notebook_3_ma_census_block_and_town_match

October 19, 2019

1 MA cencus block and town join, basic geospatial EDA

1.0.1 Goals:

- matching different geospatial systems/crs across files using shapefiles - join the census block and towns
- derive the x and y coordinates for each town for modeling from the shapefile (centroid)

1.0.2 Accomplishments:

- Derived latitude and longitude of the centroid of each of the 351 towns/municipalities in the MA opioid overdose dataset to use as the geospatial component when building a model.
- Joined towns (351 towns/cities in the MA opioid overdose dataset) and 2010 census blocks to pull in the American Community Survey demographics data and use it as predictors of opioid overdose deaths per year per town.
- Explored the merge error by comparing the 2010 population counts from the town survey shapefile and the sum after merge of the 2010 population counts from the census block shapefile

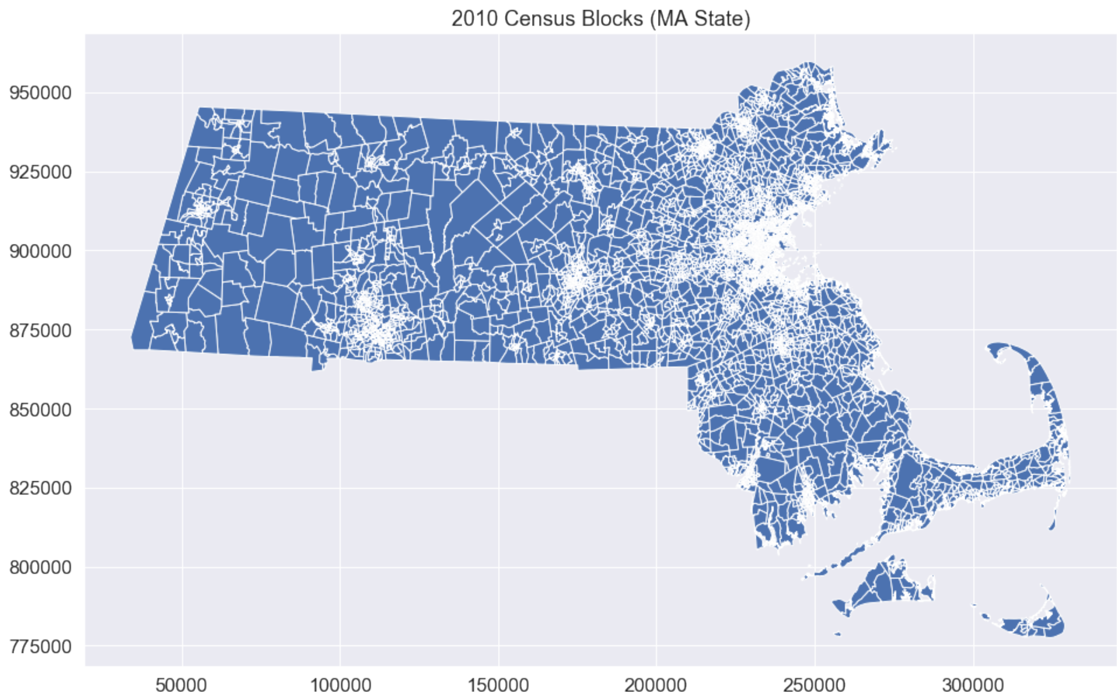
1.0.3 Outputs:

- data/tidy_data/ma_town_crs4326_coords.csv - MA tows with centroid x/y points for modeling
- data/tidy_data/census_block_town_match.csv - association of MA towns with 2010 census blocks (for ACS data merge) - not perfect, some errors, but will go with this
- data/tidy_data/census_block_town_match_2010pop_error.csv - 2010 population count errors after centroid census block and town survey join
- pdf notebook report: products/notebook_3_ma_census_block_and_town_match.pdf

[1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import geopandas as gpd
import geoplot
sns.set_style('darkgrid')
sns.set(font_scale=1.5)
```

[2]:
```
death_data = pd.read_csv("../../data/tidy_data/
↪ma_town_opioid_overdose_death_by_place_of_death_2014_to_2018.csv")
# census blocks map
census_blocks = gpd.read_file("../../data/raw_data/
↪shapefiles_and_geography_related/CENSUS2010_BLK_BG_TRCT_SHP/
↪CENSUS2010BLOCKGROUPS_POLY.shp")
town_map = gpd.read_file("../../data/raw_data/shapefiles_and_geography_related/
↪townssurvey_shp/TOWNSSURVEY_POLYM.shp")
town_map_alt = gpd.read_file("../../data/raw_data/
↪shapefiles_and_geography_related/townssurvey_shp/TOWNSSURVEY_POLY.shp")
```

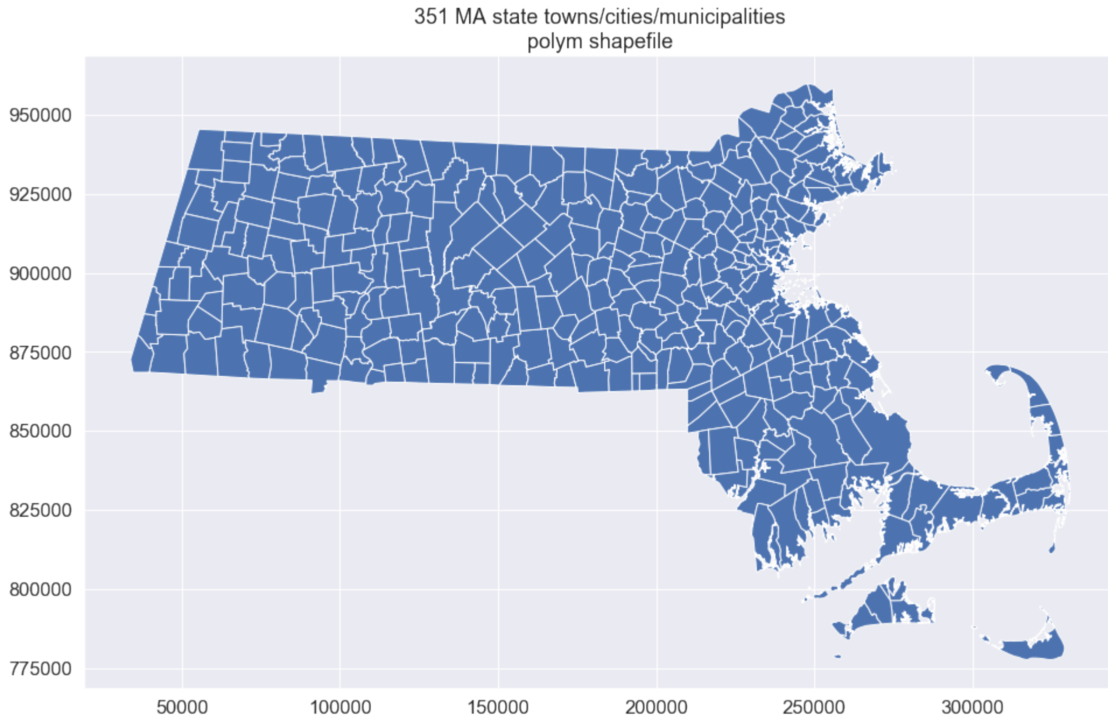
[3]:
```
census_blocks.plot(figsize=(16,10))
plt.title("2010 Census Blocks (MA State)")
plt.show()
```



Is there a visual difference between the 2 town border survey shapefiles?

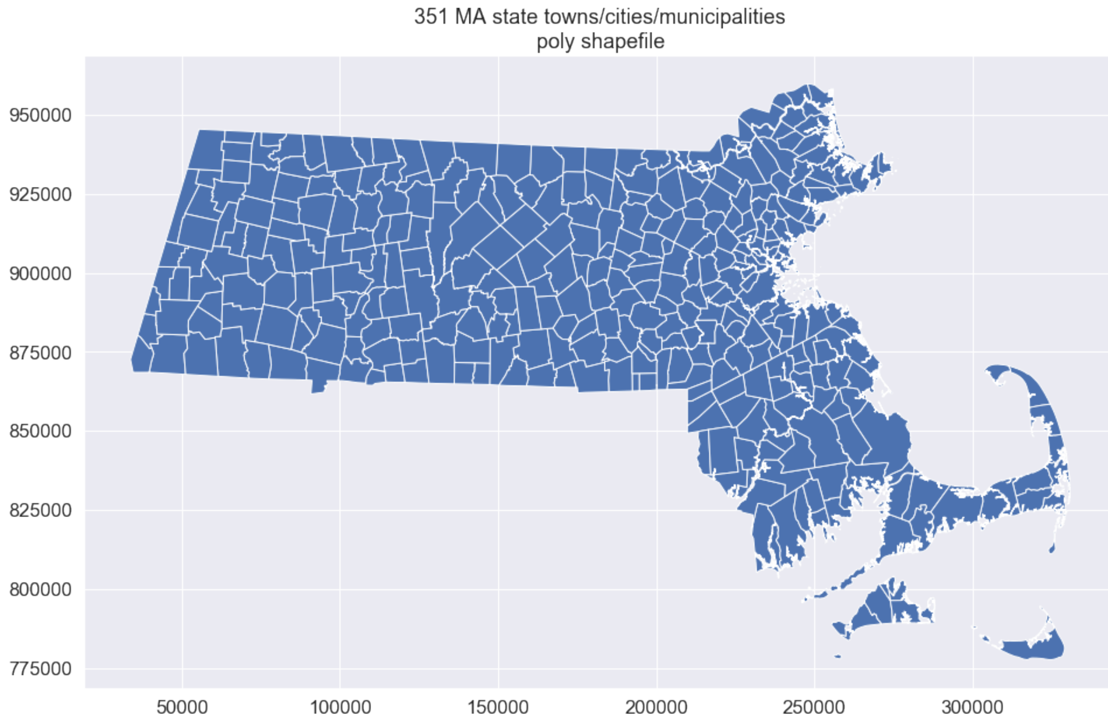
[4]:
```
print(town_map.shape)
town_map.plot(figsize=(16,10))
plt.title("351 MA state towns/cities/municipalities\npolym shapefile")
plt.show()
```

