighbourhood education data filename
nh_education_filename = "education_data.xlsx"
```

Toronto Income Data

Downloading the Data

Since the demographics data was from the 2016 census, I downloaded income data for 2015.

In [ ]:
```
# Income data source
neighbourhood_profiles_id = "neighbourhood-profiles"

# Download the neighbourhood income data file
nh_income_filename = download_file(neighbourhood_profiles_id, nh_income_filename, "CSV", downloadFiles)
```

```
Requesting: neighbourhood-profiles-2016-140-model (CSV)
Downloaded file as: income_data.csv
```

Data Cleaning and Preparation

As with all data analysis exercises, the data had to be cleaned and prepared, correcting for any deficiencies found therein. I also kept the 140 neighbourhood names and the average after-tax household incomes for each neighbourhood, discarding fields that I wasn't going to use.

To start off it can be useful to just get an idea of what the data looks like, and how the columns are arranged.

In [ ]:
```
# Read the neighbourhood data file into a DataFrame
neighbourhood_data = pd.read_csv(nh_income_filename)

# Display the first 10 rows
neighbourhood_data.head(10)
```

Out[ ]:

	_id	Category	Topic	Data Source	Characteristic	City of Toronto	Agincourt North	Agincourt South-Malvern West	Alderwood	Annex	...	Willowdale West	Willowridge-Martingrove-Richview	Wo
0	1	Neighbourhood Information	Neighbourhood Information	City of Toronto	Neighbourhood Number	NaN	129	128	20	95	...	37	7	
1	2	Neighbourhood Information	Neighbourhood Information	City of Toronto	TSNS2020 Designation	NaN	No Designation	No Designation	No Designation	No Designation	...	No Designation	No Designation	
2	3	Population	Population and dwellings	Census Profile 98-316-X2016001	Population, 2016	2,731,571	29,113	23,757	12,054	30,526	...	16,936	22,156	5
3	4	Population	Population and dwellings	Census Profile 98-316-X2016001	Population, 2011	2,615,060	30,279	21,988	11,904	29,177	...	15,004	21,343	5
4	5	Population	Population and dwellings	Census Profile 98-316-X2016001	Population Change 2011-2016	4.50%	-3.90%	8.00%	1.30%	4.60%	...	12.90%	3.80%	0
5	6	Population	Population and dwellings	Census Profile 98-316-X2016001	Total private dwellings	1,179,057	9,371	8,535	4,732	18,109	...	8,054	8,721	1
6	7	Population	Population and dwellings	Census Profile 98-316-X2016001	Private dwellings occupied by usual residents	1,112,929	9,120	8,136	4,616	15,934	...	7,549	8,509	1
7	8	Population	Population and dwellings	Census Profile 98-316-X2016001	Population density per square kilometre	4,334	3,929	3,034	2,435	10,863	...	5,820	4,007	
8	9	Population	Population and dwellings	Census Profile 98-316-X2016001	Land area in square kilometres	630.2	7.41	7.83	4.95	2.81	...	2.91	5.53	
9	10	Population	Age characteristics	Census Profile 98-316-X2016001	Children (0-14 years)	398,135	3,840	3,075	1,760	2,360	...	1,785	3,555	

10 rows × 146 columns

Based on the file documentation, I was interested in "Average after-tax income of households" for a particular year, which was located in the "Characteristic" column, so the next step was to extract that column and its data.

In [ ]:
```
# Extract the average after-tax household income for 2015
income_data = neighbourhood_data.loc[neighbourhood_data["Characteristic"]==" Average after-tax income of households in 2015 ($)"]
income_data
```

Out[ ]:

	_id	Category	Topic	Data Source	Characteristic	City of Toronto	Agincourt North	Agincourt South-Malvern West	Alderwood	Annex	...	Willowdale West	Willowridge-Martingrove-Richview	Woburn	Woodb Corri
1029	1030	Income	Income of households in 2015	Census Profile 98-316-X2016001	Average after-tax income of households in 20...	81,495	427,037	278,390	168,602	792,507	...	272,986	412,302	629,030	240,

1 rows × 146 columns

The first 7 columns weren't required for the analysis, including an average income for the entire aggregated city, so they were dropped.

In [ ]:
```
# Filter out some irrelevant columns
income_data = income_data.iloc[:, 4:]
income_data.reset_index(drop=True, inplace=True)

income_data
```

Out[ ]:

	Characteristic	City of Toronto	Agincourt North	Agincourt South-Malvern West	Alderwood	Annex	Banbury-Don Mills	Bathurst Manor	Bay Street Corridor	Bayview Village	...	Willowdale West	Willowridge-Martingrove-Richview	Woburn	Woodbine Corridor
0	Average after-tax income of households in 20...	81,495	427,037	278,390	168,602	792,507	493,486	251,583	352,218	354,894	...	272,986	412,302	629,030	240,272

1 rows × 142 columns

The last step for the income data was to convert it to a row-oriented DataFrame for later use.

In [ ]:
```
# Construct new DataFrame with the target columns
income_data_temp = pd.DataFrame()
income_data_temp["Neighbourhood"] = income_data.columns.transpose()
income_data_temp["Income"] = income_data.iloc[0, :].transpose().reset_index(drop=True)
income_data_temp
```

Out[ ]:

	Neighbourhood	Income
0	Characteristic	Average after-tax income of households in 20...
1	City of Toronto	81,495
2	Agincourt North	427,037
3	Agincourt South-Malvern West	278,390
4	Alderwood	168,602
...	...	...
137	Wychwood	239,484
138	Yonge-Eglinton	222,648
139	Yonge-St.Clair	541,217
140	York University Heights	302,358
141	Yorkdale-Glen Park	213,860

142 rows × 2 columns

Some final cleanup involved dropping the "Characteristic" and "City of Toronto" rows, neither of which are neighbourhoods.

In [ ]:
```
# Drop the characteristic row
income_data_temp.drop([0, 1], axis=0, inplace=True)
income_data = income_data_temp
income_data.head(5)
```

Out[ ]:

	Neighbourhood	Income
2	Agincourt North	427,037
3	Agincourt South-Malvern West	278,390
4	Alderwood	168,602
5	Annex	792,507
6	Banbury-Don Mills	493,486

And the income numbers were converted to numeric values.

In [ ]:
```
# Convert income to numbers
income_data["Income"] = income_data["Income"].str.replace(",", "")
income_data["Income"] = pd.to_numeric(income_data["Income"])
income_data.head(5)
```

Out[ ]:

	Neighbourhood	Income
2	Agincourt North	427037
3	Agincourt South-Malvern West	278390
4	Alderwood	168602
5	Annex	792507
6	Banbury-Don Mills	493486

Toronto Crime Data

The Toronto neighbourhood crime data is from the Open Data portal's Neighbourhood Crime Rates dataset, which compiled Toronto neighbourhood crime rates from 2014 to 2020 for the following categories:

- Assault
- Auto Theft
- Break And Enter
- Theft Over $5k (CAD)
- Homicide
- Shootings

I used crime data from 2015 so as to align with the income data, which as I mentioned earlier was also from 2015.

Definitions

I think it's safe to say that most people would be familiar with the categories listed above, but in any case I reproduced the definitions here for reference, copied from the supporting documentation file for the Open Data Portal's Neighbourhood Crime Rates microsite.

Assault
The direct or indirect application of force to another person, or the attempt or threat to apply force to another person, without that person's consent.

Auto Theft
The act of taking another person's vehicle (not including attempts). Auto Theft figures represent the number of vehicles stolen.

Break And Enter
The act of entering a place with the intent to commit an indictable offence therein.

Theft Over $5k (CAD)
The act of stealing property in excess of $5,000 (excluding auto theft).

Homicide
A homicide occurs when a person directly or indirectly, by any means, causes the death of another human being. Deaths caused by criminal negligence, suicide, or accidental or justifiable homicide (i.e self-defence) are not included.

Shootings
Any incident in which a projectile is discharged from a firearm (as defined under the Criminal Code of Canada) and injures a person. This excludes events such as suicide and police involved firearm discharges.

Downloading the Data

Next I downloaded the crime dataset from the Toronto Open Data portal.

In [ ]:
```
# Download the neighbourhood crime data file and the GeoJSON data together
nh_geodata_filename = download_file("neighbourhood-crime-rates", nh_geodata_filename, "GeoJSON", downloadFiles)
```

