Marshall.Charlie - A1

May 14, 2020

1 Analysis of 2019 Chicago E-Scooter Pilot Program

By Charlie Marshall

16 April 2020

2 Importing Libraries

[1]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas.plotting as pdplt
import geopandas as gpd
import folium
from json import dumps
import shapefile
import seaborn as sns
import numpy as np
```

3 Import Data

3.1 Data Sets:

trips - Contains location, distance and duration data for each reported ride from June 15th to Oct. 15th in the pilot program.

CDS - Includes a multitude of data for each Chicago Community Area spanning transportation accessibility, land use, water use, and many other topics. Information for each field can be found the Field Descriptor pdf.

[2]:
```
trips = pd.read_csv('E-Scooter_Trips_-_2019_Pilot.csv')
```

```
/Users/charlesmarshall/anaconda3/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3063: DtypeWarning: Columns
```

```
(10,11,14,17) have mixed types.Specify dtype option on import or set
low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

[3]:
```
CDS = pd.read_csv('ReferenceCCAProfiles20132017.csv')
```

[4]:
```
CCA = pd.read_excel('CCASF12010CMAP.xlsx', header=1)
```

[5]:
```
shape=gpd.read_file('Boundaries - Community Areas (current)/
 ↪geo_export_d7a61ee9-8086-4875-a523-1ddacfa2c097.shp')
```

4 Q1: Data Exploration

4.1 Background:

The Chicago E-Scooter Pilot Program took place from June 15th to October 15th. The pilot was mainly contained to an area West of the city consisting of only a portion of the Chicago Community Areas. E-Scooters are an emerging form of microtransportation, presenting an alternative to walking or biking for trips under 5 miles. In their current form, E-Scooter rentals are considered more convenient than bike-sharing services because scooters are able to dropped off anywhere. However, these scooters are often considered more dangerous than bikes and can take up space on sidewalks.

4.2 Problem Statement:

The Chicago E-Scooter Pilot Program mainly focused on an area West of Chicago. I wish to forcast e-scooter demand in other Community Areas across the city of Chicago. Additionally, I wish to find out what socioeconomic and demographic features of Chicago Community Areas are best predictors of demand.

4.3 Cleaning Trip Data:

The first step I took was to replace the long Trip ID column in favor of numbered rows. Additionally, I got rid of extraneous location columns which I found to be redundant.

Next, I needed to get rid of any reported scooter trips which were either too short time or distance wise. The City of Chicago used 0.25 miles as a cutoff for a valueable trip in their analysis, so I used this cutoff as well. Additionally, I limited the data to only trips over one minute long. Had these trips been kept in the dataset, they would skew the data. Lastly, the City of Chicago also eliminated "loop trips" (trips in which start and end shortly after in the same location) from their analysis, but I was not able to eliminate these trips due to the quality of the published location data.

Lastly, I needed to decide whether to use Census Tracts or Chicago Community Areas as location indicators. Datasets were created with both and pros and cons were weighed.

```
[6]: trips = trips.drop(['Trip ID','Start Centroid Latitude','Start Centroid␣
     ↪Longitude','Start Centroid Location','End Centroid Latitude','End Centroid␣
     ↪Longitude','End Centroid Location'],axis = 1)
```

```
[7]: trips.head()
```

```
[7]:                Start Time                End Time  Trip Distance  \
     0  07/01/2019 05:00:00 PM  07/01/2019 05:00:00 PM            421
     1  06/29/2019 06:00:00 PM  06/29/2019 06:00:00 PM           6318
     2  09/16/2019 01:00:00 PM  09/16/2019 01:00:00 PM             77
     3  06/24/2019 07:00:00 PM  06/24/2019 07:00:00 PM            917
     4  07/12/2019 07:00:00 PM  07/12/2019 07:00:00 PM              0

        Trip Duration  Accuracy  Start Census Tract  End Census Tract  \
     0              3         1                 NaN               NaN
     1             31         1                 NaN               NaN
     2            732        10                 NaN               NaN
     3            359        10                 NaN               NaN
     4            218         0                 NaN               NaN

        Start Community Area Number  End Community Area Number  \
     0                          NaN                        NaN
     1                          NaN                        NaN
     2                          NaN                       25.0
     3                         25.0                       25.0
     4                         21.0                       21.0

       Start Community Area Name End Community Area Name
     0                       NaN                     NaN
     1                       NaN                     NaN
     2                       NaN                  AUSTIN
     3                    AUSTIN                  AUSTIN
     4                  AVONDALE                AVONDALE
```

Limiting to specific distances and durations

```
[8]: # Minimum travel distance is 0.25 mile
     min_dist = 0.25*1609
```

```
[9]: # Minimum travel time is 60 sec (1 min)
     min_dur = 60
```

```
[10]: trips = trips[trips['Trip Distance'] > min_dist]
      trips = trips[trips['Trip Duration'] > min_dur]
```

The size of the dataset is limited to 523,700 trips.

```
[11]: len(trips)
```

[11]: 523700

4.3.1 Census Tract Data

[12]:
```
CTtrips = trips.drop(['Start Community Area Number','End Community Area␣
↪Number','Start Community Area Name','End Community Area Name'],axis = 1)
```

[13]:
```
CTtrips.head()
```

[13]:
```
                Start Time                End Time  Trip Distance  \
3   06/24/2019 07:00:00 PM  06/24/2019 07:00:00 PM            917
6   07/20/2019 06:00:00 PM  07/20/2019 06:00:00 PM           1725
9   06/30/2019 08:00:00 PM  06/30/2019 09:00:00 PM           1107
11  09/06/2019 03:00:00 PM  09/06/2019 04:00:00 PM           2977
12  07/20/2019 05:00:00 PM  07/20/2019 05:00:00 PM           1024

    Trip Duration  Accuracy  Start Census Tract  End Census Tract
3             359        10                 NaN               NaN
6            1141        10                 NaN               NaN
9            2519        10                 NaN               NaN
11            572        10                 NaN               NaN
12            539        10                 NaN               NaN
```

A large proportion the the Census Tract locations are null. So, we must remove these values because there is no way to guess location or use an average.

[14]:
```
CTtrips = CTtrips.dropna(subset=['Start Census Tract', 'End Census Tract'])
```

[15]:
```
CTtrips['Start Census Tract'].value_counts()
```

[15]:
```
1.703183e+10    69832
1.703124e+10    24377
1.703183e+10    22314
1.703124e+10    12316
1.703184e+10    12216
                …
1.703126e+10        2
1.703115e+10        2
1.703183e+10        1
1.703126e+10        1
1.703116e+10        1
Name: Start Census Tract, Length: 243, dtype: int64
```

295437 trips have recorded Census Tract locations

[16]:
```
CTtrips.shape
```

[16]: (295437, 7)

4.3.2 Chicago Community Area Data

```
[17]: CCAtrips = trips.drop(['Start Census Tract','End Census Tract'],axis = 1)
```

Removing trips will null start or end values

```
[18]: CCAtrips = CCAtrips.dropna(subset=['Start Community Area Number', 'End␣
       ↪Community Area Number'])
```

```
[19]: CCAtrips['Start Community Area Name'].value_counts()
```

```
[19]: WEST TOWN             148147
      NEAR WEST SIDE        148050
      LOGAN SQUARE           75102
      BELMONT CRAGIN         17005
      AUSTIN                 16011
      AVONDALE               14756
      HERMOSA                 9877
      HUMBOLDT PARK           9490
      LOWER WEST SIDE         9232
      NORTH LAWNDALE          7950
      IRVING PARK             6736
      SOUTH LAWNDALE          6188
      PORTAGE PARK            6184
      EAST GARFIELD PARK      6121
      WEST GARFIELD PARK      4219
      MONTCLARE               1677
      DUNNING                 1489
      NEAR NORTH SIDE          115
      LINCOLN PARK              91
      BRIDGEPORT                28
      NORTH CENTER              26
      LOOP                       3
      NEW CITY                   3
      WOODLAWN                   2
      BRIGHTON PARK              1
      GAGE PARK                  1
      Name: Start Community Area Name, dtype: int64
```

```
[20]: CCAtrips.shape
```

[20]: (488504, 9)

```
[21]: area_key = shape.iloc[:,[1,5]]
```

[22]:
```
CCAcounts = pd.DataFrame(CCAtrips['Start Community Area Name'].value_counts())
CCAcounts = CCAcounts.reset_index()
CCAcounts.columns = ['community','Count']
CCAcounts = pd.merge(CCAcounts,area_key,how = 'outer', on = 'community').
 ↪fillna(0).drop(['area_num_1'],axis = 1)
```

[23]:
```
CCAcounts.head()
```

[23]:

	community	Count
0	WEST TOWN	148147.0
1	NEAR WEST SIDE	148050.0
2	LOGAN SQUARE	75102.0
3	BELMONT CRAGIN	17005.0
4	AUSTIN	16011.0

4.4 Compare and Contrast Census Tracts vs Chicago Community Areas:

4.5 Census Tract:

4.5.1 Pros:

- A census tract is smaller than a community area, meaning there are more reported census tracts. This means that we could study more exact areas and there will be more a larger number of rows to train a model on (243 census tracts vs 26 community areas)

4.5.2 Cons:

- Not as many rows have reported location data, but still a good amount (almost 300k)
- Much harder to find socioeconomic and demographic features to be added to the dataset

4.6 Chicago Community Areas:

4.6.1 Pros:

- Lots of additional demographic data (CDS dataset), which can be used to create features
- More trips can be considered because community areas were more widely reported than census tracts.