```
(351, 17)
```



[5]:
```
print(town_map_alt.shape)
town_map_alt.plot(figsize=(16,10))
plt.title("351 MA state towns/cities/municipalities\npoly shapefile")
plt.show()
```

```
(1243, 23)
```



More rows in the poly shapefile (vs the polyM), but maps look the same?

[6]:
```
# what's inside the shapefiles?
print(census_blocks.columns)
census_blocks.head()
```

```
Index(['STATEFP10', 'COUNTYFP10', 'TRACTCE10', 'BLKGRPCE10', 'GEOID10',
       'NAMELSAD10', 'MTFCC10', 'ALAND10', 'AWATER10', 'INTPTLAT10',
       'INTPTLON10', 'AREA_SQFT', 'AREA_ACRES', 'POP100_RE', 'HU100_RE',
       'LOGPL94171', 'LOGSF1', 'LOGACS0610', 'LOGSF1C', 'SHAPE_AREA',
       'SHAPE_LEN', 'geometry'],
      dtype='object')
```

[6]:
```
  STATEFP10 COUNTYFP10 TRACTCE10 BLKGRPCE10       GEOID10     NAMELSAD10  \
0        25        023    525104          2  250235251042  Block Group 2
1        25        023    525104          4  250235251044  Block Group 4
2        25        023    525203          1  250235252031  Block Group 1
3        25        023    510100          2  250235101002  Block Group 2
4        25        023    510100          3  250235101003  Block Group 3

  MTFCC10    ALAND10  AWATER10   INTPTLAT10  ... AREA_ACRES  POP100_RE  \
0   G5030  2648651.0  119260.0  +41.9751132  ...   683.9256       1120
1   G5030  4625818.0   11563.0  +41.9677679  ...  1145.8539       2178
2   G5030  2367037.0   62136.0  +42.0051872  ...   600.2231       1540
3   G5030   686351.0       0.0  +42.1115078  ...   169.5890       1172
```

```
4   G5030   403906.0       0.0  +42.1115626  ...    99.8001        792

   HU100_RE  LOGPL94171  LOGSF1 LOGACS0610  LOGSF1C    SHAPE_AREA  \
0       404     0159334  137632    0007882  0137632  2.767760e+06
1       690     0159336  137634    0007884  0137634  4.637125e+06
2       527     0159338  137636    0007885  0137636  2.429027e+06
3       414     0159161  137459    0007745  0137459  6.863050e+05
4       290     0159162  137460    0007746  0137460  4.038783e+05

     SHAPE_LEN                                          geometry
0  8963.749528  POLYGON ((245073.4579000026 857943.9572999999,...
1  9555.602586  POLYGON ((242521.254900001 859747.8350999989, ...
2  9226.194871  POLYGON ((244276.8404999971 862120.2314999998,...
3  3338.839737  POLYGON ((241369.4518999979 874137.570700001, ...
4  3178.436538  POLYGON ((240747.7463999987 873189.6048000008,...

