plots_and_analysis

May 22, 2018

1 Plots

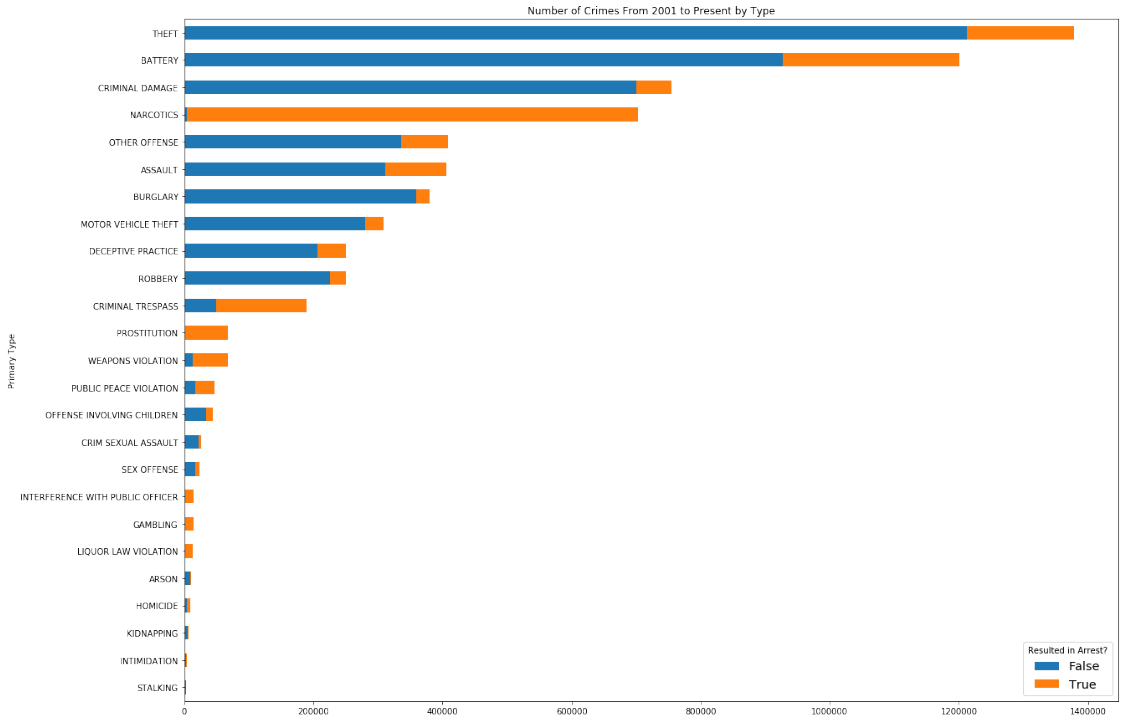
```
In [42]: import pandas as pd
         import geopandas as gpd
         import seaborn as sns
         import numpy as np
         import matplotlib.pyplot as plt
         from ipywidgets import widgets
```

```
In [3]: df = pd.read_csv("Crimes_-_2001_to_present.csv")
```

1.0.1 Distributions by Crime Type and their Descriptions

Let's first look at how crimes are distributed by type, and how each frequently each time of crime led to an actual arrest. We only look at the 25 most common types to avoid overcrowding.

```
In [4]: # helper function for sorting grouped dataframes by totals
        def sort_total(df):
                ''' takes in grouped dataframe (divided into True and False columns), returns
                return df.assign(total = lambda x: x[True] + x[False]).sort_values("total", asc


        # group by primary crime type and whether or not an arrest was made, get count with si
        type_counts = df.groupby(["Primary Type", "Arrest"]).size().unstack()
        type_counts = sort_total(type_counts)

        fig = plt.figure()
        fig.set_size_inches(20, 15)

        crime_types = fig.add_subplot(111)
        # get first 25, reverse df so that most common crime is first in plot
        type_counts.head(25).iloc[::-1].plot.barh(ax = crime_types, stacked = True, legend = Tr
        crime_types.legend(loc = "lower right", fontsize = "x-large", title = "Resulted in Arre
        plt.title("Number of Crimes From 2001 to Present by Type")
        fig.savefig("crime_types.png")
        plt.show()
```



Let's make this interactive by year to get a sense of how this changes over time.

```
In [8]: # slider for choosing what years to represent
        year_interval = widgets.IntRangeSlider(
            value = [2001, 2017],
            min = 2001,
            max = 2017,
            step = 1,
            description = "Years:",
            disabled = False,
            continuous_update = False,
            orientation = 'horizontal',
            #readout = True,
        )

        # function similar to the above that allows for a variable to choose given year interv
        def plot_dist_by_year(year_interval):
            # get df restricted to just that year interval
            (start, end) = year_interval
            interval = [year for year in range(start, end+1)]

            df_years = df[df.Year.isin(interval)]

            # group by primary crime type and whether or not an arrest was made, get count wit
            type_counts = df_years.groupby(["Primary Type", "Arrest"]).size().unstack()
```

```
        type_counts = sort_total(type_counts)

        fig = plt.figure()
        fig.set_size_inches(20, 15)

        crime_types = fig.add_subplot(111)
        # get first 25, reverse df so that most common crime is first in plot
        type_counts.head(25).iloc[::-1].plot.barh(ax = crime_types, stacked = True, legend
        crime_types.legend(loc = "lower right", fontsize = "x-large", title = "Resulted in

        if start == end:
            plt.title("Number of Crimes in {} by Type".format(start))
        else:
            plt.title("Number of Crimes From {} to {} by Type".format(start, end))
        fig.savefig("crime_types.png")
        plt.show()

    widgets.interact(plot_dist_by_year, year_interval = year_interval)
```

```
interactive(children=(IntRangeSlider(value=(2001, 2017), continuous_update=False, description=
```

```
Out[8]: <function __main__.plot_dist_by_year>
```

Beyond the obvious ordering that this graph provides in crimes, what immediately stands out is how nearly every narcotic crime resulted in an arrest. Let's look further into how just narcotic crime is distributed by description, and compare that with the distributions other types of crimes, like assault and burglarly, that seem as though they should have relatively higher arrest rates.

```
In [9]: # helper function for plotting
        def plot_description_dist_by_type(df, year_interval, primary_type):
            ''' given df and type restriction, plots distribution of crime and arrests by desc
            (start, end) = year_interval
            interval = [year for year in range(start, end+1)]

            crimes = df[df.Year.isin(interval)]

            # get counts of just narcotic crime, grouped into description, split into arrest
            crimes = crimes[crimes["Primary Type"] == primary_type].groupby(["Description", "A

            # sort narcotic crime by totals
            crimes = sort_total(crimes)

            fig2 = plt.figure()
            fig2.set_size_inches(20, 15)

            crime_description = fig2.add_subplot(111)

            crimes.head(25).iloc[::-1].plot.barh(ax = crime_description, stacked = True, legend
```

```
            crime_description.legend(loc = "lower right", fontsize = "x-large", title = "Resul

            if start == end:
                    plt.title("Number of {} Crimes in {} by Description".format(primary_type.ti
            else:
                plt.title("Number of {} Crimes from {} to {} by Description".format(primary_typ
            plt.show()
        # slider for choosing what years to represent
        year_interval = widgets.IntRangeSlider(
            value = [2001, 2017],
            min = 2001,
            max = 2017,
            step = 1,
            description = "Years:",
            disabled = False,
            continuous_update = False,
            orientation = 'horizontal',
            #readout = True,
        )

        types = list(df["Primary Type"].unique())

        type_dict = dict(zip([x.title() for x in types], types))

        primary_type = widgets.Dropdown(
            options = type_dict,
            value = "NARCOTICS",
            description = "Primary Type",
            disabled = False,
        )
        widgets.interact(plot_description_dist_by_type,
                         df = widgets.fixed(df),
                         year_interval = year_interval,
                         primary_type = primary_type)
```

```
interactive(children=(IntRangeSlider(value=(2001, 2017), continuous_update=False, description=
```

```
Out[9]: <function __main__.plot_description_dist_by_type>
```

```
In [10]: widgets.interact(plot_description_dist_by_type,
                          df = widgets.fixed(df),
                          year_interval = year_interval,
                          primary_type = primary_type)
```

```
interactive(children=(IntRangeSlider(value=(2001, 2017), continuous_update=False, description=
```

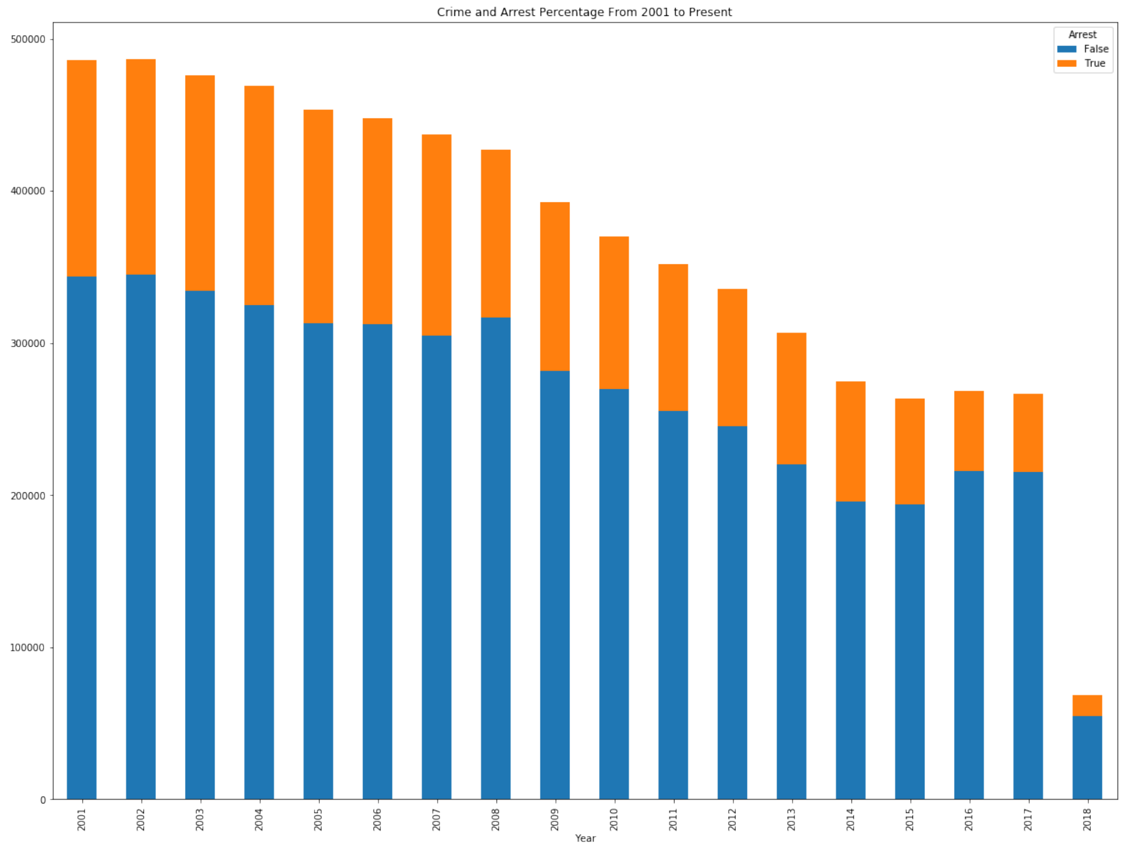
```
Out[10]: <function __main__.plot_description_dist_by_type>
```

```
In [11]: # helper function for plotting
         def plot_description_dist_by_type(df, year_interval, primary_type):
             ''' given df and type restriction, plots distribution of crime and arrests by des
             # get counts of just narcotic crime, grouped into description, split into arrest
             crimes = df[df["Primary Type"] == primary_type].groupby(["Description", "Arrest"]

             # sort narcotic crime by totals
             crimes = sort_total(crimes)

             fig2 = plt.figure()
             fig2.set_size_inches(20, 15)

             crime_description = fig2.add_subplot(111)

             crimes.head(25).iloc[::-1].plot.barh(ax = crime_description, stacked = True, legen
             crime_description.legend(loc = "lower right", fontsize = "x-large", title = "Resul
             plt.title("Number of {} Crimes from 2001 to Present by Description".format(primar
             plt.show()
```

1.1 Trends of Crime Over Time

Let's with how crime has changed over the years, starting with just raw crime numbers.

```
In [14]: arrest_by_year = df.groupby(["Year", "Arrest"]).size().unstack()
         fig = plt.figure()
         fig.set_size_inches(20, 15)

         by_year_plot = fig.add_subplot(111)

         arrest_by_year.plot.bar(ax = by_year_plot, stacked = True)
         plt.title("Crime and Arrest Percentage From 2001 to Present")
         plt.show()
```

