project_one

February 6, 2025

1 Blockbusting in the 21st Century?: Minority Move-ins and Neighborhood Home Value Appreciation

1.1 Introduction

I am using the **Fannie Mae (FNMA) & Freddie Mac (FHLMC)** data to analyze the demographics of move-ins on a census-tract level. This, with the census-level demographic data **(ACS)**, can provide an estimate of a quantity of minority "move-ins". I am seeing the extent to which this has an effect on the appreciation in home values (**Zillow ZHVI**)

My **y-variable** is census-tract property value appreciation. My main **explanatory variables** are minority move-ins and previous neighborhood demographics. I will be controlling for income.

1.2 Data Loading

We start by loading libraries. For dataframes we are using `pandas`, for plots we are using `pyplot` from `matplotlib`.

[15]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

Because the data is in `.txt` format, with no column headers, and is in many different files for different years and loan types, some light data cleaning is required. To begin, we set up short titles for each column based on the data dictionary provided, and create a list of columns we do not need.

[16]:
```
def read_columns_from_txt(filename):
    data = {}
    with open(filename, "r") as f:
        for line in f:
            key, value = line.split("=", 1)
            data[key.strip()] = eval(value.strip())
    return data

columns_data = read_columns_from_txt("data/columns.txt")
```

[17]:
```
def load_loans(years):
    loans_list = []
    for year in years:
        if year >= 2018: # The data format changes in 2018
```

```
            files = [f"data/sf/fhlmc_sf{year}c_loans.txt", f"data/sf/
↪fnma_sf{year}c_loans.txt"]
            loans_year = pd.concat(
                [pd.read_csv(file, sep=r"\s+", header=None, names=cols).
↪drop(columns=dropcols) for file in files],
                ignore_index=True
            )
        else:
            files = [f"data/sf/fhlmc_sf{year}c_loans.txt", f"data/sf/
↪fnma_sf{year}c_loans.txt"]
            loans_year = pd.concat(
                [pd.read_csv(file, sep=r"\s+", header=None, names=cols_pre2018).
↪drop(columns=dropcols_pre2018) for file in files],
                ignore_index=True
            )
        loans_year["year"] = year
        loans_list.append(loans_year)
        print(f"Processed {year}")
    return pd.concat(loans_list, ignore_index=True)
```

[18]:
```
loans = load_loans([2017])
```

```
Processed 2017
```

Next, we load in the census data to give us information about neighborhood demographics

[19]:
```
cols_tract = ['YEAR', 'TRACTA','U7J001', 'U7J002', 'U7J003', 'U7J004',␣
↪'U7J005', 'U7J006', 'U7J007', 'U7J008']
tract_data = pd.read_csv('data/census/nhgis0009_ds258_2020_zcta.
↪csv')[cols_tract]
tract_data.rename(columns={
    'YEAR': 'year',
    'TRACTA': 'census_tract',
    'U7J001': 'total_pop',
    'U7J002': 'white',
    'U7J003': 'black',
    'U7J004': 'native_american',
    'U7J005': 'asian',
    'U7J006': 'pacific_islander',
    'U7J007': 'other_race',
    'U7J008': 'two_or_more'}, inplace=True)
```

Next, we load in the Zillow data for information about home prices

[20]:
```
zhvi = pd.read_csv('data/zhvi/Zip_zhvi_uc_sfrcondo_tier_0.33_0.67_sm_sa_month␣
↪(1).csv')
```

The Zillow data is based on zip-code, but all our other data is based on census tract. As a result, we use a Crosswalk File from HUD.

[21]:
```
crosswalk = pd.read_excel('data/census/ZIP_TRACT_122024.xlsx')
```

[22]:
```
# loans groupby census tract merge into crosswalk
# crosswalk find high and low minority move-ins, etc?
# merge with zhvi
```

1.3 Summary Statistics

One potentially interesting summary statistic is the quantity of loans in 2023 in each metropolitan area. We use `groupby` to look at each individual MSA and the count of entries in the table for each to define `vol_by_msa` which is the loan volume in each MSA.