[5 rows x 22 columns]
```

[7]:
```python
print(town_map.columns)
print(town_map_alt.columns)
town_map.head()
```

```
Index(['TOWN', 'TOWN_ID', 'POP1980', 'POP1990', 'POP2000', 'POPCH80_90',
       'POPCH90_00', 'TYPE', 'FOURCOLOR', 'FIPS_STCO', 'SUM_ACRES',
       'SUM_SQUARE', 'POP2010', 'POPCH00_10', 'SHAPE_Leng', 'SHAPE_Area',
       'geometry'],
      dtype='object')
Index(['TOWN', 'TOWN_ID', 'POP1980', 'POP1990', 'POP2000', 'POPCH80_90',
       'POPCH90_00', 'TYPE', 'ISLAND', 'COASTAL_PO', 'FOURCOLOR', 'FIPS_STCO',
       'CCD_MCD', 'FIPS_PLACE', 'FIPS_MCD', 'FIPS_COUNT', 'ACRES',
       'SQUARE_MIL', 'POP2010', 'POPCH00_10', 'SHAPE_Leng', 'SHAPE_Area',
       'geometry'],
      dtype='object')
```

[7]:
```
        TOWN  TOWN_ID  POP1980  POP1990  POP2000  POPCH80_90  POPCH90_00 TYPE  \
0  WELLESLEY      317    26658    26615    26604         -43         -11    T
1    NEEDHAM      199    27310    27557    28924         247        1367    T
2  PETERSHAM      234      997     1131     1180         134          49    T
3    READING      246    22545    22539    23708          -6        1169    T
4     QUINCY      243    83682    84985    88025        1303        3040    C

   FOURCOLOR  FIPS_STCO  SUM_ACRES  SUM_SQUARE  POP2010  POPCH00_10  \
0          2      25021   6749.852      10.547    27982        1378
1          4      25021   8130.187      12.703    28886         -38
2          3      25027  43675.599      68.243     1234          54
3          3      25017   6393.727       9.990    24747        1039
4          3      25021  11080.397      17.313    92271        4246
```

```
     SHAPE_Leng    SHAPE_Area  \
0  26738.594369  2.726958e+07
1  28960.012825  3.294777e+07
2  73405.706629  1.767489e+08
3  23126.471303  2.587450e+07
4  87188.934275  4.484078e+07

                                            geometry
0  (POLYGON ((219129.012500003 897474.7045999989,...
1  POLYGON ((222991.1424999982 895149.4145999998,...
2  POLYGON ((150592.6525000036 914968.5846000016,...
3  POLYGON ((232551.0625 923891.9946000017, 23344...
4  (POLYGON ((242254.612499997 895020.5045999996,...
```

[8]:
```python
# does this column denote state?
len(set(census_blocks['STATEFP10']))
# A: yes - all from one state
```

[8]:
```
1
```

1.1 Towns shapefile EDA

- do the town names match between the opioid overdose death data and the MA towns survey shapefile?
- derive the x and y coordinates of each town for modeling

[9]:
```python
town_map['TOWN'] = town_map['TOWN'].str.lower()
len(set(town_map['TOWN']))
```

[9]:
```
351
```

[10]:
```python
print(set(town_map['TOWN']) - set(death_data['city_death']))
print(set(death_data['city_death']) - set(town_map['TOWN']))
```

```
{'north attleborough'}
{'north attleboro'}
```

[11]:
```python
# fix name
death_data['city_death'] = death_data['city_death'].replace('north attleboro',␣
↪'north attleborough')
```

[12]:
```python
town_test_merge = town_map.merge(death_data, how='left', left_on='TOWN',␣
↪right_on='city_death')
```

[13]:
```python
print(len(set(town_test_merge['TOWN'])))
print(len(set(town_test_merge['city_death'])))
print(sum(town_test_merge['TOWN'] == town_test_merge['city_death']))
# able to merge without problems
```

```
351
351
351
```

[14]:
```
town_test_merge
```

[14]:
```
           TOWN  TOWN_ID  POP1980  POP1990  POP2000  POPCH80_90  POPCH90_00  \
0     wellesley      317    26658    26615    26604         -43         -11
1       needham      199    27310    27557    28924         247        1367
2     petersham      234      997     1131     1180         134          49
3       reading      246    22545    22539    23708          -6        1169
4        quincy      243    83682    84985    88025        1303        3040
..          ...      ...      ...      ...      ...         ...         ...
346  somerville      274    75836    76210    77478         374        1268
347    pembroke      231    13882    14544    16927         662        2383
348     duxbury       82    13174    13895    14248         721         353
349     boxford       38     5751     6266     7921         515        1655
350      boston       35   570719   574283   588957        3564       14674

    TYPE  FOURCOLOR  FIPS_STCO  ...  POPCH00_10     SHAPE_Leng    SHAPE_Area  \
0      T          2      25021  ...        1378   26738.594369  2.726958e+07
1      T          4      25021  ...         -38   28960.012825  3.294777e+07
2      T          3      25027  ...          54   73405.706629  1.767489e+08
3      T          3      25017  ...        1039   23126.471303  2.587450e+07
4      C          3      25021  ...        4246   87188.934275  4.484078e+07
..   ...        ...        ...  ...         ...            ...           ...
346    C          4      25017  ...       -1724   19884.219455  1.069865e+07
347    T          4      25023  ...         910   42341.724263  6.100547e+07
348    T          1      25023  ...         811   99884.166652  6.232911e+07
349    T          3      25009  ...          44   44884.884100  6.321618e+07
350    C          1      25025  ...       28637  205312.529714  1.295200e+08

                                              geometry  city_death  2014  2015  \
0    (POLYGON ((219129.012500003 897474.7045999989,...   wellesley     0     1
1    POLYGON ((222991.1424999982 895149.4145999998,...     needham     0     1
2    POLYGON ((150592.6525000036 914968.5846000016,...   petersham     0     0
3    POLYGON ((232551.0625 923891.9946000017, 23344...     reading     2     2
4    (POLYGON ((242254.612499997 895020.5045999996,...      quincy    39    43
..                                                 ...         ...   ...   ...
346  POLYGON ((230662.7124999985 907352.6845999993,...  somerville    14    15
347  POLYGON ((259609.4624999985 874160.1746000014,...    pembroke     3     2
348  (POLYGON ((266547.2325000018 868791.5045999996...     duxbury     0     0
349  POLYGON ((233567.7524999976 943010.8845999986,...     boxford     0     0
350  (POLYGON ((235056.1525000036 904531.2545999996...      boston   167   226

    2016  2017  2018
0      0     0     0
1      2     1     0
```

```
2      0     0     0
3      4     4     1
4     36    37    35
..   ...   ...   ...
346   19    12     8
347    5     1     0
348    2     0     0
349    0     0     0
350  259   279   245

[351 rows x 23 columns]
```

[15]:
```
# convert geometry to a more recognizable crs format
town_test_merge['geometry'] = town_test_merge['geometry'].to_crs(epsg=4326)
town_test_merge.head()
```

[15]:
```
        TOWN  TOWN_ID  POP1980  POP1990  POP2000  POPCH80_90  POPCH90_00 TYPE  \
0  wellesley      317    26658    26615    26604         -43         -11    T
1    needham      199    27310    27557    28924         247        1367    T
2  petersham      234      997     1131     1180         134          49    T
3    reading      246    22545    22539    23708          -6        1169    T
4     quincy      243    83682    84985    88025        1303        3040    C

   FOURCOLOR  FIPS_STCO  ...  POPCH00_10    SHAPE_Leng    SHAPE_Area  \
0          2      25021  ...        1378  26738.594369  2.726958e+07
1          4      25021  ...         -38  28960.012825  3.294777e+07
2          3      25027  ...          54  73405.706629  1.767489e+08
3          3      25017  ...        1039  23126.471303  2.587450e+07
4          3      25021  ...        4246  87188.934275  4.484078e+07