```
Requesting: neighbourhood-crime-rates (GeoJSON)
Downloaded file as: geo_data.geojson
```

Data Cleaning and Preparation

Just as with the income data, I had to clean and prepare the crime data, dropping irrelevant columns and keeping only the columns that I was interested in.

As usual, the first step was to read the data and take a high level view to get oriented.

In [ ]:
```
# Open the neighbourhood data file
crime_data = pd.read_csv(nh_crime_filename)
crime_data.head(5)
```

Out[ ]:

	_id	OBJECTID	HoodName	HoodID	F2021_Population_Projection	Assault_2014	Assault_2015	Assault_2016	Assault_2017	Assault_2018	...	TheftfromMotorV
0	1	1	Yonge-St.Clair	97	14362	16	25	34	25	28	...	
1	2	2	York University Heights	27	30660	273	298	363	351	362	...	
2	3	3	Lansing-Westgate	38	18577	42	81	67	84	69	...	
3	4	4	Yorkdale-Glen Park	31	18100	106	137	175	163	178	...	
4	5	5	Stonegate-Queensway	16	27838	91	74	78	98	86	...	

5 rows × 134 columns

Some columns had numeric identifiers and other data which weren't useful, so they were dropped.

In [ ]:
```
# Drop first few identifier columns
crime_data = crime_data.iloc[:, 2:]
crime_data.head(5)
```

Out[ ]:

	HoodName	HoodID	F2021_Population_Projection	Assault_2014	Assault_2015	Assault_2016	Assault_2017	Assault_2018	Assault_2019	Assault_2020	...	Thef
0	Yonge-St.Clair	97	14362	16	25	34	25	28	35	23	...	
1	York University Heights	27	30660	273	298	363	351	362	382	348	...	
2	Lansing-Westgate	38	18577	42	81	67	84	69	70	100	...	
3	Yorkdale-Glen Park	31	18100	106	137	175	163	178	210	157	...	
4	Stonegate-Queensway	16	27838	91	74	78	98	86	83	105	...	

5 rows × 132 columns

In [ ]:
```
# Drop some other specific irrelevant columns
crime_data.drop(columns=["HoodID", "F2021_Population_Projection"], inplace=True)
crime_data.head(5)
```

Out[ ]:

	HoodName	Assault_2014	Assault_2015	Assault_2016	Assault_2017	Assault_2018	Assault_2019	Assault_2020	Assault_2021	Assault_Rate2014	...	Theftfroml
0	Yonge-St.Clair	16	25	34	25	28	35	23	55	127.0144	...	
1	York University Heights	273	298	363	351	362	382	348	339	950.5571	...	
2	Lansing-Westgate	42	81	67	84	69	70	100	97	261.0966	...	
3	Yorkdale-Glen Park	106	137	175	163	178	210	157	166	697.5061	...	
4	Stonegate-Queensway	91	74	78	98	86	83	105	77	354.9696	...	

5 rows × 130 columns

The neighbourhood column name didn't match the income data (i.e. "Neighbourhood" vs "HoodName"), so it had to be renamed.

Finally, the 2015 crime rates were extracted from the dataset.

In [ ]:
```
# Rename the neighbourhood name column to match income data
crime_data.rename(columns={"HoodName": "Neighbourhood"}, inplace=True)

# Keep only crime rate data from 2015 to match the income data
neighbourhood = crime_data["Neighbourhood"] # Save neighbourhood name data
crime_data = crime_data.loc[:, crime_data.columns.str.contains("Rate2015")]
crime_data.insert(0, "Neighbourhood", neighbourhood)

# Clean up column names
crime_data.columns = crime_data.columns.str.replace("_Rate2015", "") # Remove rate and year suffix
crime_data.columns = crime_data.columns.str.replace(r"([a-z])([A-Z])", "\g<1> \g<2>", regex=True) # Add spaces to terms
crime_data.columns = crime_data.columns.str.replace("Theft Over", "Theft Over $5k") # Clarify Theft Over term

# Clean up dataframe
crime_data = crime_data.reset_index(drop=True)

# Extract list of crimes
crimes = crime_data.columns.values[1:]

crime_data.head(5)
```

Out[ ]:

	Neighbourhood	Assault	Auto Theft	Break And Enter	Theft Over $5k	Homicide	Shootings	Theftfrom Motor Vehicle
0	Yonge-St.Clair	196.1246	23.53495	156.8997	39.22491	NaN	NaN	86.29482
1	York University Heights	1041.4850	349.49150	489.2881	160.76610	NaN	NaN	429.87450
2	Lansing-Westgate	494.6263	134.34290	164.8754	30.53249	NaN	NaN	183.19490
3	Yorkdale-Glen Park	902.0279	348.95970	375.2963	98.76218	6.584146	19.752440	500.39510
4	Stonegate-Queensway	288.1732	124.61540	175.2405	31.15386	NaN	7.788465	237.54820

One thing I noticed immediately was that some values were NaN, so I took take care of those right away.

In [ ]:
```
# Substitute NaN for zeros
crime_data.fillna(0, inplace=True)

crime_data.head(5)
```

Out[ ]:

	Neighbourhood	Assault	Auto Theft	Break And Enter	Theft Over $5k	Homicide	Shootings	Theftfrom Motor Vehicle
0	Yonge-St.Clair	196.1246	23.53495	156.8997	39.22491	0.000000	0.000000	86.29482
1	York University Heights	1041.4850	349.49150	489.2881	160.76610	0.000000	0.000000	429.87450
2	Lansing-Westgate	494.6263	134.34290	164.8754	30.53249	0.000000	0.000000	183.19490
3	Yorkdale-Glen Park	902.0279	348.95970	375.2963	98.76218	6.584146	19.752440	500.39510
4	Stonegate-Queensway	288.1732	124.61540	175.2405	31.15386	0.000000	7.788465	237.54820

Much better! Those values were updated to zeros.

Cross Validation

Since I was going to be using the neighbourhood names to correlate the two datasets, I had to make sure that there weren't any mismatches.

In [ ]:
```
# Get just the neighbourhood names
income_neighbourhoods = income_data["Neighbourhood"]
crime_neighbourhoods = crime_data["Neighbourhood"]
print("Income dataset has", len(income_neighbourhoods), "rows")
print("Crime dataset has", len(crime_neighbourhoods), "rows")

# Sort both lists of names
income_neighbourhoods = income_neighbourhoods.sort_values(ascending=True)
crime_neighbourhoods = crime_neighbourhoods.sort_values(ascending=True)
income_neighbourhoods.reset_index(drop=True, inplace=True)
crime_neighbourhoods.reset_index(drop=True, inplace=True)

# Find the differences between the two sets
income_crime_diffs = set(crime_neighbourhoods).difference(set(income_neighbourhoods))
print("Differences from the income dataset:", income_crime_diffs)

crime_income_diffs = set(income_neighbourhoods).difference(set(crime_neighbourhoods))
print("Differences from the crime dataset:", crime_income_diffs)
```

```
Income dataset has 140 rows
Crime dataset has 140 rows
Differences from the income dataset: {'Weston-Pellam Park', 'Mimico', 'North St.James Town', 'Cabbagetown-South St.James Town'}
Differences from the crime dataset: {'North St. James Town', 'Mimico (includes Humber Bay Shores)', 'Weston-Pelham Park', 'Cabbagetown-South St. James Town'}
```

From this comparison it could be seen that both name lists had the same number of rows, but that there were some spelling differences between the two.

If uncorrected, these would have resulted in mismatches and gaps in the data, so the next step was to fix these and re-run the comparison.

In [ ]:
```
# Correct the spelling of some neighbourhood names
income_data["Neighbourhood"].replace("Weston-Pelham Park", "Weston-Pellam Park", inplace=True)
income_data["Neighbourhood"].replace("North St. James Town", "North St.James Town", inplace=True)
income_data["Neighbourhood"].replace("Cabbagetown-South St. James Town", "Cabbagetown-South St.James Town", inplace=True)
income_data["Neighbourhood"].replace("Mimico (includes Humber Bay Shores)", "Mimico", inplace=True)