4.6.2 Cons:

- A small number of datapoints to train a regression model on

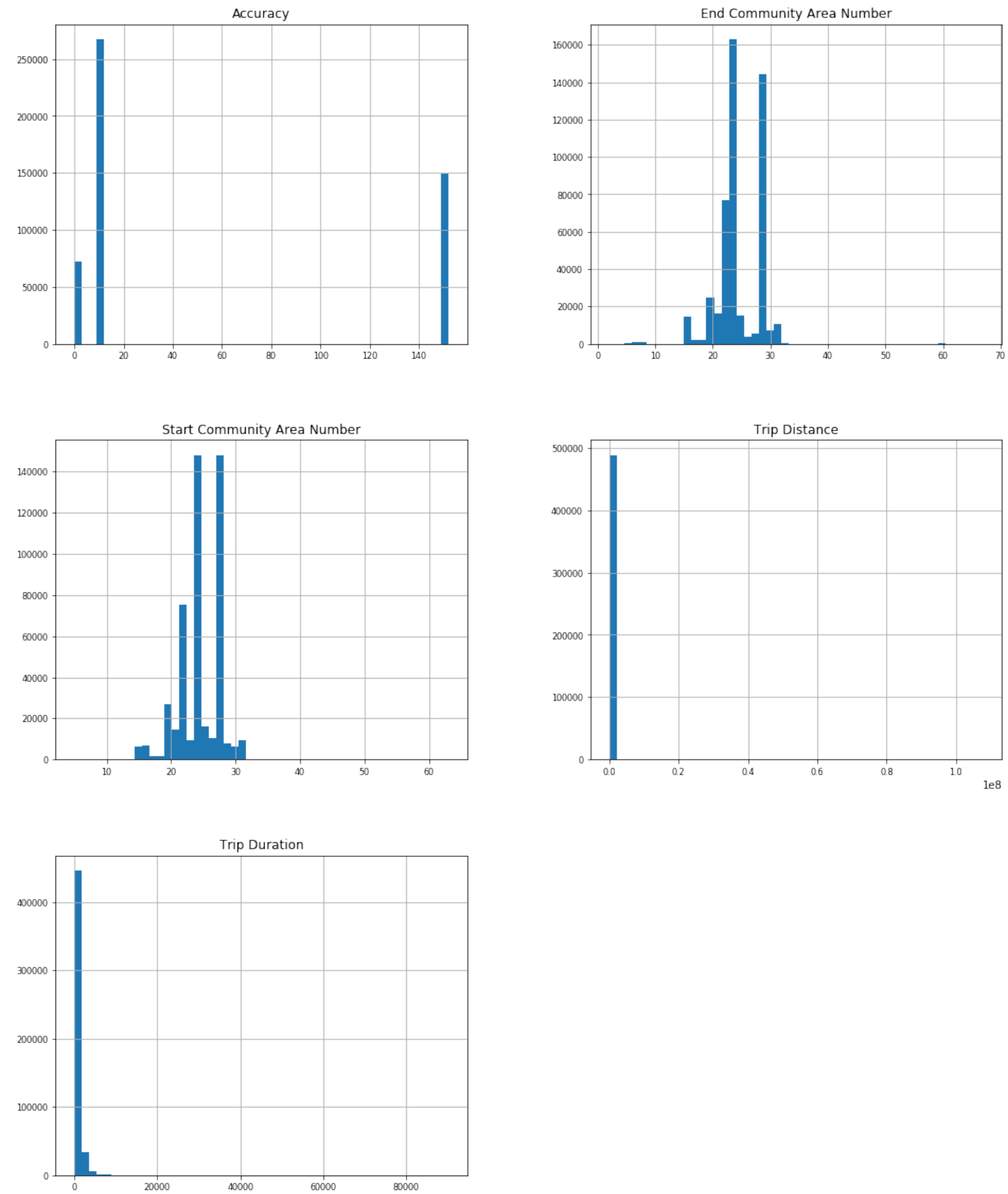
4.7 Conclusion:

In the end, because there is vastly more supporting data available, I decided to use Chicago Community Areas as my location parameter.

4.8 Data Exploration of CCA Trips:

[24]:
```
CCAtrips.hist(figsize=(16, 20), bins = 50, xlabelsize=8, ylabelsize=8)
```

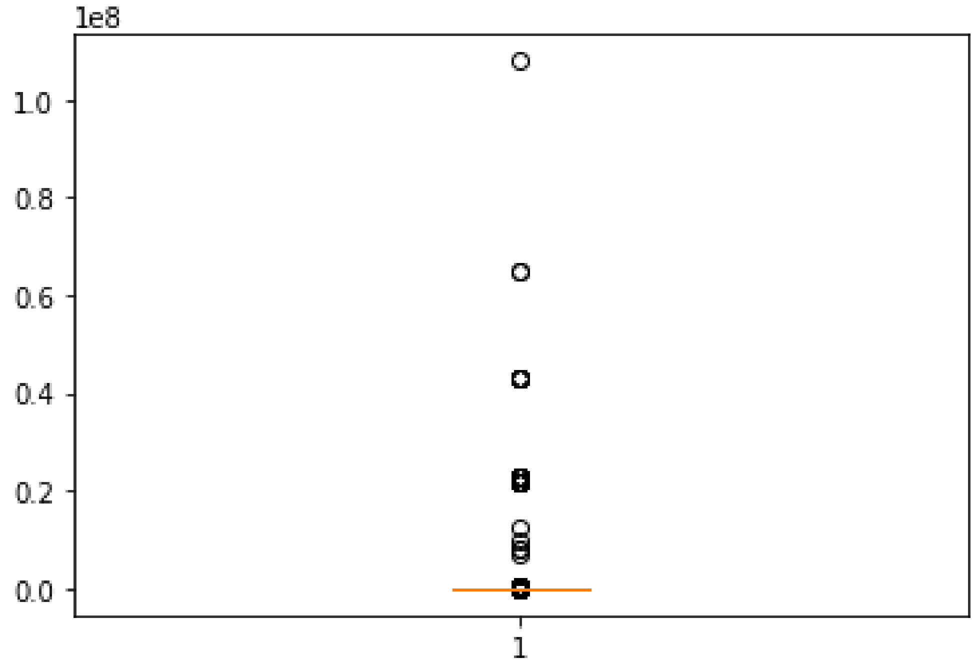
[24]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a2a56f390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a3623f550>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a30365d50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a32ae5590>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a2e394d90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a33abb5d0>]],
      dtype=object)
```



[25]:
```
plt.boxplot(CCAtrips['Trip Distance'])
```

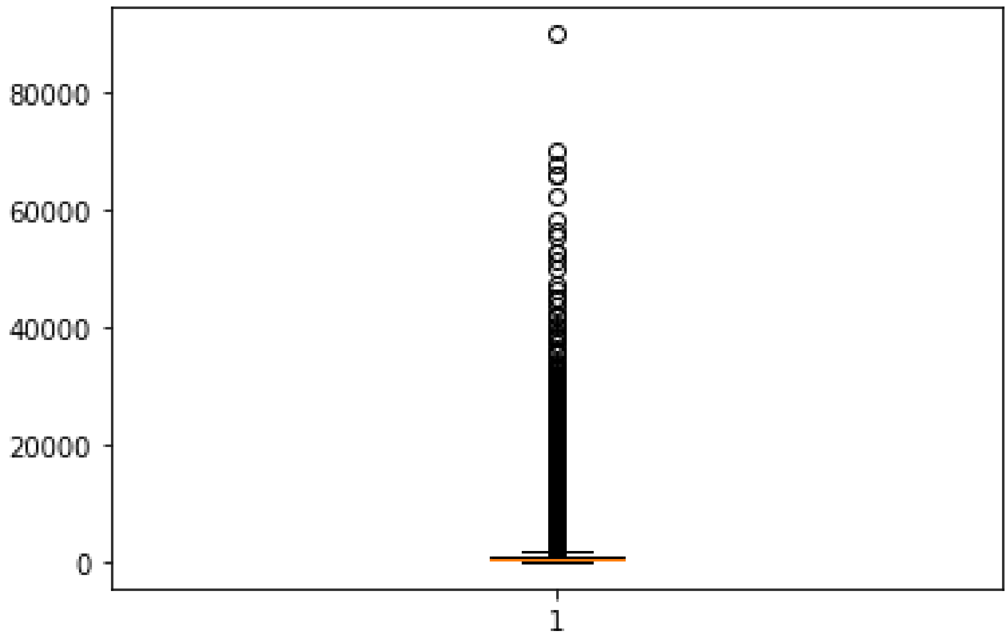
[25]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1a2a0bb9d0>,
  <matplotlib.lines.Line2D at 0x1a2a0bbf10>],
 'caps': [<matplotlib.lines.Line2D at 0x1a2a123450>,
  <matplotlib.lines.Line2D at 0x1a2a123950>],
 'boxes': [<matplotlib.lines.Line2D at 0x1a2a60eb10>],
```

```
'medians': [<matplotlib.lines.Line2D at 0x1a2a123e50>],
'fliers': [<matplotlib.lines.Line2D at 0x1a2a118cd0>],
'means': []}
```