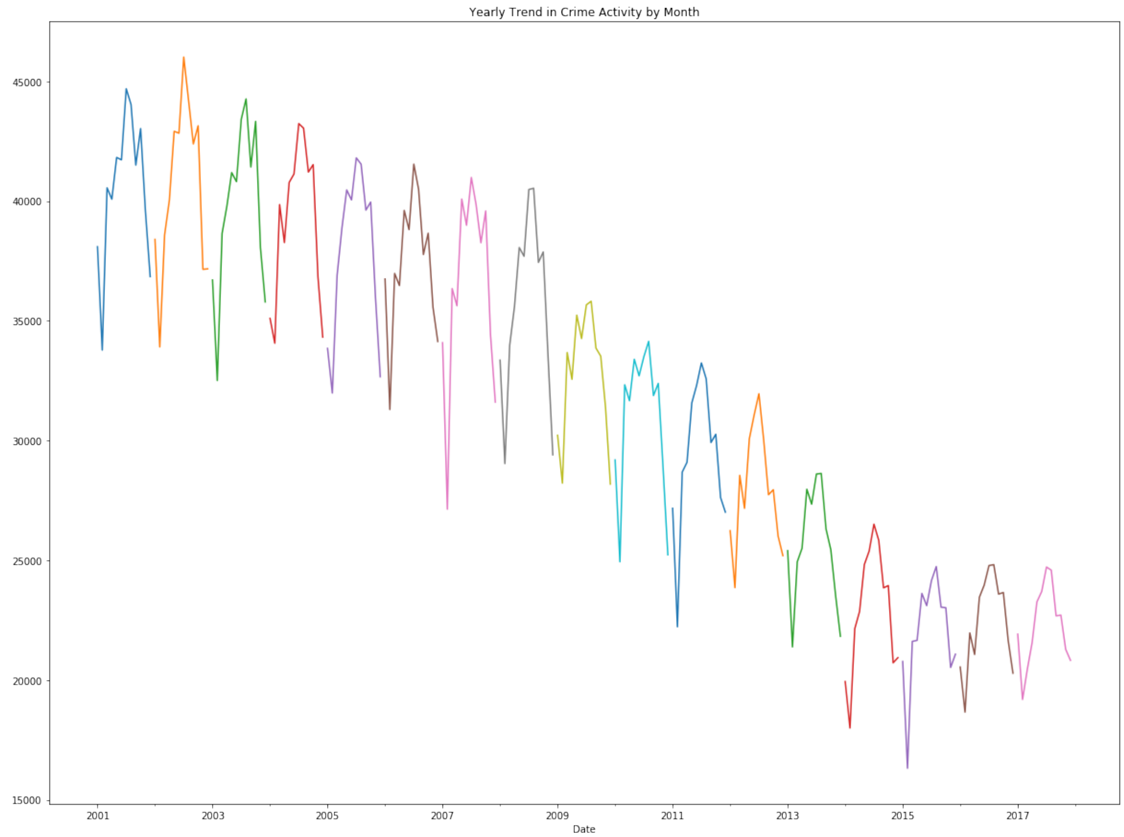


What immediately stands out is the downward trend in crime. More interestingly, while the number of crimes stayed relatively constant from 2014 to 2017 (note that this is a daily updated dataset, so 2018 is still incomplete), there was still a lower number of arrests made for those crimes.

Let's look at every year's distribution of crime per month.

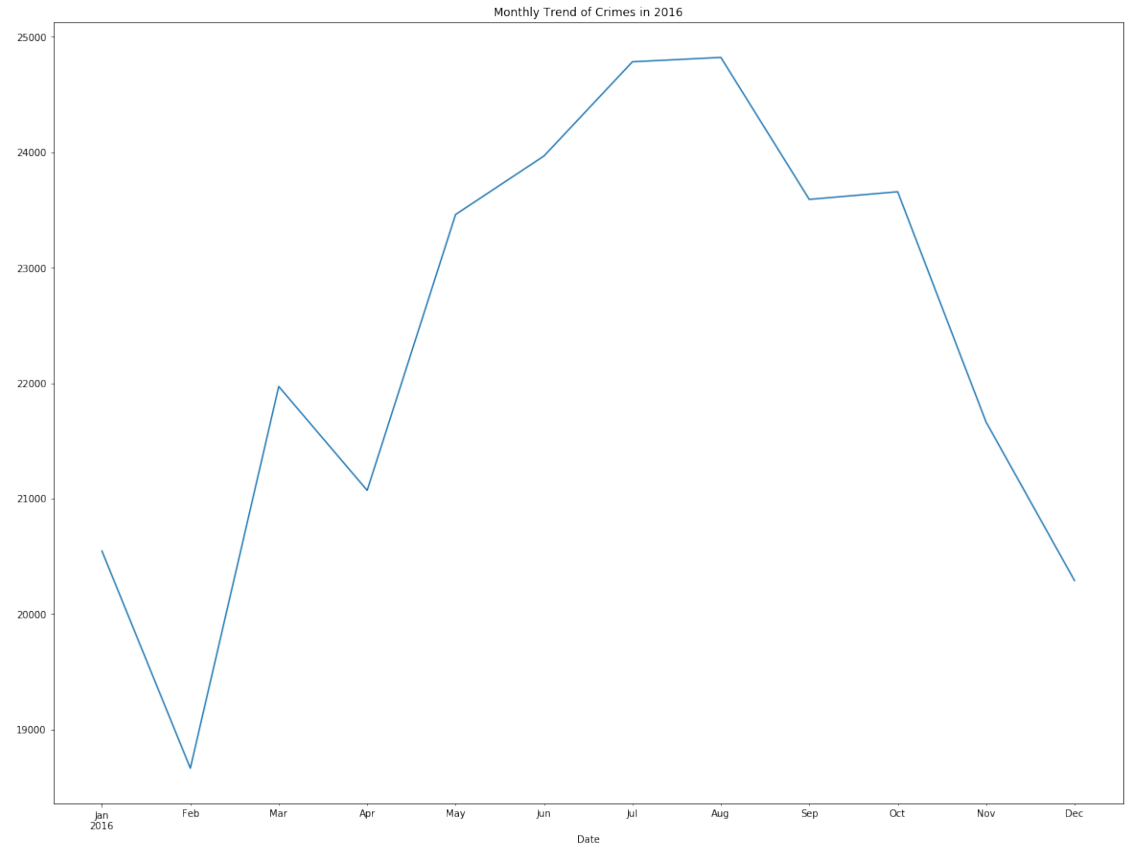
```
In [18]: df.Date = pd.to_datetime(df.Date)
```

```
In [19]: fig = plt.figure()
         fig.set_size_inches(20, 15)

         month_trends = fig.add_subplot(111)
         for year in range(2001, 2018):
             df[df["Year"] == year].set_index("Date").resample("MS").size().plot(ax = month_tre

         plt.title("Yearly Trend in Crime Activity by Month")
         plt.show()
```



Since years are seperated by colors, we can clearly see that despite the overall downward trend of crime, there is a consistent pattern of crime within each year, with each of the spikes occuring during summer months. We illustrate that more clearly by showing just 2016's plot below.

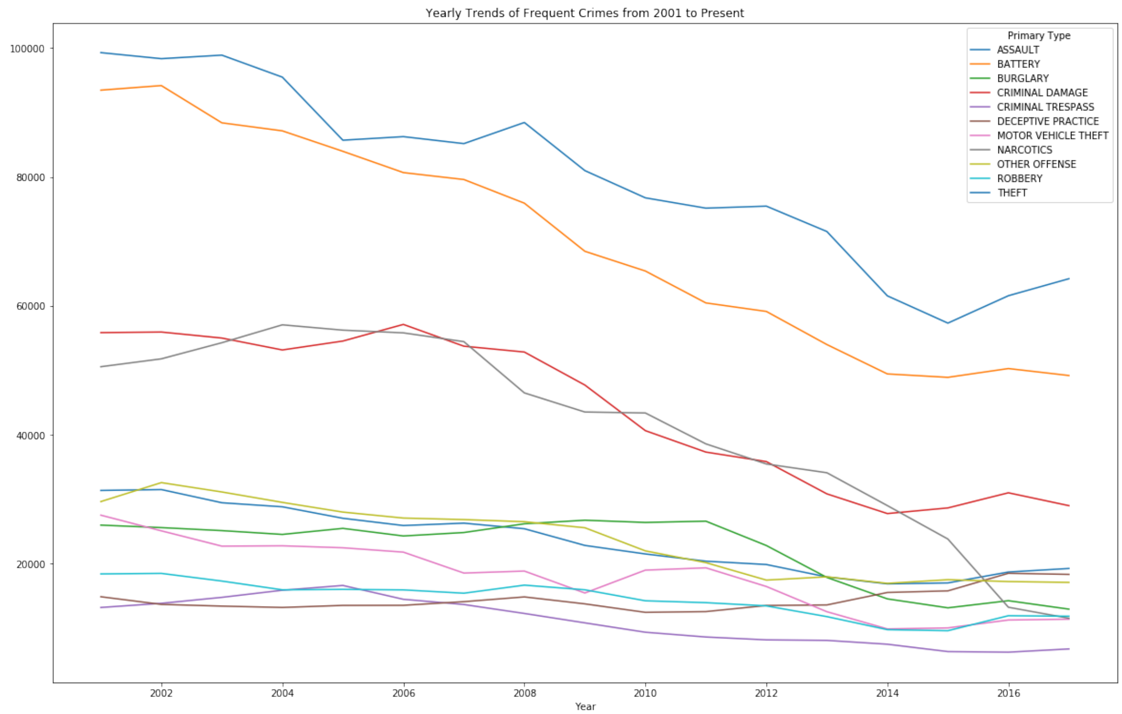
```
In [20]: fig = plt.figure()
         fig.set_size_inches(20, 15)
         crimes2016 = fig.add_subplot(111)
         df[df["Year"] == 2016].set_index("Date").resample("MS").size().plot(ax = crimes2016)

         plt.title("Monthly Trend of Crimes in 2016")
         plt.show()
```

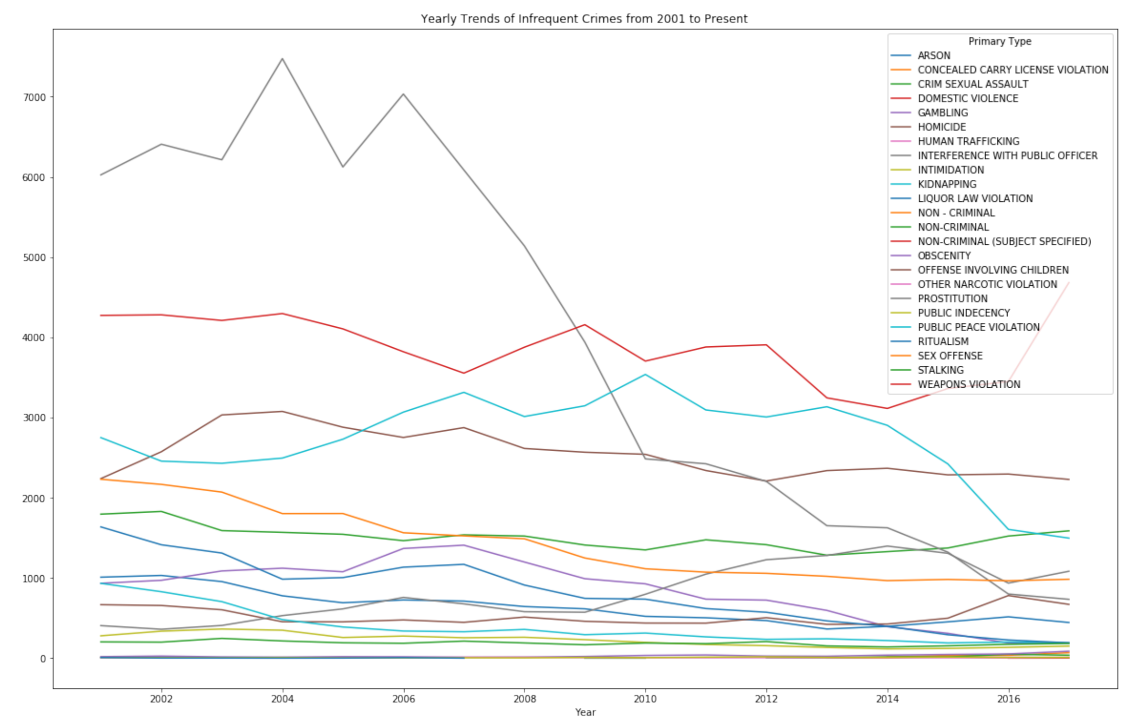


Next, let's look at yearly trends by crime type, splitting the data into two comparable groups based on crime-type frequency.