# Get just the neighbourhood names
income_neighbourhoods = income_data["Neighbourhood"]
crime_neighbourhoods = crime_data["Neighbourhood"]

# Sort both lists of names
income_neighbourhoods = income_neighbourhoods.sort_values(ascending=True)
crime_neighbourhoods = crime_neighbourhoods.sort_values(ascending=True)

# Find the differences between the two sets
income_crime_diffs = set(crime_neighbourhoods).difference(set(income_neighbourhoods))
print("Differences in income dataset:", income_crime_diffs)

crime_income_diffs = set(income_neighbourhoods).difference(set(crime_neighbourhoods))
print("Differences in crime dataset:", crime_income_diffs)
```

```
Differences in income dataset: set()
Differences in crime dataset: set()
```

Hurray, no more differences!

Geo Data

Cleaning and Preparation

Before attempting to merge and visualize the data, I had to also clean and prepare the mapping data that was loaded earlier.

As usual, the first step is to take a peek at the file contents.

In [ ]:
```
# Read the geo data file into a DataFrame
peek_geo_data = pd.read_csv(nh_geodata_filename)
peek_geo_data.head(2)
```

Out[ ]:

	_id	OBJECTID	HoodName	HoodID	F2021_Population_Projection	Assault_2014	Assault_2015	Assault_2016	Assault_2017	Assault_2018	...	TheftfromMotorV
0	1	1	Yonge-St.Clair	97	14362	16	25	34	25	28	...	
1	2	2	York University Heights	27	30660	273	298	363	351	362	...	

2 rows × 134 columns

The geo data was embedded in a file containing other data, so it must be extracted. I also renamed the neighbourhood column as before.

In [ ]:
```
# Read the file data
nh_geo_data = gpd.GeoDataFrame(peek_geo_data)

# Filter out some irrelevant columns
nh_geo_data = nh_geo_data.iloc[:, 2:]

# Drop irrelevant columns
nh_geo_data.drop(columns=["HoodID", "F2021_Population_Projection"], inplace=True)

# Rename the neighbourhood column
nh_geo_data.rename(columns={"HoodName": "Neighbourhood"}, inplace=True)

nh_geo_data.head(2)
```

Out[ ]:

	Neighbourhood	Assault_2014	Assault_2015	Assault_2016	Assault_2017	Assault_2018	Assault_2019	Assault_2020	Assault_2021	Assault_Rate2014	...	Theftf
0	Yonge-St.Clair	16	25	34	25	28	35	23	55	127.0144	...	
1	York University Heights	273	298	363	351	362	382	348	339	950.5571	...	

2 rows × 130 columns

I needed both the neighbourhood names and their associated geographical boundaries, so that they could be combined with the income and crime data later on.

In [ ]:
```
# Save some columns that we're interested in
neighbourhood = nh_geo_data["Neighbourhood"] # Save neighbourhood name data
geometry = nh_geo_data["geometry"] # Save neighbourhood geometry data
```

And as before, I need to extract only the data for 2015.

In [ ]:
```
# Keep only crime rate data from 2015 to match the income data
nh_geo_data = nh_geo_data.loc[:, nh_geo_data.columns.str.contains("Rate2015")]
nh_geo_data.insert(0, "Neighbourhood", neighbourhood)
nh_geo_data.head(5)
```

Out[ ]:

	Neighbourhood	Assault_Rate2015	AutoTheft_Rate2015	BreakAndEnter_Rate2015	TheftOver_Rate2015	Homicide_Rate2015	Shootings_Rate2015	TheftfromMot
0	Yonge-St.Clair	196.1246	23.53495	156.8997	39.22491	NaN	NaN	
1	York University Heights	1041.4850	349.49150	489.2881	160.76610	NaN	NaN	
2	Lansing-Westgate	494.6263	134.34290	164.8754	30.53249	NaN	NaN	
3	Yorkdale-Glen Park	902.0279	348.95970	375.2963	98.76218	6.584146	19.752440	
4	Stonegate-Queensway	288.1732	124.61540	175.2405	31.15386	NaN	7.788465	

And finally perform a bit of data cleanup.

In [ ]:
```
# Replace NaN values with zeros
nh_geo_data_checkpoint = nh_geo_data.fillna(0)

nh_geo_data_checkpoint = gpd.GeoDataFrame(nh_geo_data_checkpoint)
```

In [ ]:
```
geometry
```

Out[ ]:
```
0      {"type": "Polygon", "coordinates": [[[-79.3911...
1      {"type": "Polygon", "coordinates": [[[-79.5052...
2      {"type": "Polygon", "coordinates": [[[-79.4399...
3      {"type": "Polygon", "coordinates": [[[-79.4396...
4      {"type": "Polygon", "coordinates": [[[-79.4926...
                             ...
135    {"type": "Polygon", "coordinates": [[[-79.3434...
136    {"type": "Polygon", "coordinates": [[[-79.4359...
137    {"type": "Polygon", "coordinates": [[[-79.3774...
138    {"type": "Polygon", "coordinates": [[[-79.4646...
139    {"type": "Polygon", "coordinates": [[[-79.4803...
Name: geometry, Length: 140, dtype: object
```

Armed with these prepared datasets, it was time to assemble a visualization.

First Glance

The first task with all of the prepared data was to try and display the Toronto neighbourhood data as a map.

Before I could display the map however, the household income data had to combined with the map data.

For the sake of brevity, I calculated the income in 1000s of Canadian dollars.

In [ ]:
```
# Add income data in 1000s of dollars
income_data["Income 1000s"] = income_data.Income / 1000
nh_geo_data = nh_geo_data_checkpoint.merge(income_data, on="Neighbourhood", how="left")

# Add neighbourhood map geometry
nh_geo_data.insert(8, "geometry", geometry)

# Clean up column names
nh_geo_data.columns = nh_geo_data.columns.str.replace("_Rate2015", "") # Remove rate and year suffix
nh_geo_data.columns = nh_geo_data.columns.str.replace(r"([a-z])([A-Z])", "\g<1> \g<2>", regex=True) # Add spaces to terms
nh_geo_data.columns = nh_geo_data.columns.str.replace("Theft Over", "Theft Over $5k") # Clarify Theft Over term

# Clean up dataframe indices
nh_geo_data = nh_geo_data.reset_index(drop=True)

nh_geo_data.head(2)
```

Out[ ]:

	Neighbourhood	Assault	Auto Theft	Break And Enter	Theft Over $5k	Homicide	Shootings	Theftfrom Motor Vehicle	geometry	Income	Income 1000s
0	Yonge-St.Clair	196.1246	23.53495	156.8997	39.22491	0.0	0.0	86.29482	{"type": "Polygon", "coordinates": [[[-79.3911...	541217	541.217
1	York University Heights	1041.4850	349.49150	489.2881	160.76610	0.0	0.0	429.87450	{"type": "Polygon", "coordinates": [[[-79.5052...	302358	302.358

Since the mapping data was imported as a string, it had to be converted to a format that the plotter would accept.

In [ ]:
```
# Convert string-based geo data to shapes
nh_geo_data["geometry"] = nh_geo_data["geometry"].apply(lambda x : shape(json.loads(x)))

