



UPSTATE NEW YORK REAL ESTATE

Data Analysis Report

James R. Ingram II

Syracuse University School of Information Studies

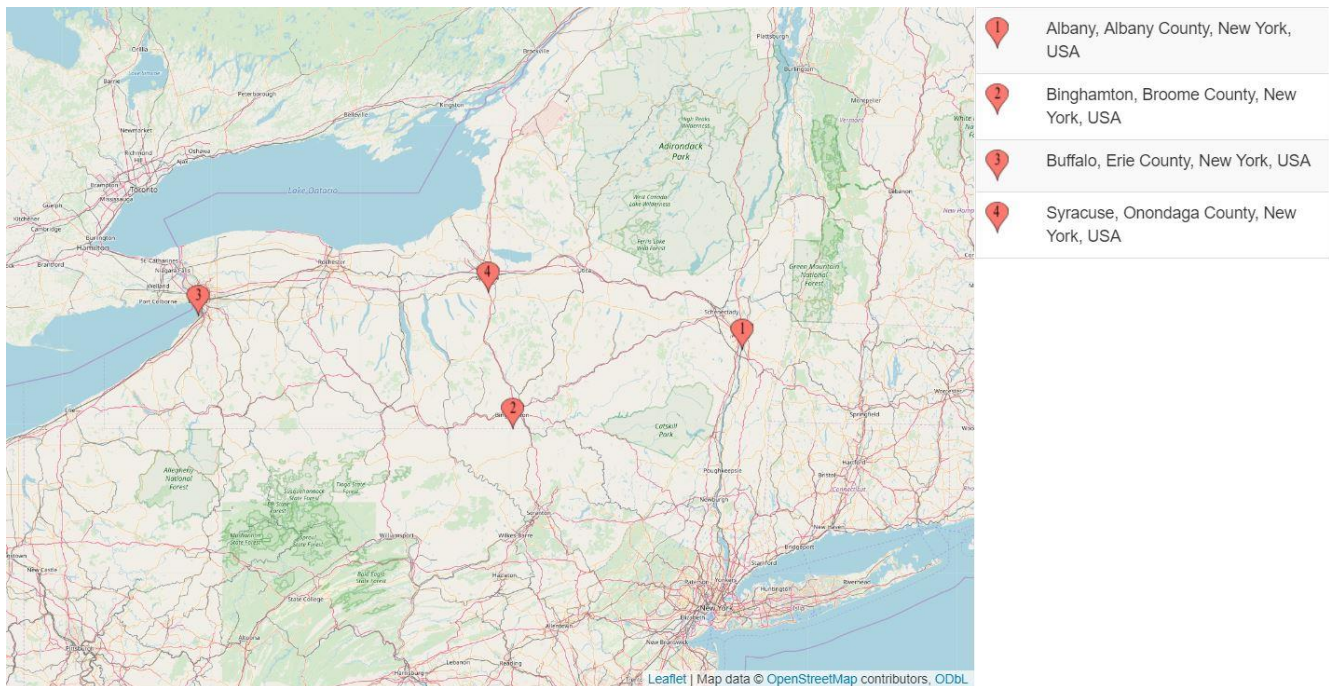
IST 652 - Scripting for Data Analysis

Prof. Debbie Landowski, PhD

Summer Quarter 2018

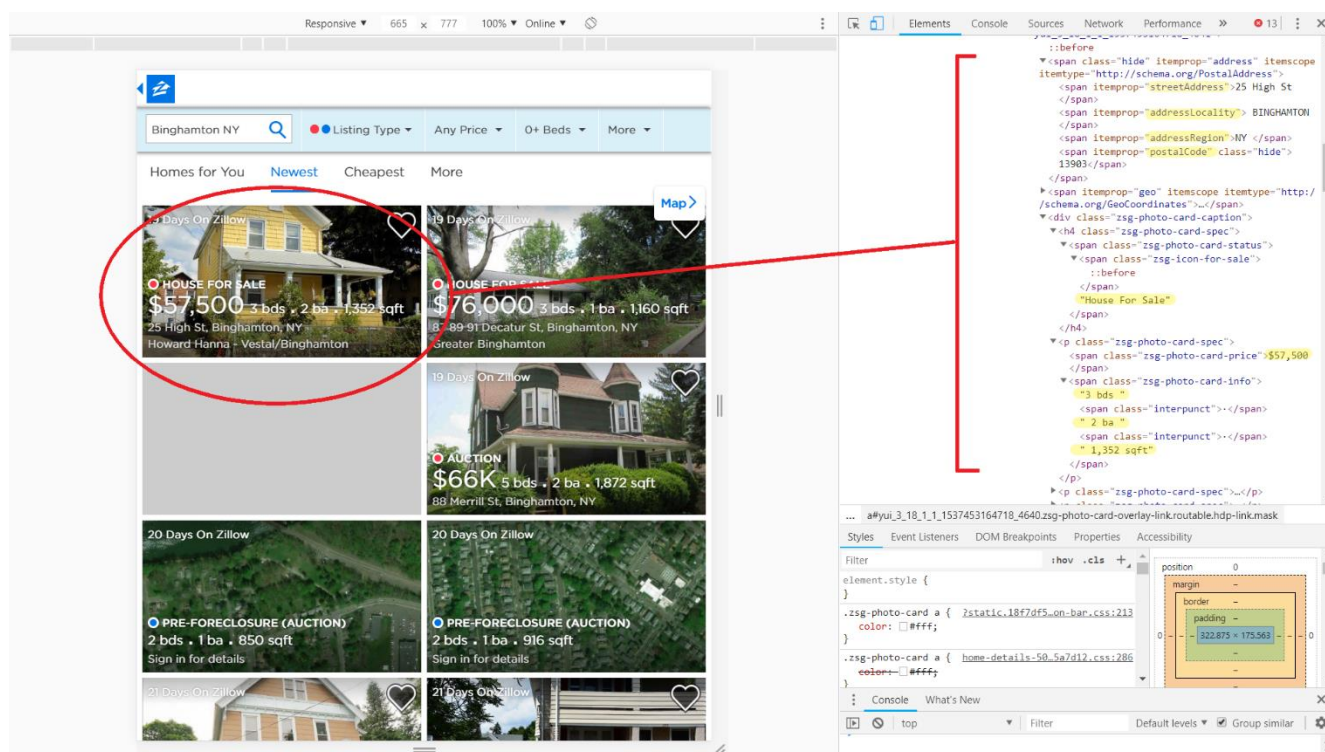
Data and its Source

Buying a home is a large investment. How does one know what properties are available and what price is reasonable? Several companies have online listings of houses for sale, including Zillow. Zillow, from the creators of Expedia, maintains an online real estate database displaying listings of over 110 million homes in the United States. For this project, real estate data from four Upstate New York cities were analyzed and compared.



The initial strategy was to write a Python program to scrape data from Zillow's website and store the results in a .csv file for future analysis. The first few attempts resulted in successful data extraction followed by a temporary block of my IP address from the website. A best practice when performing any type of web-scraping is to read a site's acceptable use policy and check if they have a robots.txt file outlining disallowed practices. Zillow.com does not permit automated scraping. There are applications and methods to bypass these restrictions, but it would be unethical to do so. Therefore, an alternative approach was created.

To avoid scraping Zillow's live server, the html files were downloaded to a local host. Data extraction was performed on the mirrored sites. Chrome's built-in developer tool was used to examine the structure of the html files and find the fields containing the desired data.



Pre-Processing Steps

The website data were parsed using BeautifulSoup's html parser and stored as variables. These variables were then added to a Python dictionary where their keys would later be assigned to column names in the .csv file. The text corresponding to the "city" field was in all capitals and had a space preceding it. To store it consistently the functions ".lstrip()", ".lower()", and ".title()" were applied. The text "SYRACUSE" was thus transformed to "Syracuse". The listing price data type was a string containing a dollar sign and commas. The "\$"s and ","s were removed, and the data were transformed to numeric. The bedroom, bathroom, and area data were located together and needed to be split. Also, area was a string containing commas. Commas were removed, and the data type was transformed to numeric. These data were written to a .csv file which would be read by the "real_estate_analyzer.py" program.

Initially, all real estate listings were analyzed; however, many inconsistencies were found when examining these values. For example, apartments for sale were listed with the total number of apartments as the bedroom number, but the price was for each room individually. Foreclosure properties, sales of only lots, and homes for sale by owner also skewed the results. Ultimately, the decision was made to analyze only listings where the title was "House For Sale."

The program checks for null values in the remaining dataset, displays the number to the user, and allows the user to choose to delete them or replace them with average values. In most

cases, the number of null values is small enough to just remove them from the dataset for analysis.

Preliminary analysis of the data showed that most variable values fell within certain ranges with a few extreme examples outside of this range. To determine which values are outliers, a formula was used to calculate the z-score of each variable. Any variable with an absolute z-score of 3 or greater was considered an outlier and was excluded from the final data set.

Program Description

The program for this project is actually three separate programs that work together. The first, `real_estate_scraper.py`, allows the user to choose the city from which to extract data.

```
Anaconda Prompt - python real_estate_scraper.py
Real Estate Scraper

Choose city:

1 Albany, NY
2 Binghamton, NY
3 Buffalo, NY
4 Syracuse, NY

Please make a selection from the list above:
>
```

The user makes a selection by inputting the number corresponding to the desired city and the program begins reading each html file, extracting the desired data, and writing them to a .csv file.

```
Anaconda Prompt - python real_estate_scaper.py
Reading html files from:
bpage1.html
bpage2.html
bpage3.html
bpage4.html
bpage5.html
bpage6.html
bpage7.html
bpage8.html
bpage9.html
bpage10.html
bpage11.html
bpage12.html
bpage13.html
bpage14.html
bpage15.html
bpage16.html
bpage17.html
bpage18.html
bpage19.html
19 files successfully read!

Writing records to file...
452 records written to 'properties-binghamton-092518.csv'

Would you like to load more files?
>
```

The file then asks the user if they would like to load any more files. If “yes” is entered, the process repeats for the next chosen city. Once completed, the file asks the user if they would like to begin analyzing the data. If the user agrees, the `real_estate_analyzer.py` program is called. If not, instructions are provided to open this file when ready. Like the scraper program, the analyzer program presents a menu to the user allowing them to choose the city to analyze.

```
Anaconda Prompt - python real_estate_scaper.py
Real Estate Data Analysis
Choose city:

1 Albany, NY
2 Binghamton, NY
3 Buffalo, NY
4 Syracuse, NY

Please make a selection from the list above:
>
```

When a city is chosen (for these examples, Binghamton, NY was selected), the program reads the data from the .csv file and stores them in a pandas dataframe. It displays the number of null values and allows the user to delete or replace them. The program also displays the number of records containing outliers that will be removed and gives a total count of the rows of data.

```
Anaconda Prompt - python real_estate_scaper.py
Retrieved 452 records from 'properties-binghamton-092518.csv'
221 records contain data on houses for sale
8 records contain null values
Delete or Replace null values?
>del

Removing 8 records containing null values...
Removing 8 records containing outliers...
...205 records remaining
Press <enter> to continue:
>
```

When the user presses enter, they are taken to a main menu screen where analysis choices can be made. The user can view a specified number of listings, examine pricing data, examine data related to the number of bedrooms, examine data related to the number of bathrooms, examine data regarding the house area (in square feet), view a correlation matrix to learn which variables are related, display several linear regression models based on these associations, and use the linear model to predict house prices.


```

Anaconda Prompt - python real_estate_scaper.py
Main Menu

1 Display Listings
2 Display Pricing Data
3 Display Bedroom Data
4 Display Bathroom Data
5 Display Area Data
6 Display Correlation Matrix
7 Display Linear Regression Model(s)
8 Predict House Price
9 Exit

Please make a selection from the list above:
>

