On the pattern of criminal activities in the city of Chicago

by Hoai Nam Nguyen

March 14, 2020

1 Introduction

The city of Chicago is the third largest city in the United States (after New York and Los Angeles) in terms of population. With a population of over 9 million people in its metropolitan area, Chicago is unsurprisingly one of the places where criminal activities are most abundant. According to the most recent survey, the rate of violent crime in Chicago is 1098.86 per 100,000 people per year. In particular, the rate of murder and non-negligent manslaughter is 24.13 per 100,000 people per year, making it a top 10 city in the US for this type of crime. The number of crimes committed has been steadily increasing for the past decade.

The city council of the city of Chicago is interested in whether there is a correlation between the types of crime that frequently occur in a community area and the popular venues that the area has to offer to its citizens. For example, in an area with many banks and jewelry stores, robbery and burglary may be popular types of crime to be committed. If there is such a correlation, we can predict the potential types of crime that may arise in a certain community area and, therefore, will have better solutions to prevent those from happening.

2 Data

2.1 Data sources

The Foursquare location data is very useful for this problem since we can retrieve the popular venues in a specific community area. In addition, we will make use of following data set https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2

This data set is publicly available and can be downloaded as a csv file. It contains all reported crimes in the city of Chicago from 2001 and is updated on a regular basis (all crimes are included except those in the last 7 days). According to this data set, there are over 7 million reported cases. For simplicity, we will only consider the cases that occur during and after 2015. This results in a subset with over 1 million cases. Each observation in the data set contains useful information about the case, including the primary type of crime, the community area in which it happened and the precise location in terms of longitude and latitude. Using this data set, we can deduce the longitude and latitude of a community area and, therefore, easily retrieve the list of venues for that area by making an API call to Foursquare.

2.2 Data cleaning

Due to the large size of the data set (over 7 million rows), I decide to read the data chunk by chunk, each of which has 1 million rows.
Then, I iterate through the chunks and append them one by one to an empty list. For each chunk,

- I filter out columns that do not play a role in the subsequent analysis. The most important columns to be considered are ID, Date, Primary Type, District, Ward, Community Area, Latitude and Longitude.
- Note that the data set contains NA entries. Since the number of such entries is negligible compared to the size of the data, I decide to remove rows that contains one or more NAs.
- As mentioned before, I only consider cases that occurred during or after 2015 due to limited computational power.

This procedure results in a data set with 8 columns and over 1 million observations. A small portion of the data set is shown below.

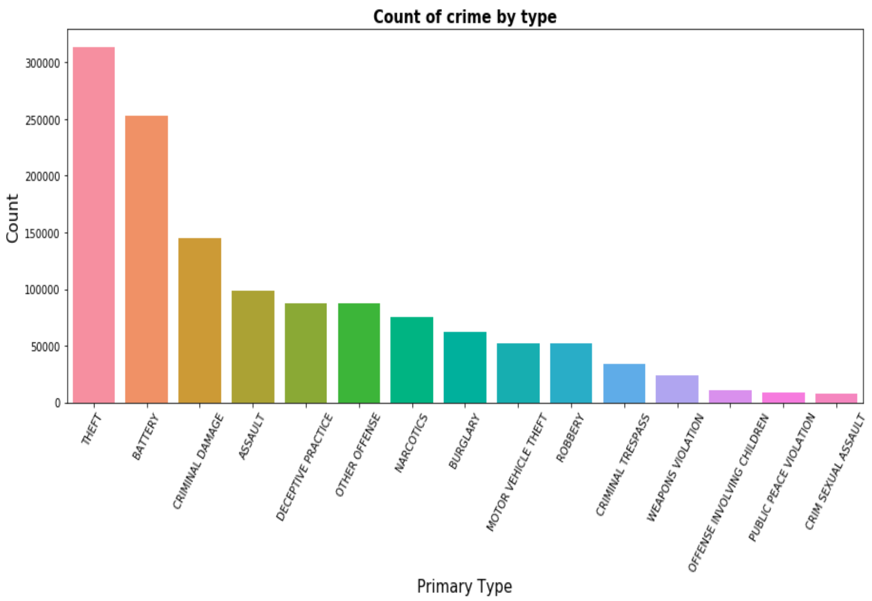
	ID	Date	Primary Type	District	Ward	Community Area	Latitude	Longitude
60329	11556037	2019-01-03 19:20:00	PUBLIC PEACE VIOLATION	16.0	41.0	76.0	42.002816	-87.906094
62255	11626027	2019-03-16 17:58:00	BATTERY	1.0	42.0	32.0	41.883369	-87.633860
62597	11622422	2019-03-12 22:00:00	THEFT	2.0	4.0	36.0	41.825347	-87.606781
62630	11625922	2019-03-14 18:42:00	BATTERY	24.0	49.0	1.0	42.016542	-87.672499
62631	11622907	2019-03-14 16:03:00	OTHER OFFENSE	2.0	4.0	36.0	41.825299	-87.606961

Finally, I convert the District, Ward and Community Area columns from float to integer.