# Set the projection on the geo data
nh_geo_data = gpd.GeoDataFrame(nh_geo_data) # Can only set CRS on a GeoDataFrame
nh_geo_data = gpd.GeoDataFrame.set_crs(nh_geo_data, "EPSG:4326", allow_override=True)

nh_geo_data.head(2)
```

Out[ ]:

	Neighbourhood	Assault	Auto Theft	Break And Enter	Theft Over $5k	Homicide	Shootings	Theftfrom Motor Vehicle	geometry	Income	Income 1000s
0	Yonge-St.Clair	196.1246	23.53495	156.8997	39.22491	0.0	0.0	86.29482	POLYGON ((-79.39118 43.68108, -79.39139 43.680...	541217	541.217
1	York University Heights	1041.4850	349.49150	489.2881	160.76610	0.0	0.0	429.87450	POLYGON ((-79.50527 43.75987, -79.50487 43.759...	302358	302.358

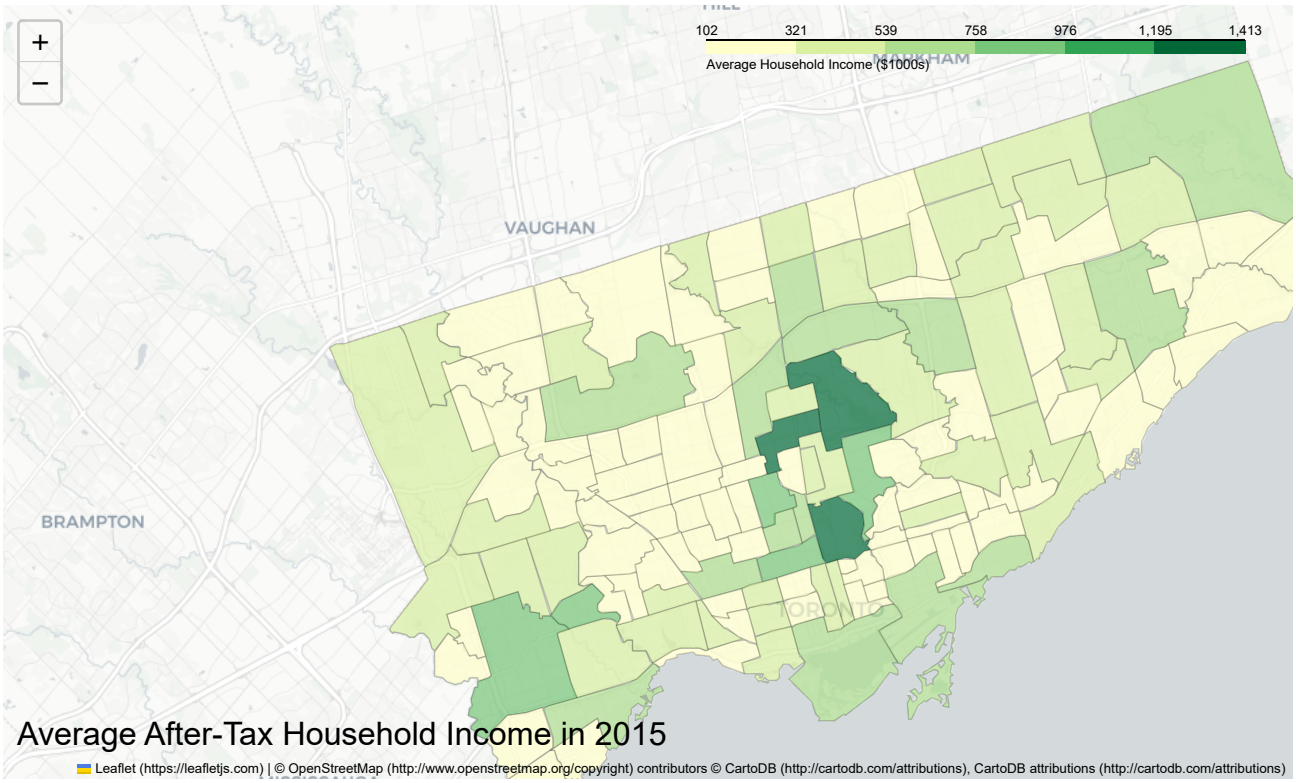
Now that the geo data had been prepared it was time to draw the interactive map, superimposing the income data as a colored map for reference. The darker the green, the higher the average household income was in 2015.

For ease of use, an informative tooltip was displayed when hovering over a particular neighbourhood.

In [ ]:
```
# Generate folium base map
map = folium.Map(location=[43.70, -79.4], tiles="CartoDB positron", zoom_start=11)

# Add the neighbourhood and income data
chloroMap = folium.Choropleth(
        geo_data=nh_geo_data,
        data=nh_geo_data,
```

```
        columns=["Neighbourhood", "Income 1000s"],
        key_on="feature.properties.Neighbourhood",
        fill_color="YlGn",
        fill_opacity=0.7,
        line_opacity=0.2,
        legend_name="Average Household Income ($1000s)",
        ).add_to(map)

# Add informational tooltips
folium.GeoJsonTooltip(fields=["Neighbourhood", "Income"], aliases=["Neighbourhood", "Income $"], localize=True).add_to(chloroMap.geojson

# Add the title
map_title = "Average After-Tax Household Income in 2015"
map.get_root().html.add_child(
        folium.Element("<h3 style=\"position: absolute; bottom: 1rem; left: 1rem; z-index: 1000;\">" + map_title + "</h3>")
        )

map
```

Out[ ]:



The map provided an easy visual overview of the distribution of low and high income neighbourhoods relative to each other, but it wasn't suitable for comparing the two extremes.

For that comparison I needed to extract a list of the 10% lowest and highest income neighbourhoods. In absolute numbers this equated to the bottom and top 14 neighbourhoods.

In [ ]:
```
# Top and bottom 10%, i.e. 14 neighbourhoods
data_size = int( income_data.Neighbourhood.count() * 0.1 )

# Extract lowest income neighbourhoods
income_min = income_data.sort_values("Income", ascending=True)
income_min = income_min.head(data_size)
income_min.reset_index(drop=True, inplace=True)

# Extract highest income neighbourhoods
income_max = income_data.sort_values("Income", ascending=False)
income_max = income_max.head(data_size)
income_max.reset_index(drop=True, inplace=True)
```

Neighbourhoods with the Lowest Income

I wanted to display each of the candidate neighbourhood data using tables, so I defined a function to do that.

In [ ]:
```
# Address some rendering issues with plotly images when exporting the notebook
pio.renderers.default = "plotly_mimetype+notebook"

# Function to display a data table
def display_table(dataframe, title, width=1000, height=None):

        # Get table dimensions
```

```
        numColumns = len(dataframe.columns)
        numRows = len(dataframe)

        # Define table layout
        if height==None:
                table_layout = dict(
                        title_text=title,
                        font=dict(color="black", size=16),
                        height=210 + len(dataframe) * 34,
                        width=width
                        )
        else:
                table_layout = dict(
                        title_text=title,
                        font=dict(color="black", size=16),
                        height=height,
                        width=width
                        )

        # Table styling
        headerColor = "grey"
        lineColor = "darkgrey"

        # Build alternating row colors
        tableRowColors = []
        for i in range(int(numRows / 2)+1):
                tableRowColors.append("white")
                tableRowColors.append("lightgrey")
        tableRowColors = [tableRowColors] * numColumns

        # Generate column formats
        columnFormats = []
        for type in dataframe.dtypes:
                if type == "int64":
                        columnFormats.append(",")
                elif type == "float64":
                        columnFormats.append(",")
                else:
                        columnFormats.append("")

        # Build the table
        data_table = go.Figure(
                data=[go.Table(

                        header=dict(
                                values=list(dataframe.columns),
                                fill_color=headerColor,
                                font=dict(color="white", size=15),
                                line_color=lineColor,
                                ),

                        cells=dict(
                                values=dataframe.transpose().values.tolist(),
                                fill_color=tableRowColors,
                                font=dict(color="black", size=12),
                                format=columnFormats,
                                height=32,
                                line_color=lineColor,
                                ),
                )],
                layout=table_layout,
        )