                                            geometry  city_death  2014  2015  \
0  (POLYGON ((-71.26791296846598 42.327527799305,...   wellesley     0     1
1  POLYGON ((-71.22114734819833 42.30648947704145...     needham     0     1
2  POLYGON ((-72.10093262757198 42.48368711606084...   petersham     0     0
3  POLYGON ((-71.10357918791566 42.56490520777559...     reading     2     2
4  (POLYGON ((-70.98752024993465 42.3045244458064...      quincy    39    43

  2016  2017  2018
0    0     0     0
1    2     1     0
2    0     0     0
3    4     4     1
4   36    37    35

[5 rows x 23 columns]
```

Need the town centroids for modeling - derive it here:

[16]:
```
ma_town_coord = pd.DataFrame({'town':town_test_merge['TOWN'].copy(), 'x':
↪town_test_merge.centroid.x, 'y': town_test_merge.centroid.y})
ma_town_coord.head()
```

[16]:
```
        town          x          y
0  wellesley -71.285441  42.304304
1    needham -71.241078  42.281368
2  petersham -72.221072  42.459687
3    reading -71.105574  42.535054
4     quincy -71.020353  42.250488
```

[17]:
```
### write to csv:
#ma_town_coord.to_csv("../../data/tidy_data/ma_town_crs4326_coords.csv",
↪index=False)
```

1.2 Census block and Towns join

[18]:
```
town_map.head()
```

[18]:
```
        TOWN  TOWN_ID  POP1980  POP1990  POP2000  POPCH80_90  POPCH90_00 TYPE  \
0  wellesley      317    26658    26615    26604         -43         -11    T
1    needham      199    27310    27557    28924         247        1367    T
2  petersham      234      997     1131     1180         134          49    T
3    reading      246    22545    22539    23708          -6        1169    T
4     quincy      243    83682    84985    88025        1303        3040    C

   FOURCOLOR  FIPS_STCO  SUM_ACRES  SUM_SQUARE  POP2010  POPCH00_10  \
0          2      25021   6749.852      10.547    27982        1378
1          4      25021   8130.187      12.703    28886         -38
2          3      25027  43675.599      68.243     1234          54
3          3      25017   6393.727       9.990    24747        1039
4          3      25021  11080.397      17.313    92271        4246

     SHAPE_Leng    SHAPE_Area  \
0  26738.594369  2.726958e+07
1  28960.012825  3.294777e+07
2  73405.706629  1.767489e+08
3  23126.471303  2.587450e+07
4  87188.934275  4.484078e+07

                                            geometry
0  (POLYGON ((219129.012500003 897474.7045999989,...
1  POLYGON ((222991.1424999982 895149.4145999998,...
2  POLYGON ((150592.6525000036 914968.5846000016,...
3  POLYGON ((232551.0625 923891.9946000017, 23344...
4  (POLYGON ((242254.612499997 895020.5045999996,...
```

1.2.1 Joining the geometries directly:

[19]:
```
# how does the intersects option do?
intersects_join = gpd.sjoin(town_map, census_blocks, how='left',␣
↪op='intersects')
```

[20]:
```
# original number of towns:
print(town_map.shape)
# number of towns in the intersects join:
print(len(set(intersects_join['TOWN'])))
print(census_blocks.shape)
print(intersects_join.shape)
print(len(set(intersects_join['GEOID10'])))
intersects_join['GEOID10'].value_counts()
```

```
(351, 17)
351
(4979, 22)
(8697, 39)
4979
```

[20]:
```
250110401004    10
250158227003     9
250110406003     9
250039322001     8
250277231003     8
                ..
250138107003     1
250277316006     1
250250922004     1
250277318003     1
250092232001     1
Name: GEOID10, Length: 4979, dtype: int64
```

[21]:
```
intersects_join_alt = gpd.sjoin(town_map_alt, census_blocks, how='left',␣
↪op='intersects')
# original number of towns:
print(town_map_alt.shape)
# number of towns in the intersects join:
print(len(set(intersects_join_alt['TOWN'])))
print(census_blocks.shape)
print(intersects_join_alt.shape)
print(len(set(intersects_join_alt['GEOID10'])))
intersects_join_alt['GEOID10'].value_counts()
```

```
(1243, 23)
351
(4979, 22)
```

```
(9932, 45)
4980
```

[21]:
```
250092221001    28
250056461042    21
250235062033    21
250092691002    21
250092701001    21
                ..
250277573001     1
250173103003     1
250173738004     1
250039002006     1
250092611012     1
Name: GEOID10, Length: 4979, dtype: int64
```

Join even messier with the poly shapefile (vs the polyM) - try the other op join options:

[22]:
```python
# how does the within option do?
within_join = gpd.sjoin(town_map, census_blocks, how='left', op='within')
# original number of towns:
print(town_map.shape)
# number of towns in the within join:
print(len(set(within_join['TOWN'])))
print(census_blocks.shape)
print(within_join.shape)
print(len(set(within_join['GEOID10'])))
within_join['GEOID10'].value_counts()
```

```
(351, 17)
351
(4979, 22)
(351, 39)
1
```

[22]:
```
Series([], Name: GEOID10, dtype: int64)
```

[23]:
```python
# how does the contains option do?
contains_join = gpd.sjoin(town_map, census_blocks, how='left', op='contains')
# original number of towns:
print(town_map.shape)
# number of towns in the contains join:
print(len(set(contains_join['TOWN'])))
print(census_blocks.shape)
print(contains_join.shape)
print(len(set(contains_join['GEOID10'])))
contains_join['GEOID10'].value_counts()
```