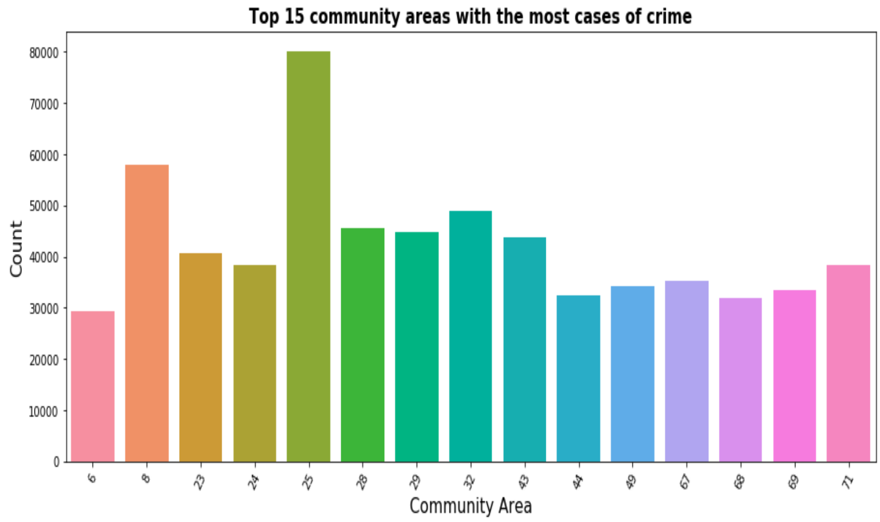
	ID	Date	Primary Type	District	Ward	Community Area	Latitude	Longitude
0	11556037	2019-01-03 19:20:00	PUBLIC PEACE VIOLATION	16	41	76	42.002816	-87.906094
1	11626027	2019-03-16 17:58:00	BATTERY	1	42	32	41.883369	-87.633860
2	11622422	2019-03-12 22:00:00	THEFT	2	4	36	41.825347	-87.606781
3	11625922	2019-03-14 18:42:00	BATTERY	24	49	1	42.016542	-87.672499
4	11622907	2019-03-14 16:03:00	OTHER OFFENSE	2	4	36	41.825299	-87.606961

3 Exploratory data analysis

A quick summary shows us the number of distinct districts, wards and community areas in the city of Chicago. There are 77 community areas and I will focus on these for the subsequent analysis. In addition, there are 33 distinct types of crime.
To explore the data set in more details, I plot a barchart that displays the frequencies of the top 15 types of crime.



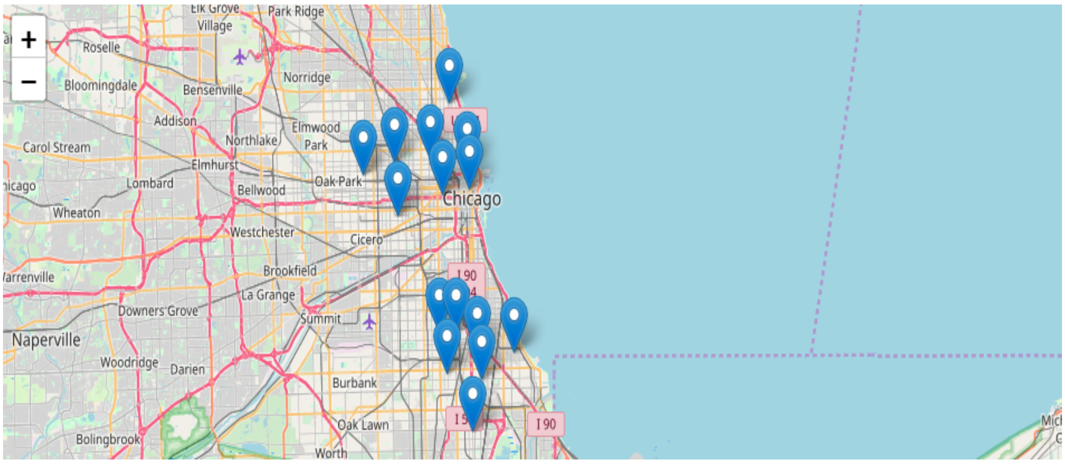
We can see that theft and battery are the most common types of crime by a large margin. Over the last 5 years, there are over 300,000 cases of theft and around 250,000 cases of battery in the city of Chicago.
Next, I group the data set by community areas and plot a barchart that displays the 15 areas with the most cases of crime.



It seems that the community area 25 has much more cases of crime compared to the rest.
Although I do not have a data set that contains the precise locations of the community areas, I can use the crime data set to estimate their latitudes and longitudes. Specifically, for each community area, I take average of the longitudes and the latitudes over all cases of crime that happened in that area. Given the size of the data set, this simple procedure will give me very good estimates. The table below gives the estimated latitude and longitude of the first 5 community areas.

Community area	Latitude	Longitude
1	42.012195	-87.670609
2	41.999862	-87.693029
3	41.966217	-87.656876
4	41.972274	-87.688504
5	41.947509	-87.682949

Using the *geopy* package, I find that the latitude and longitude of Chicago are 41.8755616 and -87.6244212 respectively. This information, in addition to the estmated locations of the community areas, allows me to visualize the 15 areas with the most cases of crime explicitly on a map as shown below.