```

Choosing option one allows the user to view property listings. The listing title, address, city, state, zip code, number of bedrooms, number of bathrooms, area, and price are displayed for the number of listings chosen.

```

Anaconda Prompt - python real_estate_scaper.py
How many listings would you like to display?
>30

  title                address          city state  zip_code  beds  baths   area   price
0  House For Sale      4187 Cheryl Dr  Binghamton  NY    13903    3.0    1.0  1170.0  132900.0
3  House For Sale       79 Mary St  Binghamton  NY    13903    3.0    1.0  1152.0   64900.0
5  House For Sale     32 Highland Ave  Binghamton  NY    13905    4.0    2.0  1908.0  119900.0
6  House For Sale     54 Lincoln Ave  Binghamton  NY    13905    3.0    2.0  1778.0  139000.0
7  House For Sale    121 Schubert St  Binghamton  NY    13905    4.0    2.0  2340.0  140000.0
9  House For Sale     244 Bevier St  Binghamton  NY    13904    2.0    2.0  1033.0  117000.0
14 House For Sale     87 Tompkins St  Binghamton  NY    13903    2.0    1.0  1407.0   60000.0
16 House For Sale      6 Schiller St  Binghamton  NY    13905    4.0    2.0  1560.0  114900.0
17 House For Sale     166 Broad Ave  Binghamton  NY    13904    3.0    1.0  1043.0   69900.0
18 House For Sale   4163 Brady Hill Rd  Binghamton  NY    13903    3.0    2.0  1402.0  179000.0
20 House For Sale    133 Seminary Ave  Binghamton  NY    13905    4.0    2.0  1690.0  124900.0
21 House For Sale     16 Bennett Ave  Binghamton  NY    13905    3.0    2.0  1480.0  149900.0
23 House For Sale      4 Gordon Pl  Binghamton  NY    13905    3.0    2.0  1512.0   89900.0
25 House For Sale      2 Spurr Ave  Binghamton  NY    13903    4.0    2.0  2017.0  130000.0
26 House For Sale     85 Kneeland Ave  Binghamton  NY    13905    3.0    2.0  1422.0  107085.0
27 House For Sale  206 Old Pennsylvania Ave  Binghamton  NY    13903    3.0    2.0  1902.0  145000.0
28 House For Sale       8 Gold St  Binghamton  NY    13904    3.0    1.0  1627.0   45000.0
30 House For Sale     38 Oakridge Dr  Binghamton  NY    13903    4.0    3.0  2421.0  169900.0
31 House For Sale    209 E Frederick St  Binghamton  NY    13904    4.0    1.0  1702.0   89900.0
32 House For Sale    102 Chapin St  Binghamton  NY    13905    5.0    2.0  1218.0   85000.0
33 House For Sale     16 Linda Dr  Binghamton  NY    13905    3.0    3.0  2302.0  119900.0
35 House For Sale     12 Duane Ave  Binghamton  NY    13903    4.0    2.0  1554.0   84200.0
36 House For Sale     155 Mary St  Binghamton  NY    13903    3.0    1.0  1280.0   49400.0
37 House For Sale      6 Sherwood Ave  Binghamton  NY    13903    2.0    2.0  1126.0  120000.0
38 House For Sale     76 Kneeland Ave  Binghamton  NY    13905    4.0    2.0  1944.0  169900.0
40 House For Sale     97 Burr Ave  Binghamton  NY    13903    4.0    2.0  3168.0  139000.0
41 House For Sale     52 Bigelow St  Binghamton  NY    13904    3.0    2.0  1800.0   79900.0
42 House For Sale     69 Kendall Ave  Binghamton  NY    13903    4.0    1.0  1141.0  123900.0
43 House For Sale    28 Brookfield Rd  Binghamton  NY    13903    3.0    2.0  1883.0  179900.0
45 House For Sale    111 Beethoven St  Binghamton  NY    13905    5.0    2.0  2016.0  125000.0

Press <enter> to return to main menu:
>

```

Once done reviewing the listings, the user returns to the main menu and can make another choice. Screenshots from the remaining options will be displayed in the analysis methods section below.

Methods of Analysis

The remaining options make use of exploratory data analysis and linear regression modeling to allow the user to gain more insight into houses for sale in the selected city. EDA figures are displayed to the user and saved as image files. In addition, a log file is automatically updated with information about file creations, model creations, and program errors.

Data Exploration

The main strategy was to use exploratory data analysis to examine the frequency, distribution, range, and central tendency for the variables present in the real estate data. The variables under investigation include price, beds, baths, and area.

Pricing Data

```
Anaconda Prompt - python real_estate_scaper.py
Pricing Data

1 Descriptive Statistics
2 Display Pricing Boxplot
3 Display Pricing Frequency Histogram
4 Return to Main Menu

Please make a selection from the list above:
>
```

Selecting option one displays data regarding the min, mean, median, and max values of the houses for the selected city.

```
Anaconda Prompt - python real_estate_scaper.py
Housing Costs:

The minumum price is: $17000.00
The average price is: $121174.51
The median price is: $99500.00
The maximum price is: $424900.00

Press <enter> to return to menu:
>
```


Option two shows a graphical representation of the range and central tendency of the price data.

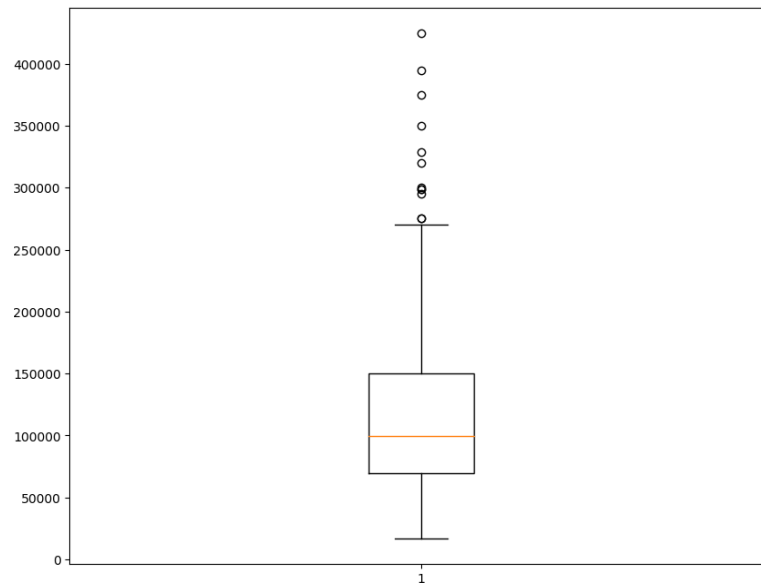


Figure 1: Housing Prices (USD)

Selecting option three displays a histogram showing a frequency distribution of the pricing data. The data appear to be normally distributed with a right-skew. Most prices fall between about \$25,000 and \$275,000 with a few higher priced homes skewing the results. This skewing is also revealed in the summary statistics above where the mean value is greater than the median value.

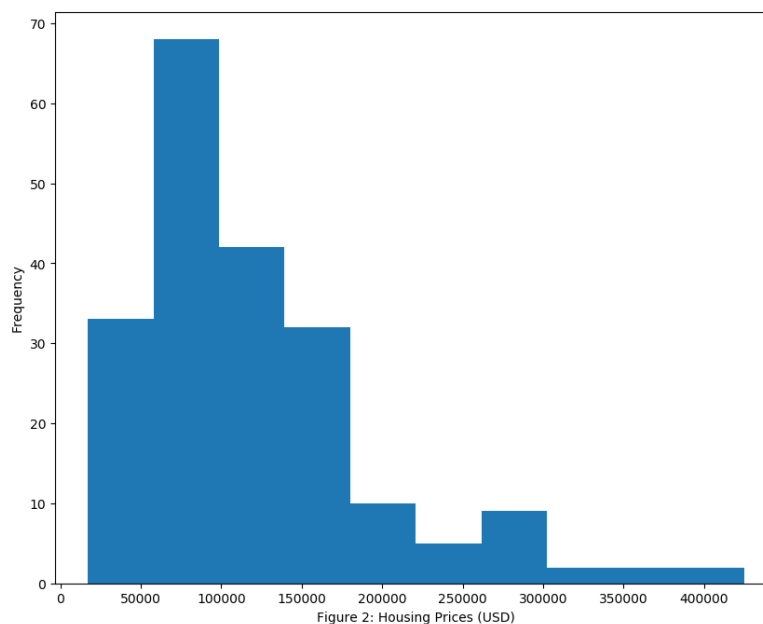


Figure 2: Housing Prices (USD)

Option four brings the user back to the main menu where the next variable can be examined.

Bedroom Data

```
Anaconda Prompt - python real_estate_scaper.py
Bedroom Data

1 Descriptive Statistics
2 Display Bedroom Boxplot
3 Display Bedroom Frequency Histogram
4 Display Relationship Between Bedrooms and Price
5 Return to Main Menu

Please make a selection from the list above:
>
```

EDA of bedroom data revealed a range of 1 to 6 bedrooms, with median and mean values of 3. In addition to displaying information about the range of values, average prices associated with these values are displayed.

```
Anaconda Prompt - python real_estate_scaper.py
Bedrooms:

The minimum number of bedrooms is: 1
The average number of bedrooms is: 3
The median number of bedrooms is: 3
The maximum number of bedrooms is: 6

The average price for a 1 bedroom home is: $ 74900.00
The average price for a 3 bedroom home is: $ 110909.72
The average price for a 3 bedroom home is: $ 110909.72
The average price for a 6 bedroom home is: $ 135500.00

Press <enter> to return to menu:
>
```

Again, a boxplot is shown to graphically illustrate the range and central tendency of the bedroom data.

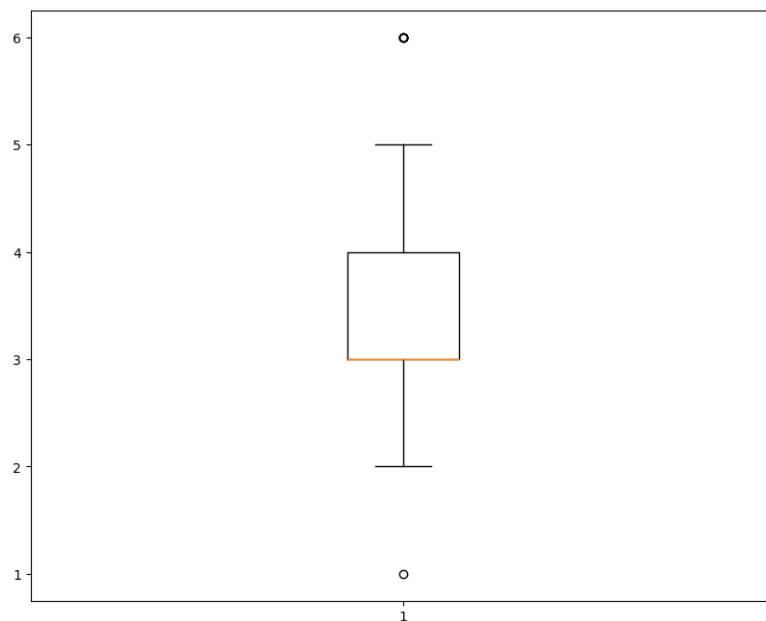


Figure 3: Number of Bedrooms

A frequency distribution histogram shows a normal distribution with most homes having three bedrooms. The mean and median number of bedrooms are equal and there is no significant skew.

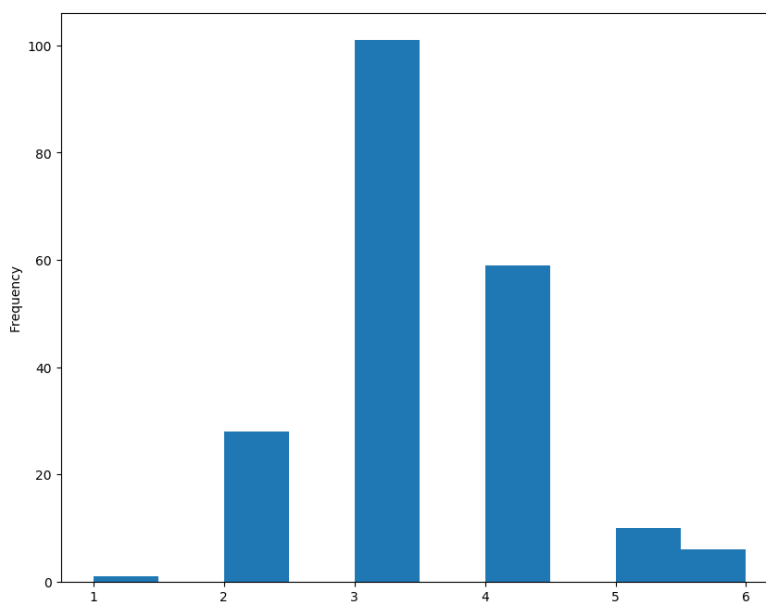


Figure 4: Number of Bedrooms

The user is also presented with the option to display a scatterplot showing the relationship between the number of bedroom and the house price. As seen in the figure below, there appears to be a somewhat positive correlation between the values, as expected. However, homes with six bedrooms tend to have lower prices which may affect how useful this variable would be in model building.

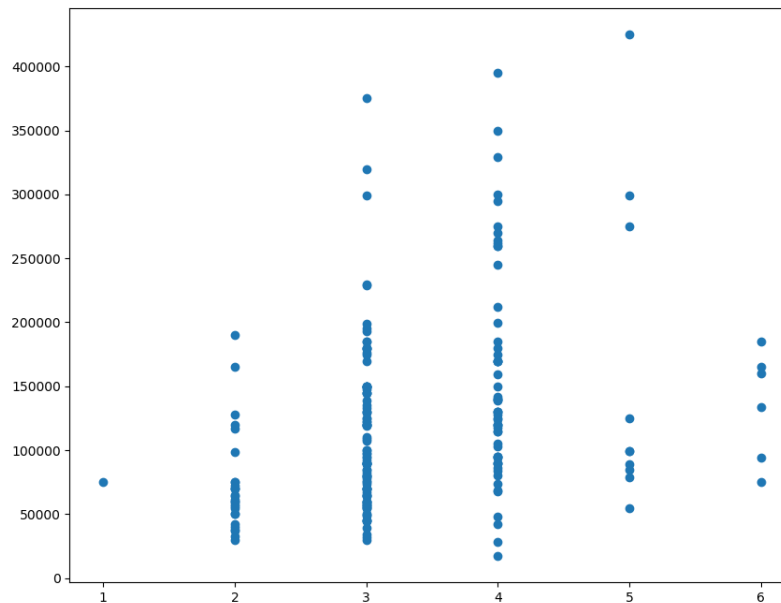


Figure 5: Relationship between number of bedrooms and price

Bathroom Data

After viewing the bedroom data, the user returns to the main menu and is able to select another option, such as bathroom data.

```
Anaconda Prompt - python real_estate_scaper.py
Bathroom Data

1 Descriptive Statistics
2 Display Bathroom Boxplot
3 Display Bathroom Frequency Histogram
4 Display Relationship Between Bathrooms and Price
5 Return to Main Menu

Please make a selection from the list above:
>
```

Displaying the summary statistics shows that the number of bathrooms (for Binghamton, NY) ranges between 1 and 4, with a mean value of 1 and a median value of 2. Average prices for homes with these numbers of bathrooms is also displayed.