In [21]:
```
# get frequent crime types and infrequent crime types
frequent_crimes =  df["Primary Type"].value_counts().head(11).index
infrequent_crimes = df["Primary Type"].value_counts().tail(-11).index

# get value counts for each
frequent_trends = df[df["Primary Type"].isin(frequent_crimes)].groupby(["Year", "Prima
infrequent_trends = df[df["Primary Type"].isin(infrequent_crimes)].groupby(["Year", "I

# plot
fig = plt.figure()
fig.set_size_inches(20,28)

freq = fig.add_subplot(2,1,1)
frequent_trends.iloc[:-1].plot(ax = freq)
freq.set_title("Yearly Trends of Frequent Crimes from 2001 to Present")

infreq = fig.add_subplot(2,1,2)
infrequent_trends.iloc[:-1].plot(ax = infreq)
```

```python
infreq.set_title("Yearly Trends of Infrequent Crimes from 2001 to Present")
plt.show()
```

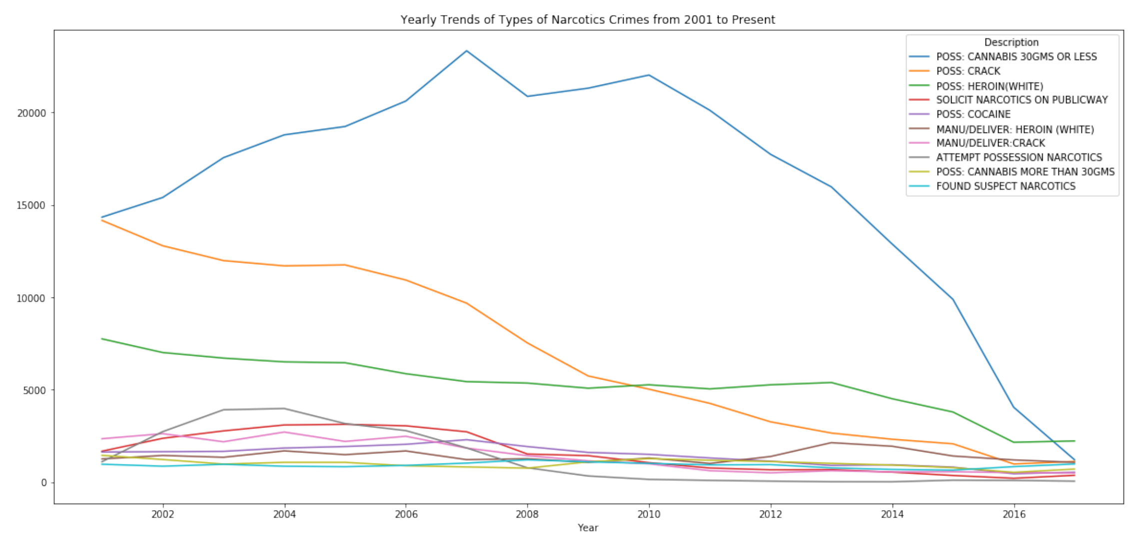




So the more frequent crime types seem to have substantially more pronounced downward trends, whereas the less frequent crime types remain at least relatively stationary with the exception of prostituion.

Let's again break down individual crime types into their respective descriptions, to get a better understanding how the breakdown of how crimes trended over time.

```
In [22]: def plot_by_description(df, primary_type):
             crimes = df[df["Primary Type"] == primary_type].groupby(["Year", "Description"]).s

             # get top 10 descriptions
             major = crimes.sum(0).sort_values(ascending = False).head(10).index
             crimes = crimes[major]

             # plot
             fig = plt.figure()
             fig.set_size_inches(20,20)

             desc = fig.add_subplot(2,1,1)
             crimes.iloc[:-1].plot(ax = desc)
             desc.set_title("Yearly Trends of Types of {} Crimes from 2001 to Present".format(s

             plt.show()

         plot_by_description(df, "NARCOTICS")
```



Interestingly, the rise in cannibis possession crimes dramatically increased until 2007, then exponentially dropped after that (especially after 2010). Also of note is how crack posession charges also consistently dropped throughout the years, but posession of heroin (white), remained relatively constant throughout.

```
In [23]: primary_type = widgets.Dropdown(
             options = type_dict,
```

```
            value = "NARCOTICS",
            description = "Primary Type",
            disabled = False,
        )

        widgets.interact(plot_by_description, df = widgets.fixed(df), primary_type = primary_
```

```
interactive(children=(Dropdown(description='Primary Type', index=3, options={'Battery': 'BATTE
```

```
Out[23]: <function __main__.plot_by_description>
```

1.1.1 Geographical Plots

To start, let's get a raw geographical distribution of crimes from just last year using a heatmap. We use the python package "gmaps" for interfacing with the Google Maps API and creating interactive geographical plots.

```
In [24]: import gmaps

         # load api key from file
         f = open("api_key")
         key = f.read().split("\n")[0]
         f.close()

         # pass key to gmaps
         gmaps.configure(key)
```

```
In [25]: # get list of latitude and longitude pairs for crimes from last year
         def heatmap_for_year(year):
             ''' given a year, outputs a heatmap '''
             crimes_year = list(zip(df[(df.Year == year) & (df.Latitude.notnull())].Latitude, d

             chi_coords = (41.8781, -87.6298)
             chi = gmaps.figure(center = chi_coords, zoom_level = 10)
             heatmap_layer = gmaps.heatmap_layer(crimes_year)

             # set intensity of heatmap for zooming (set by experimentation)
             heatmap_layer.max_intensity = 1500

             chi.add_layer(heatmap_layer)
             return chi
```

```
In [26]: heatmap_for_year(2008)
```

```
Figure(layout=FigureLayout(height='420px'))
```

In [27]:
```
# adapted from the following gmaps tutorial:
# http://jupyter-gmaps.readthedocs.io/en/latest/app_tutorial.html#reacting-to-user-ac

from IPython.display import display
import ipywidgets as widgets


class HeatmapByYear(object):
    """
    A Jupyter widget that allows for the interactive display of heatmaps for Chicago 

    Takes in dataframe of just coordinates (Latitude, Longitude) integer tuples, grou

    """

    def __init__(self, df):
        self._df = df
        self._heatmap = None
        self._slider = None
        initial_year = min(self._df['Year'])

        title_widget = widgets.HTML(
            '<h3>Distribution of Crime in Chicago by Year</h3>'
        )

        map_figure = self._render_map(initial_year)
        controls = self._render_controls(initial_year)
        self._container = widgets.VBox([title_widget, controls, map_figure])

    def render(self):
        display(self._container)

    def _on_year_change(self, change):
        year = self._slider.value
        self._heatmap.locations = self._locations_for_year(year)
        self._total_box.value = self._total_crimes_text_for_year(year)
        return self._container

    def _render_map(self, initial_year):
        chi_coords = (41.8781, -87.6298)
        fig = gmaps.figure(center = chi_coords, zoom_level = 10)
        self._heatmap = gmaps.heatmap_layer(
            self._locations_for_year(initial_year),
            max_intensity=1500
        )
        fig.add_layer(self._heatmap)
        return fig
```

```
    def _render_controls(self, initial_year):
        self._slider = widgets.IntSlider(
            value=initial_year,
            min=min(self._df['Year']),
            max=max(self._df['Year']),
            description='Year',
            continuous_update=False
        )
        self._total_box = widgets.Label(
            value=self._total_crimes_text_for_year(initial_year)
        )
        self._slider.observe(self._on_year_change, names='value')
        controls = widgets.HBox(
            [self._slider, self._total_box],
            layout={'justify_content': 'space-between'}
        )
        return controls

    def _locations_for_year(self, year):
        return self._df[(self._df['Year'] == year) & (self._df["Latitude"].notnull())]

    def _total_crimes_for_year(self, year):
        return int(self._df[self._df['Year'] == year]['Year'].count())

    def _total_crimes_text_for_year(self, year):
        return '{} total crimes'.format(self._total_crimes_for_year(year))
```

```
In [28]: HeatmapByYear(df).render()
```

```
VBox(children=(HTML(value='<h3>Distribution of Crime in Chicago by Year</h3>'), HBox(children=
```

So this shows that the reduction in crime through the years has largely ignored certain communities on the west side and the south side; as well as the loop, though it's obvious that that is a factor of its relative population and as future plots show, the crimes there are much less serious.

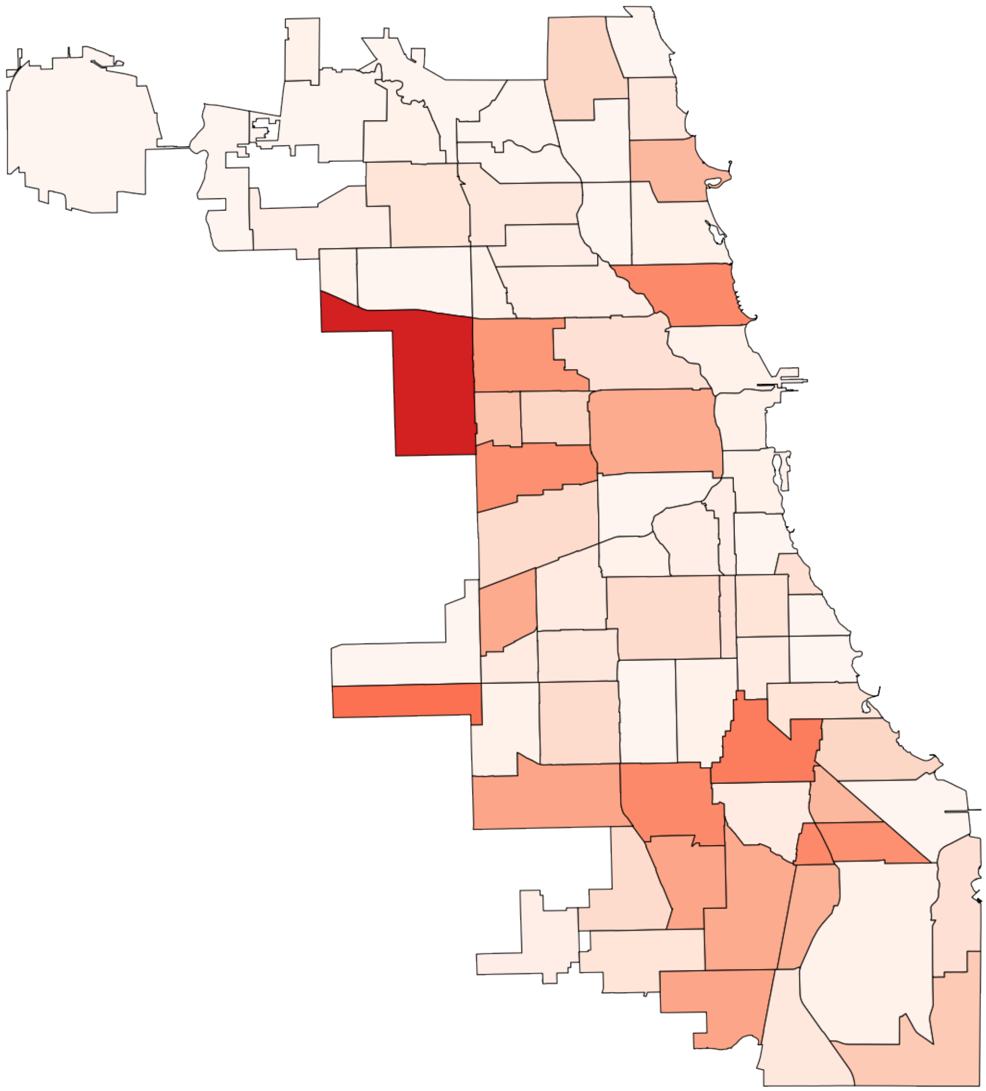
Let's look the distribution of homicides now, and rather than looking at a heatmap, let's construct a choropleth to compare the relative distribution of homicides by community area. We look at two years, 2016, the year of the massive homicide spike, and 2011, a baseline that for future census-data derived comparisons remains statistically relevant (since the ACS-5 is an average over 5 years; see census data notebook for details).