It seems that these community areas are divided into 2 distinct clusters.
I then make API calls to Foursquare to retrieve the list of popular venues for each community area. In this case, I set the search radius to be 750 and I limit the result set to 100 venues per community area. In total, 2885 venues are retrieved, which are classified into 293 distinct categories. A small portion of the list is given below.

	Community Area	Area Latitude	Area Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	1	42.012195	-87.670609	Taqueria & Restaurant Cd. Hidalgo	42.011634	-87.674484	Mexican Restaurant
1	1	42.012195	-87.670609	El Famous Burrito	42.010421	-87.674204	Mexican Restaurant
2	1	42.012195	-87.670609	R Public House	42.016032	-87.668571	Sports Bar
3	1	42.012195	-87.670609	Taste Food & Wine	42.016086	-87.668488	Wine Shop
4	1	42.012195	-87.670609	Morse Fresh Market	42.008087	-87.667041	Grocery Store
5	1	42.012195	-87.670609	Bark Place	42.010080	-87.675223	Pet Store
6	1	42.012195	-87.670609	The Common Cup	42.007797	-87.667901	Coffee Shop
7	1	42.012195	-87.670609	Smack Dab	42.009291	-87.666201	Bakery
8	1	42.012195	-87.670609	Luzzat	42.015952	-87.668774	Indian Restaurant
9	1	42.012195	-87.670609	Romanian Kosher Sausage Co.	42.012765	-87.674692	Deli / Bodega

I am interested in the most popular venues in each area. In particular, I construct a table that displays the top 10 venues in each community area, ranking from the most common to the least common. The first 10 rows, which corresponds to the first 10 community areas, of the table is given below.

	Community Area	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	1	Mexican Restaurant	Sandwich Place	Park	Pizza Place	Bar	Train Station	Bakery	Chinese Restaurant	Coffee Shop	Donut Shop
1	2	Grocery Store	American Restaurant	Chinese Restaurant	Fast Food Restaurant	Fried Chicken Joint	Bar	Bus Station	Discount Store	Sandwich Place	Donut Shop
2	3	Coffee Shop	Pizza Place	Bar	Mexican Restaurant	Chinese Restaurant	Grocery Store	Fast Food Restaurant	Sandwich Place	Discount Store	Donut Shop
3	4	Bar	Sandwich Place	Café	Cosmetics Shop	Pizza Place	Bus Station	Mexican Restaurant	Italian Restaurant	Gym / Fitness Center	Donut Shop
4	5	Pizza Place	American Restaurant	Sandwich Place	Bar	Coffee Shop	Mexican Restaurant	Breakfast Spot	Bus Station	Bakery	Chinese Restaurant
5	6	Bakery	Sandwich Place	Coffee Shop	Pizza Place	Mexican Restaurant	Italian Restaurant	Bar	Café	Donut Shop	Park
6	7	Coffee Shop	Pizza Place	American Restaurant	Gym / Fitness Center	Bar	Italian Restaurant	Fried Chicken Joint	Sandwich Place	Breakfast Spot	Cosmetics Shop
7	8	American Restaurant	Coffee Shop	Italian Restaurant	Gym / Fitness Center	Breakfast Spot	Pizza Place	Bakery	Bar	Mexican Restaurant	Grocery Store
8	9	Italian Restaurant	Bar	Park	Mexican Restaurant	American Restaurant	Coffee Shop	Pizza Place	Bakery	Grocery Store	Bus Station
9	10	Park	Pizza Place	Bar	Donut Shop	Fast Food Restaurant	Bus Station	Café	Chinese Restaurant	Discount Store	American Restaurant

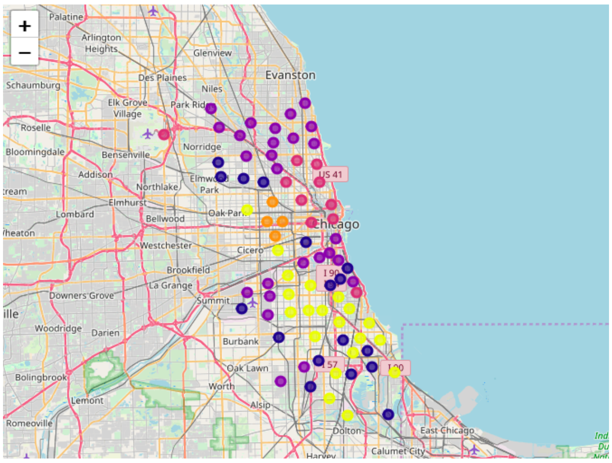
4 Modeling

4.1 K-means clustering

In this section, I will first try to group the community areas into distinct clusters according to the patterns of criminal activities that they exhibit.
For each community area, I calculate the percentage for each type of crime. That is, I count how many times a particular type of crime happened in that area and then divide the count by the total number of cases. As a result, the percentages in each row should sum to 1. This gives me a data frame with 77 rows and 34 columns. A small portion of the data frame is displayed below.