```
(351, 17)
351
```

```
(4979, 22)
(2608, 39)
2467
```

[23]:
```
250092526023    1
250173578002    1
250250701017    1
250010120014    1
250277311021    1
               ..
250251403006    1
250214002002    1
250250502001    1
250251203012    1
250039011001    1
Name: GEOID10, Length: 2466, dtype: int64
```

1.2.2 Notes:

- intersects - assigns census blocks to too many towns
- within - terrible
- contains - too many census blocks lost

1.2.3 Alternative geometry joining strategy:

- convert the geometry of the census blocks to points (centroids) - join using contains (no need to worry about borders and geometry overlap)
- check how well this join works by comparing 2010 population totals by town

[24]:
```
centroid_blocks = census_blocks.copy()
# change geometry
centroid_blocks['geometry'] = centroid_blocks['geometry'].centroid
print(centroid_blocks.shape)
centroid_blocks['geometry'].head()
```

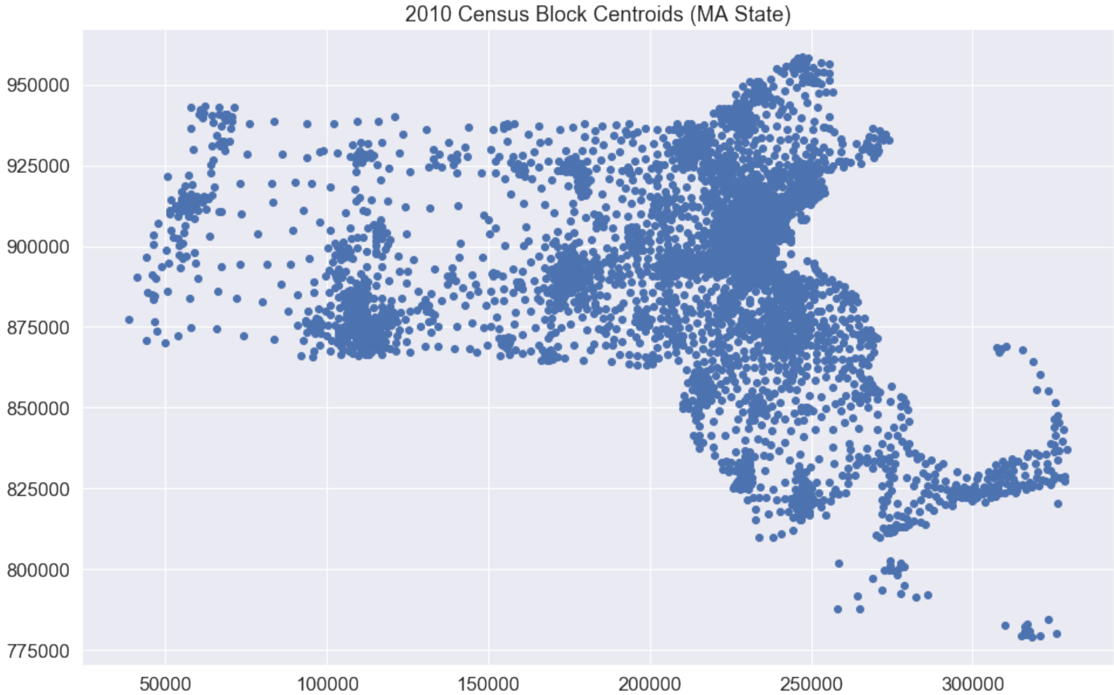
```
(4979, 22)
```

[24]:
```
0    POINT (243856.7307279073 858308.2978385694)
1    POINT (242004.2566174036 857829.7623223617)
2    POINT (244160.0989660571 861561.0733446195)
3    POINT (241130.6851021084 873574.4539211198)
4     POINT (240478.7043746133 873576.766094962)
Name: geometry, dtype: object
```

Centroid shape file still has the same data, but the geometry is different - switched from polygon to point geometries.

Map plot of result:

[25]:
```
centroid_blocks.plot(figsize=(16,10))
plt.title("2010 Census Block Centroids (MA State)")
plt.show()
```



[26]:
```
cent_join = gpd.sjoin(town_map, centroid_blocks, how='left', op='contains')
# number of towns in the contains join:
print(len(set(cent_join['TOWN'])))
print(census_blocks.shape)
print(cent_join.shape)
print(len(set(cent_join['GEOID10'])))
cent_join['GEOID10'].value_counts()
```

```
351
(4979, 22)
(4958, 39)
4955
```

[26]:
```
250092526023    1
250214012001    1
250251402024    1
250092523005    1
250277574001    1
               ..
250173131013    1
250138110005    1
```

```
250214002001    1
250173822002    1
250039011001    1
Name: GEOID10, Length: 4954, dtype: int64
```

[27]:
```
print(len(set(census_blocks['GEOID10'])))
print(len(set(cent_join['GEOID10'])))
```

```
4979
4955
```

[28]:
```
cent_join_inter = gpd.sjoin(town_map, centroid_blocks, how='left',␣
 ↪op='intersects')
# number of towns in the contains join:
print(len(set(cent_join_inter['TOWN'])))
print(census_blocks.shape)
print(cent_join_inter.shape)
print(len(set(cent_join_inter['GEOID10'])))
cent_join_inter['GEOID10'].value_counts()
```

```
351
(4979, 22)
(4958, 39)
4955
```

[28]:
```
250092526023    1
250214012001    1
250251402024    1
250092523005    1
250277574001    1
               ..
250173131013    1
250138110005    1
250214002001    1
250173822002    1
250039011001    1
Name: GEOID10, Length: 4954, dtype: int64
```

[29]:
```
cent_join_within = gpd.sjoin(town_map, centroid_blocks, how='left', op='within')
# number of towns in the contains join:
print(len(set(cent_join_within['TOWN'])))
print(census_blocks.shape)
print(cent_join_within.shape)
print(len(set(cent_join_within['GEOID10'])))
cent_join_within['GEOID10'].value_counts()
```

```
351
(4979, 22)
```

```
(351, 39)
1
```

[29]:
```
Series([], Name: GEOID10, dtype: int64)
```

For the centroid join: * within not a good option * contains and intersects give similar results - go with the contains option * lost 21 centroids (4979 in original file vs 4955 after merge), but better than other options

To validate the join: compare 2010 population count column in the town survey shapefile with the sum of the 2010 census block populations (tallied by town after merge) - do they match up?

[30]:
```python
cent_join_sub = cent_join[['TOWN', 'POP2010', 'SHAPE_Area', 'GEOID10', 'SHAPE_AREA', 'POP100_RE']].copy()
cent_join_sub.head()
```

[30]:

	TOWN	POP2010	SHAPE_Area	GEOID10	SHAPE_AREA	POP100_RE
0	wellesley	27982	2.726958e+07	250214041003	6.424854e+05	935.0
0	wellesley	27982	2.726958e+07	250214042012	1.167837e+06	989.0
0	wellesley	27982	2.726958e+07	250214042013	1.182595e+06	968.0
0	wellesley	27982	2.726958e+07	250214041002	1.079832e+06	1145.0
0	wellesley	27982	2.726958e+07	250214042014	5.306549e+05	664.0

[31]:
```python
# add up census block 2010 population counts by town:
cent_join_block_sum = cent_join_sub.groupby('TOWN').sum()[['POP100_RE']].reset_index()
print(cent_join_block_sum.shape)
#result:
cent_join_block_sum.head()
```