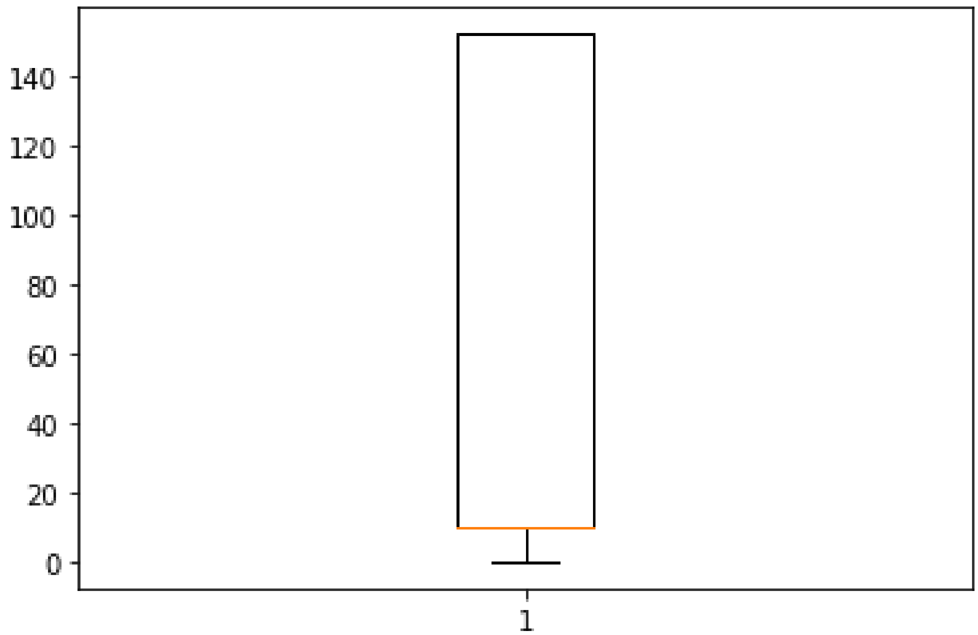
[26]:
```
plt.boxplot(CCAtrips['Trip Duration'])
```

[26]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1a2a1bc190>,
  <matplotlib.lines.Line2D at 0x1a2a1bc750>],
 'caps': [<matplotlib.lines.Line2D at 0x1a2a1bcc50>,
  <matplotlib.lines.Line2D at 0x1a2a1b65d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1a2a1b6590>],
 'medians': [<matplotlib.lines.Line2D at 0x1a2a1c46d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1a2a1c4bd0>],
 'means': []}
```



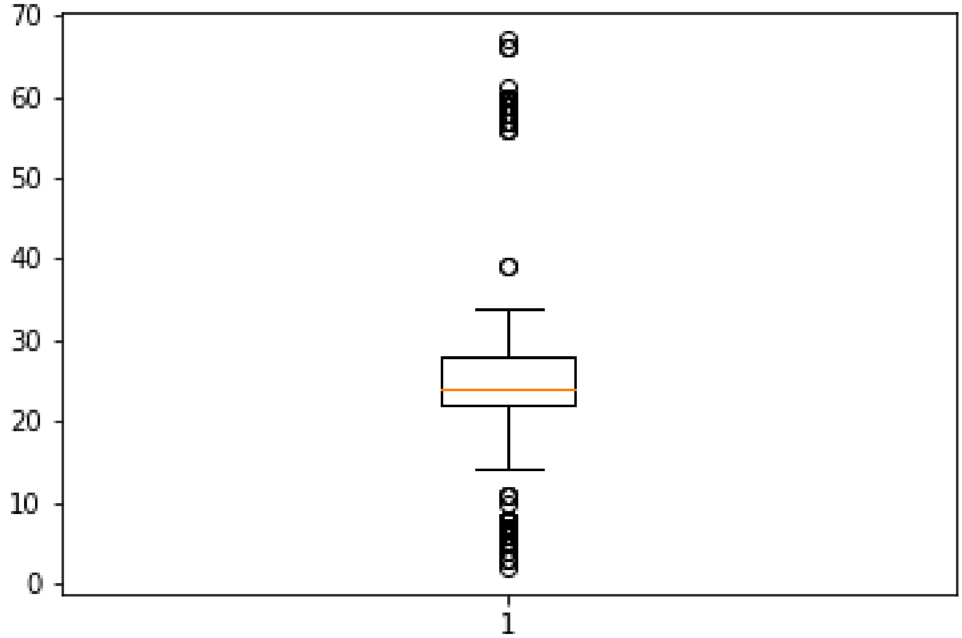
[27]:
```
plt.boxplot(CCAtrips['Accuracy'])
```

[27]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1a2a5de490>,
  <matplotlib.lines.Line2D at 0x1a2a5dea50>],
 'caps': [<matplotlib.lines.Line2D at 0x1a2a5def50>,
  <matplotlib.lines.Line2D at 0x1a2a5e6490>],
 'boxes': [<matplotlib.lines.Line2D at 0x1a2a588890>],
 'medians': [<matplotlib.lines.Line2D at 0x1a2a5e69d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1a2a5e6ed0>],
 'means': []}
```



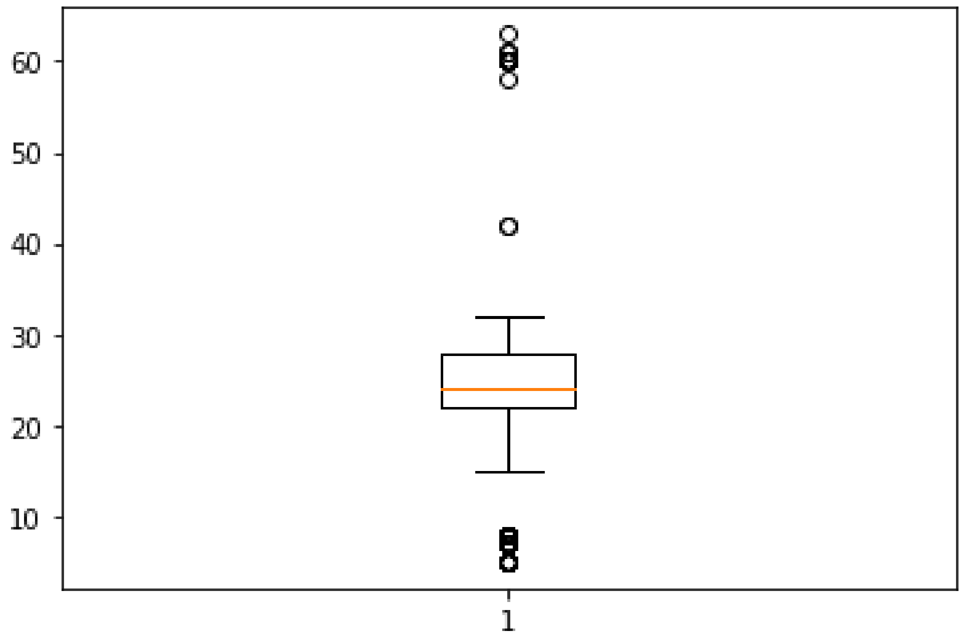
[28]:
```
plt.boxplot(CCAtrips['End Community Area Number'])
```

[28]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1a2b9317d0>,
  <matplotlib.lines.Line2D at 0x1a2b931d50>],
 'caps': [<matplotlib.lines.Line2D at 0x1a2b925c10>,
  <matplotlib.lines.Line2D at 0x1a2b934790>],
 'boxes': [<matplotlib.lines.Line2D at 0x1a2b925bd0>],
 'medians': [<matplotlib.lines.Line2D at 0x1a2b934cd0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1a2b931dd0>],
 'means': []}
```



[29]:
```
plt.boxplot(CCAtrips['Start Community Area Number'])
```

[29]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1a2db65ad0>,
  <matplotlib.lines.Line2D at 0x1a2db59f10>],
 'caps': [<matplotlib.lines.Line2D at 0x1a2db6a5d0>,
  <matplotlib.lines.Line2D at 0x1a2db6aad0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1a2db59ed0>],
 'medians': [<matplotlib.lines.Line2D at 0x1a2db65b90>],
 'fliers': [<matplotlib.lines.Line2D at 0x1a2db73550>],
 'means': []}
```



4.9 CCA Additional Data

[30]:
```
CDS['GEOG'] = CDS['GEOG'].str.upper()
CDS = CDS.rename(columns = {'GEOG':'community'})
```