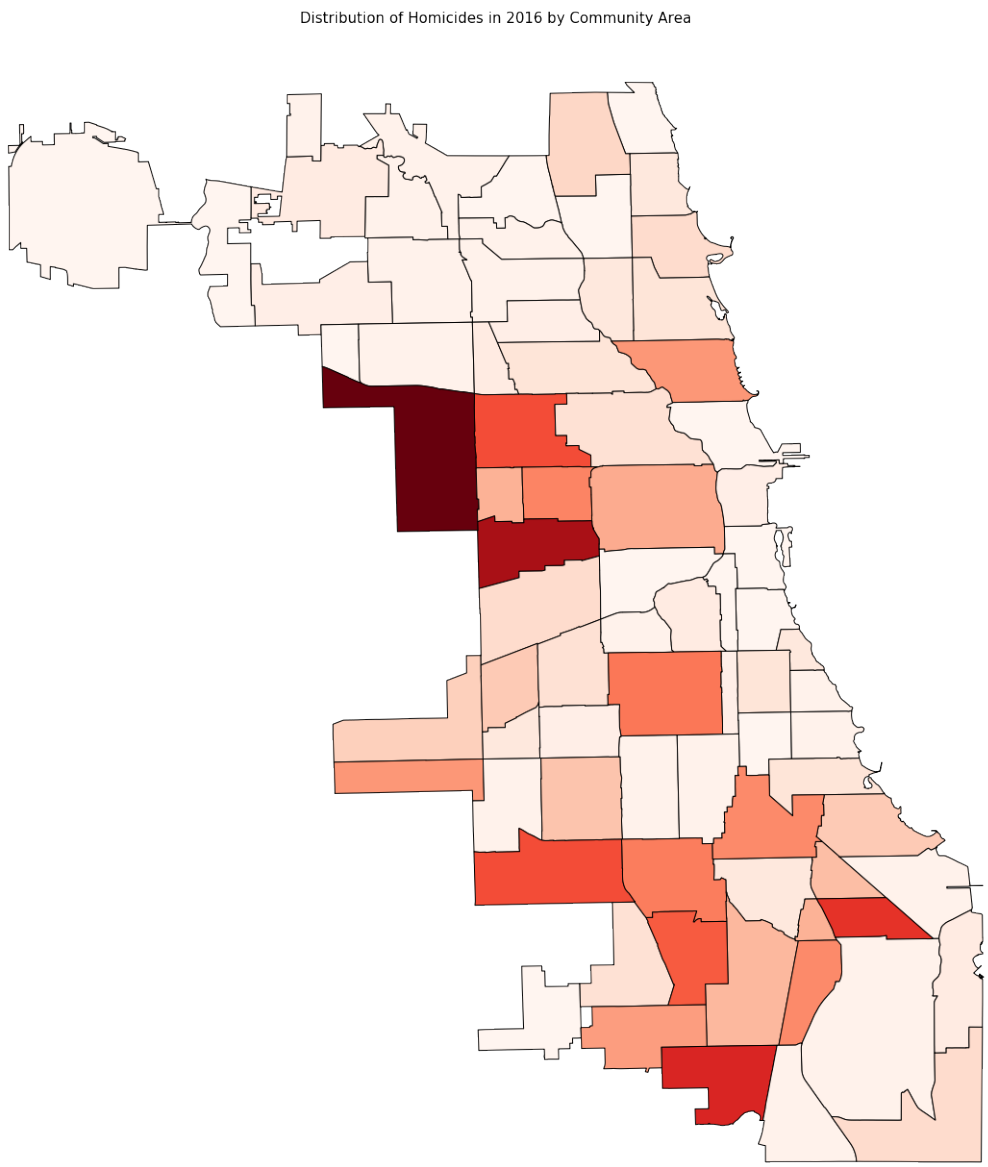
```
In [40]: homicides_by_community = df[df["Primary Type"] == "HOMICIDE"].groupby(["Year","Communi



         def plot_homicides_per_year(year):
             offset = year - 2018
             com["Homicides"] = homicides_by_community.iloc[offset]
```

```
    fig = plt.figure()
    fig.set_size_inches(20,50)

    choro = fig.add_subplot(2,1,1)
    com.plot(ax = choro, column = "Homicides", cmap = "Reds", vmin = 0, vmax = 50, edg
    plt.title("Distribution of Homicides in {} by Community Area".format(year), fontsi
    plt.axis("off")
    plt.show()
```

In [44]:
```
com = gpd.read_file("Boundaries - Community Areas (current).geojson")

plot_homicides_per_year(2011)
plot_homicides_per_year(2016)
```

```
/Users/ashwin/anaconda3/lib/python3.6/site-packages/matplotlib/colors.py:489: RuntimeWarning: i
  np.copyto(xa, -1, where=xa < 0.0)
```


Distribution of Homicides in 2011 by Community Area



The color's are on the same scale so as to be comparable. What is notable here is that, clearly, areas that were bad in 2011 only got worse in 2016.

Also, the same concentration towards the west and south sides occured as with the general crimes (the main difference obviously being the lack of homicides in the loop versus an extreme amount of crime in the loop).

Let's compare that to socioeconomic census data from the same period.

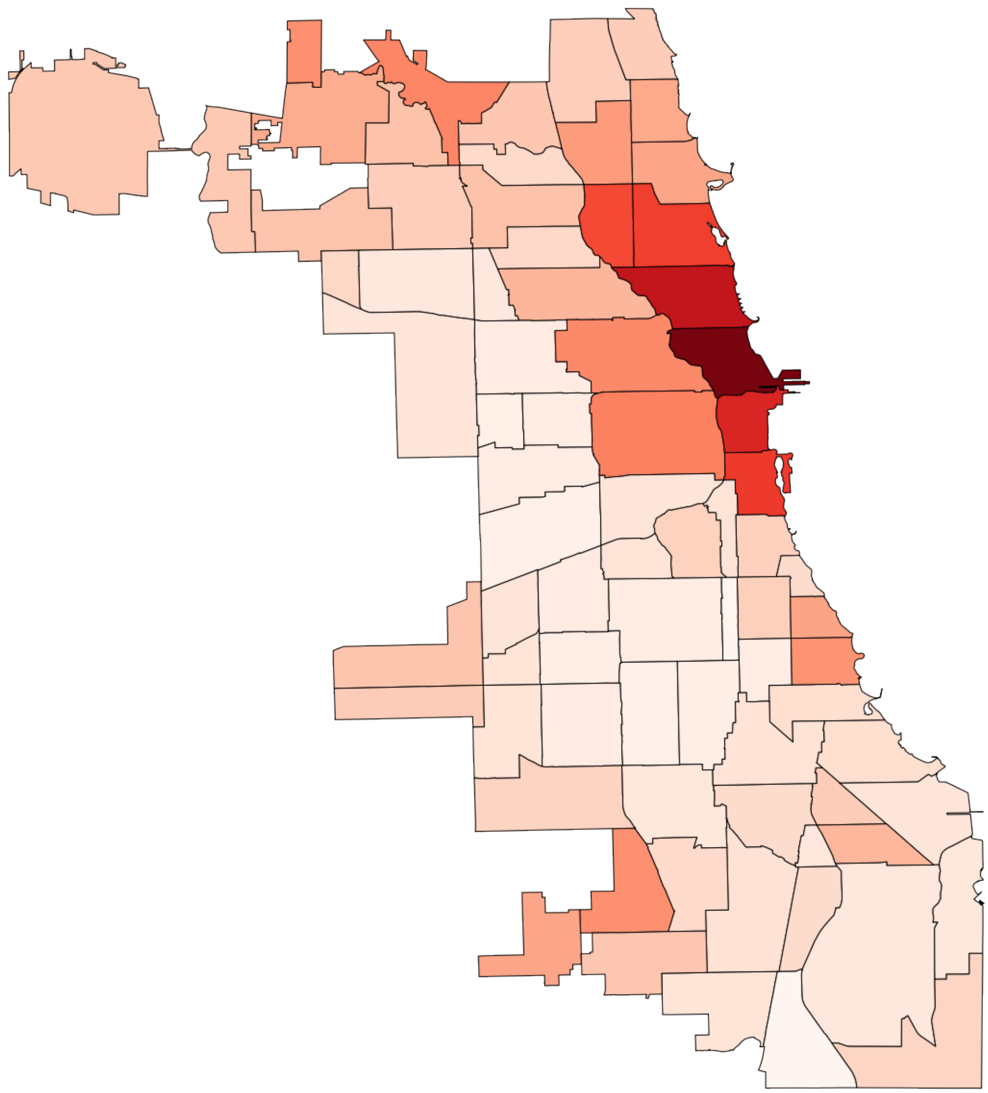
```
In [50]: socio_2011 = pd.read_csv("census/socio_2011.csv", index_col = "Community Area")
         socio_2016 = pd.read_csv("census/socio_2016.csv", index_col = "Community Area")

         hardship_indices = pd.read_csv("census/hardship_indices.csv", index_col = "Community A
```

```
In [88]: # load another geopandas df for plotting socioeconomic data by community areas
         socio_comm = gpd.read_file("Boundaries - Community Areas (current).geojson")
         socio_comm = socio_comm.set_index("area_num_1")
         socio_comm.index = socio_comm.index.astype(int)

In [126]: def plot_socio_data(column, comms = socio_comm):
              ''' creates choropleth plots of given column for multiple years; gets accurate s
              # get bounds
              vmin = min(socio_2011[column].min(), socio_2016[column].max())
              vmax = max(socio_2011[column].max(), socio_2016[column].max())

              comms[column] = socio_2011[column]

              fig = plt.figure()
              fig.set_size_inches(20,50)

              choro = fig.add_subplot(2,1,1)
              comms.plot(ax = choro, column = column, vmin = vmin, vmax = vmax, cmap = "Reds",
              plt.title("Distribution of {} in 2011 by Community Area".format(column.replace("I
              plt.axis("off")

              comms[column] = socio_2016[column]

              fig = plt.figure()
              fig.set_size_inches(20,50)

              choro = fig.add_subplot(2,1,1)
              comms.plot(ax = choro, column = column, cmap = "Reds", edgecolor = "black", linew
              plt.title("Distribution of {} in 2016 by Community Area".format(column.replace("I
              plt.axis("off")
              plt.show()
```

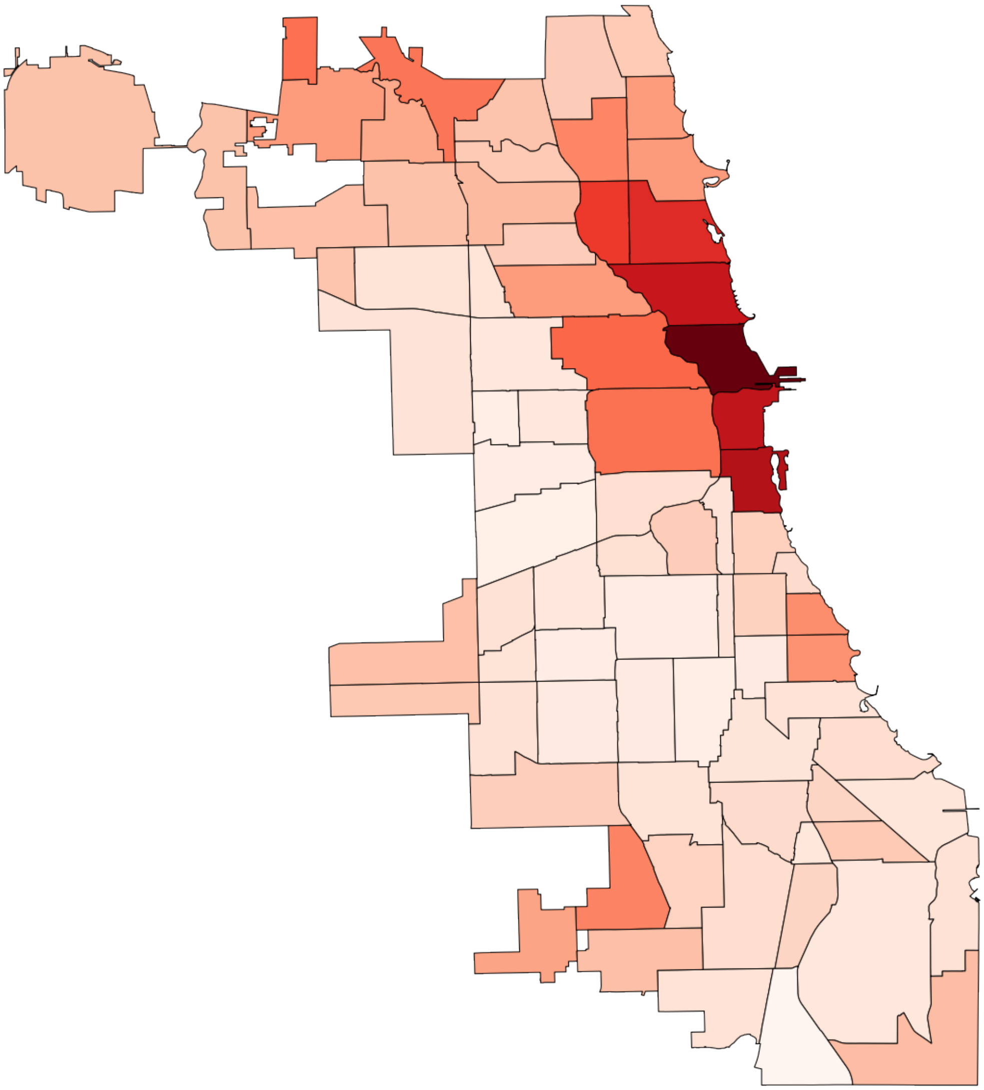
Note that for income, darker indicates more income per capita, i.e., is the opposite of the other variables where darker indicates more hardship. This is left as is instead of reversed to more clearly show the concentration of wealth, since it's easier to make out darker sections from a majority of light as opposed to the opposite.

```
In [127]: plot_socio_data("Estimated Income Per Capita")
```