Anaconda Prompt - python real_estate_scaper.py

Bathrooms:

The minimum number of bathrooms is: 1

The average number of bathrooms is: 1

The median number of bathrooms is: 2

The maximum number of bathrooms is: 4

The average price for a 1 bathroom home is: \$ 71520.80

The average price for a 1 bathroom home is: \$ 71520.80

The average price for a 2 bathroom home is: \$ 112380.43

The average price for a 4 bathroom home is: \$ 286070.00

Press <enter> to return to menu:

>

The boxplot illustrates a graphical representation of the above-mentioned range and central tendency.

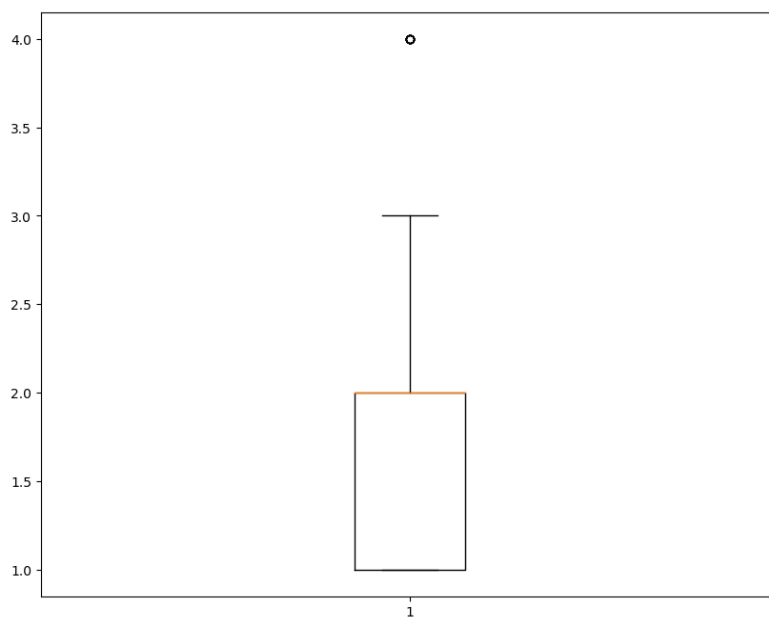
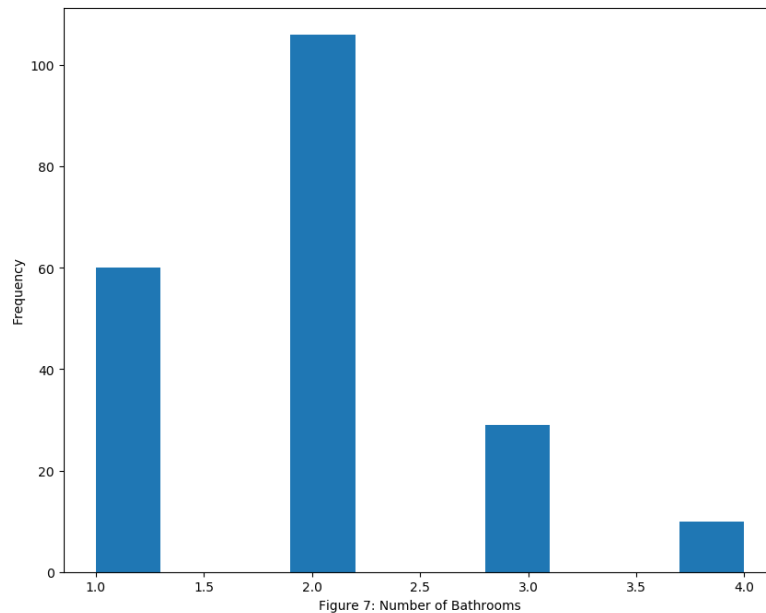
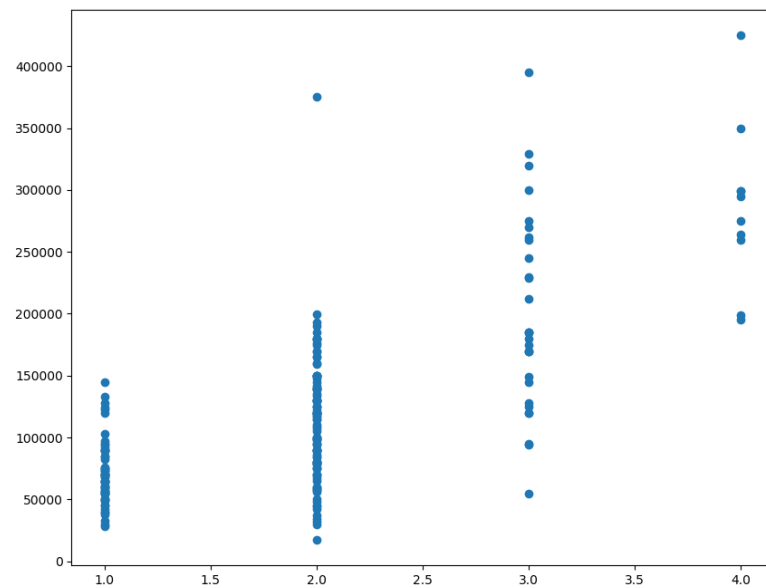


Figure 6: Number of Bathrooms

The frequency histogram reflects the information above with a slightly left-skewed normal distribution (median > mean). Most homes have one or two bathrooms.



The relationship between the number of bathrooms is illustrated by the scatterplot below, which shows a strong positive correlation. Number of bathrooms may be a good predictor variable for a house price.



Area (in square feet) Data

The area of the homes is examined next after returning to the main menu and choosing option 5. As before, the user is presented with a menu of items to view data associated with this variable.


```
Anaconda Prompt - python real_estate_scaper.py
Area Data

1 Descriptive Statistics
2 Display Area Boxplot
3 Display Area Frequency Histogram
4 Display Relationship Between Area and Price
5 Return to Main Menu

Please make a selection from the list above:
>
```

Displaying the summary statistics shows a range of 700 to 5322 square feet. The average area is higher than the median value, suggesting a right-skew. Average price appears to be positively correlated with size.

```
Anaconda Prompt - python real_estate_scaper.py
Area:

The minimum area is: 700 sq ft
The average area is: 1879 sq ft
The median area is: 1690 sq ft
The maximum area is: 5322 sq ft

The average price for a 700 square foot home is: $ 72500.00
The average price for a 2000 square foot home is: $ 179900.00
The average price for a 5322 square foot home is: $ 350000.00

Press <enter> to return to menu:
>
```

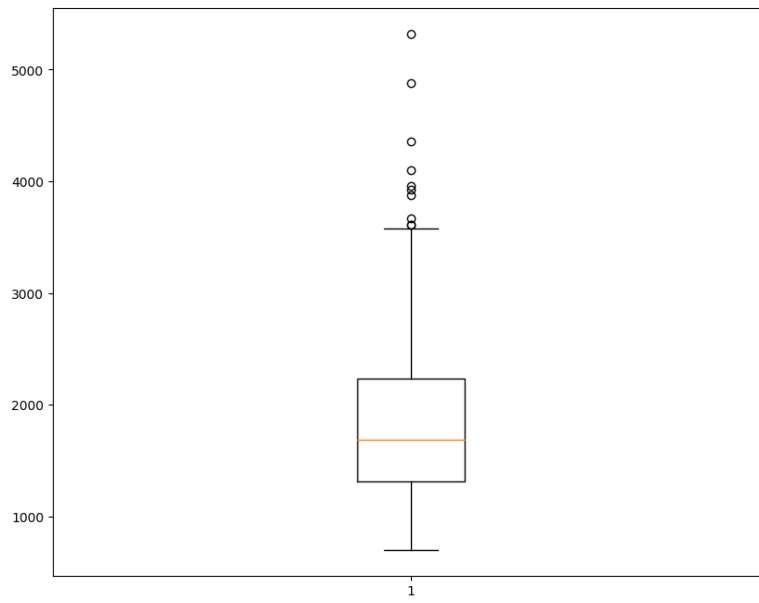


Figure 9: Area (sq ft)

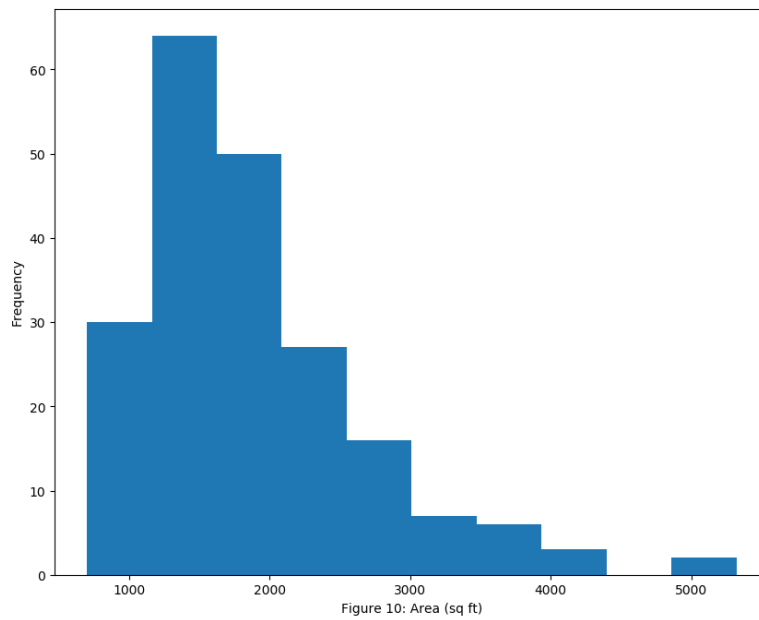


Figure 10: Area (sq ft)



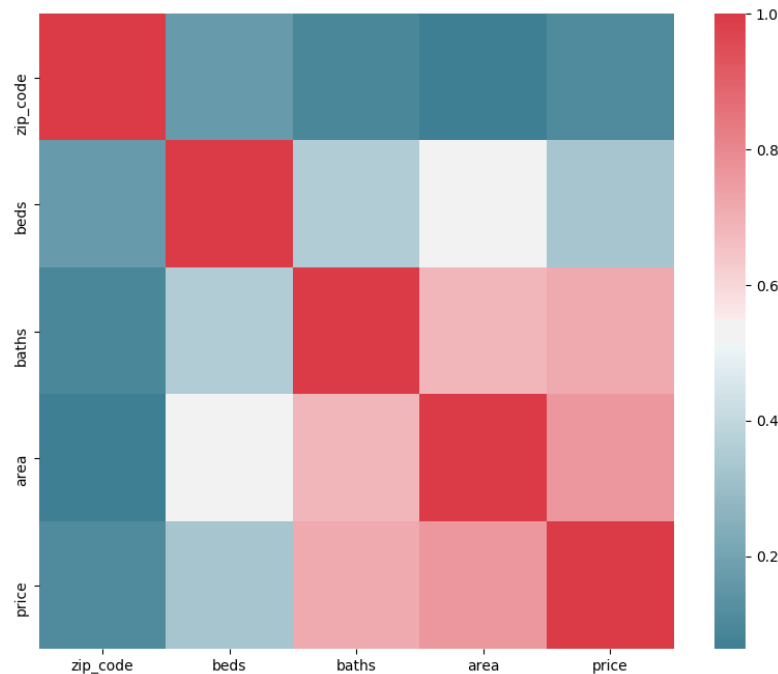
The graphical displays corroborate the summary statistics. The boxplot displays the range and central tendency and shows that a number of higher area values fall outside of the normal range. The size distribution is normally distributed with a right tail. The scatterplot demonstrates a strong positive linear correlation between area and price.

Correlation of Variables

To better understand the relationship between the variables and which variables have the greatest impact on price, a correlation matrix was created. This matrix is displayed in both numerical and graphical forms to the viewer.

Correlation Matrix:

	zip_code	beds	baths	area	price
zip_code	1.000000	0.168300	0.094918	0.062963	0.108883
beds	0.168300	1.000000	0.356516	0.534382	0.329186
baths	0.094918	0.356516	1.000000	0.682691	0.711326
area	0.062963	0.534382	0.682691	1.000000	0.762306
price	0.108883	0.329186	0.711326	0.762306	1.000000



As can be seen, there are relatively strong correlations between number of bathrooms and price, and the overall area and price, with area being stronger. Additionally, the number of bedrooms and bathrooms correlate with the area, unsurprisingly. What this means is that there is likely collinearity between beds, baths, and area. In the next section on building a linear regression model, the most parsimonious model can be created using only area as the independent variable and price as the dependent variable.