[31]:
```
CDS = pd.merge(CCAcounts,CDS,on='community')
```

[32]:
```
CDS.head()
```

[32]:
```
         community     Count  2000_POP  2010_POP  TOT_POP  UND19  A20_34  \
0        WEST TOWN  148147.0     87435     82236    84502  13570   37651
1   NEAR WEST SIDE  148050.0     46419     54881    62872  11897   26796
2     LOGAN SQUARE   75102.0     82715     72791    73046  15288   27569
3  BELMONT CRAGIN   17005.0     78144     78743    79910  24695   17843
4           AUSTIN   16011.0    117527     98514    95260  26485   20778

   A35_49  A50_64  A65_74  …  highly_walkable_pop_pct  \
0   19528    8662    3234  …                 1.000000
1   12859    6779    2876  …                 1.000000
2   16595    8685    2888  …                 1.000000
3   17467   12996    4248  …                 1.000000
4   17210   18261    7751  …                 0.997556

   highly_walkable_emp_pct  assoc_plus_pct  in_lbr_frc_pct  \
```

```
0                  0.999832        0.694456        0.896667
1                  0.903230        0.701478        0.788495
2                  1.000000        0.576617        0.852237
3                  1.000000        0.181722        0.784830
4                  0.906326        0.215805        0.686323

   pct_pop_access_4_acres_per_1k  pct_pop_access_10_acres_per_1k  \
0                       0.095257                             0.0
1                       0.063530                             0.0
2                       0.000000                             0.0
3                       0.000000                             0.0
4                       0.000000                             0.0

   impervious_acres_per_hh  modhigh_ta_pop_pct  modhigh_ta_emp_pct  nonsov_pct
0                 0.057823                 1.0                 1.0    0.587723
1                 0.109953                 1.0                 1.0    0.636903
2                 0.057365                 1.0                 1.0    0.524368
3                 0.082042                 1.0                 1.0    0.347700
4                 0.090247                 1.0                 1.0    0.416497

[5 rows x 232 columns]
```

[33]:
```python
colnames = CDS.columns
print(colnames)
```

```
Index(['community', 'Count', '2000_POP', '2010_POP', 'TOT_POP', 'UND19',
       'A20_34', 'A35_49', 'A50_64', 'A65_74',
       …
       'highly_walkable_pop_pct', 'highly_walkable_emp_pct', 'assoc_plus_pct',
       'in_lbr_frc_pct', 'pct_pop_access_4_acres_per_1k',
       'pct_pop_access_10_acres_per_1k', 'impervious_acres_per_hh',
       'modhigh_ta_pop_pct', 'modhigh_ta_emp_pct', 'nonsov_pct'],
      dtype='object', length=232)
```

[34]:
```python
CDS.describe()
```

[34]:
```
               Count       2000_POP      2010_POP       TOT_POP  \
count      77.000000      77.000000     77.000000      77.000000
mean     6344.207792   37606.077922  35007.766234   35347.233766
std     25049.690302   24446.866243  22400.350739   22947.281631
min         0.000000    3294.000000   2876.000000    2254.000000
25%         0.000000   18165.000000  18109.000000   19019.000000
50%         0.000000   33694.000000  31028.000000   29929.000000
75%        28.000000   52723.000000  48743.000000   46278.000000
max    148147.000000  117527.000000  98514.000000  100470.000000

              UND19        A20_34        A35_49        A50_64       A65_74  \
```

```
count     77.000000     77.000000     77.000000     77.000000    77.000000
mean    8530.753247   9680.337662   7108.519481   5904.428571  2365.116883
std     5491.609061   9120.785819   4839.836352   3574.085339  1503.713990
min      381.000000    472.000000    349.000000    437.000000   229.000000
25%     4601.000000   3799.000000   3537.000000   3325.000000  1458.000000
50%     7554.000000   6459.000000   5599.000000   5094.000000  2161.000000
75%    11696.000000  12625.000000   9773.000000   8134.000000  2935.000000
max    26485.000000  49608.000000  19828.000000  18261.000000  8850.000000

            A75_84  …  highly_walkable_pop_pct  highly_walkable_emp_pct  \
count    77.000000  …                77.000000                77.000000
mean   1248.064935  …                 0.926932                 0.872398
std     789.873213  …                 0.173809                 0.238128
min      48.000000  …                 0.000000                 0.000000
25%     715.000000  …                 0.940533                 0.854935
50%    1092.000000  …                 1.000000                 0.995529
75%    1590.000000  …                 1.000000                 1.000000
max    4138.000000  …                 1.000000                 1.000000

       assoc_plus_pct  in_lbr_frc_pct  pct_pop_access_4_acres_per_1k  \
count       77.000000       77.000000                      77.000000
mean         0.372207        0.765484                       0.233190
std          0.207805        0.071177                       0.328929
min          0.106823        0.579680                       0.000000
25%          0.215805        0.729986                       0.000000
50%          0.327608        0.778427                       0.095257
75%          0.470011        0.805830                       0.276763
max          0.853270        0.906218                       1.000000

       pct_pop_access_10_acres_per_1k  impervious_acres_per_hh  \
count                       77.000000                77.000000
mean                         0.042198                 0.121383
std                          0.167956                 0.110569
min                          0.000000                 0.026010
25%                          0.000000                 0.072342
50%                          0.000000                 0.100300
75%                          0.000000                 0.118719
max                          1.000000                 0.767034

       modhigh_ta_pop_pct  modhigh_ta_emp_pct  nonsov_pct
count           77.000000           77.000000   77.000000
mean             0.980183            0.962884    0.447339
std              0.121431            0.158907    0.131273
min              0.013487            0.041012    0.172889
25%              1.000000            1.000000    0.347700
50%              1.000000            1.000000    0.465169
75%              1.000000            1.000000    0.529383
```

```
max              1.000000            1.000000    0.752025

[8 rows x 211 columns]
```

4.10 Creating Relevant Features:

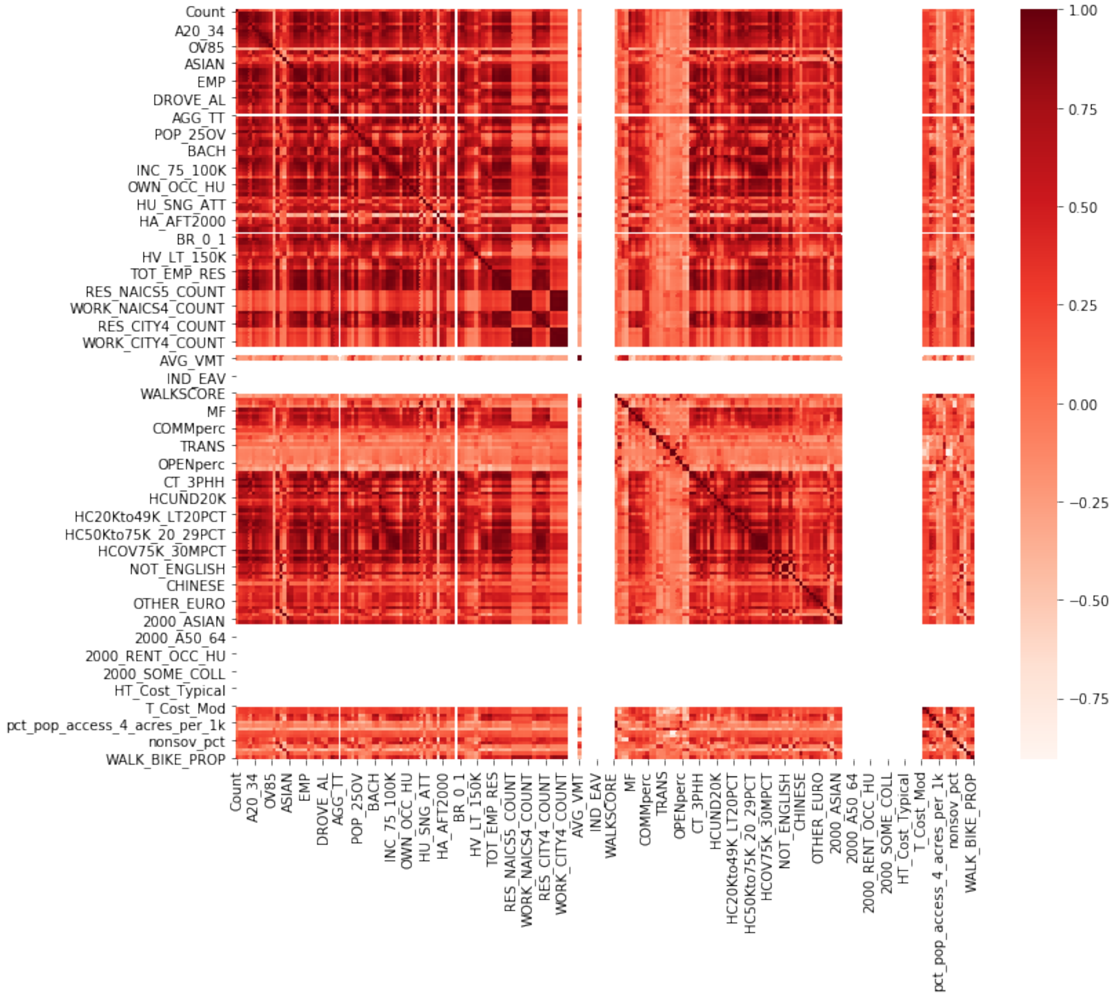
[35]:
```python
# Percentage of people in an area who identify as a certain race:
CDS['WHITE_PERC'] = (CDS['WHITE']/CDS['TOT_POP'])*100
CDS['HISP_PERC'] = (CDS['HISP']/CDS['TOT_POP'])*100
CDS['BLACK_PERC'] = (CDS['BLACK']/CDS['TOT_POP'])*100
CDS['ASIAN_PERC'] = (CDS['ASIAN']/CDS['TOT_POP'])*100

CDS['WALK_BIKE_PROP'] = CDS['WALK_BIKE']/CDS['TOT_POP']
```

5 Q2: Data Visualization

5.1 Connection

I will use a correlation heatmap to analyze the correlation between different variables. This should help me initially eliminate some features which many be redundant in a regression model and illuminate so features which might be related to high demand/ridership.