        # Display the table
        data_table.show()
```

Next, I called the function to display the lowest income neighbourhoods.

In [ ]:
```
income_table = income_min[["Neighbourhood", "Income"]]
display_table(income_table, "Neighbourhoods with the Lowest Household Income ($)", 700)
```

Neighbourhoods with the Lowest Household Income ($)

Neighbourhood	Income
Taylor-Massey	102,259
Regent Park	103,250
Beechborough-Greenbrook	109,880
Rustic	113,995
Elms-Old Rexdale	123,119
Broadview North	126,349
Weston-Pellam Park	127,491
Dufferin Grove	128,586
Etobicoke West Mall	132,460
Caledonia-Fairbank	132,879
Oakridge	134,303
Rexdale-Kipling	134,305
Long Branch	137,480
Thistletown-Beaumond Heights	140,050

Neighbourhoods with the Highest Income

I did the same for top 10% of neighbourhoods having the highest incomes.

In [ ]:
```
income_table = income_max[["Neighbourhood", "Income"]]
display_table(income_table, "Neighbourhoods with the Highest Income", 700)
```

Neighbourhoods with the Highest Income

Neighbourhood	Income
Rosedale-Moore Park	1,413,132
Bridle Path-Sunnybrook-York Mills	1,392,010
Lawrence Park South	1,216,585
Kingsway South	900,624
Islington-City Centre West	854,623
Leaside-Bennington	841,619
Annex	792,507
Forest Hill South	784,645
Rouge	729,154
Bedford Park-Nortown	720,203
South Riverdale	662,651
Waterfront Communities-The Island	662,333
The Beaches	659,192
St.Andrew-Windfields	649,291

Analysis of the Data

Neighbourhoods with Highest Crime Rates

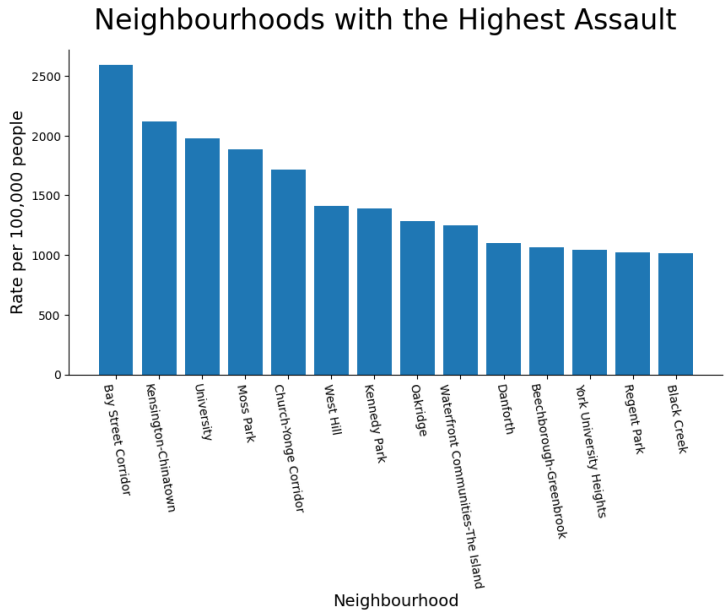
The next steps of course were to extract the 10% of neighbourhoods which had the highest crime rates.

Since there were several categories of crime (i.e. Assault, Auto Theft, etc), I decided to plot a chart for each one individually and restricted the output to the top 10% for readability, rather than displaying all 140 neighbourhoods each time.

Beside each chart I added the related table of statistics for reference.

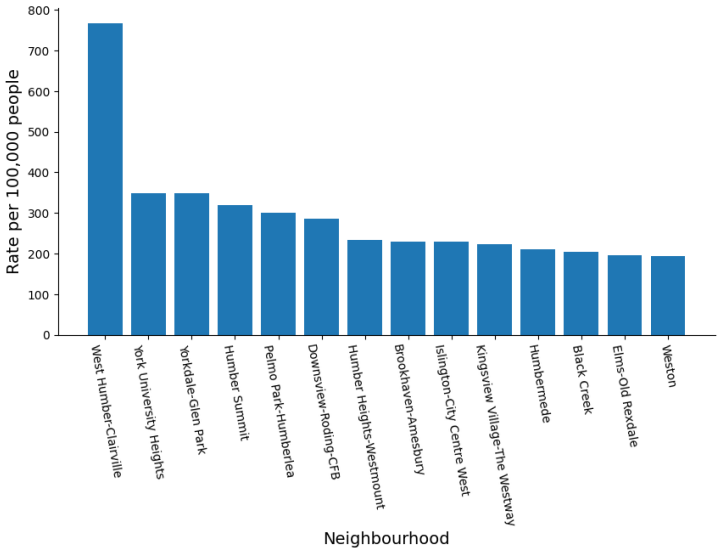
In [ ]:
```
# Establish chart dimensions
chart_width = 10
chart_height = 5

# Define table row colors
tableRowColors = []
for i in range(7):
        tableRowColors.append([ "lightgray", "lightgray"])
        tableRowColors.append([ "white", "white" ])

# Define table header colors
tableHeaderColors = [ "gray", "gray" ]

# Plot the chart for each crime
for crime in crimes:

        # Sort the crime data
        specific_crime_data = crime_data.sort_values(crime, ascending=False)
        specific_crime_data = specific_crime_data[["Neighbourhood", crime]].head(data_size)

        # Round the rate to the nearest whole number
        specific_crime_data[crime] = specific_crime_data[crime].round(0).astype(int)

        # Plot the chart
        plt.figure(figsize=(chart_width, chart_height))
        plt.bar(specific_crime_data.Neighbourhood, specific_crime_data[crime])
        plt.xticks(rotation=-80)

        # Hide the chart spines
        plt.gca().spines["right"].set_color("none")
        plt.gca().spines["top"].set_color("none")
```

```
        # Prepare the table data
        cell_text = []
        for row in range(len(specific_crime_data)):
                cell_text.append(specific_crime_data.iloc[row])

        # Render the statistic table
        statsTable = plt.table(
                cellText=cell_text,
                colLabels=specific_crime_data.columns,
                colWidths=[0.6, 0.3],
                cellColours=tableRowColors,
                cellLoc="center",
                colColours=tableHeaderColors,
                bbox = [1.1, -0.3, 0.8, 1.2],
                )

        # Set the header font color
        statsTable[0, 0].get_text().set_color("white")
        statsTable[0, 1].get_text().set_color("white")

        # Set table font size and scaling
        statsTable.auto_set_font_size(False)
        statsTable.set_fontsize(14)
        statsTable.scale(0.9, 1.75)

        # Display the axis labels
        plt.xlabel("Neighbourhood", fontsize=14)
        plt.ylabel("Rate per 100,000 people", fontsize=14)