Regression Modeling

The program now allows the user to create a linear regression model by choosing which variable(s) to include as the independent variable.

```
Anaconda Prompt - python real_estate_scaper.py
Display Linear Regression Model:

1 Beds, Baths, and Area vs. Price
2 Beds vs. Price
3 Baths vs. Price
4 Area vs. Price
5 Return to Main Menu

Please make a selection from the list above:
>
```

On their own, each variable (number of bedrooms, number of bathrooms, and area) had a positive correlation with price; however, as previously mentioned, a model of area vs. price is

the most parsimonious. Once the variable is selected, a multiple linear regression model is built using ordinary least squares (OLS) from Python's statsmodel package.

Anaconda Prompt - python real_estate_scaper.py

```

                                OLS Regression Results
=====
Dep. Variable:                price    R-squared:                0.885
Model:                        OLS      Adj. R-squared:           0.885
Method:                    Least Squares  F-statistic:             1573.
Date:                Tue, 25 Sep 2018    Prob (F-statistic):       7.83e-98
Time:                16:33:51           Log-Likelihood:          -2501.0
No. Observations:                205     AIC:                    5004.
Df Residuals:                    204     BIC:                    5007.
Df Model:                        1
Covariance Type:                nonrobust
=====
               coef      std err          t      P>|t|      [0.025      0.975]
-----
area             65.4174      1.650     39.658      0.000      62.165      68.670
=====
Omnibus:                 18.811    Durbin-Watson:           1.996
Prob(Omnibus):            0.000    Jarque-Bera (JB):         34.028
Skew:                    0.479    Prob(JB):                 4.08e-08
Kurtosis:                 4.751    Cond. No.                  1.00
=====

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Press <enter> to return to model menu:
>

```

The area vs. price model, illustrated above, explains approximately 89% of the variance from the mean and has a very high F-statistic. The program allows the user to create other models in order to determine which is best for the given dataset. Once the best model is found it can be used to predict housing prices.

Price Prediction (Using Regression Model)

Based on the linear relationship between the area variable and price, the linear regression model is used to predict the average price of a home based on user inputted criteria. The program prompts the user to enter the chosen independent variable and the variable value. It then calculates and displays the predicted price.

```

Anaconda Prompt - python real_estate_scaper.py
Home Price Prediction

Choose independent variable:
1 Beds
2 Baths
3 Area

Please make a selection from the list above:
>3

Enter area (sqft):
>2000

Estimated Price: $130834.76

Make another prediction? (y/n)
>

```

It is important to note that there are many more variables not accounted for in the model that can have a significant impact on a home price. Also, it is important to remember that it is not prudent to make predictions of a response variable based on explanatory variables outside of the range within the original dataset.

Comparing Data from all the Cities

After playing with predictions, the user can return to the main menu. Once all of the data analysis tools have been explored, they can choose to exit the analyzer program. If all of the cities' data have been previously scraped and examined, the user can choose to compare their results.

```

Anaconda Prompt - python real_estate_scaper.py
Main Menu

1 Display Listings
2 Display Pricing Data
3 Display Bedroom Data
4 Display Bathroom Data
5 Display Area Data
6 Display Correlation Matrix
7 Display Linear Regression Model(s)
8 Predict House Price
9 Exit

Please make a selection from the list above:
>9

Analyze another city?
>no

Would you like to compare all cities?
>yes

```

If this option is chosen, the program will call the third Python program, `real_estate_compare.py`. This program pulls each of the four previously-created .csv files into pandas dataframes, performs the same data-cleaning steps as before, and merges them into one dataframe. The user is presented with a menu of options to choose which variable to compare among the four cities.

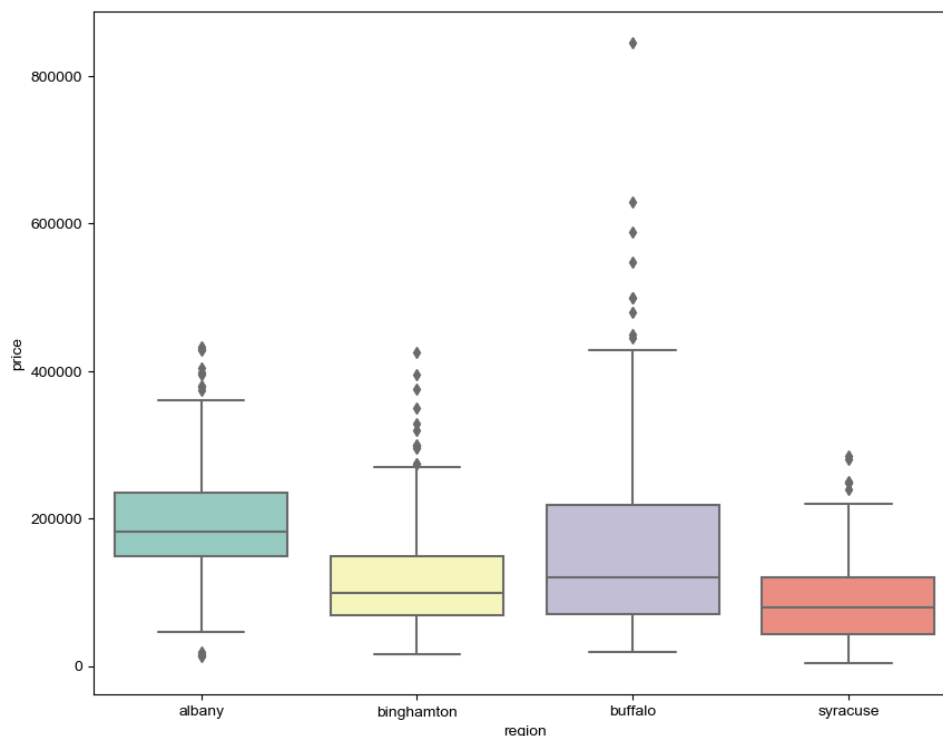
```
Anaconda Prompt - python real_estate_scaper.py
Compare All City Real Estate Data

1 Display Pricing Data
2 Display Bedroom Data
3 Display Bathroom Data
4 Display Area Data
5 Exit

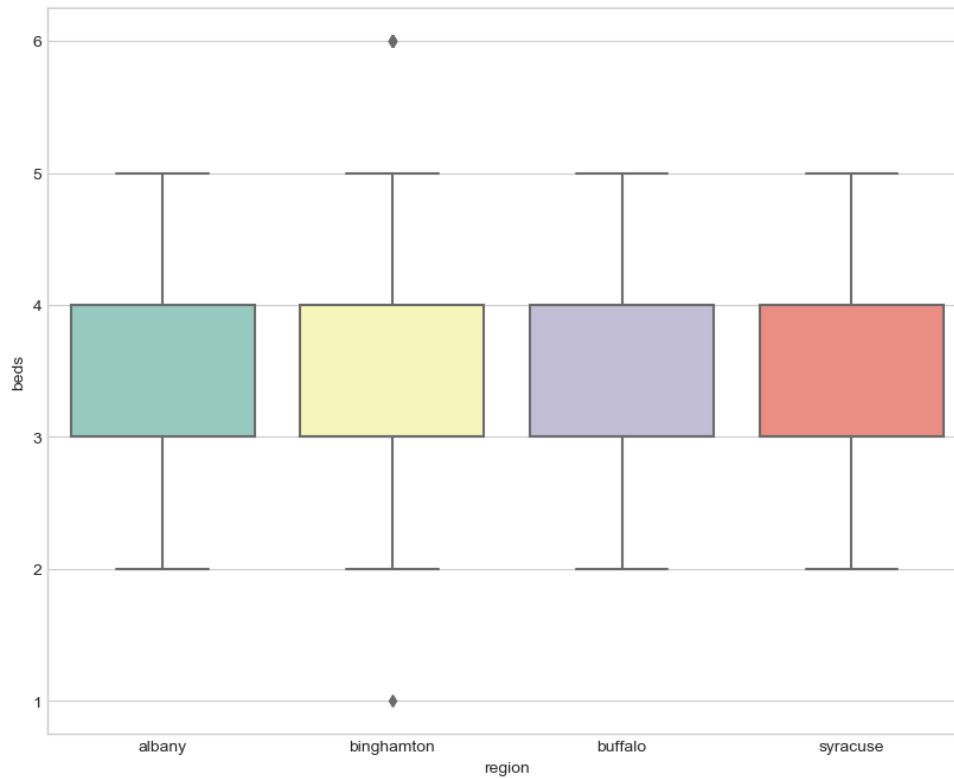
Please make a selection from the list above:
>
```

Each option displays side-by-side boxplots comparing the variable in question. Each figure is displayed below:

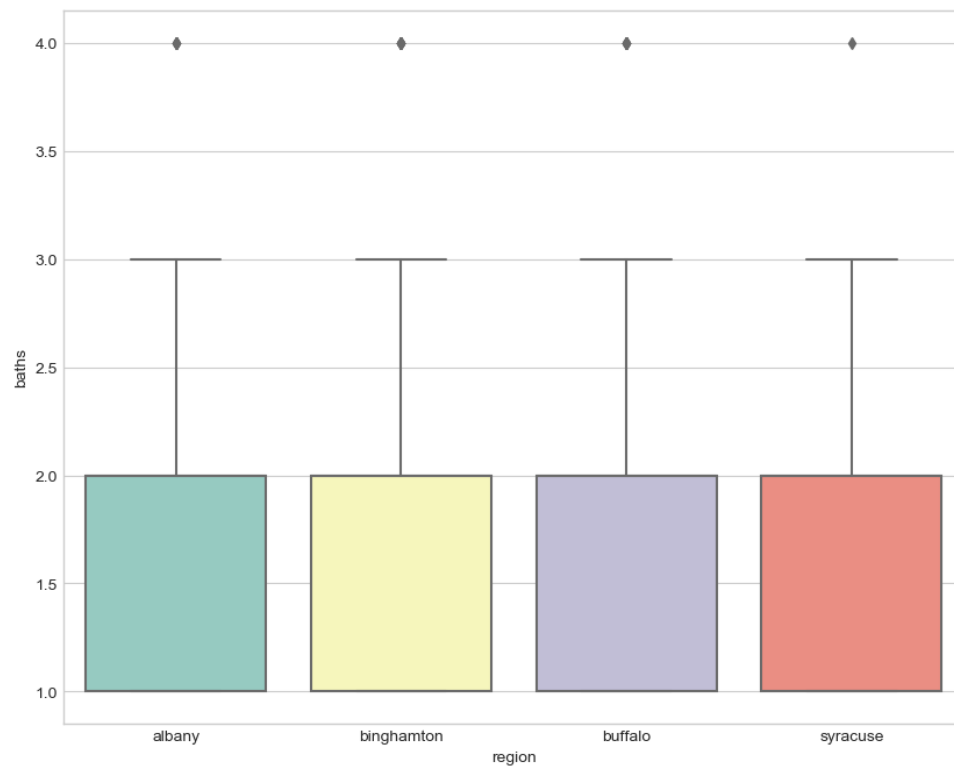
Pricing Data



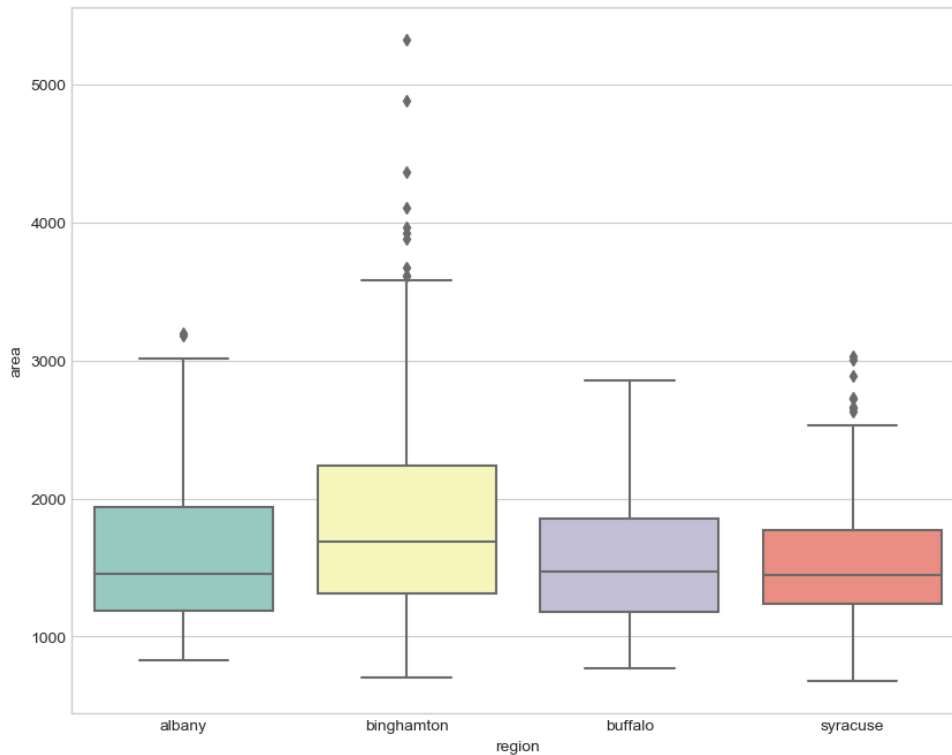
Bedroom Data



Bathroom Data



Area Data



Conclusions

Real-estate data can be easily scraped from websites and used for analysis. This data can give users a great deal of information about the types of homes on the market within a certain region. These data can also provide details about trends in prices based on identified variables such as the number of rooms and size of the home. Models can be built to predict the average cost of a home based on these criteria, with the understanding that these models are incomplete and only represent linear models of often non-completely linear relationships.