[36]:
```python
fig=plt.figure(figsize=(12,10))
cor = CDS.corr()
sns.heatmap(cor,cmap=plt.cm.Reds)
plt.show()
```




```
[37]: cor_target = abs(cor[colnames[1]])
      #Selecting highly correlated features
      relevant_features = cor_target[cor_target>0.5]
      relevant_features
```

```
[37]: Count              1.000000
      HU_3_4UN           0.549562
      RES_CITY4_COUNT    0.539519
      RES_CITY5_COUNT    0.502492
      MIX                0.622276
      Name: Count, dtype: float64
```

I think this visualization type is informative because it allows us to make a little bit of sense out of the over 200 features in the dataset. From the correlation data, we were able to find that HU_3_4UN, RES_CITY4_COUNT, RES_CITY5_COUNT, and MIX are all highly correlated with the count/ridership data. The meaning behind those features is as follows:

```
HU_3_4UN - number of 3 or 4 unit housing types
RES_CITY4_COUNT - Employment of Community residents by residence location 4
RES_CITY5_COUNT - Employment of Community residents by residence location 5
MIX - Mixed Use Acres
```

Additionally, the colors of the heatmap allow us to not just how well the features relate to the dependent variable, but also to other features. On the heatmap, there are lots of areas which are eaither white or deep red. According to the lengend, white means highly negative correlation and deep red means highly positive correlation. Highly positively or negatively correlated features can most likely be subsituted in favor of choosing one feature which would explain the whole group. This could be a useful tool in reducing the dimensions when creating a model.

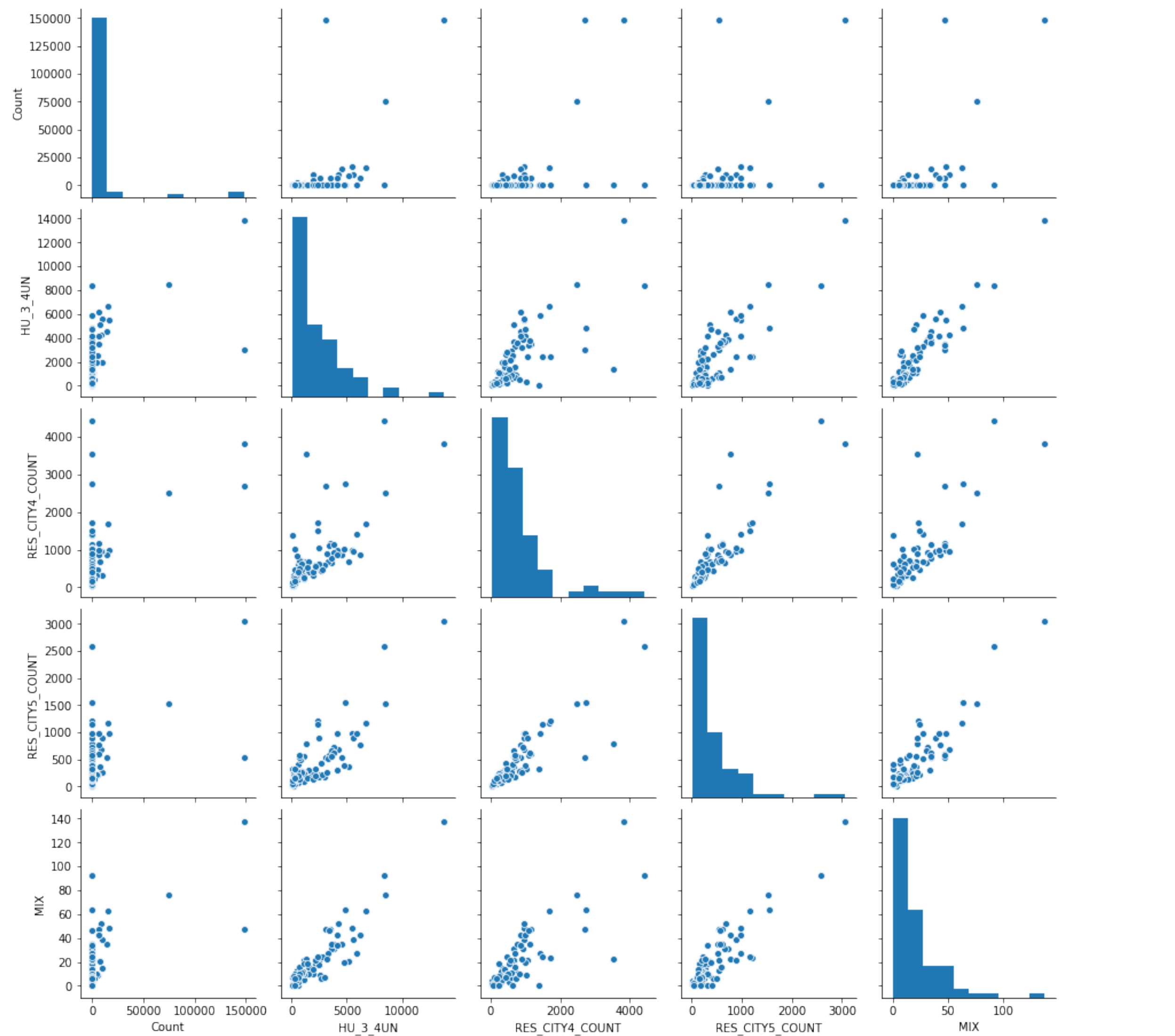
5.2 Distribution

Exploring the distributions of important features as decided by their correlations.

[38]:
```
selectCDS = CDS[['Count','HU_3_4UN','RES_CITY4_COUNT','RES_CITY5_COUNT','MIX']]
```

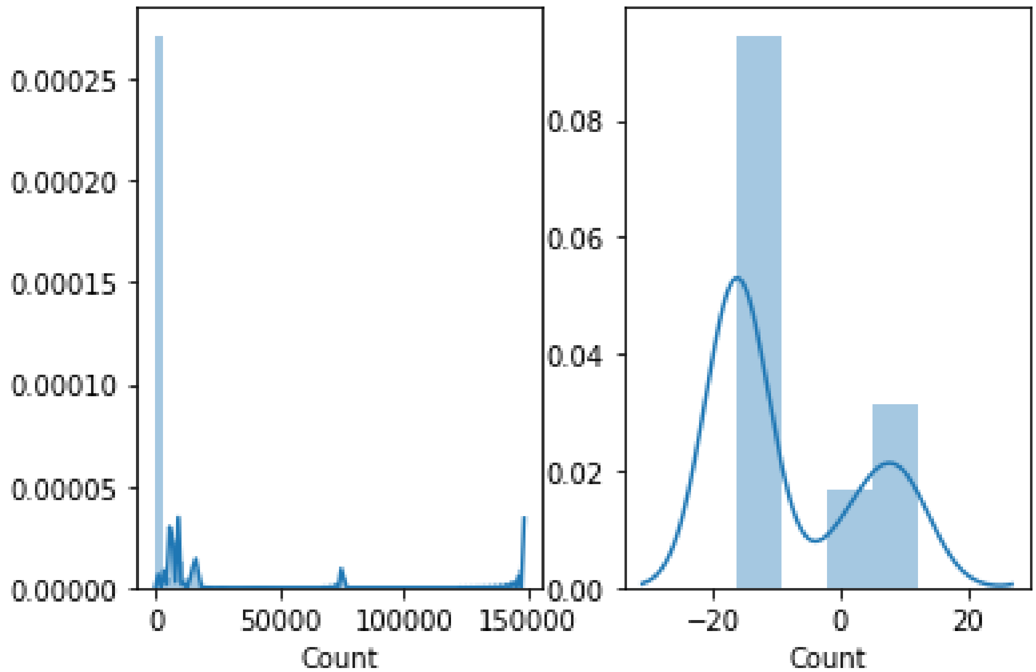
[39]:
```
sns.pairplot(selectCDS, palette="husl")
```