        # Set the title
        plt.suptitle("Neighbourhoods with the Highest " + crime, fontsize=24)

        # Display the charts
        plt.show()
```



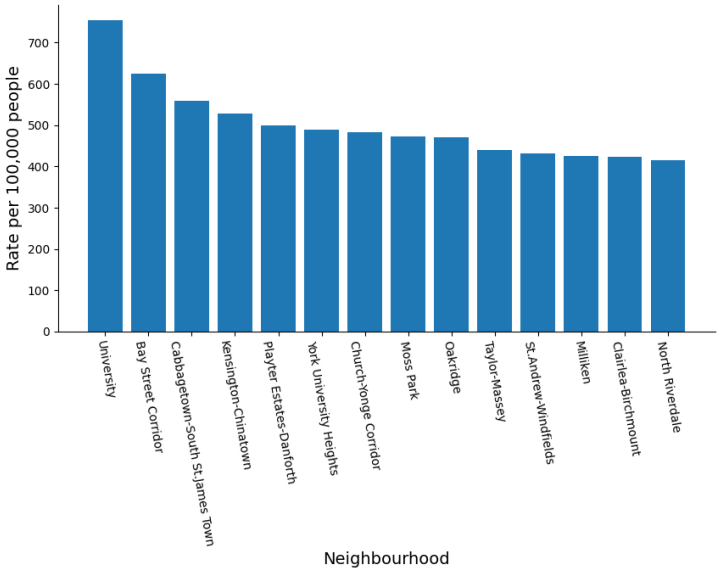
Neighbourhood	Assault
Bay Street Corridor	2594
Kensington-Chinatown	2118
University	1974
Moss Park	1887
Church-Yonge Corridor	1713
West Hill	1411
Kennedy Park	1392
Oakridge	1287
Waterfront Communities-The Island	1247
Danforth	1098
Beechborough-Greenbrook	1066
York University Heights	1041
Regent Park	1025
Black Creek	1016

Neighbourhoods with the Highest Auto Theft



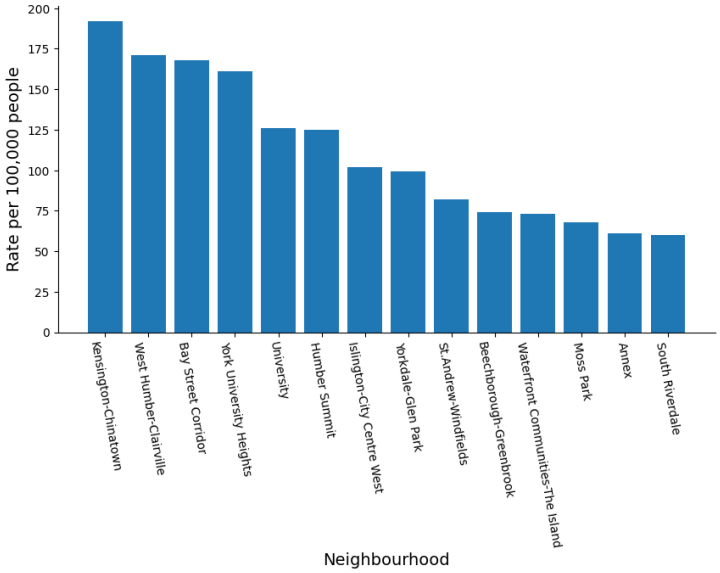
Neighbourhood	Auto Theft
West Humber-Clairville	767
York University Heights	349
Yorkdale-Glen Park	349
Humber Summit	320
Pelmo Park-Humberlea	301
Downsview-Roding-CFB	286
Humber Heights-Westmount	233
Brookhaven-Amesbury	230
Islington-City Centre West	229
Kingsview Village-The Westway	222
Humbermede	211
Black Creek	205
Elms-Old Rexdale	195
Weston	194

Neighbourhoods with the Highest Break And Enter



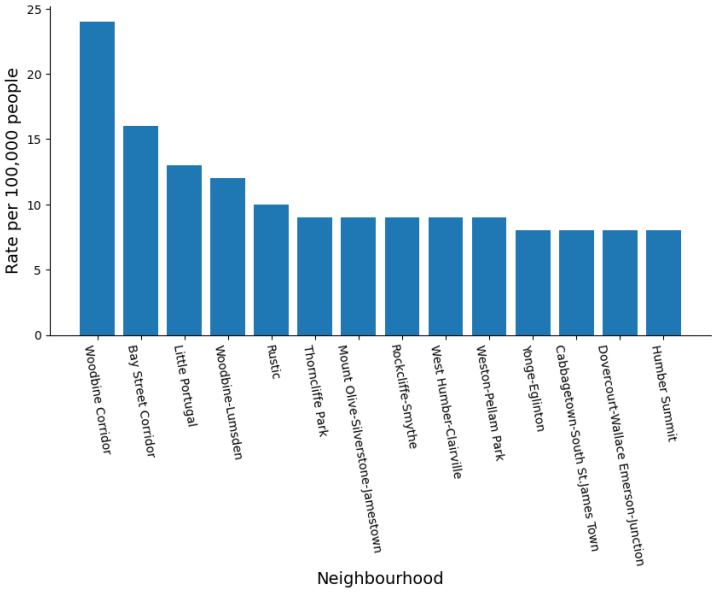
Neighbourhood	Break And Enter
University	754
Bay Street Corridor	624
Cabbagetown-South St.James Town	559
Kensington-Chinatown	527
Playter Estates-Danforth	499
York University Heights	489
Church-Yonge Corridor	482
Moss Park	473
Oakridge	471
Taylor-Massey	439
St.Andrew-Windfields	431
Milliken	425
Clairlea-Birchmount	424
North Riverdale	414

Neighbourhoods with the Highest Theft Over $5k



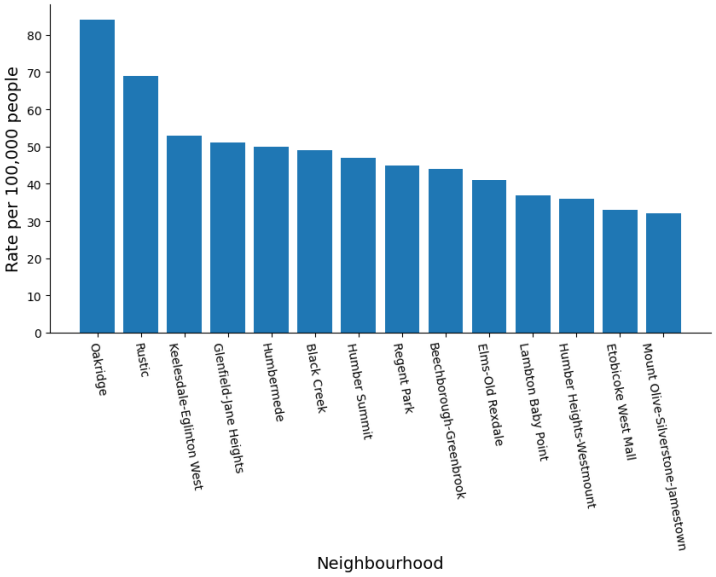
Neighbourhood	Theft Over $5k
Kensington-Chinatown	192
West Humber-Clairville	171
Bay Street Corridor	168
York University Heights	161
University	126
Humber Summit	125
Islington-City Centre West	102
Yorkdale-Glen Park	99
St.Andrew-Windfields	82
Beechborough-Greenbrook	74
Waterfront Communities-The Island	73
Moss Park	68
Annex	61
South Riverdale	60

Neighbourhoods with the Highest Homicide



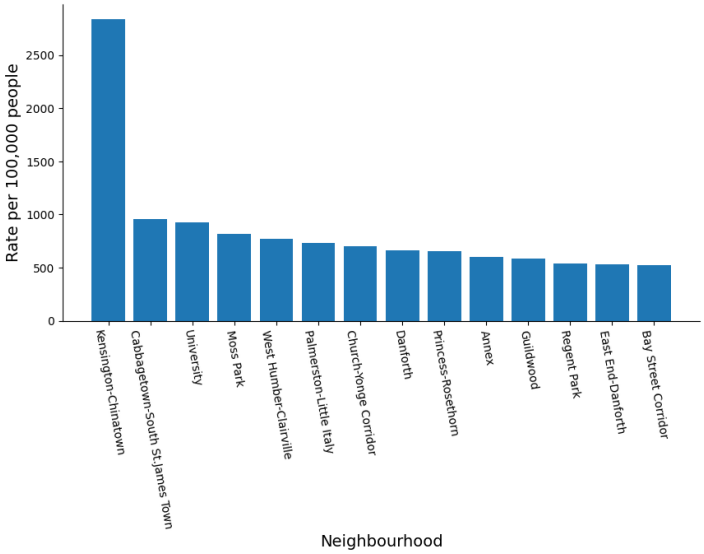
Neighbourhood	Homicide
Woodbine Corridor	24
Bay Street Corridor	16
Little Portugal	13
Woodbine-Lumsden	12
Rustic	10
Thorncliffe Park	9
Mount Olive-Silverstone-Jamestown	9
Rockcliffe-Smythe	9
West Humber-Clairville	9
Weston-Pellam Park	9
Yonge-Eglinton	8
Cabbagetown-South St.James Town	8
Dovercourt-Wallace Emerson-Junction	8
Humber Summit	8

Neighbourhoods with the Highest Shootings



Neighbourhood	Shootings
Oakridge	84
Rustic	69
Keelesdale-Eglinton West	53
Glenfield-Jane Heights	51
Humbermede	50
Black Creek	49
Humber Summit	47
Regent Park	45
Beechborough-Greenbrook	44
Elms-Old Rexdale	41
Lambton Baby Point	37
Humber Heights-Westmount	36
Etobicoke West Mall	33
Mount Olive-Silverstone-Jamestown	32

Neighbourhoods with the Highest Theftfrom Motor Vehicle



Neighbourhood	Theftfrom Motor Vehicl
Kensington-Chinatown	2836
Cabbagetown-South St.James Town	954
University	930
Moss Park	819
West Humber-Clairville	773
Palmerston-Little Italy	730
Church-Yonge Corridor	699
Danforth	665
Princess-Rosethorn	652
Annex	602
Guildwood	583
Regent Park	539
East End-Danforth	529
Bay Street Corridor	527

From these charts, I could already start to see some neighbourhood names appear more than once, but rather than simply eyeballing the lists I combined the two datasets to see how they correlated with each other.

In [ ]:
```
# Combine the crime data with the income data, for matching neighbourhoods
crime_income_df = pd.merge(crime_data, income_data, on="Neighbourhood")
crime_income_df.head(3)
```

Out[ ]:

	Neighbourhood	Assault	Auto Theft	Break And Enter	Theft Over $5k	Homicide	Shootings	Theftfrom Motor Vehicle	Income	Income 1000s
0	Yonge-St.Clair	196.1246	23.53495	156.8997	39.22491	0.0	0.0	86.29482	541217	541.217
1	York University Heights	1041.4850	349.49150	489.2881	160.76610	0.0	0.0	429.87450	302358	302.358
2	Lansing-Westgate	494.6263	134.34290	164.8754	30.53249	0.0	0.0	183.19490	356122	356.122

Map of the Highest-Crime Neighbourhoods

Before I started to work on the correlations, I thought it might be interesting to visualize the relative locations of the highest-crime neighbourhoods, in case there was a geographical relationship that wasn't immediately apparent simply by examining the raw numbers.

To accomplish this, I first merged the geo dataset with the crime dataset and since there were several crime categories, I constructed a map data layer for each crime type. This provided a visually interesting and interactive way to see how these neighbourhoods were distributed across the city.

For reference purposes I also added income as one of the map layers.

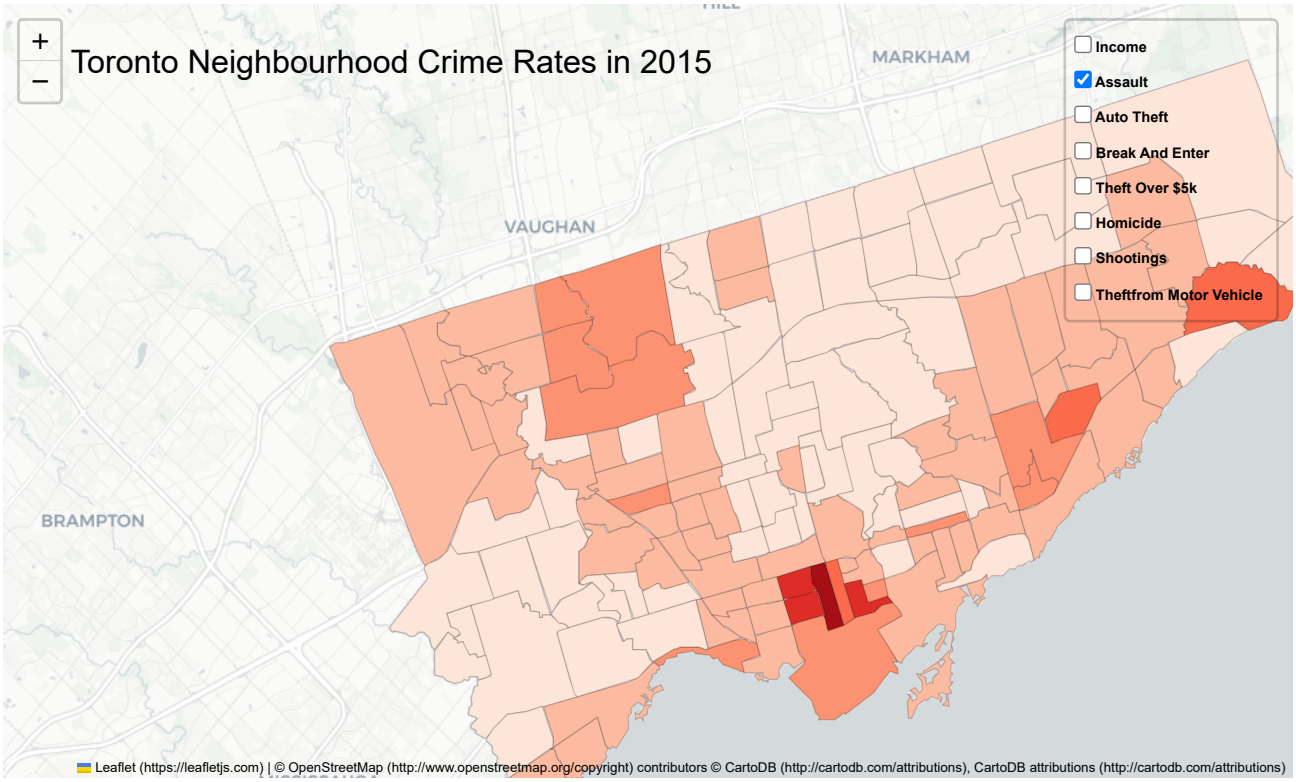
In [ ]:
```
# Generate folium base map
crime_map = folium.Map(location=[43.70, -79.4], tiles=None, zoom_start=11)