Distribution of Income Per Capita in 2011 by Community Area

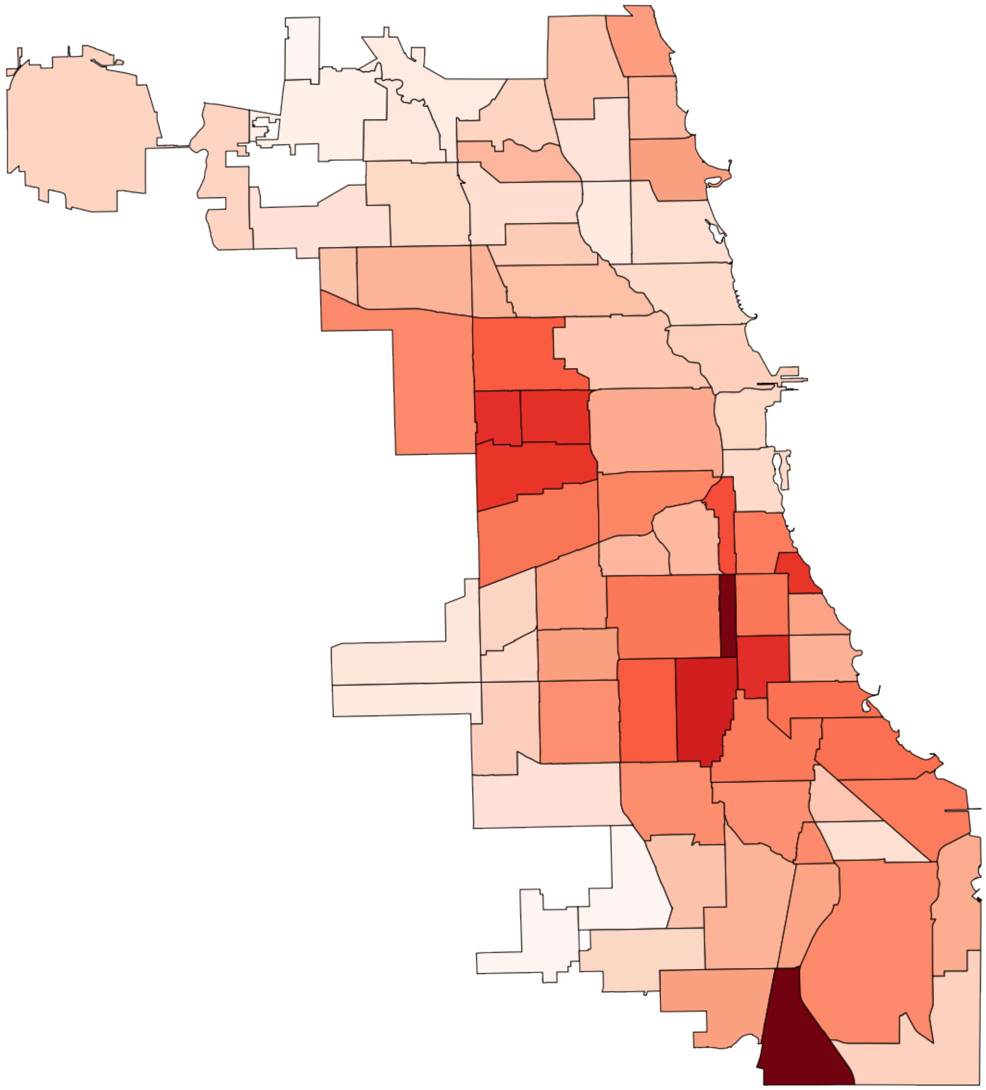


Distribution of Income Per Capita in 2016 by Community Area

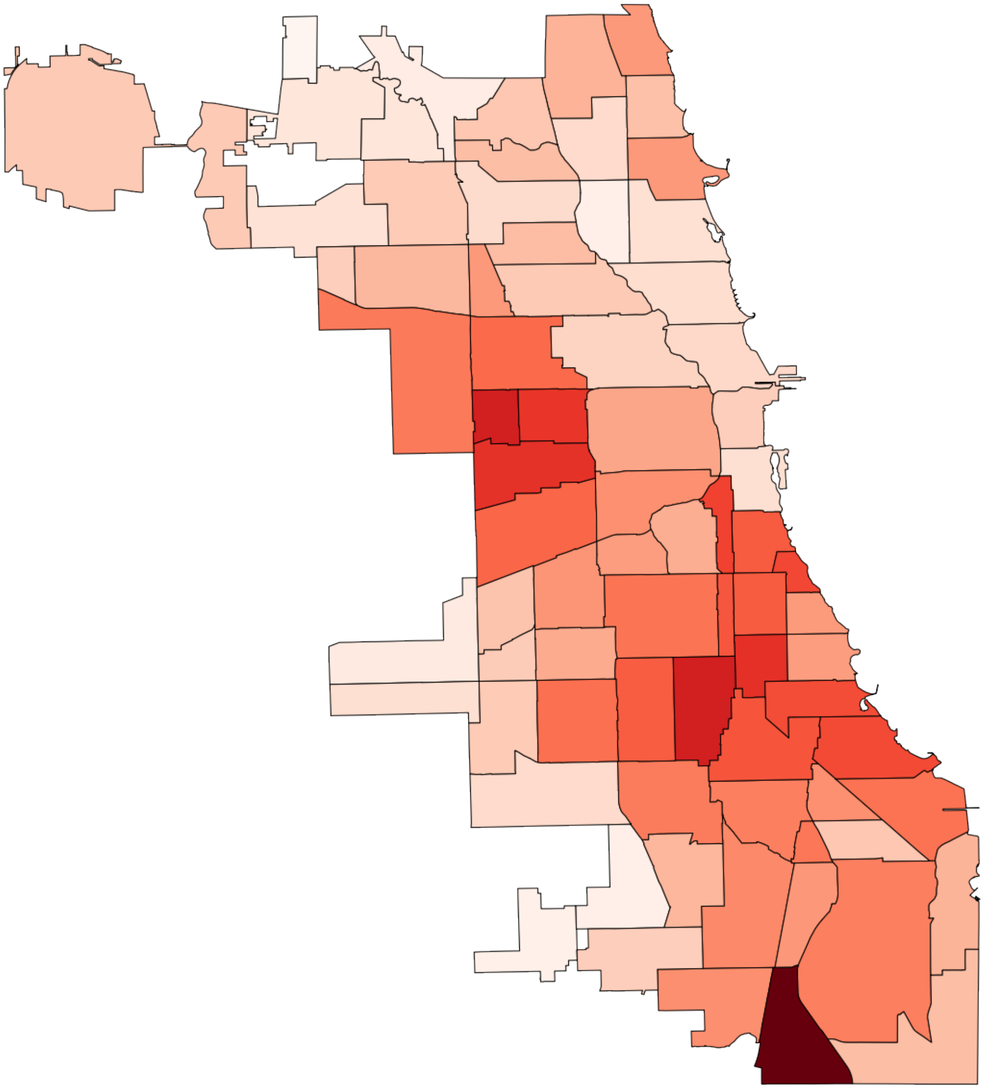


```
In [121]: plot_socio_data("Estimated Percentage of Households with Income Below Poverty Line in
```

Distribution of Estimated Percentage of Households with Income Below Poverty Line in Last 12 Months in 2011 by Community Area