Community Area	ARSON	ASSAULT	BATTERY	BURGLARY	CONCEALED CARRY LICENSE VIOLATION	CRIM SEXUAL ASSAULT	CRIMINAL DAMAGE	CRIMINAL TRESPASS	DECEPTIVE PRACTICE	...	OTHER NARCOTIC VIOLATION	OTHER OFFENSE
1	0.000566	0.068413	0.188459	0.046175	0.000000	0.007825	0.125399	0.037424	0.062699	...	0.000051	0.059456
2	0.001109	0.065912	0.172573	0.065386	0.000175	0.005429	0.132349	0.027264	0.078055	...	0.000000	0.070057
3	0.000617	0.073823	0.193834	0.039798	0.000056	0.010146	0.086435	0.035762	0.102130	...	0.000056	0.057567
4	0.000627	0.058996	0.162786	0.058160	0.000209	0.009293	0.115067	0.029863	0.102851	...	0.000104	0.058996
5	0.001464	0.045248	0.093279	0.077317	0.000000	0.004979	0.110119	0.028555	0.118026	...	0.000000	0.044370
6	0.000918	0.041417	0.135779	0.055495	0.000170	0.008773	0.082427	0.033358	0.117077	...	0.000068	0.036181
7	0.001005	0.031331	0.087098	0.054213	0.000137	0.005526	0.100388	0.024389	0.102215	...	0.000000	0.026307
8	0.000449	0.041657	0.120675	0.017584	0.000155	0.007369	0.052546	0.030389	0.142487	...	0.000017	0.028370
9	0.002946	0.077320	0.183358	0.043446	0.000000	0.005155	0.134757	0.037555	0.134021	...	0.000000	0.092784
10	0.001831	0.071060	0.152272	0.065568	0.000000	0.004993	0.132967	0.037777	0.110501	...	0.000000	0.076552

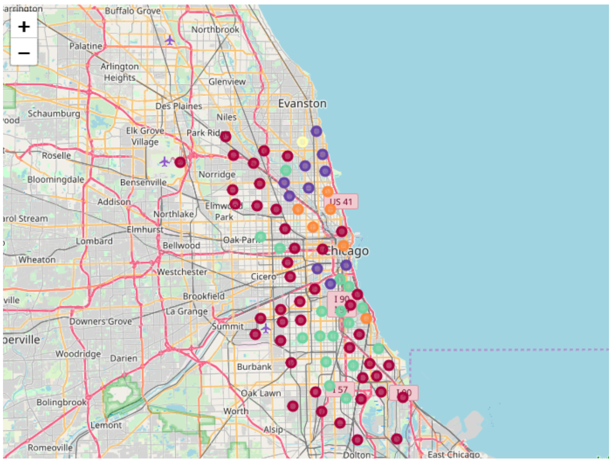
Using this result, I can build a clustering model using the K-means algorithm. In this case, I pick k = 5. I then combine the clustering result with the estimated locations of the community areas to produce the following visualization.



Next, I build another clustering model that is based on the types of venues that the community areas have to offer. As mentioned earlier, I made API calls to Foursquare to retrieve a list of popular venues in each community area. The search radius was 750 and the result set was limited to 100 venues per area. This resulted in a list of 2885 venues with 293 categories of venue. Since most categories of venue are under-represented, I decide to only include categories that appear more than 30 times across all community areas. After a few steps of pre-processing, I obtain a 77 x 25 data frame. Part of this data frame is displayed below.

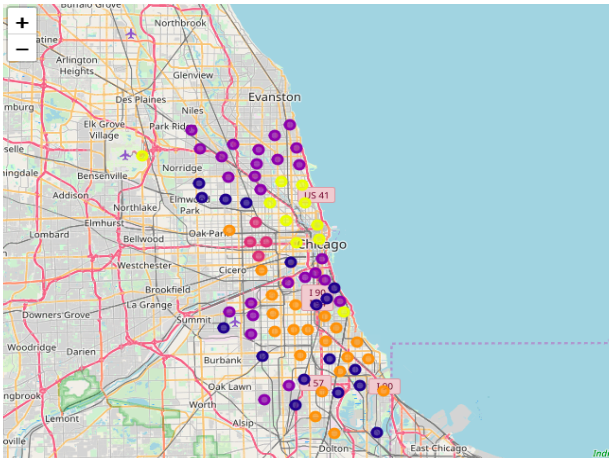
Community Area	American Restaurant	Bakery	Bar	Breakfast Spot	Bus Station	Café	Chinese Restaurant	Coffee Shop	Convenience Store	...	Grocery Store	Ice Cream Shop	Indian Restaurant	Italian Restaurant	Mexican Restaurant
1	2	2	3	1	1	1	3	2	0	...	2	0	1	0	7
2	2	0	1	0	0	0	0	0	2	...	4	0	21	0	0
3	1	0	4	1	0	1	3	7	1	...	2	0	0	0	3
4	1	0	5	0	4	3	1	1	1	...	1	1	1	2	2
5	3	1	3	2	4	1	1	3	2	...	1	0	0	1	2
6	0	4	2	1	0	2	0	4	0	...	1	2	1	2	3
7	3	1	3	2	0	1	0	6	0	...	1	2	0	2	1
8	6	2	2	3	0	1	0	3	0	...	1	0	0	3	1
9	1	1	3	0	1	0	0	0	0	...	1	0	0	4	1
10	1	0	2	0	1	1	1	0	0	...	0	0	0	0	1

Note that one of the columns is Community Area. In other words, there are only 24 categories of venue that appear more than 30 times across all 77 community areas.
I standardize the data frame above and build a K-means clustering model with k = 5. The clustering result is displayed in the map below.