# Add tile layer separately
folium.TileLayer("CartoDB positron", control=False).add_to(crime_map)

# Assemble the base map data
base_crime_map_data = nh_geo_data[["Neighbourhood", "Income"]].copy()
base_crime_map_data["Income"] = (base_crime_map_data["Income"] / base_crime_map_data["Income"].max()) * 100
base_crime_map_data = gpd.GeoDataFrame(nh_geo_data)

# Add base map
base_crime_map = folium.Choropleth(
        geo_data=base_crime_map_data,
        data=base_crime_map_data,
        columns=["Neighbourhood", "Income"],
        control=True,
        key_on="feature.properties.Neighbourhood",
        fill_color="YlGn",
        fill_opacity=1,
        line_opacity=0.2,
        name="Income",
        overlay=True,
        show=False
        ).add_to(crime_map)

# Remove the base layer legend since it's unnecessary
for key in base_crime_map._children:
        if key.startswith('color_map'):
                del(base_crime_map._children[key])

# Build a data layer for each individual crime dataset
first = True
for crime in crimes:

        # Sort the crime data
        individual_crime_data = crime_data.sort_values(crime, ascending=False).copy()

        # Scale crime rate to percentage
        individual_crime_data[crime] = (individual_crime_data[crime] / individual_crime_data[crime].max()) * 100

        # Round the rate to the nearest whole number
        individual_crime_data[crime] = individual_crime_data[crime].round(0).astype(int)

        # Keep only required columns
        individual_crime_data = individual_crime_data[["Neighbourhood", crime]]

        # Merge the geo dataset with the crime dataset
        crime_geo_data = pd.merge(individual_crime_data, nh_geo_data[["Neighbourhood", "geometry"]], on="Neighbourhood")
        crime_geo_data = gpd.GeoDataFrame(crime_geo_data)

        # Create and add the map layer
        crime_chloroMap = folium.Choropleth(
                geo_data=crime_geo_data,
                data=crime_geo_data,
                columns=["Neighbourhood", crime],
                key_on="feature.properties.Neighbourhood",
                fill_color="Reds",
                fill_opacity=1,
                highlight=True,
                line_opacity=0.2,
                legend_name="Highest Rate of " + crime,
                name=crime,
```

```
                overlay=True,
                show=first
                ).add_to(crime_map)

        # Remove the layer legends since we don't really need them
        for key in crime_chloroMap._children:
                if key.startswith('color_map'):
                        del(crime_chloroMap._children[key])

        # Add the tooltip for this layer
        folium.GeoJsonTooltip(fields=["Neighbourhood", crime], aliases=["Neighbourhood", "Crime Rate %"], localize=True).add_to(crime_ch

        # Only show the first layer at first
        first=False

# Add a layer control to the map
crime_map_layer_control = folium.LayerControl(collapsed=False)
crime_map_layer_control.add_to(crime_map)

# Add tooltips to identify neighbourhoods
folium.GeoJsonTooltip(fields=["Neighbourhood", "Income"], aliases=["Neighbourhood", "Income $"], localize=True).add_to(base_crime_map.ge

# Add the title
crime_map_title = "Toronto Neighbourhood Crime Rates in 2015"
crime_map.get_root().html.add_child(
        folium.Element("<h3 style=\"position: absolute; top: 1rem; left: 5rem; z-index: 1000;\">" + crime_map_title + "</h3>")
        )
```

Out[ ]:
```
<branca.element.Element at 0x15cc311e810>
```

In [ ]:
```
crime_map
```

Out[ ]:



Although certain categories of crime appeared in clusters in particular regions (.e.g Auto Theft), there were more likely to be multiple hotspots throughout the city. One thing that could be observed however, is that neighbourhoods with the highest levels of crime tended to be adjacent to one or more other neighbourhoods that were similarly affected.

As usual, human activity isn't necessarily constrained to artificial boundaries and discrete regions.

Crime Rates in Low Income Neighbourhoods

Now that the datasets had been combined it was time to see how they correlated.

For each crime category I printed out the names of neighbourhoods that were correspondingly found in the low income list, and then I did the same for the high income list.

In [ ]:
```
# Correlate crime rates with lowest income neighbourhoods
for crime in crimes:

        # Extract the top 10% crime list for each category
```

```
    crime_max = crime_income_df.sort_values(crime, ascending=False)[["Neighbourhood", crime]].head(data_size)

    # Merge the crime data with the income data
    crime_max_low_income = crime_max.merge(income_min, on=["Neighbourhood"])
    crime_max_low_income.drop("Income 1000s", axis=1, inplace=True)

    # Round the crime rate
    crime_max_low_income[crime] = crime_max_low_income[crime].round()

    # Heading
    title ="Neighbourhoods with Highest Rates of " + crime + " and also Lowest Income"

    # Only print results if the category isn't empty
    if len(crime_max_low_income) == 0:
            print("There are no Low-Income neighbourhoods correlated with " + crime)
    else:
            display_table(crime_max_low_income, title)
    print("")
```

Neighbourhoods with Highest Rates of Assault and also Lowest Income

Neighbourhood	Assault	Income
Oakridge	1,287	134,303
Beechborough-Greenbrook	1,066	109,880
Regent Park	1,025	103,250

Neighbourhoods with Highest Rates of Auto Theft and also Lowest Income

Neighbourhood	Auto Theft	Income
Elms-Old Rexdale	195	123,119

Neighbourhoods with Highest Rates of Break And Enter and also Lowest Income

Neighbourhood	Break And Enter	Income
Oakridge	471	134,303
Taylor-Massey	439	102,259

Neighbourhoods with Highest Rates of Theft Over $5k and also Lowest Income

Neighbourhood	Theft Over $5k	Income
Beechborough-Greenbrook	74	109,880

Neighbourhoods with Highest Rates of Homicide and also Lowest Income

Neighbourhood	Homicide	Income
Rustic	10	113,995
Weston-Pellam Park	9	127,491

Neighbourhoods with Highest Rates of Shootings and also Lowest Income

Neighbourhood	Shootings	Income
Oakridge	84	134,303
Rustic	69	113,995
Regent Park	45	103,250
Beechborough-Greenbrook	44	109,880
Elms-Old Rexdale	41	123,119
Etobicoke West Mall	33	132,460

Neighbourhoods with Highest Rates of Theftfrom Motor Vehicle and also Lowest Inc

Neighbourhood	Theftfrom Motor Vehicle	Income
Regent Park	539	103,250

Crime Rates in High-Income Neighbourhoods

And as promised, I repeated the exercise for the top 10% of high-income neighbourhoods.

In [ ]:
```
# Correlate with highest income neighbourhoods
for crime in crimes:

        # Extract the top 10% crime list for each category
        crime_max = crime_income_df.sort_values(crime, ascending=False)[["Neighbourhood", crime]].head(data_size)

        # Merge the crime data with the income data
        crime_max_high_income = crime_max.merge(income_max, on=["Neighbourhood"])
        crime_max_high_income.drop("Income 1000s", axis=1, inplace=True)

        # Round the crime rate
        crime_max_high_income[crime] = crime_max_high_income[crime].round()

        # Heading
        title="Neighbourhoods with Highest Rates of " + crime + " and also Highest Income"

        # Only print results if the category isn't empty
        if len(crime_max_high_income) != 0:
                display_table(crime_max_high_income, title)
```

```
        else:
                print("There are no High-Income neighbourhoods correlated with " + crime)
```

Neighbourhoods with Highest Rates of Assault and also Highest Income

Neighbourhood	Assault	Income
Waterfront Communities-The Island	1,247	662,333

Neighbourhoods with Highest Rates of Auto Theft and also Highest Income

Neighbourhood	Auto Theft	Income
Islington-City Centre West	229	854,623

Neighbourhoods with Highest Rates of Break And Enter and also Highest Income

Neighbourhood	Break And Enter	Income
St.Andrew-Windfields	431	649,291

Neighbourhoods with Highest Rates of Theft Over $5k and also Highest Income

Neighbourhood	Theft Over $5k	Income
Islington-City Centre West	102	854,623
St.Andrew-Windfields	82	649,291
Waterfront Communities-The Island	73	662,333
Annex	61	792,507
South Riverdale	60	662,651

```
There are no High-Income neighbourhoods correlated with Homicide
There are no High-Income neighbourhoods correlated with Shootings
```

Neighbourhoods with Highest Rates of Theftfrom Motor Vehicle and also Highest Inc

Neighbourhood	Theftfrom Motor Vehicle	Income
Annex	602	792,507

Correlation of Crime and Income

Crime vs Low Income Neighbourhoods

We could immediately see that each crime category was correlated with at least one neighbourhood from the low-income category. In fact when I counted them up 8 out of the 14 (57%) of the lowest-income neighbourhoods were represented here.

We could also see that low-income neighbourhoods disproportionately experienced the most shooting crime, with residents of the neighbourhood of **Rustic** being doubly at risk from both shooting and homicide (whether by gunshot or otherwise).

The correlation also showed that the most crime-prone neighbourhood was **Beechborough-Greenbrook**, appearing in three of the crime categories, the highest representation of the lowest-income neighbourhoods in this comparison.

Crime vs High Income Neighbourhoods

Although the highest-income neighbourhoods were not spared the effects of criminal activity in this comparison, the picture looked very different for a resident of one of these more affluent districts.

The first most obvious difference was that **none** of the highest-income neighbourhoods were represented in either the **Homicide** or **Shootings** categories. A resident of one of these neighbourhoods was statistically unlikely to be shot or killed, at least while they were close to home.

The only crime category that showed significant representation was the **Theft Over $5k** category, albeit with only 5 out of 14 highest-income neighbourhoods (35%) appearing in the list. I leave it up to your imagination or further exploration as to why this might be the case, but note that the only other two categories that appeared in this list were **Auto Theft** and **Break and Enter**. I should point out however, that the latter two were only minimally represented.

The two neighbourhoods that appeared most in this comparison were **Islington-City Centre West** and **St.Andrew-Windfields**, in **Auto Theft** and **Break and Enter** respectively, and both also appeared in **Theft Over $5k**. I imagine that these categories are correlated somehow (e.g theft of an automobile is surely valued in excess of $5k), but more specific crime data would be required to determine the relationship, if any. Regrettably, it didn't appear that Toronto Police Services published data at this level of specificity for public consumption (possibly due to privacy issues) and so I wasn't able to further explore the question.

Conclusions

From the examination of both the lowest- and highest-income households vs highest crime rates, it seemed clear that a resident of one of the lowest-income neighbourhoods had a 57% chance of living in an area that experienced some of the worst crime in the city, and in particular violent crime (i.e. Assault, Homicide, Shootings).

This stood in sharp contrast to residents of the highest-income neighbourhoods, of which only 37% were represented in any crime category, and were not at all represented in any of the violent crime categories. On the other hand, a resident of one of these neighbourhoods did have a somewhat higher chance of having their belongings misappropriated through unlawful means.

It was clear then, that there was a statistically significant correlation between the two factors, with lower-income neighbourhoods clearly experiencing more crime than higher-income neighbourhoods overall. A neighbourhood's geographical location in the city relative to others did not seem to be a significant factor.

And finally, some low-income neighbourhoods did not correlate prominently with those having the highest crime rates. This gap couldn't be explained in the limited scope of this investigation, and would most likely require a much broader set of data.

Caveats

Although I noted a correlation between a neighbourhood's household income and its crime rates, there were some gaps. Although more than half of the lowest-income neighbourhoods appeared in the highest crime charts, not all did. This indicates that there were other factors at play that were not captured by this analysis. That could be an avenue for further exploration, e.g. availability of education, job opportunities, other societal factors.

I'm also not passing any kind of moral judgement here, the data simply suggests there's a correlation and nothing more, i.e. if you live in a certain neighbourhood then statistically you can expect these outcomes. More talented people than I would have to explore the sociatal implications, and how to address them.