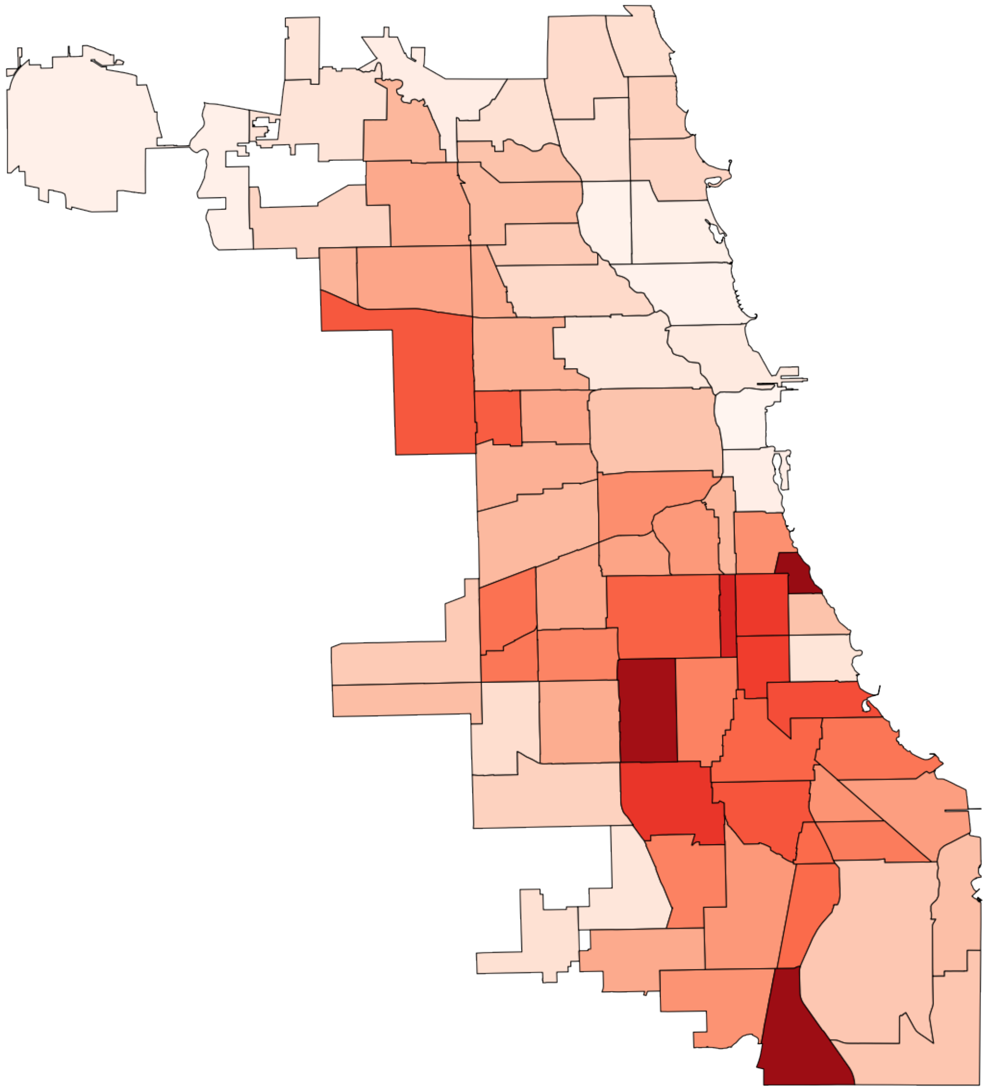


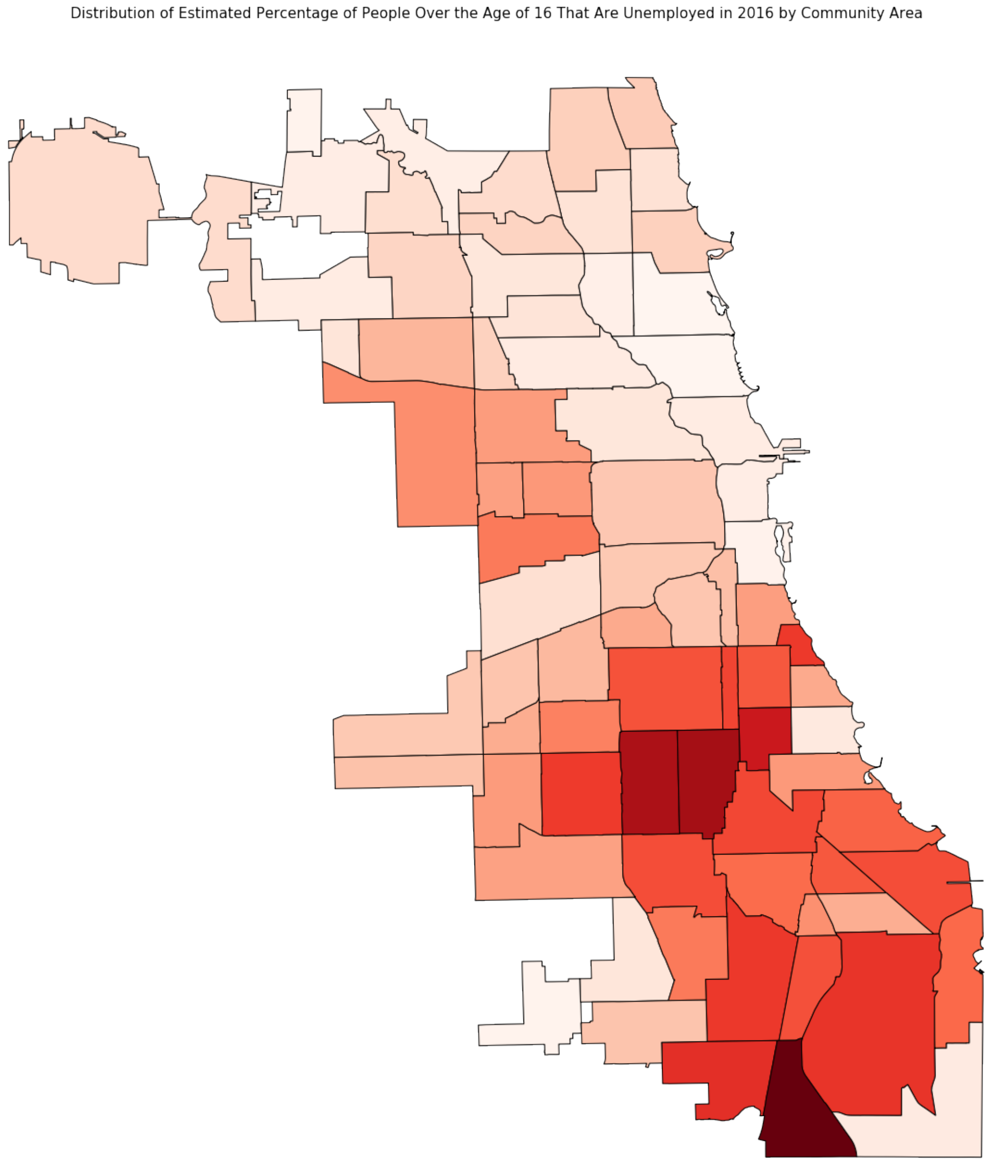
Distribution of Estimated Percentage of Households with Income Below Poverty Line in Last 12 Months in 2016 by Community Area



```
In [122]: plot_socio_data("Estimated Percentage of People Over the Age of 16 That Are Unemploye
```

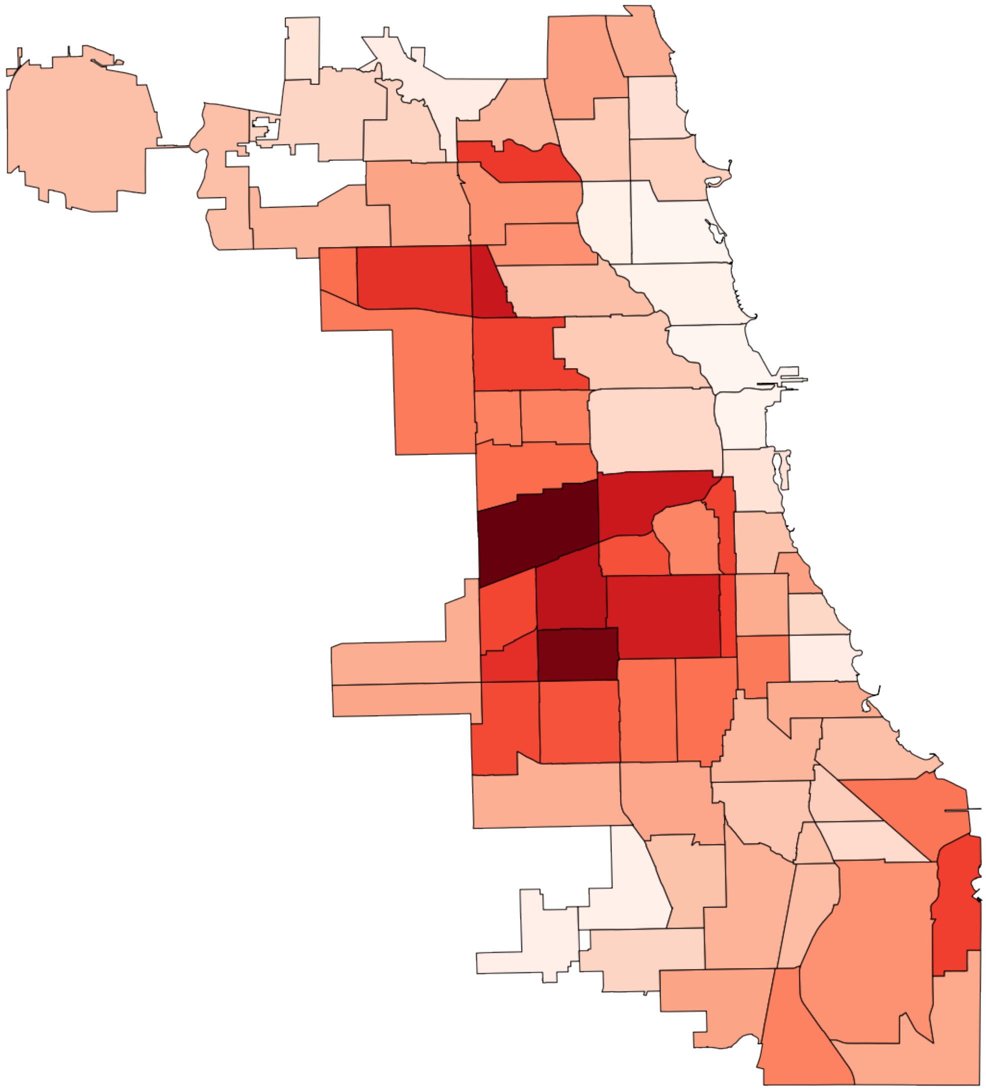
Distribution of Estimated Percentage of People Over the Age of 16 That Are Unemployed in 2011 by Community Area

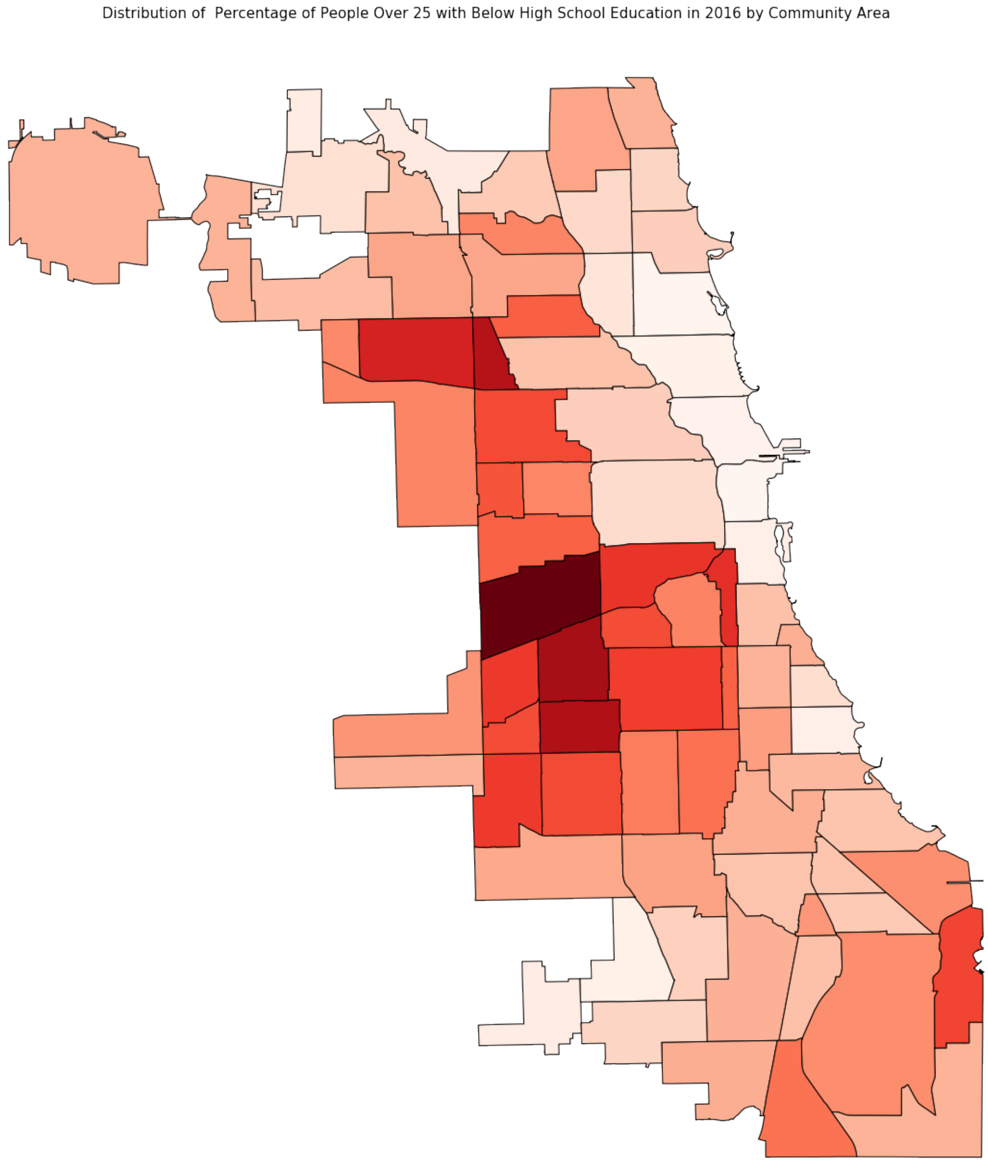





```
In [130]: plot_socio_data("Estimated Percentage of People Over 25 with Below High School Educa
```

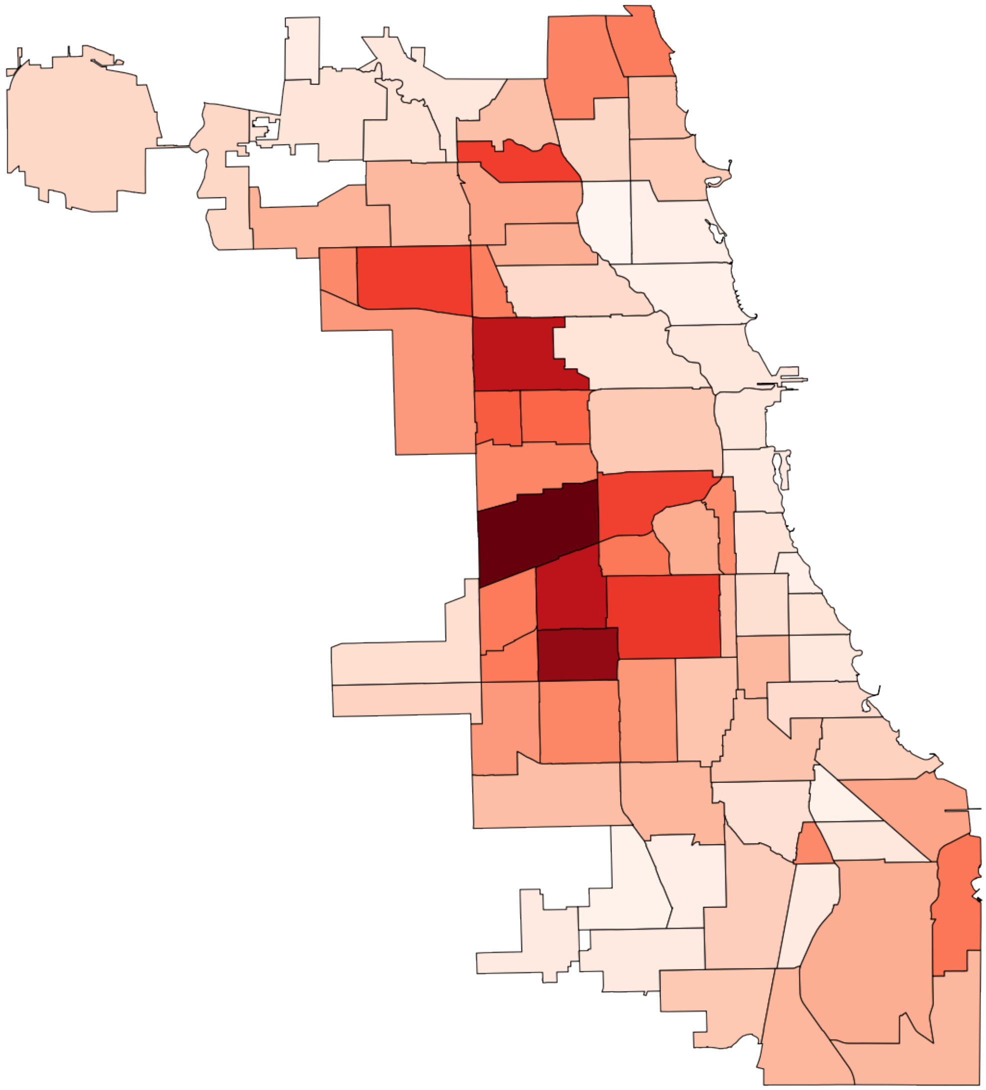
Distribution of Percentage of People Over 25 with Below High School Education in 2011 by Community Area