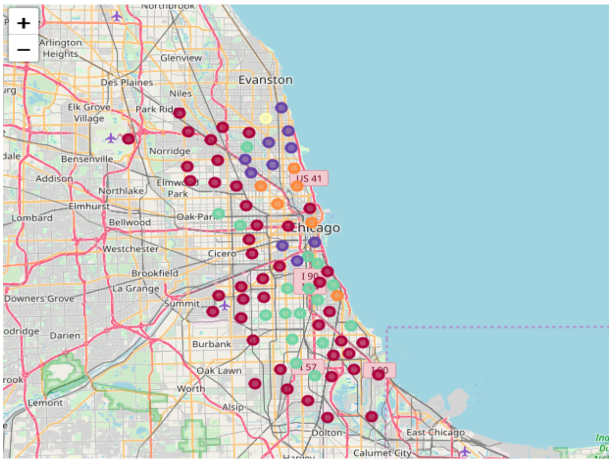


4.2 Hierarchical clustering

I repeat the same procedure, except that hierarchical (agglomerative) clustering is used instead of the K-means algorithm. In this case, I set the number of clusters to be 5. The clustering result based on criminal patterns is displayed in the following map.



Similarly, I create a visualization for the clustering result based on popular venues.



5 Results

In this section, I will perform chi-squared test of independence to see if there is a correlation between the clustering results. Specifically, I consider the following hypotheses:

H_0: Cluster labels based on crime and cluster labels based on venue are NOT associated.

H_a: Cluster labels based on crime and cluster labels based on venue are associated.

5.1 K-means clustering

The clustering results based on the K-means algorithm gives us the following two-way contingency table:

		Venue labels				
		label = 0	label = 1	label = 2	label = 3	label = 4
Crime labels	label = 0	0	6	0	4	10
	label = 1	1	8	0	4	4
	label = 2	9	3	1	10	3
	label = 3	2	0	7	1	0
	label = 4	0	3	0	0	1

The expected frequencies can be calculated according to the following formula:

$$E_{ij} = \frac{\sum_i O_{ij} \sum_j O_{ij}}{\sum_i \sum_i O_{ij}}$$

where O_{ij} are the observed frequencies in the contingency table below. In other words,

$$E_{ij} = \frac{\text{(sum of row i)}\times\text{(sum of column j)}}{\text{table total}}$$

In this particular case, the table total is 77 because there are 77 community areas. Using this formula, the expected frequencies are given by:

		Venue labels				
		label = 0	label = 1	label = 2	label = 3	label = 4
Crime labels	label = 0	3.12	5.19	2.08	4.94	4.68
	label = 1	2.65	4.42	1.77	4.19	3.97
	label = 2	4.05	6.75	2.70	6.42	6.08
	label = 3	1.56	2.60	1.04	2.47	2.34
	label = 4	0.62	1.04	0.42	0.99	0.94

The chi-squared test statistics is calculated as follow:

$$\chi^2 = \sum_i \sum_j \frac{(E_{ij} - O_{ij})^2}{E_{ij}}$$

In this case, we have that $\chi^2 = 75.9$.
The degree of freedom is given by (#rows - 1)(#columns - 1) = $(5-1)(5-1) = 16$. Therefore, under the null hypothesis, we have that $\chi^2 \sim \chi^2_{16}$. This leads to a p-value of 9×10^{-10}. Since the p-value is much smaller than any reasonable significance level (usually 5%), we reject the null hypothesis and conclude that there is a correlation between crime labels and venue labels.

5.2 Hierarchical clustering

I repeat the same thing using the clustering results from hierarchical clustering. The contingency table is given by:

		Venue labels				
		label = 0	label = 1	label = 2	label = 3	label = 4
Crime labels	label = 0	8	1	0	0	1
	label = 1	0	14	1	2	0
	label = 2	3	13	2	2	6
	label = 3	0	3	0	1	0
	label = 4	0	12	1	6	1

Then, the expected frequencies are given below:

		Venue labels				
		label = 0	label = 1	label = 2	label = 3	label = 4
Crime labels	label = 0	1.43	5.58	0.52	1.43	1.04
	label = 1	2.43	9.49	0.88	2.43	1.77
	label = 2	3.71	14.52	1.35	3.71	2.7
	label = 3	0.57	2.23	0.21	0.57	0.42
	label = 4	2.86	11.17	1.04	2.86	2.08

The test statistic is 56.51 and the degree of freedom is still 16. This leads to a p-value of 2.0036×10^{-6}. As a result, we also reject the null hypothesis H_0.

6 Discussion

In this project, I have investigated the relationship between the pattern of criminal activities and the popular venues in a community area in the city of Chicago. The main question was to know whether such a relationship existed.

To answer this question, I first built a clustering model based on incidents of crime that had occurred during and after 2015. Then, I constructed another clustering model based on the popular venues that a community area had to offer. Both K-means and agglomerative algorithms were used. By conducting a chi-squared test of independence, I arrived at the conclusion that the relationship existed.