```
(351, 2)
```

[31]:

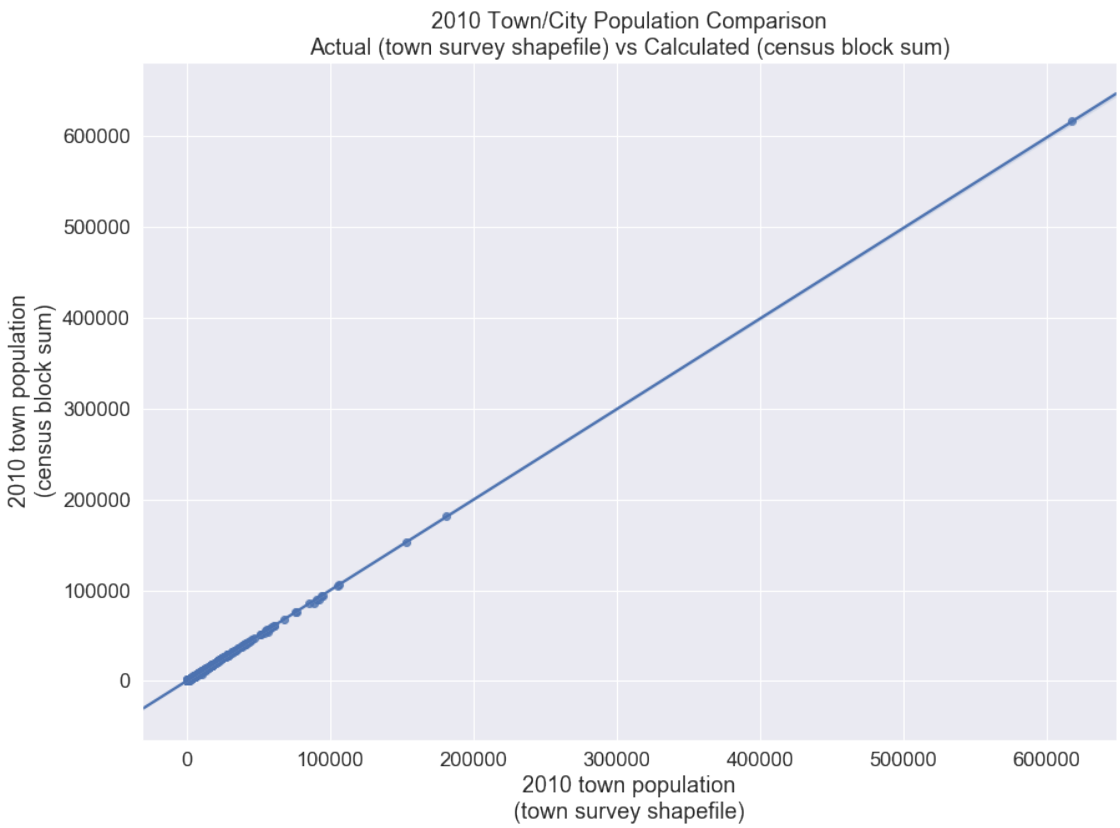
	TOWN	POP100_RE
0	abington	15985.0
1	acton	21924.0
2	acushnet	10303.0
3	adams	8485.0
4	agawam	27621.0

[32]:
```python
# join back to df with population counts from the town survey shapefile
cent_join_sum_to_exp = cent_join_sub[['TOWN', 'POP2010']].drop_duplicates().merge(cent_join_block_sum, on='TOWN', how='inner')
cent_join_sum_to_exp.head()
```

[32]:

	TOWN	POP2010	POP100_RE
0	wellesley	27982	27982.0
1	needham	28886	28886.0
2	petersham	1234	1234.0
3	reading	24747	24747.0
4	quincy	92271	89703.0

[33]:
```
# plot result:
plt.figure(figsize=(14, 10))
sns.regplot(x='POP2010', y='POP100_RE', data=cent_join_sum_to_exp)
plt.xlabel('2010 town population\n(town survey shapefile)')
plt.ylabel('2010 town population\n(census block sum)')
plt.title('2010 Town/City Population Comparison\nActual (town survey shapefile)␣
↪vs Calculated (census block sum)')
plt.show()
```



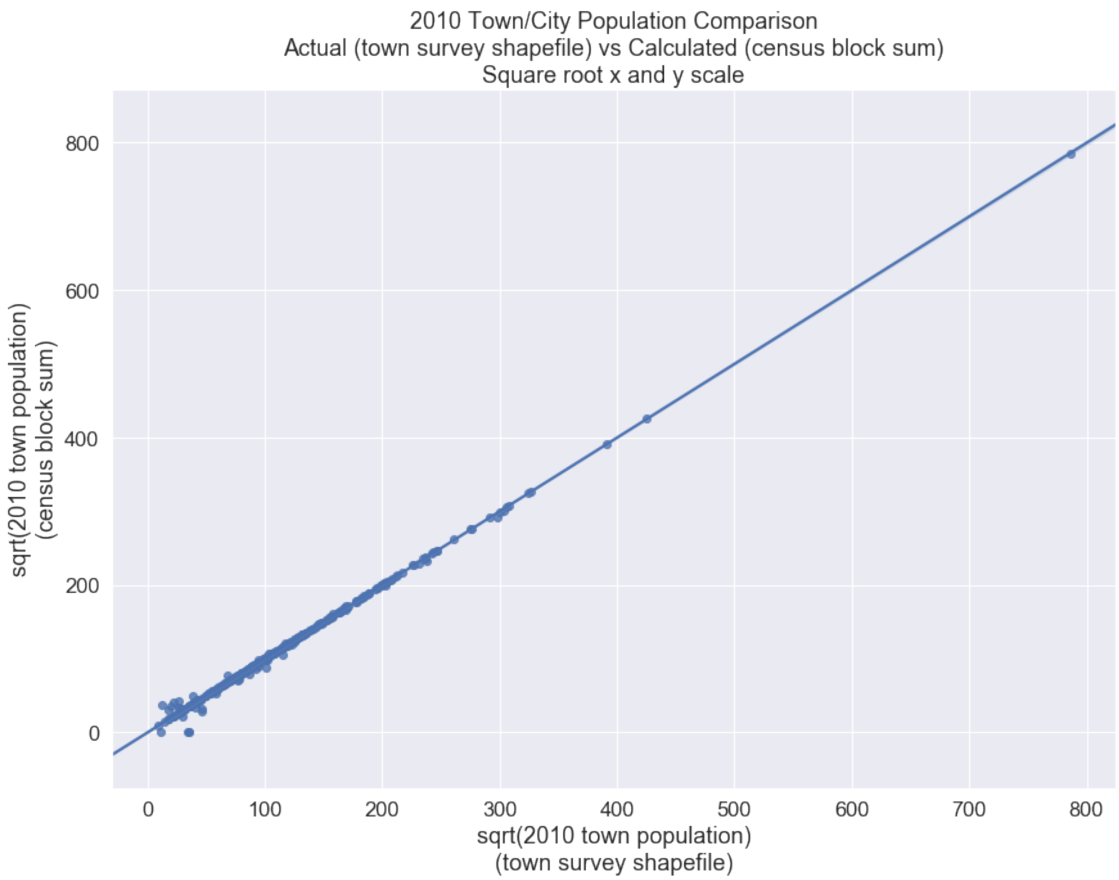
Towns/cities with high populations dominate the plot - convert to square root to change scale

[34]:
```
cent_join_sum_sqrt = cent_join_sum_to_exp.copy()
cent_join_sum_sqrt.set_index('TOWN', inplace=True)
cent_join_sum_sqrt = cent_join_sum_sqrt.apply(np.sqrt)
cent_join_sum_sqrt.head()
```

[34]:

TOWN	POP2010	POP100_RE
wellesley	167.278211	167.278211
needham	169.958819	169.958819
petersham	35.128336	35.128336
reading	157.311792	157.311792
quincy	303.761420	299.504591

[35]:
```
# plot result:
plt.figure(figsize=(14, 10))
sns.regplot(x='POP2010', y='POP100_RE', data=cent_join_sum_sqrt)
plt.xlabel('sqrt(2010 town population)\n(town survey shapefile)')
plt.ylabel('sqrt(2010 town population)\n(census block sum)')
plt.title('2010 Town/City Population Comparison\nActual (town survey shapefile)␣
↪vs Calculated (census block sum)\nSquare root x and y scale')
plt.show()
```



Some towns with low populations seem to have pretty large errors

[36]:
```
cent_join_sum_to_exp['error'] = cent_join_sum_to_exp['POP100_RE'] -␣
↪cent_join_sum_to_exp['POP2010']
cent_join_sum_to_exp['percent_error'] = (abs(cent_join_sum_to_exp['POP100_RE']␣
↪- cent_join_sum_to_exp['POP2010']) * 100) / cent_join_sum_to_exp['POP2010']
display(cent_join_sum_to_exp.sort_values('error', ascending=False))
display(cent_join_sum_to_exp.sort_values('percent_error', ascending=False))
```

	TOWN	POP2010	POP100_RE	error	percent_error
22	north brookfield	4680	6003.0	1323.0	28.269231