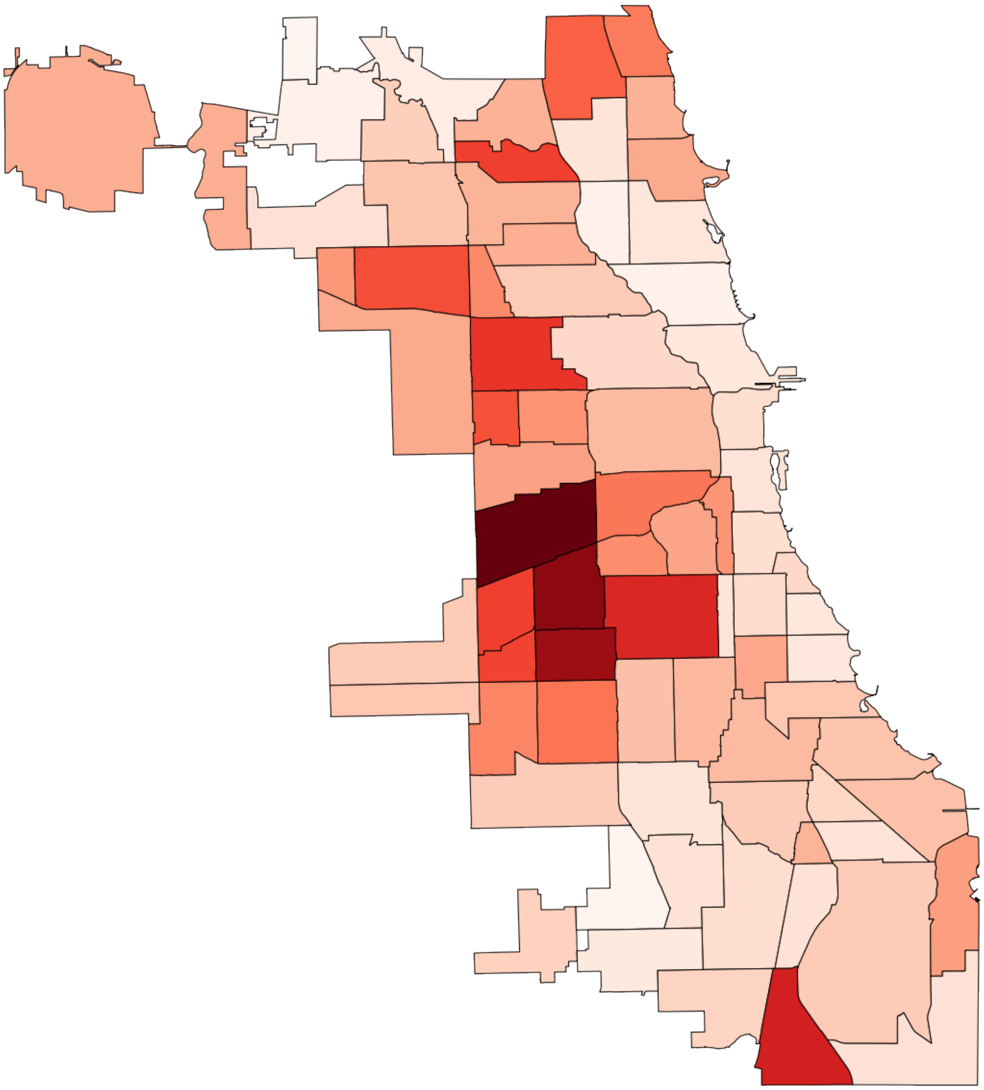



```
In [131]: plot_socio_data("Estimated Percentage of Housing Units with More Than One Person Per
```

Distribution of Percentage of Housing Units with More Than One Person Per Room in 2011 by Community Area

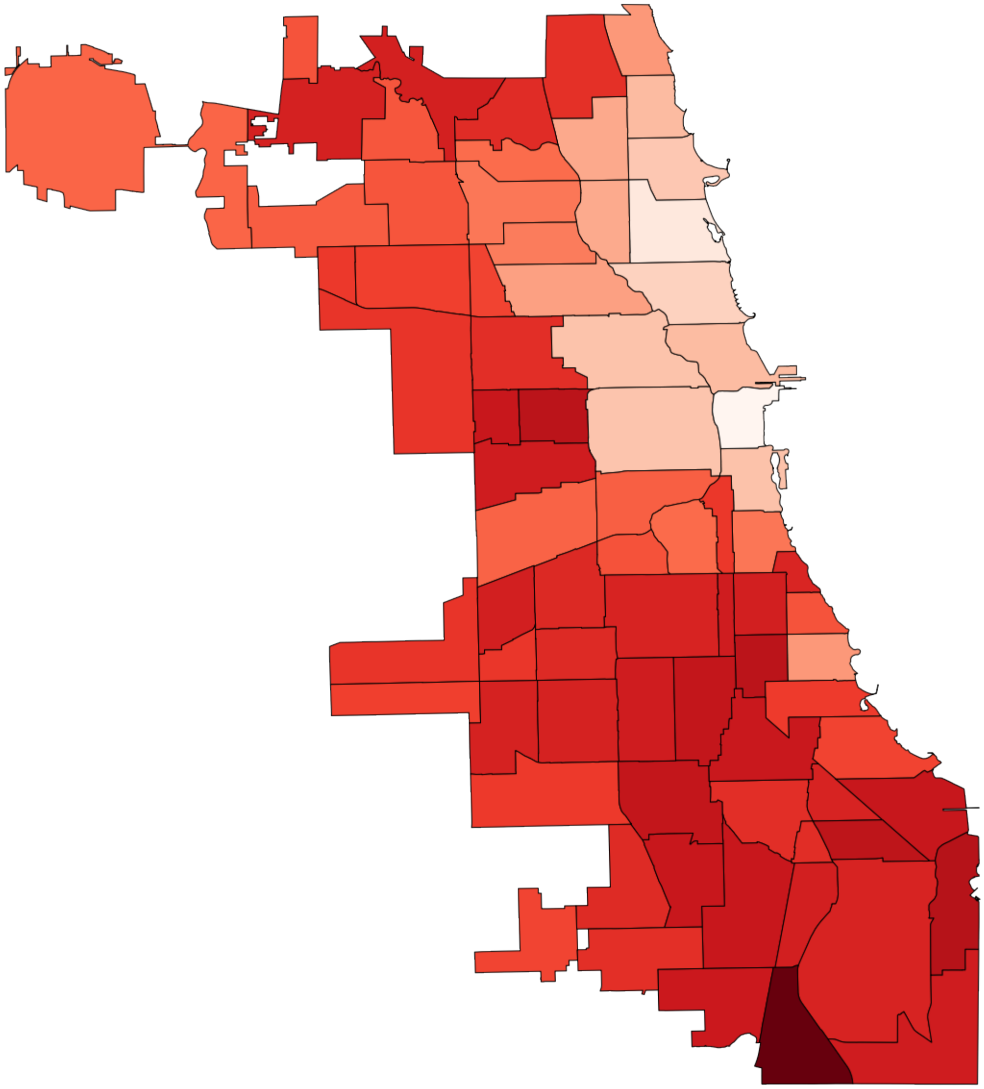


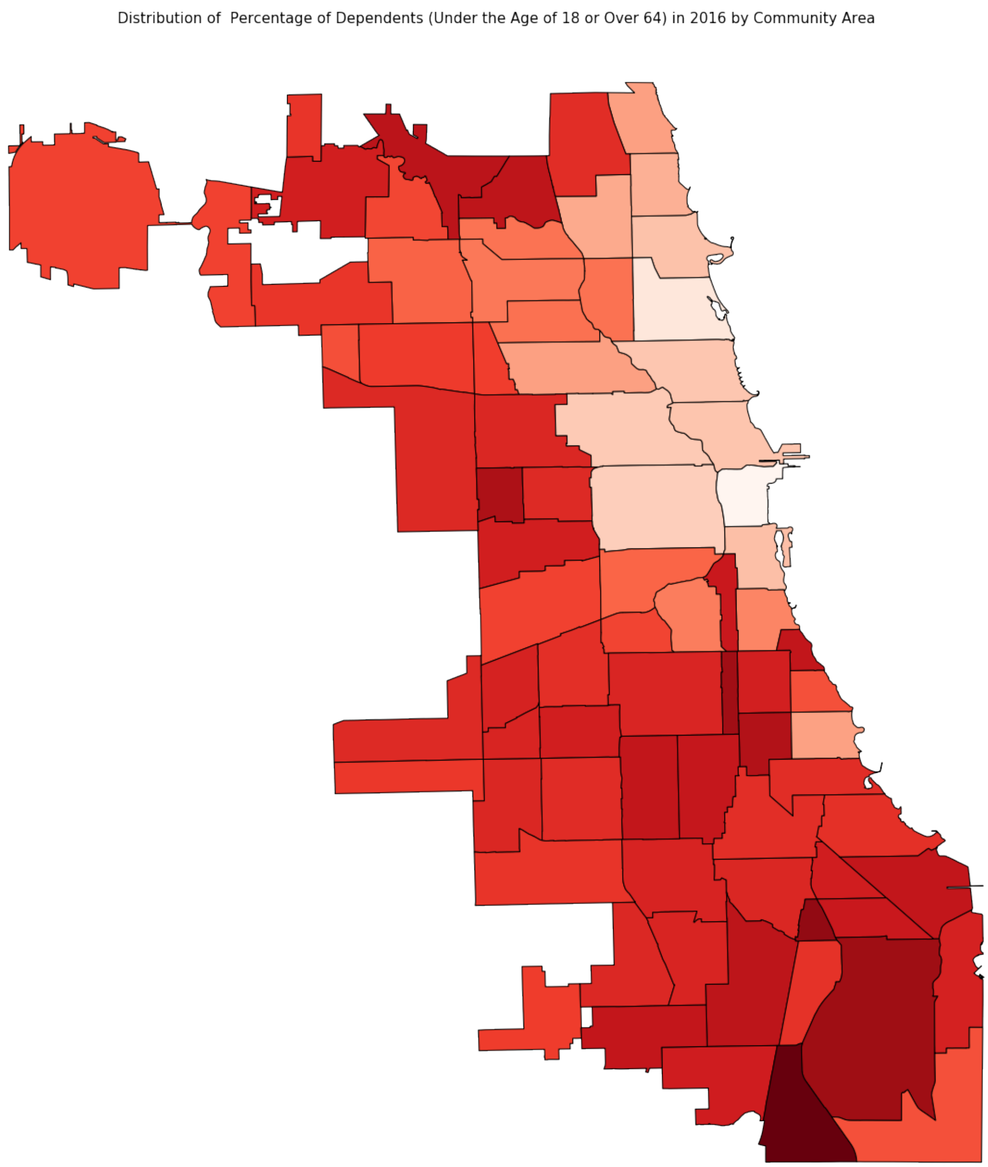
Distribution of Percentage of Housing Units with More Than One Person Per Room in 2016 by Community Area



```
In [133]: plot_socio_data("Estimated Percentage of Dependents (Under the Age of 18 or Over 64)"
```

Distribution of Percentage of Dependents (Under the Age of 18 or Over 64) in 2011 by Community Area





There is a lot of data to interpret here, but the overarching story is one of inequality: the well off areas in Chicago are, as expected, concentrated around the loop and going up along the lake to the north side.