In both K-means and hierarchical models, the number of clusters was k = 5. There are, however, methods that we can use to help select a value for k (e.g the Elbow method). We may also try a non-parametric clustering algorithm (e.g DBSCAN) to avoid the need to specify the number of clusters. In addition, there are a few other cities (e.g. San Francisco) that also make crime data publicly available. We can conduct the same study on these cities to see if the observed relationship in Chicago is a widespread phenomenon.

7 Conclusion

From the study, given a community area, we can look at the types of venue and predict something useful about criminal activities in this area. This allows the police to be better prepared to cope with future cases of crime.

Appendix

```python
import numpy as np
import pandas as pd

from scipy import stats

import json

from geopy.geocoders import Nominatim

import requests

from pandas.io.json import json_normalize

import matplotlib.cm as cm
import matplotlib.colors as colors
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import StandardScaler

import folium

import datetime as dt
```

```python
chicago_chunk = pd.read_csv("Chicago_Crime.csv", chunksize =
    1000000)
```

```python
chunk_list = []
features = ['ID', 'Date', 'Primary Type', 'District', 'Ward',
     'Community Area', 'Latitude', 'Longitude']

for chunk in chicago_chunk:
    chunk_filter = chunk.loc[:,features]
    chunk_filter.dropna(inplace = True)
    chunk_filter['Date'] = pd.to_datetime(chunk_filter['Date'
    ], format = '%m/%d/%Y %I:%M:%S %p')
    chunk_filter = chunk_filter.loc[chunk_filter['Date'] >= '
    01/01/2015',:]
    chunk_list.append(chunk_filter)

chicago_data = pd.concat(chunk_list)
```

```python
print(chicago_data.shape)
chicago_data.head()
```

```
chicago_data.reset_index(drop = True, inplace = True)
chicago_data['District'] = chicago_data['District'].astype('int32')
chicago_data['Ward'] = chicago_data['Ward'].astype('int32')
chicago_data['Community Area'] = chicago_data['Community Area'].astype('int32')
chicago_data.head()
```

```
print('There are', len(chicago_data['District'].unique()), 'districts in Chicago')
print('There are', len(chicago_data['Ward'].unique()), 'wards in Chicago')
print('There are', len(chicago_data['Community Area'].unique()), 'community areas in Chicago')
print('There are', len(chicago_data['Primary Type'].unique()), 'types of crime in Chicago')
```

```
crime_count = pd.DataFrame(chicago_data['Primary Type'].value_counts()).reset_index()
crime_count.columns = ['Primary Type', 'Count']

plt.figure(figsize = (15,6))
ax = sns.barplot(x = 'Primary Type', y = 'Count', data = crime_count.head(15))
ax.set_xticklabels(ax.get_xticklabels(), rotation = 60)
ax.set_xlabel(xlabel = 'Primary Type', fontsize = 15)
ax.set_ylabel(ylabel = 'Count', fontsize = 15)
ax.set_title('Count of crime by type', fontdict = {'fontweight':'bold', 'fontsize':15})
plt.show()
```

```
community_area_count = pd.DataFrame(chicago_data['Community Area'].value_counts()).reset_index()
community_area_count.columns = ['Community Area', 'Count']

plt.figure(figsize = (15,6))
ax = sns.barplot(x = 'Community Area', y = 'Count', data = community_area_count.head(15))
ax.set_xticklabels(ax.get_xticklabels(), rotation = 60)
ax.set_xlabel(xlabel = 'Community Area', fontsize = 15)
ax.set_ylabel(ylabel = 'Count', fontsize = 15)
ax.set_title('Top 15 community areas with the most cases of crime', fontdict = {'fontweight':'bold', 'fontsize':15})
plt.show()
```

```
chicago_grouped = chicago_data.groupby('Community Area')

community_area_loc = pd.DataFrame(chicago_grouped['Latitude', 'Longitude'].mean())
```

```
community_area_loc.reset_index(inplace = True)
community_area_loc.head(10)
```

```
community_area_merged = community_area_count.join(community_area_loc.set_index('Community Area'),
                                                  on = 'Community Area')
community_area_merged.head(15)
```

```
address = 'Chicago, IL'

geolocator = Nominatim(user_agent = "chicago_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude

print('The geographical coordinates of Chicago are {}, {}.'.format(latitude, longitude))
```

```
map_clusters = folium.Map(location = [latitude, longitude], zoom_start = 10)

for lat, lon, com, count in zip(community_area_merged.head(15)['Latitude'],
                                community_area_merged.head(15)['Longitude'],
                                community_area_merged.head(15)['Community Area'],
                                community_area_merged.head(15)['Count']):
     folium.Marker([lat, lon],
                  popup = 'Community area: ' + str(com) + '; Number of cases: ' + str(count)).add_to(map_clusters)

map_clusters
```

```
crime_proportion = chicago_grouped['Primary Type'].value_counts()/chicago_grouped['Primary Type'].count()

chicago_crime_prop = pd.DataFrame(crime_proportion)
chicago_crime_prop.columns = ['Count']
chicago_crime_prop.reset_index(inplace = True)
chicago_crime_prop = chicago_crime_prop.pivot_table(index = 'Community Area', columns = 'Primary Type')
chicago_crime_prop.columns = chicago_crime_prop.columns.droplevel()
chicago_crime_prop.fillna(0, inplace = True)
chicago_crime_prop.reset_index(inplace = True)