In addition to analyzing data for each city individually, comparisons can be made among the cities. Despite the differences between the four cities selected for this project, there is relative homogeneity among them. They all share similar bedroom, bathroom, and area characteristics.

The variable with the largest variance is price. Future work could focus on examining outside data sources (such as population size/density, average income, average education level, crime rate, etc.) to see what variables can best explain the variance in home values.

Python Code

real_estate_scraper.py

```
'''
Name of file: real_estate_scraper.py
Date created: August 29, 2018
Date last updated: September 25, 2018
Created by: James Ingram
Purpose of Program: Scrape real estate data from Zillow.com, parse it with BeautifulSoup, and write
the output to a csv file
'''

# imports necessary packages, libraries, and dependencies
from bs4 import BeautifulSoup
import unicodedcsv as csv
import codecs
import datetime
import os
import logging

# Web scraping from live website if !robots.txt
# from urllib import request
# html = request.urlopen(pageurl).read().decode('utf-8')
# soup = BeautifulSoup(html, 'html.parser')

def main_menu():
    # To avoid scraping data from live web server each page was downloaded onto localhost
    # Data from zillow.com, search term = "Albany, NY"
    alb_urls = ["apage1.html", "apage2.html", "apage3.html", "apage4.html", "apage5.html",
"apage6.html", "apage7.html", "apage8.html", "apage9.html", "apage10.html", "apage11.html",
"apage12.html", "apage13.html", "apage14.html", "apage15.html", "apage16.html", "apage17.html",
"apage18.html", "apage19.html", "apage20.html"]
    # Data from zillow.com, search term = "Binghamton, NY"
    bing_urls = ["bpage1.html", "bpage2.html", "bpage3.html", "bpage4.html", "bpage5.html",
"bpage6.html", "bpage7.html", "bpage8.html", "bpage9.html", "bpage10.html", "bpage11.html",
"bpage12.html", "bpage13.html", "bpage14.html", "bpage15.html", "bpage16.html", "bpage17.html",
"bpage18.html", "bpage19.html"]
    # Data from zillow.com, search term = "Buffalo, NY"
    buff_urls = ["bupage1.html", "bupage2.html", "bupage3.html", "bupage4.html", "bupage5.html",
"bupage6.html", "bupage7.html", "bupage8.html", "bupage9.html", "bupage10.html", "bupage11.html",
"bupage12.html", "bupage13.html", "bupage14.html", "bupage15.html", "bupage16.html",
"bupage17.html", "bupage18.html", "bupage19.html", "bupage20.html"]
    # Data from zillow.com, search term = "Syracuse, NY"
    syr_urls = ["spage1.html", "spage2.html", "spage3.html", "spage4.html", "spage5.html",
"spage6.html", "spage7.html", "spage8.html", "spage9.html", "spage10.html", "spage11.html",
"spage12.html", "spage13.html", "spage14.html", "spage15.html", "spage16.html", "spage17.html",
"spage18.html", "spage19.html", "spage20.html"]

    # create empty list to store data from scraped html documents
```

```

properties_list = []
# set initial count to 0
writecount = 0
# create date format to append to .csv filename
now = datetime.datetime.today().strftime("%m%d%y")

# function to scrape data from saved html documents
def pagescraper(url):
    # reads html document
    f = codecs.open(url, 'r')
    # use BeautifulSoup to parse html
    soup = BeautifulSoup(f, 'html.parser')
    # finds div and class tags associated with desired data
    all = soup.find_all("div", {"class": "zsg-photo-card-content"})
    # iterates through data to extract desired fields. If data is missing, makes it null value
    and sends debug report to log file
    for item in all:
        try:
            raw_title = item.find("h4").text
        except:
            raw_title = None
            logging.debug("Listing title not found")
        try:
            raw_address = item.find("span", {"itemprop": "streetAddress"}).text
        except:
            raw_address = None
            logging.debug("Listing address not found")
        try:
            raw_city =
item.find("span", {"itemprop": "addressLocality"}).text.lstrip().lower().title()
        except:
            raw_city = None
            logging.debug("Listing city not found")
        try:
            raw_state = item.find("span", {"itemprop": "addressRegion"}).text
        except:
            raw_state = None
            logging.debug("Listing state not found")
        try:
            raw_zip_code = str(item.find("span", {"itemprop": "postalCode"}).text)
        except:
            raw_zip_code = None
            logging.debug("Listing zip code not found")
        try:
            raw_price = int(item.find("span", {"class": "zsg-photo-card-
price"}).text.replace("$", "").replace(", ", ""))
        except:
            raw_price = None

```



```

        logging.debug("Listing price not found")
    try:
        raw_bed_info = int(item.find("span", {"class": "zsg-photo-card-
info"}).text.split()[0])
    except:
        raw_bed_info = None
        logging.debug("Listing bedroom data not found")
    try:
        raw_bath_info = int(item.find("span", {"class": "zsg-photo-card-
info"}).text.split()[3])
    except:
        raw_bath_info = None
        logging.debug("Listing bathroom data not found")
    try:
        raw_sqft_info = int(item.find("span", {"class": "zsg-photo-card-
info"}).text.split()[6].replace(", ", ""))
    except:
        raw_sqft_info = None
        logging.debug("Listing area data not found")

    # append items to properties list
    properties = {
        'title': raw_title,
        'address': raw_address,
        'city': raw_city,
        'state': raw_state,
        'zip_code': raw_zip_code,
        'price': raw_price,
        'beds': raw_bed_info,
        'baths': raw_bath_info,
        'area': raw_sqft_info,
        'region': fname
    }
    properties_list.append(properties)
return properties_list

# creates log file and sets logging parameters
logging.basicConfig(
    filename='real_estate.log',
    level=logging.DEBUG,
    format='%(asctime)s %(levelname)s %(module)s - %(funcName)s: %(message)s',
    datefmt="%Y-%m-%d %H:%M:%S")
try:
    os.system('cls')
except:
    os.system('clear')

print("Real Estate Scraper\n")

```

```

print("Choose city:\n")
def menu(list,question):
    for item in list:
        print(1 + list.index(item), item)
    return input(question)
while True:
    items = ["Albany, NY", "Binghamton, NY", "Buffalo, NY", "Syracuse, NY"]
    choice = menu(items,"\nPlease make a selection from the list above:\n>")
    try:
        choice = int(choice)
        break
    except:
        print("Invalid selection, please try again\n")
        continue
print()

# User makes a selection
if choice == 1:
    urls = alb_urls
    fname = "albany"
elif choice == 2:
    urls = bing_urls
    fname = "binghamton"
elif choice == 3:
    urls = buff_urls
    fname = "buffalo"
elif choice == 4:
    urls = syr_urls
    fname = "syracuse"
else:
    main_menu()

try:
    os.system('cls')
except:
    os.system('clear')
print("Reading html files from:")
filecount = 0
# iterates through list of html documents and runs pagescraper function
for url in urls:
    print(url)
    pagescraper(url)
    filecount +=1
print(filecount, "files successfully read!")

print("\nWriting records to file...")
fh = open("properties-" + fname + "-" + now + ".csv", "wb")
# writes extracted data to .csv file

```

```

try:
    with fh as csvfile:
        fieldnames =
['title','address','city','state','zip_code','price','beds','baths','area','region']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()
        for row in properties_list:
            writecount += 1
            writer.writerow(row)
        fh.close()
        logging.info("Records successfully written to file")
        print(writecount, "records written to 'properties-" + fname + "-" + now + ".csv'")
except:
    logging.debug("Records not written to file")
    print("Error: records could not be written to file!")
    quit()
rescrape = input("\nWould you like to load more files?\n>")
if rescrape.lower().startswith("y"):
    main_menu()
else:
    analyze = input("\nWould you like to analyze the records now?\n>")
    if analyze.lower().startswith("y"):
        os.system("python real_estate_analyzer.py")
    else:
        print('\nTo analyze the records, run "real_estate_analyzer.py"')
        print("Goodbye!")
        quit()

#####
# Program execution begins here
#####
main_menu()

```

real_estate_analyzer.py

```
'''
Name of file: real_estate_analyzer.py
Date created: August 29, 2018
Date last updated: September 25, 2018
Created by: James Ingram
Purpose of Program: Create pandas dataframe from csv file (output of real_estate_scraper.py) and
perform exploratory data analysis
'''

# imports necessary packages, libraries, and dependencies
import datetime
import logging
import os
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from scipy import stats

# created date format for filename
now = datetime.datetime.today().strftime("%m%d%y")

# creates log file and sets logging parameters
logging.basicConfig(
    filename='real_estate.log',
    level=logging.DEBUG,
    format='%(asctime)s %(levelname)s %(module)s - %(funcName)s: %(message)s',
    datefmt="%Y-%m-%d %H:%M:%S")

# function to show menu and run all other functions
def main_program():
    # function to display a menu of options linked to other functions
    def display_options():
        # clears console screen for Windows (cls) or Linux (clear)
        try:
            os.system('cls')
        except:
            os.system('clear')
        # Displays menu and prompts user for input
        print("Main Menu\n")
        def menu(list,question):
            for item in list:
                print(1 + list.index(item), item)
            return input(question)
```

```

while True:
    items = ["Display Listings", "Display Pricing Data", "Display Bedroom Data", "Display Bathroom Data", "Display Area Data", "Display Correlation Matrix", "Display Linear Regression Model(s)", "Predict House Price", "Exit"]
    choice = menu(items, "\nPlease make a selection from the list above:\n>")
    try:
        choice = int(choice)
        break
    except:
        print("Invalid selection, please try again\n")
        continue
print("")
# User makes a selection
if choice == 1:
    display_listings()
elif choice == 2:
    display_prices()
elif choice == 3:
    display_beds()
elif choice == 4:
    display_baths()
elif choice == 5:
    display_area()
elif choice == 6:
    display_comatrix()
elif choice == 7:
    display_linear()
elif choice == 8:
    predict_linear()
elif choice == 9:
    exit_choice = input("Analyze another city?\n>")
    if exit_choice.lower().startswith('n'):
        compare = input("\nWould you like to compare all cities?\n>")
        if compare.lower().startswith('y'):
            try:
                os.system('python real_estate_compare.py')
            except:
                print("\nUnable to load program at this time")
                print("Goodbye!")
                quit()
        else:
            try:
                os.system('cls')
            except:
                os.system('clear')
            print("Goodbye!")
            quit()
    else:

```



```

        main_program()
    else:
        print("Invalid selection, please try again")
        display_options()
# function to display real estate listings in pandas dataframe
def display_listings():
    try:
        os.system('cls')
    except:
        os.system('clear')
    # prompts user to enter number of listings to display
    while True:
        list_num = input("How many listings would you like to display?\n>")
        try:
            list_num = int(list_num)
            break
        except:
            print("Error:",list_num,"is not a valid number")
            continue
    # displays listings
    print(df2[:list_num])
    # prompts user to return to menu screen
    input("\nPress <enter> to return to main menu:\n>")
    display_options()

# function to display price data to user
def display_prices():
    # clears console screen for Windows (cls) or Linux (clear)
    try:
        os.system('cls')
    except:
        os.system('clear')
    # Displays menu and prompts user for input
    print("Pricing Data\n")
    def menu(list,question):
        for item in list:
            print(1 + list.index(item), item)
        return input(question)
    while True:
        items = ["Descriptive Statistics","Display Pricing Boxplot", "Display Pricing Frequency Histogram", "Return to Main Menu"]
        choice = menu(items,"\nPlease make a selection from the list above:\n>")
        try:
            choice = int(choice)
            break
        except:
            print("Invalid selection, please try again\n")
            continue