[39]:
```
<seaborn.axisgrid.PairGrid at 0x1a2f9bb150>
```



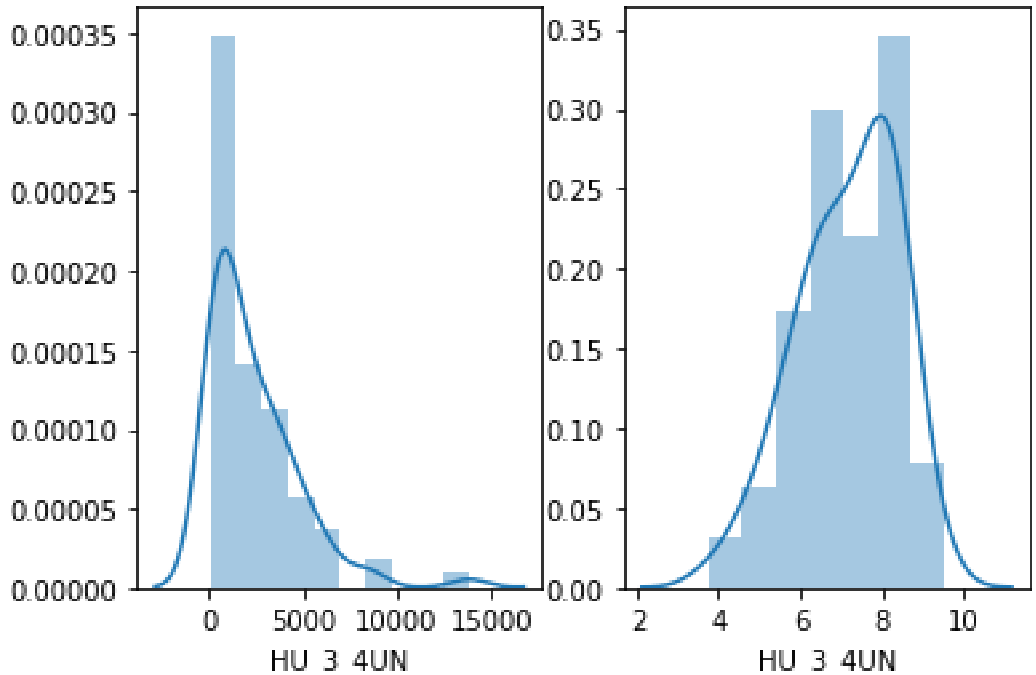
[40]:
```
fig, ax =plt.subplots(1,2)
#sns.countplot(df['batting'], ax=ax[0])
#sns.countplot(df['bowling'], ax=ax[1])
sns.distplot(selectCDS['Count'], ax=ax[0])
sns.distplot(np.log(selectCDS['Count'] + .0000001), ax=ax[1])
fig.show()
```

```
/Users/charlesmarshall/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```



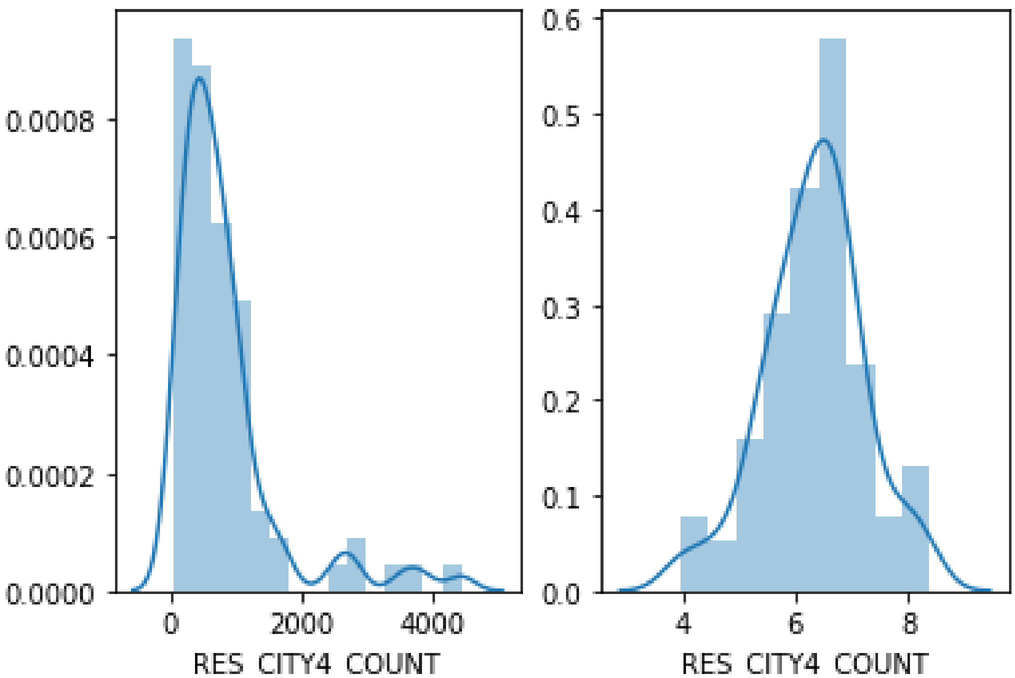
[41]:
```
fig, ax =plt.subplots(1,2)
#sns.countplot(df['batting'], ax=ax[0])
#sns.countplot(df['bowling'], ax=ax[1])
sns.distplot(selectCDS['HU_3_4UN'], ax=ax[0])
sns.distplot(np.log(selectCDS['HU_3_4UN'] + .0000001), ax=ax[1])
fig.show()
```

```
/Users/charlesmarshall/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```



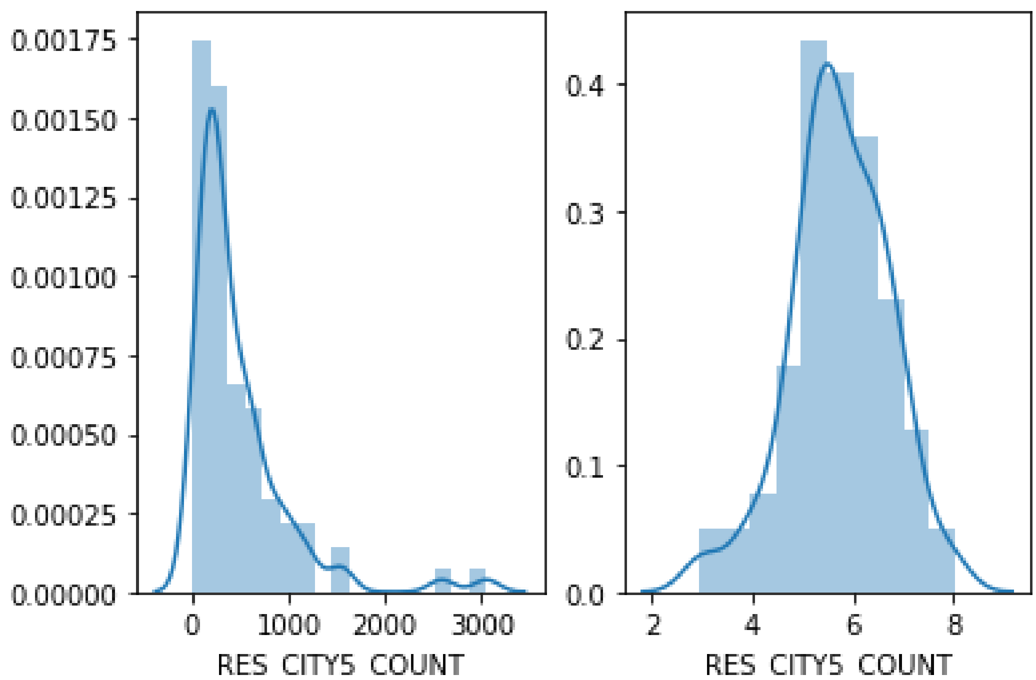
[42]:
```
fig, ax =plt.subplots(1,2)
#sns.countplot(df['batting'], ax=ax[0])
#sns.countplot(df['bowling'], ax=ax[1])
sns.distplot(selectCDS['RES_CITY4_COUNT'], ax=ax[0])
sns.distplot(np.log(selectCDS['RES_CITY4_COUNT'] + .0000001), ax=ax[1])
fig.show()
```

```
/Users/charlesmarshall/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```



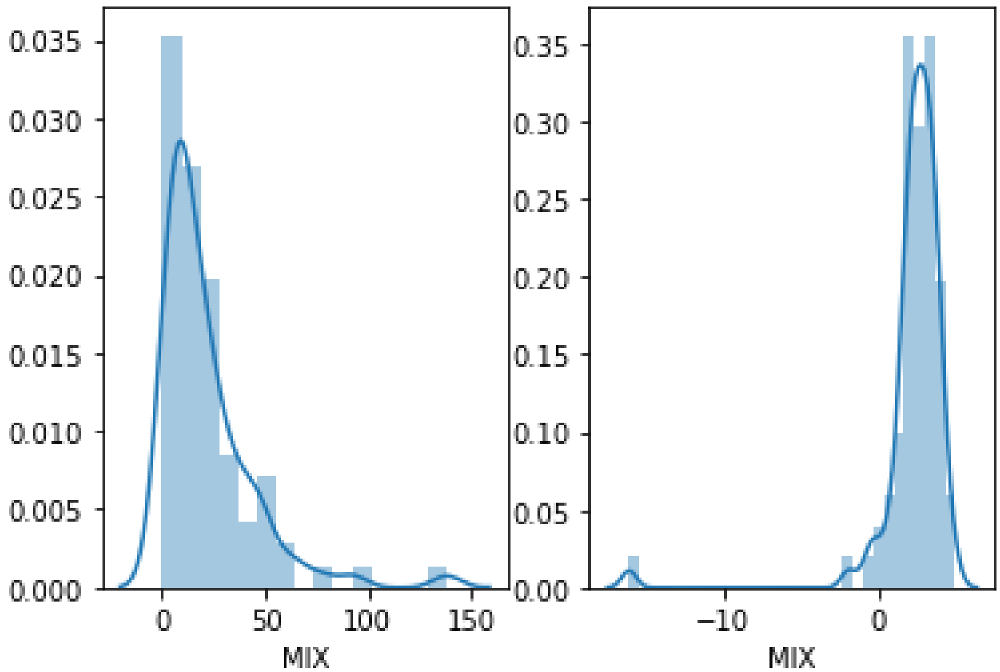
[43]:
```
fig, ax =plt.subplots(1,2)
#sns.countplot(df['batting'], ax=ax[0])
#sns.countplot(df['bowling'], ax=ax[1])
sns.distplot(selectCDS['RES_CITY5_COUNT'], ax=ax[0])
sns.distplot(np.log(selectCDS['RES_CITY5_COUNT'] + .0000001), ax=ax[1])
fig.show()
```