```
112  mount washington      167      1392.0  1225.0      733.532934
63         middlefield      521      1677.0  1156.0      221.880998
271             leyden      711      1822.0  1111.0      156.258790
52          marshfield    25132     26097.0   965.0        3.839726
..                 ...      ...         ...     ...             ...
154            ipswich    13175     10919.0 -2256.0       17.123340
159           plymouth    56468     54143.0 -2325.0        4.117376
4               quincy    92271     89703.0 -2568.0        2.783106
114               hull    10293      7584.0 -2709.0       26.318857
86          fall river    88857     84996.0 -3861.0        4.345184

[351 rows x 5 columns]
```

```
                  TOWN  POP2010  POP100_RE   error  percent_error
112  mount washington      167     1392.0  1225.0     733.532934
63         middlefield      521     1677.0  1156.0     221.880998
62                rowe      393     1220.0   827.0     210.432570
59              hawley      337      897.0   560.0     166.172107
271             leyden      711     1822.0  1111.0     156.258790
..                 ...      ...        ...     ...            ...
129           brewster     9820     9820.0     0.0       0.000000
126            florida      752      752.0     0.0       0.000000
124        belchertown    14649    14649.0     0.0       0.000000
123        marlborough    38499    38499.0     0.0       0.000000
175             woburn    38120    38120.0     0.0       0.000000

[351 rows x 5 columns]
```

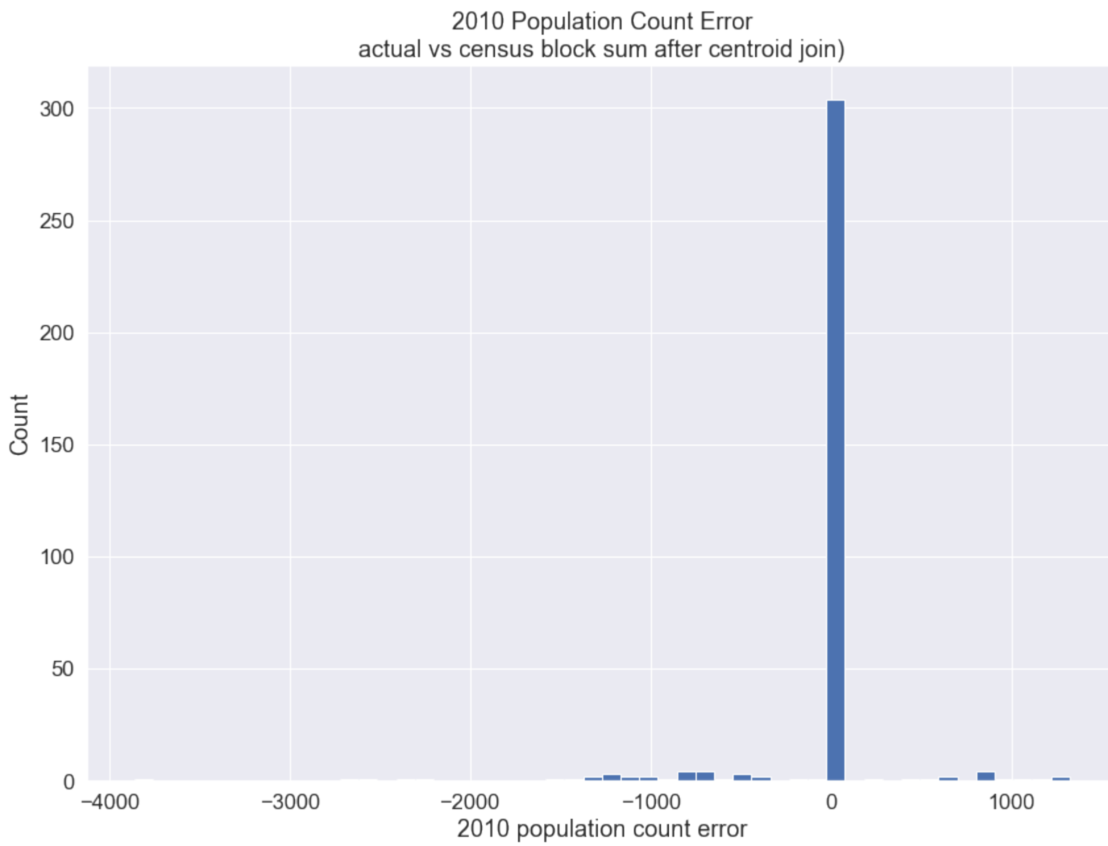
[37]:
```
# average percent error and distributions:
cent_join_sum_to_exp[['error', 'percent_error']].describe()
```

[37]:

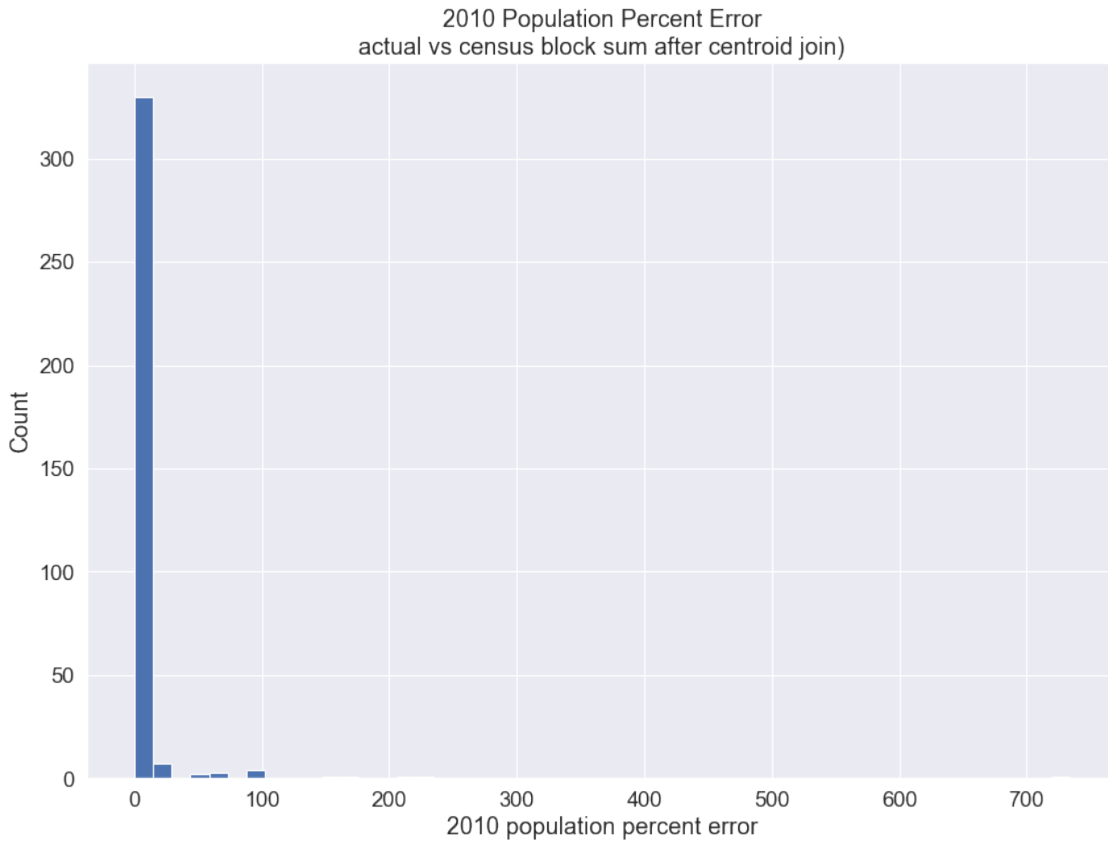
	error	percent_error
count	351.000000	351.000000
mean	-73.572650	7.051235
std	455.502353	45.602087
min	-3861.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	0.000000	0.000000
max	1323.000000	733.532934

[38]:
```
plt.figure(figsize=(14,10))
cent_join_sum_to_exp['error'].hist(bins=50)
plt.xlabel('2010 population count error')
plt.ylabel('Count')
plt.title('2010 Population Count Error\nactual vs census block sum after␣
 ↪centroid join)')
```