```

```

print("")
# User makes a selection
if choice == 1:
    try:
        os.system('cls')
    except:
        os.system('clear')
    # calculates and displays min, mean, median, and max prices
    min_price = '{0:.2f}'.format(df2['price'].min())
    avg_price = '{0:.2f}'.format(df2['price'].mean())
    med_price = '{0:.2f}'.format(df2['price'].median())
    max_price = '{0:.2f}'.format(df2['price'].max())
    print("Housing Costs:\n")
    print("The minumum price is: $" + min_price)
    print("The average price is: $" + avg_price)
    print("The median price is: $" + med_price)
    print("The maximum price is: $" + max_price)
    print()
    input("Press <enter> to return to menu:\n")
    display_prices()
elif choice == 2:
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Displaying Housing Prices Boxplot...")
    # creates figure showing boxplot of housing prices
    fig1 = plt.figure(1, figsize=(10,8))
    ax1 = fig1.add_subplot(111)
    ax1.set_xlabel("Figure 1: Housing Prices (USD)")
    ax1.get_yaxis().tick_left()
    bp = ax1.boxplot(df2['price'])
    # displays figure to user
    plt.show()
    # saves figure to file
    try:
        fig1.savefig('fig1-' + fname + '.png', bbox_inches='tight')
        # displays file creation to user and appends log file
        print("Created file: fig1.png")
        logging.info("Figure 1 successfully created")
    except:
        print("Unable to save file: fig1.png")
        logging.debug("Figure 1 not saved")
    input("\nPress <enter> to return to menu:\n")
    display_prices()
elif choice == 3:
    try:
        os.system('cls')

```

```

except:
    os.system('clear')
print("Displaying Housing Prices Frequency Histogram")
# creates figure showing histogram of housing price distribution
fig2 = plt.figure(2, figsize=(10,8))
ax2 = fig2.add_subplot(111)
ax2.set_xlabel("Figure 2: Housing Prices (USD)")
ax2.set_ylabel("Frequency")
ht = ax2.hist(df2['price'])
# displays figure to user
plt.show()
# saves figure to file
try:
    fig2.savefig('fig2-' + fname + '.png', bbox_inches='tight')
    print("Created file: fig2.png")
    logging.info("Figure 2 successfully created")
except:
    print("Unable to save file: fig2.png")
    logging.debug("Figure 2 not saved")
    input("\nPress <enter> to return to menu:\n>")
    display_prices()
elif choice == 4:
    display_options()
else:
    print("Invalid selection, please try again")
    display_prices()

# function to display bedroom data to user
def display_beds():
    # clears console screen for Windows (cls) or Linux (clear)
    try:
        os.system('cls')
    except:
        os.system('clear')
    # Displays menu and prompts user for input
    print("Bedroom Data\n")
    def menu(list,question):
        for item in list:
            print(1 + list.index(item), item)
        return input(question)
    while True:
        items = ["Descriptive Statistics","Display Bedroom Boxplot", "Display Bedroom Frequency Histogram", "Display Relationship Between Bedrooms and Price", "Return to Main Menu"]
        choice = menu(items,"\nPlease make a selection from the list above:\n>")
        try:
            choice = int(choice)
            break
        except:

```

```

        print("Invalid selection, please try again\n")
        continue
print("")
# User makes a selection
if choice == 1:
    try:
        os.system('cls')
    except:
        os.system('clear')
    # calculates and displays min, mean, median, and max bedrooms
    min_beds = int(df2['beds'].min())
    avg_beds = int(df2['beds'].mean())
    med_beds = int(df2['beds'].median())
    max_beds = int(df2['beds'].max())
    print("Bedrooms:\n")
    print("The minimum number of bedrooms is:",min_beds)
    print("The average number of bedrooms is:",avg_beds)
    print("The median number of bedrooms is:",med_beds)
    print("The maximum number of bedrooms is:",max_beds)
    print()
    # displays average price for min, mean, median, and max bedrooms
    print("The average price for a", min_beds, "bedroom home is:
$, '{0:.2f}'.format(df2[df2['beds']==min_beds]['price'].mean()))
    print("The average price for a", avg_beds, "bedroom home is:
$, '{0:.2f}'.format(df2[df2['beds']==avg_beds]['price'].mean()))
    print("The average price for a", med_beds, "bedroom home is:
$, '{0:.2f}'.format(df2[df2['beds']==med_beds]['price'].mean()))
    print("The average price for a", max_beds, "bedroom home is:
$, '{0:.2f}'.format(df2[df2['beds']==max_beds]['price'].mean()))
    print()
    input("Press <enter> to return to menu:\n")
    display_beds()
elif choice == 2:
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Displaying Bedrooms Boxplot...")
    # creates figure displaying boxplot of bedrooms
    fig3 = plt.figure(3, figsize=(10,8))
    ax3 = fig3.add_subplot(111)
    ax3.set_xlabel("Figure 3: Number of Bedrooms")
    ax3.get_yaxis().tick_left()
    bp2 = ax3.boxplot(df2['beds'])
    # displays figure to user
    plt.show()
    # saves figure to file
    try:

```

```

        fig3.savefig('fig3-' + fname + '.png', bbox_inches='tight')
        print("Created file: fig3.png")
        logging.info("Figure 3 successfully created")
    except:
        print("Unable to save file: fig3.png")
        logging.debug("Figure 3 not saved")
    input("\nPress <enter> to return to menu:\n>")
    display_beds()
elif choice == 3:
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Displaying Bedrooms Frequency Histogram...")
    # creates figure displaying histogram of bedroom frequency distribution
    fig4 = plt.figure(4, figsize=(10,8))
    ax4 = fig4.add_subplot(111)
    ax4.set_xlabel("Figure 4: Number of Bedrooms")
    ax4.set_ylabel("Frequency")
    ht2 = ax4.hist(df2['beds'])
    # displays figure to user
    plt.show()
    # saves figure to file
    try:
        fig4.savefig('fig4-' + fname + '.png', bbox_inches='tight')
        print("Created file: fig4.png")
        logging.info("Figure 4 successfully created")
    except:
        print("Unable to save file: fig4.png")
        logging.debug("Figure 4 not saved")
    input("\nPress <enter> to return to menu:\n>")
    display_beds()
elif choice == 4:
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Displaying Relationship Between Bedrooms and Price...")
    # creates figure displaying relationship between number of bedrooms and price
    fig5 = plt.figure(5, figsize=(10,8))
    X = df2['beds']
    Y = df2['price']
    plt.scatter(X,Y)
    # displays figure to user
    plt.show()
    # saves figure to file
    try:
        fig5.savefig('fig5-' + fname + '.png', bbox_inches='tight')

```



```

        print("Created file: fig5.png")
        logging.info("Figure 5 successfully created")
    except:
        print("Unable to save file: fig5.png")
        logging.debug("Figure 5 not saved")
    input("\nPress <enter> to return to menu:\n>")
    display_beds()
elif choice == 5:
    display_options()
else:
    print("Invalid selection, please try again")
    display_beds()

# function to display bathroom data to user
def display_baths():
    # clears console screen for Windows (cls) or Linux (clear)
    try:
        os.system('cls')
    except:
        os.system('clear')
    # Displays menu and prompts user for input
    print("Bathroom Data\n")
    def menu(list,question):
        for item in list:
            print(1 + list.index(item), item)
        return input(question)
    while True:
        items = ["Descriptive Statistics","Display Bathroom Boxplot", "Display Bathroom
Frequency Histogram", "Display Relationship Between Bathrooms and Price", "Return to Main Menu"]
        choice = menu(items,"\nPlease make a selection from the list above:\n>")
        try:
            choice = int(choice)
            break
        except:
            print("Invalid selection, please try again\n")
            continue
    print("")
    # User makes a selection
    if choice == 1:
        try:
            os.system('cls')
        except:
            os.system('clear')
        # calculates and displays min, mean, median, and max number of bathrooms
        min_baths = int(df2['baths'].min())
        avg_baths = int(df2['baths'].mean())
        med_baths = int(df2['baths'].median())
        max_baths = int(df2['baths'].max())

```

```

print("Bathrooms:\n")
print("The minimum number of bathrooms is:",min_baths)
print("The average number of bathrooms is:",avg_baths)
print("The median number of bathrooms is:",med_baths)
print("The maximum number of bathrooms is:",max_baths)
print()
# displays average price for properties with min, mean, median, and max number of
bathrooms
print("The average price for a", min_baths, "bathroom home is:
$', '{0:.2f}'.format(df2[df2['baths']==min_baths]['price'].mean()))
print("The average price for a", avg_baths, "bathroom home is:
$', '{0:.2f}'.format(df2[df2['baths']==avg_baths]['price'].mean()))
print("The average price for a", med_baths, "bathroom home is:
$', '{0:.2f}'.format(df2[df2['baths']==med_baths]['price'].mean()))
print("The average price for a", max_baths, "bathroom home is:
$', '{0:.2f}'.format(df2[df2['baths']==max_baths]['price'].mean()))
print()
input("Press <enter> to return to menu:\n")
display_baths()
elif choice == 2:
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Displaying Bathrooms Boxplot...")
    # creates figure displaying boxplot of bathroom data
    fig6 = plt.figure(6, figsize=(10,8))
    ax6 = fig6.add_subplot(111)
    ax6.set_xlabel("Figure 6: Number of Bathrooms")
    ax6.get_yaxis().tick_left()
    bp3 = ax6.boxplot(df2['baths'])
    # displays figure to user
    plt.show()
    # saves figure to file
    try:
        fig6.savefig('fig6-' + fname + '.png', bbox_inches='tight')
        print("Created file: fig6.png")
        logging.info("Figure 6 successfully created")
    except:
        print("Unable to save file: fig6.png")
        logging.debug("Figure 6 not saved")
    input("\nPress <enter> to return to menu:\n")
    display_baths()
elif choice == 3:
    try:
        os.system('cls')
    except:
        os.system('clear')

```

```

print("Displaying Bathrooms Frequency Histogram...")
# creates figure displaying histogram of bathroom frequency distribution
fig7 = plt.figure(7, figsize=(10,8))
ax7 = fig7.add_subplot(111)
ax7.set_xlabel("Figure 7: Number of Bathrooms")
ax7.set_ylabel("Frequency")
ht3 = ax7.hist(df2['baths'])
# displays plot to user
plt.show()
# saves figure to file
try:
    fig7.savefig('fig7-' + fname + '.png', bbox_inches='tight')
    print("Created file: fig7.png")
    logging.info("Figure 7 successfully created")
except:
    print("Unable to save file: fig7.png")
    logging.debug("Figure 7 not saved")
input("\nPress <enter> to return to menu:\n>")
display_baths()
elif choice == 4:
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Displaying Relationship Between Bathrooms and Price...")
    # create figure displaying relationship between number of bathrooms and price
    fig8 = plt.figure(8, figsize=(10,8))
    X = df2['baths']
    Y = df2['price']
    plt.scatter(X,Y)
    # displays figure to user
    plt.show()
    # saves figure to file
    try:
        fig8.savefig('fig8-' + fname + '.png', bbox_inches='tight')
        print("Created file: fig8.png")
        logging.info("Figure 8 successfully created")
    except:
        print("Unable to save file: fig8.png")
        logging.debug("Figure 8 not saved")
    input("\nPress <enter> to return to menu:\n>")
    display_baths()
elif choice == 5:
    display_options()
else:
    print("Invalid selection, please try again")
    display_baths()

```

```

# function to display property area data
def display_area():
    # clears console screen for Windows (cls) or Linux (clear)
    try:
        os.system('cls')
    except:
        os.system('clear')
    # Displays menu and prompts user for input
    print("Area Data\n")
    def menu(list,question):
        for item in list:
            print(1 + list.index(item), item)
        return input(question)
    while True:
        items = ["Descriptive Statistics","Display Area Boxplot", "Display Area Frequency
Histogram", "Display Relationship Between Area and Price", "Return to Main Menu"]
        choice = menu(items,"\nPlease make a selection from the list above:\n>")
        try:
            choice = int(choice)
            break
        except:
            print("Invalid selection, please try again\n")
            continue
    print("")
    # User makes a selection
    if choice == 1:
        try:
            os.system('cls')
        except:
            os.system('clear')
        # calculates and displays min, mean, median, and max areas (in sqft)
        min_area = int(df2['area'].min())
        avg_area = int(df2['area'].mean())
        med_area = int(df2['area'].median())
        max_area = int(df2['area'].max())
        print("Area:\n")
        print("The minimum area is:",min_area,"sq ft")
        print("The average area is:",avg_area,"sq ft")
        print("The median area is:",med_area,"sq ft")
        print("The maximum area is:",max_area, "sq ft")
        print()
        # displays average price for properties with min, mean, median, and max number of
bathrooms
        print("The average price for a", min_area, "square foot home is:
$", '{0:.2f}'.format(df2[df2['area']==min_area]['price'].mean()))
        print("The average price for a 2000 square foot home is:
$", '{0:.2f}'.format(df2[df2['area']==2000]['price'].mean()))

```

```

        print("The average price for a", max_area, "square foot home is:
$, '{0:.2f}'".format(df2[df2['area']==max_area]['price'].mean()))
    print()
    input("Press <enter> to return to menu:\n>")
    display_area()
elif choice == 2:
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Displaying Area Boxplot...")
    # creates figure displaying boxplot of area data
    fig9 = plt.figure(9, figsize=(10,8))
    ax9 = fig9.add_subplot(111)
    ax9.set_xlabel("Figure 9: Area (sq ft)")
    ax9.get_yaxis().tick_left()
    bp4 = ax9.boxplot(df2['area'])
    # displays figure to user
    plt.show()
    # saves figure to file
    try:
        fig9.savefig('fig9-' + fname + '.png', bbox_inches='tight')
        print("Created file: fig9.png")
        logging.info("Figure 9 successfully created")
    except:
        print("Unable to save file: fig9.png")
        logging.debug("Figure 9 not saved")
    input("\nPress <enter> to return to menu:\n>")
    display_area()
elif choice == 3:
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Displaying Area Frequency Histogram...")
    # creates figure displaying histogram of area frequency
    fig10 = plt.figure(10, figsize=(10,8))
    ax10 = fig10.add_subplot(111)
    ax10.set_xlabel("Figure 10: Area (sq ft)")
    ax10.set_ylabel("Frequency")
    ht4 = ax10.hist(df2['area'])
    # displays figure to user
    plt.show()
    # saves figure to file
    try:
        fig10.savefig('fig10-' + fname + '.png', bbox_inches='tight')
        print("Created file: fig10.png")
        logging.info("Figure 10 successfully created")