```
/Users/charlesmarshall/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```



[44]:
```
fig, ax =plt.subplots(1,2)
#sns.countplot(df['batting'], ax=ax[0])
#sns.countplot(df['bowling'], ax=ax[1])
sns.distplot(selectCDS['MIX'], ax=ax[0])
sns.distplot(np.log(selectCDS['MIX'] + .0000001), ax=ax[1])
fig.show()
```

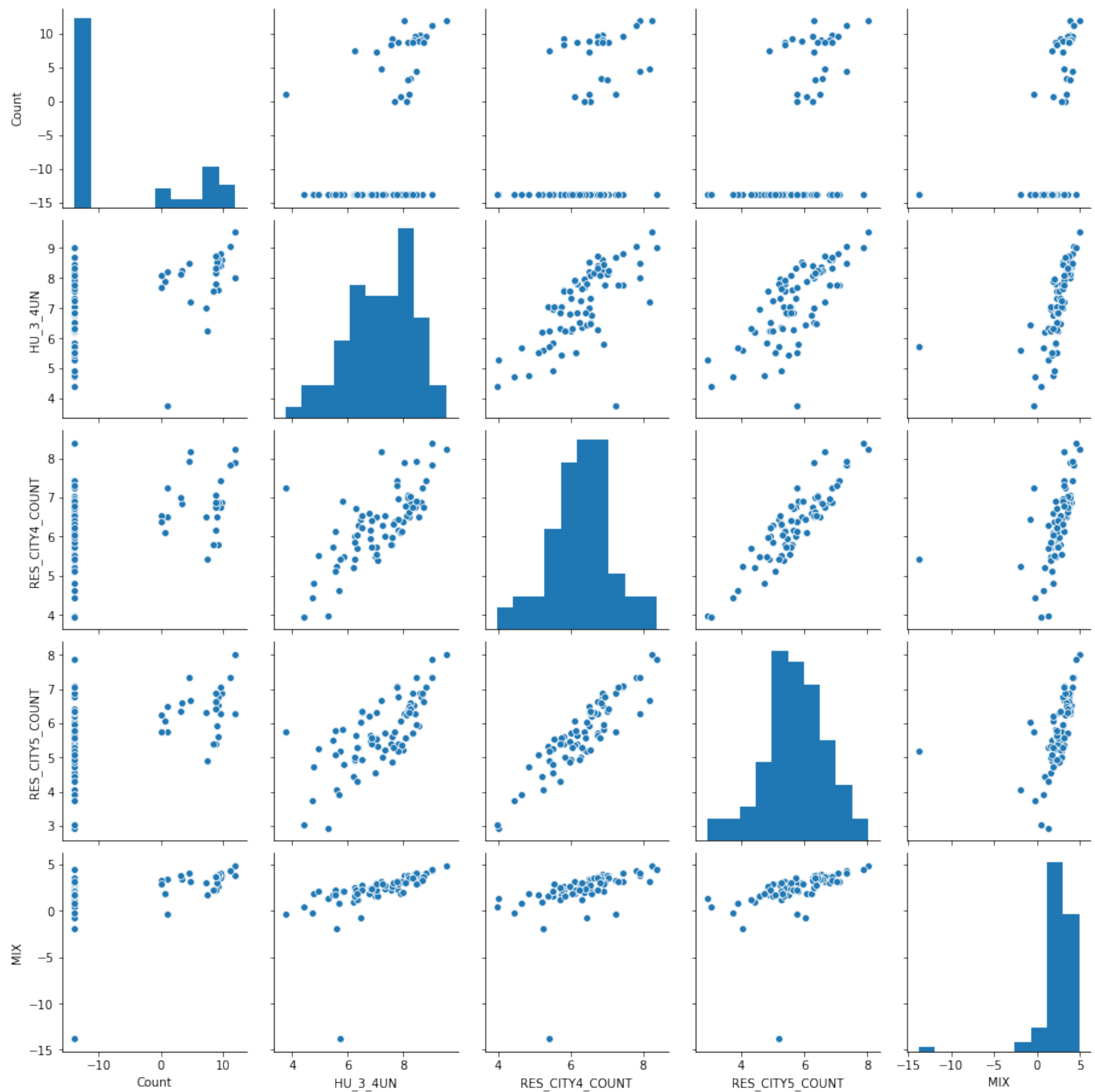
```
/Users/charlesmarshall/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```



[45]:
```
logCDS = np.log(selectCDS + .000001)
```

[46]:
```
sns.pairplot(logCDS, palette="husl")
```

[46]: `<seaborn.axisgrid.PairGrid at 0x1a2f8a6bd0>`



Exploring the distributions of the data was extremely intersting. First, I created a dataframe with the features that were recognized as having a high correlation with the counts/ridership data. When I plotted all of them using a seaborn pairplot, I found that all of the datasets were extremely right skewed. To remedy this, I fit all the columns to with a log transformation. Once the log transformation was applied, all the data looked nearly normally distributed. The counts data resembles a bi-modal normal distribution when the log transformation is applied. Having normally distributed data should be very helpful when modeling because it will allow me to standardize to improve my model.

6 Q3: Data Visualization and Analysis

1) For this question I decided to experiment with the Folium library in Python. I decided to use this tool because I wanted to find a way to visually express the difference between various Chicago Community Areas (CCA) using geospatial data. When I researched, Folium was recommended as an easy to use GIS tool in Python. My experience while using the tool was relatively good. One of the hardest parts was getting the CCA map into the right file format, a GeoJson file so that it could be used in Folium. Other than that, the classes I used, mainly the GeoJson and Cloropleth classes were well described and examples were given in the folium documentation.

[47]:
```
shape.head()
```

[47]:
```
   area area_num_1 area_numbe  comarea  comarea_id        community  \
0   0.0         35         35      0.0         0.0          DOUGLAS
1   0.0         36         36      0.0         0.0          OAKLAND
2   0.0         37         37      0.0         0.0      FULLER PARK
3   0.0         38         38      0.0         0.0  GRAND BOULEVARD
4   0.0         39         39      0.0         0.0          KENWOOD

   perimeter    shape_area     shape_len  \
0        0.0  4.600462e+07  31027.054510
1        0.0  1.691396e+07  19565.506153
2        0.0  1.991670e+07  25339.089750
3        0.0  4.849250e+07  28196.837157
4        0.0  2.907174e+07  23325.167906

                                            geometry
0  POLYGON ((-87.60914 41.84469, -87.60915 41.844…
1  POLYGON ((-87.59215 41.81693, -87.59231 41.816…
2  POLYGON ((-87.62880 41.80189, -87.62879 41.801…
3  POLYGON ((-87.60671 41.81681, -87.60670 41.816…
4  POLYGON ((-87.59215 41.81693, -87.59215 41.816…
```

[48]:
```
shape.columns
```

[48]:
```
Index(['area', 'area_num_1', 'area_numbe', 'comarea', 'comarea_id',
       'community', 'perimeter', 'shape_area', 'shape_len', 'geometry'],
      dtype='object')
```

[49]:
```
chicago_map = folium.Map(location=[41.8781, -87.6298],zoom_start=11)
```

[50]:
```
state_geo = os.path.join('/Users/charlesmarshall/Desktop/CIVE495/FinalProject/
 ↪','CCAoutline.json')
```

6.1 E-Scooter Pilot Area Visualizations

2)

I decided to use folium maps to display ridership and other demographic data. Since the E-Scooters were distributed every morning throughout the pilot area, which ran through 17 diverse and distinct community areas, I wanted to focus specifically on performance and demographic data in this specific area. The hardest part of the process was using the folium library for the first time and running into a lot of errors and problems that a beginning user of the library would make. I was able to reference articles and documentation to fix a lot of my problems, but some of them were specific to what I was trying to do so I had to figure them out on my own. One thing I found particularly challenging was overlaying the choropleth and pilot area layers which allowed me to create a distinction between the pilot area and other community areas.

Overall, I found the visulaization tools to be helpful in spearheading analysis. Looking at the ridership map, three areas on the east side of the pilot area, closest to the downtown area saw the highest overall ridership by a wide margin. From analyzing the demograph data, these three areas also had the youngest median age (early-mid 20s) and were near the top of the highest median income (around $70k). This means that there might be a connection between ridership, age, and income. It would make sense that younger people are more likely to use electic scooters because they have a reputation as not the safest form of micro-transit. Additionally, people with more income would have more disposable income and might be willing to spend it on a leisure scooter ride or try a new form of transit.

However, although these factors seem to show promise, thhe high ridership in these areas might be due to other factors. For instance these three areas are close to the city and therefore probably see more tourist traffic than areas further away from the city. Tourists could take advantage of using e-scooters to get around the city with ease because they might not have a car. It would be worth it to further explore adding features about tourist traffic in each area when building the model in the future.

6.1.1 Pilot Area

[51]:
```
pilot_area = ['AUSTIN','HUMBOLDT PARK','WEST TOWN','WEST GARFIELD PARK',
              'EAST GARFIELD PARK','NEAR WEST SIDE','NORTH LAWNDALE',
              'SOUTH LAWNDALE','LOWER WEST SIDE','MONTCLARE','BELMONT CRAGIN',
              'HERMOSA','AVONDALE','LOGAN SQUARE','DUNNING','PORTAGE␣
 ↪PARK','IRVING PARK']
```

[52]:
```
chicago_map = folium.Map(location=[41.8781, -87.6298],zoom_start=11)