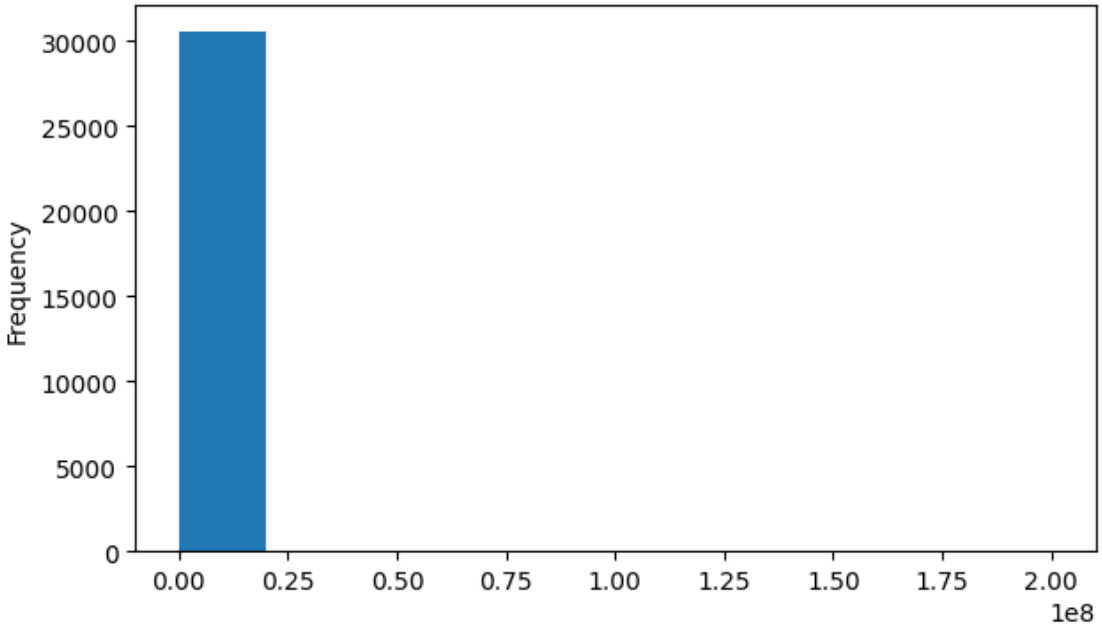
[23]:
```
loans[['tract_median_income', 'borrower_income', 'property_appraisal_value']].
 ↪describe()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[23], line 1
----> 1 loans[['tract_median_income', 'borrower_income',␣
 ↪'property_appraisal_value']].describe()

File c:\Users\emers\anaconda3\Lib\site-packages\pandas\core\frame.py:4108, in␣
 ↪DataFrame.__getitem__(self, key)
   4106     if is_iterator(key):
   4107         key = list(key)
-> 4108     indexer = self.columns._get_indexer_strict(key, "columns")[1]
   4110 # take() does not accept boolean indexers
   4111 if getattr(indexer, "dtype", None) == bool:

File c:\Users\emers\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:
 ↪6200, in Index._get_indexer_strict(self, key, axis_name)
   6197 else:
   6198     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 6200 self._raise_if_missing(keyarr, indexer, axis_name)
   6202 keyarr = self.take(indexer)
   6203 if isinstance(key, Index):
   6204     # GH 42790 - Preserve name from an Index

File c:\Users\emers\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:
 ↪6252, in Index._raise_if_missing(self, key, indexer, axis_name)
   6249     raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   6251 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
-> 6252 raise KeyError(f"{not_found} not in index")

KeyError: "['borrower_income', 'property_appraisal_value'] not in index"
```

1.4 Plots & Figures

[16]:
```
loans.groupby('census_tract_2020')["borrower_income"].mean().
 ↪sort_values(ascending=False).plot(kind='hist', figsize=(7, 4))
```

[16]:
```
<Axes: ylabel='Frequency'>
```



[17]:
```
# vol_by_msa.head(10).plot(kind='bar', figsize=(7, 4))
# plt.title('Top 10 MSAs by 2023 Multifamily Loan Count')
# plt.xlabel('MSA Code')
# plt.ylabel('Loan Count')
# plt.show()
```

[18]:
```
# msa_filtered = mf_loans_2023[mf_loans_2023['msa_name'].isin(["'Omaha-Council␣
 ↪Bluffs, NE-IA MSA'", "'Kansas City, MO-KS MSA'"])]
# msa_filtered.groupby('msa_name')["prop_val"].mean().
 ↪sort_values(ascending=False).plot(kind='bar', figsize=(7, 4))
# plt.title('Average Multifamily Property Value for Selected MSAs, 2023')
# plt.xlabel('MSA Name')
# plt.ylabel('Average Property Value')
# plt.show()
```

[19]:
```
# msas_2023_values = sf_loans_2023.groupby('Metropolitan Statistical Area (MSA)␣
 ↪Code')["Property Valuation Amount"].mean().sort_values(ascending=False)
# msas_2018_values = sf_loans_2018.groupby('Metropolitan Statistical Area (MSA)␣
 ↪Code')["Property Valuation Amount"].mean().sort_values(ascending=False)
```

```
# msa_values_by_year = pd.concat([msas_2023_values, msas_2018_values], axis=1)
# msa_values_by_year.columns = ['2023', '2018']
# plt.scatter(msa_values_by_year['2023'], msa_values_by_year['2018'])
# plt.xlabel('2023 Property Valuation Amount')
# plt.ylabel('2018 Property Valuation Amount')
# plt.title('Property Valuation Amount Comparison: 2023 vs 2018')
# plt.show()
```

[20]:
```
# msa_value_change = msa_values_by_year['2023'] - msa_values_by_year['2018']
# msa_value_change.sort_values(ascending=False).plot(kind='hist', figsize=(7,␣
↪4))
# plt.title('Change in Property Valuation Amount by MSA: 2023 vs 2018')
# plt.xlabel('Change in Property Valuation Amount')
# plt.ylabel('Count')
```

1.5 Conclusion

1.6 References

Steven Manson, Jonathan Schroeder, David Van Riper, Katherine Knowles, Tracy Kugler, Finn Roberts, and Steven Ruggles. IPUMS National Historical Geographic Information System: Version 19.0 [dataset]. Minneapolis, MN: IPUMS. 2024. http://doi.org/10.18128/D050.V19.0