While all of these variables broadly correlate to the homicide distribution, within the restricted subset of worse off areas in Chicago (not considering the loop and north side neighborhoods), none of these variables can account for the substantial difference in homicides in the Austin community area and other socioeconomically similar areas, for example. In other words, clearly none of these socioeconomic variables can account for the spike in homicides in 2016.

1.1.2 Police Complaints

Another possible explanation that we now examine, is police-community relationships. We take COPA data, aggregate it from police beats to community area, and see if there's a stronger correlation between number of complaints and homicide rates. In particular, we look at the percentage of complaints per total crime rate, as areas with higher crime incidence would be expected to have higher complaints regardless of any bias.

```
In [135]: police_complaints = pd.read_csv("COPA_Cases_-_By_Involved_Officer.csv")

/Users/ashwin/anaconda3/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2728: Dtyp
  interactivity=interactivity, compiler=compiler, result=result)


In [136]: police_complaints.shape

Out[136]: (87871, 16)

In [137]: police_complaints.head()

Out[137]:     LOG_NO          COMPLAINT_DATE ASSIGNMENT  CASE_TYPE CURRENT_STATUS  \
          0  1008899  09/01/2007 12:34:36 AM       IPRA  Complaint         Closed
          1  1008901  09/01/2007 04:16:33 AM       IPRA  Complaint         Closed
          2  1008909  09/01/2007 02:59:50 PM       IPRA  Complaint         Closed
          3  1008909  09/01/2007 02:59:50 PM       IPRA  Complaint         Closed
          4  1008910  09/01/2007 03:06:08 PM       IPRA  Complaint         Closed

                       CURRENT_CATEGORY   FINDING_CODE POLICE_SHOOTING         BEAT  \
          0               Miscellaneous   NO AFFIDAVIT              No          433
          1               Miscellaneous  NOT SUSTAINED              No         1933
          2  Death or Injury In Custody      UNFOUNDED              No         1132
          3  Death or Injury In Custody      UNFOUNDED              No         1132
          4               Miscellaneous  NOT SUSTAINED              No  1135 | 1134

            RACE_OF_INVOLVED_OFFICER SEX_OF_INVOLVED_OFFICER AGE_OF_INVOLVED_OFFICER  \
          0                      NaN                     NaN                     NaN
          1                    White                    Male                   30-39
          2                    White                    Male                   20-29
          3                    White                    Male                   20-29
          4                    White                  Female                   30-39

            YEARS_ON_FORCE_OF_INVOLVED_OFFICER  COMPLAINT_HOUR  COMPLAINT_DAY  \
          0                                NaN               0              7
          1                                5-9               4              7
          2                                5-9              14              7
          3                                0-4              14              7
          4                              10-14              15              7

             COMPLAINT_MONTH
```

```
0             9
1             9
2             9
3             9
4             9
```

In [139]: police_complaints.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87871 entries, 0 to 87870
Data columns (total 16 columns):
LOG_NO                              87871 non-null int64
COMPLAINT_DATE                      87871 non-null object
ASSIGNMENT                          87871 non-null object
CASE_TYPE                           32137 non-null object
CURRENT_STATUS                      32137 non-null object
CURRENT_CATEGORY                    32065 non-null object
FINDING_CODE                        30339 non-null object
POLICE_SHOOTING                     32137 non-null object
BEAT                                32137 non-null object
RACE_OF_INVOLVED_OFFICER            26222 non-null object
SEX_OF_INVOLVED_OFFICER             26222 non-null object
AGE_OF_INVOLVED_OFFICER             26222 non-null object
YEARS_ON_FORCE_OF_INVOLVED_OFFICER  26222 non-null object
COMPLAINT_HOUR                      87871 non-null int64
COMPLAINT_DAY                       87871 non-null int64
COMPLAINT_MONTH                     87871 non-null int64
dtypes: int64(4), object(12)
memory usage: 10.7+ MB
```

In [140]: police_complaints.isnull().sum()

```
Out[140]: LOG_NO                                  0
          COMPLAINT_DATE                          0
          ASSIGNMENT                              0
          CASE_TYPE                           55734
          CURRENT_STATUS                      55734
          CURRENT_CATEGORY                    55806
          FINDING_CODE                        57532
          POLICE_SHOOTING                     55734
          BEAT                                55734
          RACE_OF_INVOLVED_OFFICER            61649
          SEX_OF_INVOLVED_OFFICER             61649
          AGE_OF_INVOLVED_OFFICER             61649
          YEARS_ON_FORCE_OF_INVOLVED_OFFICER  61649
          COMPLAINT_HOUR                          0
          COMPLAINT_DAY                           0
```

```
COMPLAINT_MONTH                                    0
dtype: int64
```

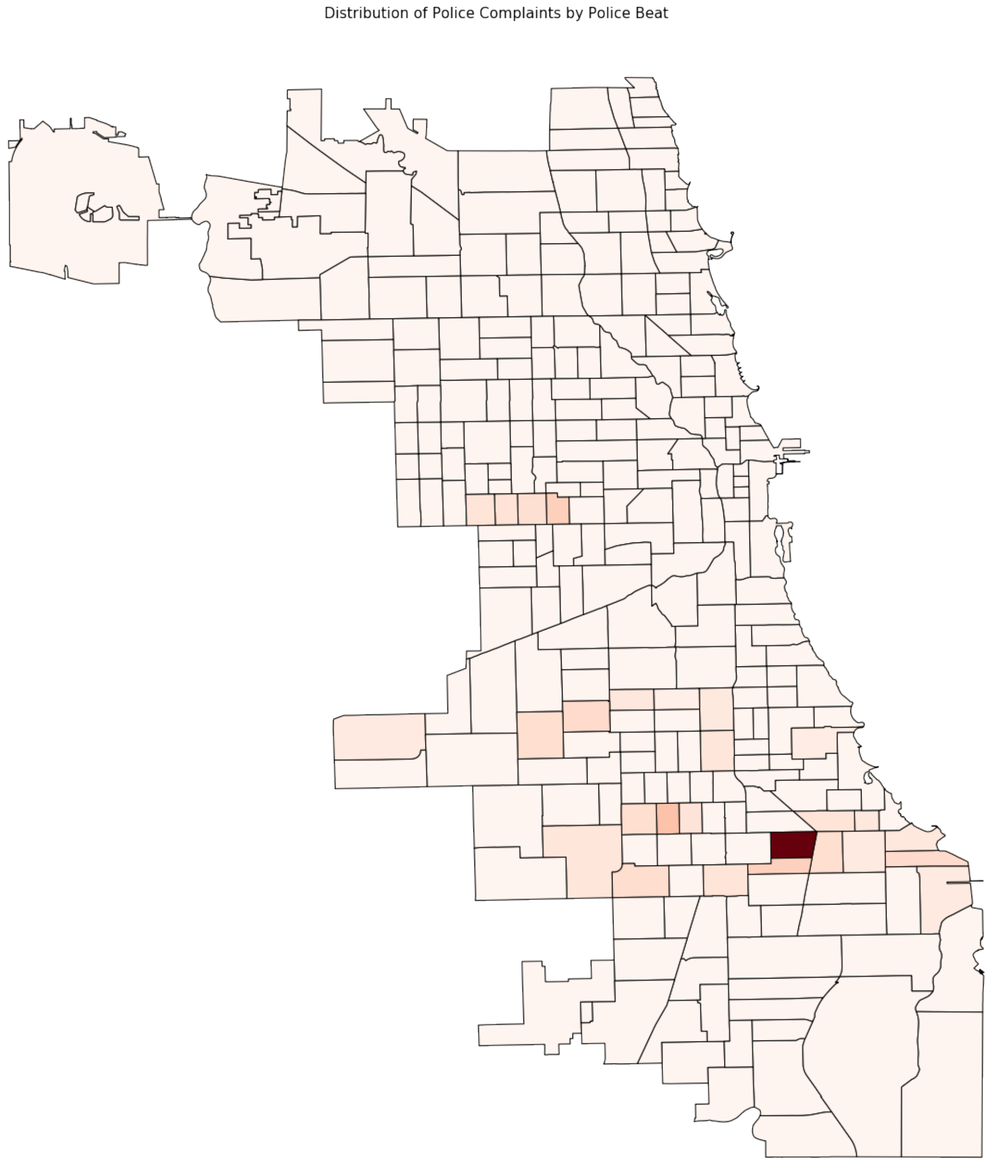
So, unfortunately, this dataset is extremely incomplete. As with any other analysis, the only way to properly justify dropping these rows, and considering the filled-in subset of data as representative of the set as a whole, would be if the distribution of complaints missing data was completely random. We tentatively make that assumption, being mindful of the fact that it's almost certainly not true, and extremely difficult to prove otherwise. We do this for the sake of demonstrating the method that will be applicable whenever this data is either filled in by COPA, or requested via FOIA.

```
In [141]: known_complaints = police_complaints[police_complaints.BEAT.notnull()]
```

```
In [142]: known_complaints.head()
```

```
Out[142]:      LOG_NO          COMPLAINT_DATE ASSIGNMENT  CASE_TYPE CURRENT_STATUS  \
          0  1008899  09/01/2007 12:34:36 AM       IPRA  Complaint         Closed
          1  1008901  09/01/2007 04:16:33 AM       IPRA  Complaint         Closed
          2  1008909  09/01/2007 02:59:50 PM       IPRA  Complaint         Closed
          3  1008909  09/01/2007 02:59:50 PM       IPRA  Complaint         Closed
          4  1008910  09/01/2007 03:06:08 PM       IPRA  Complaint         Closed

                          CURRENT_CATEGORY   FINDING_CODE POLICE_SHOOTING         BEAT  \
          0              Miscellaneous   NO AFFIDAVIT              No          433
          1              Miscellaneous  NOT SUSTAINED              No         1933
          2  Death or Injury In Custody      UNFOUNDED              No         1132
          3  Death or Injury In Custody      UNFOUNDED              No         1132
          4              Miscellaneous  NOT SUSTAINED              No  1135 | 1134

            RACE_OF_INVOLVED_OFFICER SEX_OF_INVOLVED_OFFICER AGE_OF_INVOLVED_OFFICER  \
          0                      NaN                     NaN                     NaN
          1                    White                    Male                   30-39
          2                    White                    Male                   20-29
          3                    White                    Male                   20-29
          4                    White                  Female                   30-39

            YEARS_ON_FORCE_OF_INVOLVED_OFFICER  COMPLAINT_HOUR  COMPLAINT_DAY  \
          0                                NaN               0              7
          1                                5-9               4              7
          2                                5-9              14              7
          3                                0-4              14              7
          4                              10-14              15              7

             COMPLAINT_MONTH
          0                9
          1                9
          2                9
          3                9
          4                9
```

```
In [150]: police_districts = gpd.read_file("Boundaries - Police Districts (current).geojson")
          police_beat = gpd.read_file("Boundaries - Police Beats (current).geojson")

In [161]: known_complaints.loc[:, "BEAT"] = known_complaints.BEAT.map(lambda x: x.split("|")[0

/Users/ashwin/anaconda3/lib/python3.6/site-packages/pandas/core/indexing.py:537: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s


In [163]: known_complaints.BEAT = known_complaints.BEAT.astype(int)

/Users/ashwin/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py:3643: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self[name] = value


In [166]: complaints_by_beat = known_complaints.groupby(["BEAT"]).size()

In [175]: crimes_by_beat = df.groupby("Beat").size()

In [176]: percentange_of_crimes_by_beat = complaints_by_beat / crimes_by_beat

In [179]: police_beat["Percentage of Complaints"] = percentange_of_crimes_by_beat.fillna(0)

In [182]: fig = plt.figure()
          fig.set_size_inches(20,50)

          choro = fig.add_subplot(2,1,1)
          police_beat.plot(ax = choro, column = "Percentage of Complaints", cmap = "Reds", edge
          plt.title("Distribution of Police Complaints by Police Beat", fontsize = 15)
          plt.axis("off")
          plt.show()

/Users/ashwin/anaconda3/lib/python3.6/site-packages/matplotlib/colors.py:489: RuntimeWarning: i
  np.copyto(xa, -1, where=xa < 0.0)
```



Again, this is currently not at all informative, because the dataset is extremely incomplete, but the same technique would be worthwhile to look at for a completed dataset.

A similar technique that won't be informative here because of the limited dataset, but would be extremely relevant to the distribution of homicides in 2016 would be look to at the distribution of complaints in 2015 in comparison to the distribution of complaints in 2015.