```

```

except:
    print("Unable to save file: fig10.png")
    logging.debug("Figure 10 not saved")
    input("\nPress <enter> to return to menu:\n>")
    display_area()
elif choice == 4:
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Displaying Relationship Between Area and Price...")
    # creates figure displaying relationship between area and price
    fig11 = plt.figure(11, figsize=(10,8))
    X = df2['area']
    Y = df2['price']
    plt.scatter(X,Y)
    # displays figure to user
    plt.show()
    # saves figure to file
    try:
        fig11.savefig('fig11-' + fname + '.png', bbox_inches='tight')
        print("Created file: fig11.png")
        logging.info("Figure 11 successfully created")
    except:
        print("Unable to save file: fig11.png")
        logging.debug("Figure 11 not saved")
        input("\nPress <enter> to return to menu:\n>")
        display_area()
elif choice == 5:
    display_options()
else:
    print("Invalid selection, please try again")
    display_area()

# function to create and display correlation matrix
def display_comatrix():
    try:
        os.system('cls')
    except:
        os.system('clear')

    # create and display correlation matrix text values
    print("Correlation Matrix:\n")
    print(df2.corr())

    # display heatmap correlation matrix
    f, ax = plt.subplots(figsize=(10,8))
    corr= df2.corr()

```



```

sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
cmap=sns.diverging_palette(220,10, as_cmap=True), square=True, ax=ax)
plt.show()
logging.info("Correlation matrix created")
f.savefig('fig12-' + fname + '.png', bbox_inches='tight')
# prompts user to return to main menu
input("\nPress <enter> to return to main menu:\n>")
display_options()

# function to create and display linear regression models
def display_linear():
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Display Linear Regression Model:\n")

def linear_all():
    try:
        os.system('cls')
    except:
        os.system('clear')
    # sets "price" as dependent variable
    target = pd.DataFrame(df2, columns=["price"])
    # sets "beds", "baths", and "area" as independent variables
    X = df2[["beds", "baths", "area"]]
    y = target["price"]
    # fits the model
    model = sm.OLS(y,X).fit()
    # displays model summary table
    print(model.summary())
    logging.info("Linear regression model with beds, baths, and area vs. price created")
    # return to menu
    input("\nPress <enter> to return to model menu:\n>")
    display_linear()

def linear_beds():
    try:
        os.system('cls')
    except:
        os.system('clear')
    # sets "price" as dependent variable
    target = pd.DataFrame(df2, columns=["price"])
    # sets "beds" as independent variables
    X = df2[["beds"]]
    y = target["price"]
    # fits the model
    model = sm.OLS(y,X).fit()

```

```

# displays model summary table
print(model.summary())
logging.info("Linear regression model with beds vs. price created")
# return to menu
input("\nPress <enter> to return to model menu:\n>")
display_linear()

def linear_baths():
    try:
        os.system('cls')
    except:
        os.system('clear')
    # sets "price" as dependent variable
    target = pd.DataFrame(df2, columns=["price"])
    # sets "baths" as independent variables
    X = df2[["baths"]]
    y = target["price"]
    # fits the model
    model = sm.OLS(y,X).fit()
    # displays model summary table
    print(model.summary())
    logging.info("Linear regression model with baths vs. price created")
    # return to menu
    input("\nPress <enter> to return to model menu:\n>")
    display_linear()

def linear_area():
    try:
        os.system('cls')
    except:
        os.system('clear')
    # sets "price" as dependent variable
    target = pd.DataFrame(df2, columns=["price"])
    # sets "area" as independent variables
    X = df2[["area"]]
    y = target["price"]
    # fits the model
    model = sm.OLS(y,X).fit()
    # displays model summary table
    print(model.summary())
    logging.info("Linear regression model with area vs. price created")
    # return to menu
    input("\nPress <enter> to return to model menu:\n>")
    display_linear()

def menu(list,question):
    for item in list:
        print(1 + list.index(item), item)

```

```

        return input(question)
    while True:
        items = ["Beds, Baths, and Area vs. Price", "Beds vs. Price", "Baths vs. Price", "Area
vs. Price", "Return to Main Menu" ]
        choice = menu(items, "\nPlease make a selection from the list above:\n>")
        try:
            choice = int(choice)
            break
        except:
            print("Invalid selection, please try again\n")
            continue
    print()

# User makes a selection
if choice == 1:
    linear_all()
elif choice == 2:
    linear_beds()
elif choice == 3:
    linear_baths()
elif choice == 4:
    linear_area()
elif choice == 5:
    display_options()
else:
    display_linear()

# function to predict housing price from user input (based on linear regression model)
def predict_linear():
    try:
        os.system('cls')
    except:
        os.system('clear')
    print("Home Price Prediction\n")
    print("Choose independent variable:")
    def menu(list, question):
        for item in list:
            print(1 + list.index(item), item)
        return input(question)
    while True:
        items = ["Beds", "Baths", "Area"]
        choice = menu(items, "\nPlease make a selection from the list above:\n>")
        try:
            choice = int(choice)
            break
        except:
            print("Invalid selection, please try again\n")
            continue

```

```

print()

# User makes a selection
if choice == 1:
    X = df2[["beds"]]
    pred_text = "Enter number of bedrooms:\n>"
elif choice == 2:
    X = df2[["baths"]]
    pred_text = "Enter number of bathrooms:\n>"
elif choice == 3:
    X = df2[["area"]]
    pred_text = "Enter area (sqft):\n>"
else:
    predict_linear()

target = pd.DataFrame(df2, columns=["price"])
y = target["price"]
model = sm.OLS(y,X).fit()
X2 = []

while True:
    pred_var = input(pred_text)
    try:
        X2.append(int(pred_var))
        pred2 = model.predict(X2)
        for i in pred2:
            print("\nEstimated Price: $"+"{:.2f}".format(i))
        break
    except:
        print("Invalid entry\n")
        continue
    again = input("\nMake another prediction? (y/n)\n>")
    if again.lower().startswith("y"):
        predict_linear()
    else:
        input("\nPress <enter> to return to main menu:\n>")
        display_options()

try:
    os.system('cls')
except:
    os.system('clear')

print("Real Estate Data Analysis")
print("Choose city:\n")
def menu(list,question):
    for item in list:
        print(1 + list.index(item), item)
    return input(question)

```

```

while True:
    items = ["Albany, NY", "Binghamton, NY", "Buffalo, NY", "Syracuse, NY"]
    choice = menu(items, "\nPlease make a selection from the list above:\n>")
    try:
        choice = int(choice)
        break
    except:
        print("Invalid selection, please try again\n")
        continue
print()

# User makes a selection
if choice == 1:
    fname = "albany"
elif choice == 2:
    fname = "binghamton"
elif choice == 3:
    fname = "buffalo"
elif choice == 4:
    fname = "syracuse"
else:
    menu()

try:
    os.system('cls')
except:
    os.system('clear')

# reads .csv file created by real_estate_scraper.py and creates Pandas dataframe
try:
    df = pd.read_csv("properties-" + fname + "-" + now + ".csv")
    # displays number of records retrieved from file
    print("Retrieved", len(df), "records from 'properties-" + fname + "-" + now + ".csv'")
    logging.info("Records successfully retrieved from file")
except:
    print("Unable to load files")
    logging.debug("Records not retrieved from file")
    quit()

# extracts data where title is "House For Sale"
df = df[df['title']=="House For Sale"]
# gets count of "House For Sale" records
house_count = len(df)
# displays count to user
print(house_count, "records contain data on houses for sale")
# sets column names, removes null values, and stores as new dataframe
df2 = df[['title', 'address', 'city', 'state', 'zip_code', 'beds', 'baths', 'area', 'price']].dropna()
# calculates number of null values based on difference between both dataframes

```

```

na_count = abs(len(df) - len(df2))
# displays number of null values to user
print(na_count,"records contain null values")
# prompts user to delete rows with null values or replace them with average values
del_null = input("Delete or Replace null values?\n>")
if del_null.lower().startswith('r'):
    # replaces null values with average values
    df2 = df.fillna(df.mean())
    # finds outliers (values with z-score >= 3) and excludes them from dataframe
    df2 = df2[(np.abs(stats.zscore(df2["price"])) < 2)]
    df2 = df2[(np.abs(stats.zscore(df2["beds"])) < 2)]
    df2 = df2[(np.abs(stats.zscore(df2["baths"])) < 2)]
    df2 = df2[(np.abs(stats.zscore(df2["area"])) < 2)]
    new_count = len(df2)
    print("\nReplacing", na_count,"null values with average values...")
    outlier_count = len(df) - len(df2)
    print("Removing",outlier_count, "records containing outliers...")
    print("..." + str(new_count) + " records remaining")
    logging.info("Records containing null values replaced")
    input("Press <enter> to continue:\n>")
    display_options()
elif del_null.lower().startswith('d'):
    # uses dataframe where null values have been deleted
    # deletes outliers based on z-score
    df2 = df2[(np.abs(stats.zscore(df2["price"])) < 3)]
    df2 = df2[(np.abs(stats.zscore(df2["beds"])) < 3)]
    df2 = df2[(np.abs(stats.zscore(df2["baths"])) < 3)]
    df2 = df2[(np.abs(stats.zscore(df2["area"])) < 3)]
    new_count = len(df) - na_count
    outlier_count = new_count - len(df2)
    print("\nRemoving",na_count,"records containing null values...")
    print("Removing",outlier_count,"records containing outliers...")
    print("..." + str(len(df2)) + " records remaining")
    logging.info("Records containing null values removed")
    input("Press <enter> to continue:\n>")
    display_options()
else:
    print("Invalid input")
    input("Press <enter> to continue:\n>")
    main_program()

```

```

#####
# Program execution begins here
#####
main_program()

```


real_estate_compare.py

```
'''
Name of file: real_estate_compare.py
Date created: September 25, 2018
Date last updated: September 25, 2018
Created by: James Ingram
Purpose of Program: Compare house pricing statistics between various cities in Upstate, NY based on
real estate data collected from zillow.com
'''

import datetime
import os
import logging
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# create date format for filename
now = datetime.datetime.today().strftime("%m%d%y")

# kills running analyzer program
import subprocess as sp
extProc1 = sp.Popen(['python', 'real_estate_scraper.py'])
extProc2 = sp.Popen(['python', 'real_estate_analyzer.py'])
sp.Popen.terminate(extProc1)
sp.Popen.terminate(extProc2)
# clears console (Windows or Linux)
try:
    os.system('cls')
except:
    os.system('clear')

# reads .csv file created by real_estate_scraper.py and creates Pandas dataframes
try:
    df1 = pd.read_csv("properties-albany-" + now + ".csv")
    df2 = pd.read_csv("properties-binghamton-" + now + ".csv")
    df3 = pd.read_csv("properties-buffalo-" + now + ".csv")
    df4 = pd.read_csv("properties-syracuse-" + now + ".csv")
except:
    print("Unable to load files")
    quit()

# extracts data where title is "House For Sale"
df1 = df1[df1['title']=="House For Sale"]
# sets column names, removes null values, and stores as new dataframe
df1 = df1[['title', 'address', 'city', 'state', 'zip_code', 'beds', 'baths', 'area', 'price',
'region']].dropna()
```

```

# removes outliers from dataframe
df1 = df1[(np.abs(stats.zscore(df1["price"]))) < 3]
df1 = df1[(np.abs(stats.zscore(df1["beds"]))) < 3]
df1 = df1[(np.abs(stats.zscore(df1["baths"]))) < 3]
df1 = df1[(np.abs(stats.zscore(df1["area"]))) < 3]
# extracts data where title is "House For Sale"
df2 = df2[df2['title']=="House For Sale"]
# sets column names, removes null values, and stores as new dataframe
df2 = df2[['title', 'address', 'city', 'state', 'zip_code', 'beds', 'baths', 'area', 'price',
'region']].dropna()
# removes outliers from dataframe
df2 = df2[(np.abs(stats.zscore(df2["price"]))) < 3]
df2 = df2[(np.abs(stats.zscore(df2["beds"]))) < 3]
df2 = df2[(np.abs(stats.zscore(df2["baths"]))) < 3]
df2 = df2[(np.abs(stats.zscore(df2["area"]))) < 3]
# extracts data where title is "House For Sale"
df3 = df3[df3['title']=="House For Sale"]
# sets column names, removes null values, and stores as new dataframe
df3 = df3[['title', 'address', 'city', 'state', 'zip_code', 'beds', 'baths', 'area', 'price',
'region']].dropna()
# removes outliers from dataframe
df3 = df3[(np.abs(stats.zscore(df3["price"]))) < 3]
df3 = df3[(np.abs(stats.zscore(df3["beds"]))) < 3]
df3 = df3[(np.abs(stats.zscore(df3["baths"]))) < 3]
df3 = df3[(np.abs(stats.zscore(df3["area"]))) < 3]
# extracts data where title is "House For Sale"
df4 = df4[df4['title']=="House For Sale"]
# sets column names, removes null values, and stores as new dataframe
df4 = df4[['title', 'address', 'city', 'state', 'zip_code', 'beds', 'baths', 'area', 'price',
'region']].dropna()
# removes outliers from dataframe
df4 = df4[(np.abs(stats.zscore(df4["price"]))) < 3]
df4 = df4[(np.abs(stats.zscore(df4["beds"]))) < 3]
df4 = df4[(np.abs(stats.zscore(df4["baths"]))) < 3]
df4 = df4[(np.abs(stats.zscore(df4["area"]))) < 3]
# concatenates dataframes
frames = [df1, df2, df3, df4]
result = pd.concat(frames)

# Displays menu and prompts user for input
print("Compare All City Real Estate Data\n")
def options():
    def menu(list,question):
        for item in list:
            print(1 + list.index(item), item)
        return input(question)
    while True:

```

```

        items = ["Display Pricing Data", "Display Bedroom Data", "Display Bathroom Data", "Display
Area Data", "Exit"]
        choice = menu(items, "\nPlease make a selection from the list above:\n>")
        try:
            choice = int(choice)
            break
        except:
            print("\nInvalid selection, please try again\n")
            continue
    print("")
    # user makes a selection
    if choice == 1:
        try:
            os.system('cls')
        except:
            os.system('clear')
        # creates figure displaying price data
        f, ax = plt.subplots(1, figsize=(10,8))
        sns.set_style("whitegrid")
        ax = sns.boxplot(x="region", y="price", data = result, palette="Set3")
        plt.show()
        f.savefig('price_compare.png', bbox_inches='tight')
        input("Press <enter> to return to menu:\n>")
        options()
    elif choice == 2:
        try:
            os.system('cls')
        except:
            os.system('clear')
        # creates figure displaying bedroom data
        f, ax = plt.subplots(1, figsize=(10,8))
        sns.set_style("whitegrid")
        ax = sns.boxplot(x="region", y="beds", data = result, palette="Set3")
        plt.show()
        f.savefig('beds_compare.png', bbox_inches='tight')
        input("Press <enter> to return to menu:\n>")
        options()
    elif choice == 3:
        try:
            os.system('cls')
        except:
            os.system('clear')
        # creates figure displaying bathroom data
        f, ax = plt.subplots(1, figsize=(10,8))
        sns.set_style("whitegrid")
        ax = sns.boxplot(x="region", y="baths", data = result, palette="Set3")
        plt.show()
        f.savefig('baths_compare.png', bbox_inches='tight')

```

```

        input("Press <enter> to return to menu:\n>")
        options()
    elif choice == 4:
        try:
            os.system('cls')
        except:
            os.system('clear')
        # creates figure displaying area data
        f, ax = plt.subplots(1, figsize=(10,8))
        sns.set_style("whitegrid")
        ax = sns.boxplot(x="region", y="area", data = result, palette="Set3")
        plt.show()
        f.savefig('area_compare.png', bbox_inches='tight')
        input("Press <enter> to return to menu:\n>")
        options()
    elif choice == 5:
        try:
            os.system('cls')
        except:
            os.system('clear')
        print("Goodbye!")
        quit()
    else:
        print("Invalid selection, please try again\n")
        options()
options()

```

Output Files

properties-fname-YYMMDD.csv

The image below shoes sample data from the .csv file generated from the `real_estate_scraper.py` program. This data is read into `real_estate_analyzer.py` and transformed into a pandas dataframe.

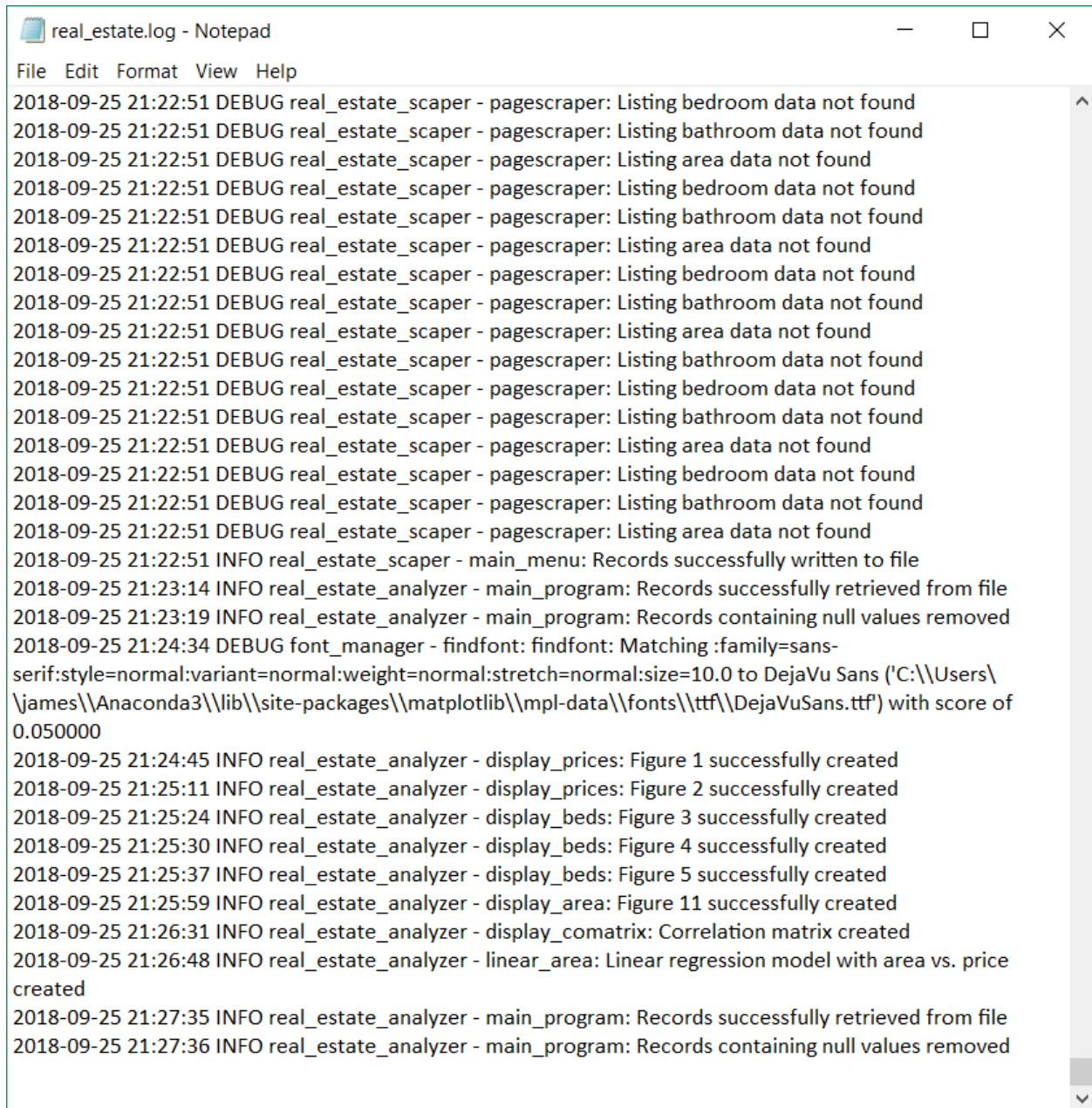
	A	B	C	D	E	F	G	H	I	J
1	title	address	city	state	zip_code	price	beds	baths	area	region
2	House For Sale	4187 Cheryl Dr	Binghamton	NY	13903	132900	3	1	1170	binghamton
3	Lot/Land For Sale	50 Hotchkiss St	Binghamton	NY	13903	25000				binghamton
4	For Sale by Owner	201 Murray St	Binghamton	NY	13905	65000	6	2	2248	binghamton
5	House For Sale	79 Mary St	Binghamton	NY	13903	64900	3	1	1152	binghamton
6	Apartment For Sale	157 Hawley St	Binghamton	NY	13901	34900				binghamton
7	House For Sale	32 Highland Ave	Binghamton	NY	13905	119900	4	2	1908	binghamton
8	House For Sale	54 Lincoln Ave	Binghamton	NY	13905	139000	3	2	1778	binghamton
9	House For Sale	121 Schubert St	Binghamton	NY	13905	140000	4	2	2340	binghamton
10	House For Sale	69 Highland Ave	Binghamton	NY	13905	65000	3	2	16088	binghamton
11	House For Sale	244 Bevier St	Binghamton	NY	13904	117000	2	2	1033	binghamton
12	Apartment For Sale	18 Morgan St	Binghamton	NY	13901	37500				binghamton
13	House For Sale	42 Spring Forest Ave	Binghamton	NY	13905	44516	3	10	1386	binghamton
14	Foreclosure	10 Bellevue Ave	Binghamton	NY	13905	142500				binghamton
15	Apartment For Sale	61 Lake Ave	Binghamton	NY	13905	72500				binghamton
16	House For Sale	87 Tompkins St	Binghamton	NY	13903	60000	2	1	1407	binghamton
17	Auction	20 Bigelow St	Binghamton	NY	13904		3	1	1624	binghamton
18	House For Sale	6 Schiller St	Binghamton	NY	13905	114900	4	2	1560	binghamton
19	House For Sale	166 Broad Ave	Binghamton	NY	13904	69900	3	1	1043	binghamton
20	House For Sale	4163 Brady Hill Rd	Binghamton	NY	13903	179000	3	2	1402	binghamton
21	Auction	5 Carhart Ave	Binghamton	NY	13905		3	2	1732	binghamton
22	House For Sale	133 Seminary Ave	Binghamton	NY	13905	124900	4	2	1690	binghamton
23	House For Sale	16 Bennett Ave	Binghamton	NY	13905	149900	3	2	1480	binghamton
24	Auction	57 Bevier St	Binghamton	NY	13901		3	2	2610	binghamton
25	House For Sale	4 Gordon Pl	Binghamton	NY	13905	89900	3	2	1512	binghamton
26	For Sale by Owner	10 Foland Rd	Binghamton	NY	13903	725000	5	5	5600	binghamton
27	House For Sale	2 Spurr Ave	Binghamton	NY	13903	130000	4	2	2017	binghamton
28	House For Sale	85 Kneeland Ave	Binghamton	NY	13905	107085	3	2	1422	binghamton
29	House For Sale	206 Old Pennsylvania Av	Binghamton	NY	13903	145000	3	2	1902	binghamton

Image files

All figures created by the program are saved as .png files and are labeled with the figure number and the city. For example The first figure for Binghamton, NY will be saved as “fig1-binghamton.png”.

real_estate.log

The program generates a log file which keeps track of program usage, file creation, errors, and more.



```

real_estate.log - Notepad
File Edit Format View Help
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bedroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bathroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing area data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bedroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bathroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing area data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bedroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bathroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing area data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bathroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bedroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bathroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing area data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bedroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing bathroom data not found
2018-09-25 21:22:51 DEBUG real_estate_scaper - pagescraper: Listing area data not found
2018-09-25 21:22:51 INFO real_estate_scaper - main_menu: Records successfully written to file
2018-09-25 21:23:14 INFO real_estate_analyzer - main_program: Records successfully retrieved from file
2018-09-25 21:23:19 INFO real_estate_analyzer - main_program: Records containing null values removed
2018-09-25 21:24:34 DEBUG font_manager - findfont: findfont: Matching :family=sans-serif:style=normal:variant=normal:weight=normal:stretch=normal:size=10.0 to DejaVu Sans ('C:\\Users\\james\\Anaconda3\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans.ttf') with score of 0.050000
2018-09-25 21:24:45 INFO real_estate_analyzer - display_prices: Figure 1 successfully created
2018-09-25 21:25:11 INFO real_estate_analyzer - display_prices: Figure 2 successfully created
2018-09-25 21:25:24 INFO real_estate_analyzer - display_beds: Figure 3 successfully created
2018-09-25 21:25:30 INFO real_estate_analyzer - display_beds: Figure 4 successfully created
2018-09-25 21:25:37 INFO real_estate_analyzer - display_beds: Figure 5 successfully created
2018-09-25 21:25:59 INFO real_estate_analyzer - display_area: Figure 11 successfully created
2018-09-25 21:26:31 INFO real_estate_analyzer - display_comatrix: Correlation matrix created
2018-09-25 21:26:48 INFO real_estate_analyzer - linear_area: Linear regression model with area vs. price created
2018-09-25 21:27:35 INFO real_estate_analyzer - main_program: Records successfully retrieved from file
2018-09-25 21:27:36 INFO real_estate_analyzer - main_program: Records containing null values removed
  
```

Syracuse University
DataScience@Syracuse