```

```
print(chicago_crime_prop.shape)
chicago_crime_prop.head(10)
```

```
kclusters = 5

chicago_crime_cluster = chicago_crime_prop.drop('Community
    Area', axis = 1)

kmeans_crime = KMeans(n_clusters = kclusters, random_state =
    0).fit(chicago_crime_cluster)

crime_labels = kmeans_crime.labels_
crime_labels[0:10]
```

```
chicago_crime_prop['Cluster Labels'] = crime_labels

chicago_merged = community_area_loc
chicago_merged = chicago_merged.join(chicago_crime_prop.
    set_index('Community Area'), on = 'Community Area')
chicago_merged.head(10)
```

```
map_clusters = folium.Map(location = [latitude, longitude],
    zoom_start = 10)

x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.plasma(np.linspace(0, 1, len(ys)))
plasma = [colors.rgb2hex(i) for i in colors_array]

markers_colors = []
for lat, lon, com, cluster in zip(chicago_merged['Latitude'],
     chicago_merged['Longitude'],
                                  chicago_merged['Community
    Area'], chicago_merged['Cluster Labels']):
    label = folium.Popup(str(com) + ' Cluster ' + str(cluster
    ), parse_html = True)
    folium.CircleMarker(
        [lat, lon],
        radius = 5,
        popup = label,
        color = plasma[cluster - 1],
        fill = True,
        fill_color = plasma[cluster - 1],
        fill_opacity = 0.7).add_to(map_clusters)

map_clusters
```

```
CLIENT_ID = 'AADZNFJL102ZBDR5UT4GCADHPFMT0GURPVQ5TFKTMPCHJZBV
    '
```

```python
CLIENT_SECRET = '04202ITCIA3HYLNLGU5NRPDX1WU55CUAIZELH5P0SQEIXSXP'
VERSION = '20180605'

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```python
LIMIT = 100

def getNearbyVenues(areas, latitudes, longitudes, radius = 750):

    venues_list = []
    for area, lat, lng in zip(areas, latitudes, longitudes):
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        results = requests.get(url).json()["response"]['groups'][0]['items']

        venues_list.append([(
            area,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['Community Area',
                  'Area Latitude',
                  'Area Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

```

```
    return(nearby_venues)
```

```
chicago_venues = getNearbyVenues(areas = community_area_loc['Community Area'],
                                 latitudes = community_area_loc['Latitude'],
                                 longitudes = community_area_loc['Longitude'])
```

```
print(chicago_venues.shape)
chicago_venues.head(10)
```

```
chicago_dummy = pd.get_dummies(chicago_venues[['Venue Category']], prefix = "", prefix_sep = "")
chicago_dummy = pd.concat([chicago_venues['Community Area'], chicago_dummy], axis = 1)
print(chicago_dummy.shape)
chicago_dummy.head(100)
```

```
chicago_dummy_grouped = chicago_dummy.groupby('Community Area').sum().reset_index()

total_venues = pd.DataFrame(chicago_dummy_grouped.iloc[:,1:].sum(axis = 0) > 30).reset_index()
total_venues.columns = ['Venue','Popular']
unpopular_venues = total_venues[total_venues['Popular'] == False]['Venue'].to_list()

chicago_dummy_grouped.drop(unpopular_venues, axis = 1, inplace = True)
print(chicago_dummy_grouped.shape)
chicago_dummy_grouped.head(10)
```

```
def most_common(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending = False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

```
num_top_venues = 10
indicators = ['st', 'nd', 'rd']
columns = ['Community Area']

for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind + 1, indicators[ind]))
```

```
    except:
        columns.append('{}th Most Common Venue'.format(ind + 1))

area_venues_sorted = pd.DataFrame(columns = columns)
area_venues_sorted['Community Area'] = chicago_dummy_grouped['Community Area']

for ind in np.arange(chicago_dummy_grouped.shape[0]):
    area_venues_sorted.iloc[ind, 1:] = most_common(chicago_dummy_grouped.iloc[ind, :], num_top_venues)

area_venues_sorted.head(10)
```

```
kclusters = 5

chicago_venue_cluster = chicago_dummy_grouped.drop('Community Area', axis = 1)
chicago_venue_cluster = StandardScaler().fit_transform(chicago_venue_cluster)

kmeans_venue = KMeans(n_clusters = kclusters, random_state = 0).fit(chicago_venue_cluster)

venue_labels = kmeans_venue.labels_
venue_labels[0:10]
```

```
chicago_dummy_grouped['Cluster Labels'] = venue_labels

chicago_venue_merged = community_area_loc
chicago_venue_merged = chicago_venue_merged.join(chicago_dummy_grouped.set_index('Community Area'), on = 'Community Area')
chicago_venue_merged.head(10)
```