```
plt.show()
```



[39]:
```
plt.figure(figsize=(14,10))
cent_join_sum_to_exp['percent_error'].hist(bins=50)
plt.xlabel('2010 population percent error')
plt.ylabel('Count')
plt.title('2010 Population Percent Error\nactual vs census block sum after␣
 ↪centroid join)')
plt.show()
```



[40]:
```
# towns with over 5% population count error after centroid merge:
print(cent_join_sum_to_exp[cent_join_sum_to_exp['percent_error'] > 5].shape)
cent_join_sum_to_exp[cent_join_sum_to_exp['percent_error'] > 5].
 ↪sort_values('percent_error', ascending=False)
```

```
(31, 5)
```

[40]:

	TOWN	POP2010	POP100_RE	error	percent_error
112	mount washington	167	1392.0	1225.0	733.532934
63	middlefield	521	1677.0	1156.0	221.880998
62	rowe	393	1220.0	827.0	210.432570
59	hawley	337	897.0	560.0	166.172107
271	leyden	711	1822.0	1111.0	156.258790
179	charlemont	1266	0.0	-1266.0	100.000000
38	egremont	1225	0.0	-1225.0	100.000000
264	monroe	121	0.0	-121.0	100.000000
127	worthington	1156	0.0	-1156.0	100.000000
282	heath	706	1182.0	476.0	67.422096
230	east brookfield	2183	860.0	-1323.0	60.604672
236	gill	1500	2393.0	893.0	59.533333
143	bernardston	2129	1018.0	-1111.0	52.184124

```
214           chilmark      866      457.0   -409.0       47.228637
95             colrain     1671     1195.0   -476.0       28.485937
22    north brookfield     4680     6003.0   1323.0       28.269231
114               hull    10293     7584.0  -2709.0       26.318857
87             warwick      780      971.0    191.0       24.487179
72              nahant     3410     2781.0   -629.0       18.445748
155           cohasset     7542     6229.0  -1313.0       17.409175
154            ipswich    13175    10919.0  -2256.0       17.123340
305            orleans     5890     5072.0   -818.0       13.887946
222            chatham     6125     5288.0   -837.0       13.665306
299             erving     1800     1609.0   -191.0       10.611111
100           montague     8437     7544.0   -893.0       10.584331
125           freetown     8870     8030.0   -840.0        9.470124
128            norwell    10506     9541.0   -965.0        9.185227
196          lakeville    10602    11442.0    840.0        7.923033
176             sutton     8963     9661.0    698.0        7.787571
348            duxbury    15059    14069.0   -990.0        6.574142
37           nantucket    10172     9650.0   -522.0        5.131734
```

[41]:
```
len(cent_join_sum_to_exp[cent_join_sum_to_exp['error'] != 0]['TOWN'])
```

[41]:
```
52
```

[42]:
```
cent_join_sub.head()
```

[42]:
```
        TOWN  POP2010    SHAPE_Area       GEOID10    SHAPE_AREA  POP100_RE
0  wellesley    27982  2.726958e+07  250214041003  6.424854e+05      935.0
0  wellesley    27982  2.726958e+07  250214042012  1.167837e+06      989.0
0  wellesley    27982  2.726958e+07  250214042013  1.182595e+06      968.0
0  wellesley    27982  2.726958e+07  250214041002  1.079832e+06     1145.0
0  wellesley    27982  2.726958e+07  250214042014  5.306549e+05      664.0
```

[43]:
```
cent_join_to_export = cent_join_sub[['TOWN', 'GEOID10']].copy()
cent_join_to_export['TOWN'] = cent_join_to_export['TOWN'].str.lower()
cent_join_to_export.head()
```

[43]:
```
        TOWN       GEOID10
0  wellesley  250214041003
0  wellesley  250214042012
0  wellesley  250214042013
0  wellesley  250214041002
0  wellesley  250214042014
```

[51]:
```
cent_join_sum_to_exp.columns = ['TOWN', 'town_actual_2010_pop',␣
 ↪'block_est_2010_pop', 'count_error', 'percent_error']
cent_join_sum_to_exp.head()
```

[51]:
```
        TOWN  town_actual_2010_pop  block_est_2010_pop  count_error  \
0  wellesley                 27982             27982.0          0.0
1    needham                 28886             28886.0          0.0
2  petersham                  1234              1234.0          0.0
```

```
3    reading                  24747             24747.0          0.0
4     quincy                  92271             89703.0      -2568.0

   percent_error
0       0.000000
1       0.000000
2       0.000000
3       0.000000
4       2.783106
```

[52]:
```
#cent_join_to_export.to_csv("../../data/tidy_data/census_block_town_match.csv",␣
↪index=False)
#cent_join_sum_to_exp.to_csv("../../data/tidy_data/
↪census_block_town_match_2010pop_error.csv", index=False)
```