folium.GeoJson(state_geo,style_function=lambda feature: {
        'fillColor': 'blue' if feature['properties']['community'] in pilot_area␣
 ↪else 'Grey',
        'color': 'black',
        'weight': 2,
        'dashArray': '5, 5'
    }).add_to(chicago_map)
```

```
folium.LayerControl().add_to(chicago_map)

chicago_map
```

[52]: <folium.folium.Map at 0x1a31156110>

6.1.2 Ridership by Community Area During Pilot Period:

[53]:
```
chicago_map = folium.Map(location=[41.8781, -87.6298],zoom_start=11)

folium.Choropleth(
    geo_data=state_geo,
    data = CDS,
    columns = ['community','Count'],
    key_on = 'feature.properties.community',

    legend_name='Number of Rides During Pilot Period',
    highlight=True
).add_to(chicago_map)

folium.GeoJson(state_geo,style_function=lambda feature: {
        'fillColor': 'none' if feature['properties']['community'] in pilot_area␣
 ↪else 'Grey',
        'fillOpacity': 0.5,
        'color': 'black',
        'weight': 2,
        'dashArray': '5, 5'
    }).add_to(chicago_map)

folium.LayerControl().add_to(chicago_map)

chicago_map
```

[53]: <folium.folium.Map at 0x1a335aed90>

6.2 Demographics

6.2.1 Comparing Population between Community Areas:

[54]:
```
chicago_map = folium.Map(location=[41.8781, -87.6298],zoom_start=11)

folium.Choropleth(
    geo_data=state_geo,
```

```
    data = CDS,
    columns = ['community','TOT_POP'],
    key_on = 'feature.properties.community',
    legend_name='Total Population',
    highlight=True
).add_to(chicago_map)

folium.GeoJson(state_geo,style_function=lambda feature: {
        'fillColor': 'none' if feature['properties']['community'] in pilot_area␣
 ↪else 'Grey',
        'fillOpacity': 0.5,
        'color': 'black',
        'weight': 2,
        'dashArray': '5, 5'
    }).add_to(chicago_map)

folium.LayerControl().add_to(chicago_map)

chicago_map
```

[54]: <folium.folium.Map at 0x1a36ac8610>

6.2.2 Comparing Racial Diversity between Community Areas:

[55]:
```
chicago_map = folium.Map(location=[41.8781, -87.6298],zoom_start=11)

folium.Choropleth(
    geo_data=state_geo,
    data = CDS,
    columns = ['community','WHITE_PERC'],
    key_on = 'feature.properties.community',
    legend_name='Percent White (%)',
    highlight=True
).add_to(chicago_map)

folium.GeoJson(state_geo,style_function=lambda feature: {
        'fillColor': 'none' if feature['properties']['community'] in pilot_area␣
 ↪else 'Grey',
        'fillOpacity': 0.5,
        'color': 'black',
        'weight': 2,
        'dashArray': '5, 5'
    }).add_to(chicago_map)

folium.LayerControl().add_to(chicago_map)
```

```
chicago_map
```

[55]: <folium.folium.Map at 0x1a383a01d0>

[56]:
```
chicago_map = folium.Map(location=[41.8781, -87.6298],zoom_start=11)

folium.Choropleth(
    geo_data=state_geo,
    data = CDS,
    columns = ['community','BLACK_PERC'],
    key_on = 'feature.properties.community',
    legend_name='Proportion Black',
    highlight=True
).add_to(chicago_map)

folium.GeoJson(state_geo,style_function=lambda feature: {
        'fillColor': 'none' if feature['properties']['community'] in pilot_area␣
 ↪else 'Grey',
        'fillOpacity': 0.5,
        'color': 'black',
        'weight': 2,
        'dashArray': '5, 5'
    }).add_to(chicago_map)

folium.LayerControl().add_to(chicago_map)

chicago_map
```

[56]: <folium.folium.Map at 0x1a2fd13950>

[57]:
```
chicago_map = folium.Map(location=[41.8781, -87.6298],zoom_start=11)

folium.Choropleth(
    geo_data=state_geo,
    data = CDS,
    columns = ['community','HISP_PERC'],
    key_on = 'feature.properties.community',
    legend_name='Proportion Hispanic',
    highlight=True
).add_to(chicago_map)

folium.GeoJson(state_geo,style_function=lambda feature: {
        'fillColor': 'none' if feature['properties']['community'] in pilot_area␣
 ↪else 'Grey',
        'fillOpacity': 0.5,
        'color': 'black',
        'weight': 2,
```

```
        'dashArray': '5, 5'
    }).add_to(chicago_map)

folium.LayerControl().add_to(chicago_map)

chicago_map
```

[57]: <folium.folium.Map at 0x1a3cb01b90>

6.2.3 Comparing Median Age between Community Areas:

[58]:
```
chicago_map = folium.Map(location=[41.8781, -87.6298],zoom_start=11)

folium.Choropleth(
    geo_data=state_geo,
    data = CDS,
    columns = ['community','MED_AGE'],
    key_on = 'feature.properties.community',
    legend_name='Median Age',
    highlight=True
).add_to(chicago_map)

folium.GeoJson(state_geo,style_function=lambda feature: {
        'fillColor': 'none' if feature['properties']['community'] in pilot_area␣
 ↪else 'Grey',
        'fillOpacity': 0.5,
        'color': 'black',
        'weight': 2,
        'dashArray': '5, 5'
    }).add_to(chicago_map)

folium.LayerControl().add_to(chicago_map)

chicago_map
```

[58]: <folium.folium.Map at 0x1a3f37ea90>

6.2.4 Comparing Median Income between Community Areas:

[59]:
```
chicago_map = folium.Map(location=[41.8781, -87.6298],zoom_start=11)

folium.Choropleth(
    geo_data=state_geo,
    data = CDS,
```

```
    columns = ['community','MEDINC'],
    key_on = 'feature.properties.community',
    legend_name='Median Income',
    highlight=True
).add_to(chicago_map)

folium.GeoJson(state_geo,style_function=lambda feature: {
        'fillColor': 'none' if feature['properties']['community'] in pilot_area␣
 ↪else 'Grey',
        'fillOpacity': 0.5,
        'color': 'black',
        'weight': 2,
        'dashArray': '5, 5'
    }).add_to(chicago_map)

folium.LayerControl().add_to(chicago_map)

chicago_map
```

[59]: <folium.folium.Map at 0x1a418882d0>

6.2.5 Proportion of Trips to work via non-SOV modes:

An SOV is a single occupancy vehicle. Popular non-SOV vehicles include public transit, carpooling, busses, bikes, and walking. Non-SOV percentage is an indicator of mobility in an area.

[60]:
```
chicago_map = folium.Map(location=[41.8781, -87.6298],zoom_start=11)

folium.Choropleth(
    geo_data=state_geo,
    data = CDS,
    columns = ['community','nonsov_pct'],
    key_on = 'feature.properties.community',
    legend_name='Proportion of trips to work via non-SOV modes',
    highlight=True
).add_to(chicago_map)

folium.GeoJson(state_geo,style_function=lambda feature: {
        'fillColor': 'none' if feature['properties']['community'] in pilot_area␣
 ↪else 'Grey',
        'fillOpacity': 0.5,
        'color': 'black',
        'weight': 2,
        'dashArray': '5, 5'
    }).add_to(chicago_map)
```

```
folium.LayerControl().add_to(chicago_map)

chicago_map
```

[60]: <folium.folium.Map at 0x1a3cf011d0>

6.3 Further Analysis/Next Steps:

3) Three interesing questions about the dataset:

```
1) Given the ride counts and the Community Area demographic data for the pilot area, can

2) What factors (demographic, mobility metrics, socioeconomic, etc) are most significant

3) There are a number of community areas within the pilot area which did not have a very
```

[ ]:

```
```