```
map_venue_clusters = folium.Map(location = [latitude, longitude], zoom_start = 10)

x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.Spectral(np.linspace(0, 1, len(ys)))
spectral = [colors.rgb2hex(i) for i in colors_array]

markers_colors = []
for lat, lon, com, cluster in zip(chicago_venue_merged['Latitude'], chicago_venue_merged['Longitude'],
                                  chicago_venue_merged['Community Area'], chicago_venue_merged['Cluster Labels']):
    label = folium.Popup(str(com) + ' Cluster ' + str(cluster), parse_html = True)
```

```
    folium.CircleMarker(
        [lat, lon],
        radius = 5,
        popup = label,
        color = spectral[cluster - 1],
        fill = True,
        fill_color = spectral[cluster - 1],
        fill_opacity = 0.7).add_to(map_venue_clusters)

map_venue_clusters
```

```
contingency_table = np.zeros((5,5))
for i in np.arange(0,5):
    for j in np.arange(0,5):
        contingency_table[i,j] = np.sum((crime_labels == i) & (venue_labels == j))
contingency_table = contingency_table.astype('int32')
print('The observed contingency table is given by\n\n', contingency_table,'\n')
chi2, p, dof, ex = stats.chi2_contingency(contingency_table, correction = False)
print('The expected frequencies are given in the following table\n\n', np.round(ex,2), '\n')
print('The test statistic is', np.round(chi2,2))
print('The degree of freedom is', dof)
print('The p-value is', np.round(p,10))
```

```
n_clusters = 5

agglom = AgglomerativeClustering(n_clusters, linkage = 'ward')
chicago_crime_agglom = agglom.fit(chicago_crime_cluster)

crime_labels = chicago_crime_agglom.labels_
crime_labels[0:10]
```

```
chicago_crime_prop.drop('Cluster Labels', axis = 1, inplace = True)
chicago_crime_prop['Cluster Labels'] = crime_labels

chicago_merged = community_area_loc
chicago_merged = chicago_merged.join(chicago_crime_prop.set_index('Community Area'), on = 'Community Area')
chicago_merged.head(10)
```

```
map_clusters = folium.Map(location = [latitude, longitude], zoom_start = 10)

x = np.arange(n_clusters)
```

```
ys = [i + x + (i*x)**2 for i in range(n_clusters)]
colors_array = cm.plasma(np.linspace(0, 1, len(ys)))
plasma = [colors.rgb2hex(i) for i in colors_array]

markers_colors = []
for lat, lon, com, cluster in zip(chicago_merged['Latitude'],
     chicago_merged['Longitude'],
                                    chicago_merged['Community
    Area'], chicago_merged['Cluster Labels']):
     label = folium.Popup(str(com) + ' Cluster ' + str(cluster
    ), parse_html = True)
     folium.CircleMarker(
         [lat, lon],
         radius = 5,
         popup = label,
         color = plasma[cluster - 1],
         fill = True,
         fill_color = plasma[cluster - 1],
         fill_opacity = 0.7).add_to(map_clusters)

map_clusters
```

```
n_clusters = 5

agglom = AgglomerativeClustering(n_clusters, linkage = 'ward'
    )
chicago_venue_agglom = agglom.fit(chicago_venue_cluster)

venue_labels = chicago_venue_agglom.labels_
venue_labels[0:10]
```

```
chicago_dummy_grouped.drop('Cluster Labels', axis = 1,
    inplace = True)
chicago_dummy_grouped['Cluster Labels'] = venue_labels

chicago_venue_merged = community_area_loc
chicago_venue_merged = chicago_venue_merged.join(
    chicago_dummy_grouped.set_index('Community Area'), on = '
    Community Area')
chicago_venue_merged.head(10)
```

```
map_venue_clusters = folium.Map(location = [latitude,
    longitude], zoom_start = 10)

x = np.arange(n_clusters)
ys = [i + x + (i*x)**2 for i in range(n_clusters)]
colors_array = cm.Spectral(np.linspace(0, 1, len(ys)))
spectral = [colors.rgb2hex(i) for i in colors_array]

```

```
markers_colors = []
for lat, lon, com, cluster in zip(chicago_venue_merged['
    Latitude'], chicago_venue_merged['Longitude'],
                                    chicago_venue_merged['
    Community Area'], chicago_venue_merged['Cluster Labels']):
    label = folium.Popup(str(com) + ' Cluster ' + str(cluster
    ), parse_html = True)
    folium.CircleMarker(
        [lat, lon],
        radius = 5,
        popup = label,
        color = spectral[cluster - 1],
        fill = True,
        fill_color = spectral[cluster - 1],
        fill_opacity = 0.7).add_to(map_venue_clusters)

map_venue_clusters
```

```
contingency_table = np.zeros((5,5))
for i in np.arange(0,5):
    for j in np.arange(0,5):
        contingency_table[i,j] = np.sum((crime_labels == i) &
     (venue_labels == j))
contingency_table = contingency_table.astype('int32')
print('The observed contingency table is given by\n\n',
    contingency_table,'\n')
chi2, p, dof, ex = stats.chi2_contingency(contingency_table,
    correction = False)
print('The expected frequencies are given in the following
    table\n\n', np.round(ex,2), '\n')
print('The test statistic is', np.round(chi2,2))
print('The degree of freedom is', dof)
print('The p-value is', np.round(p,10))
```