Analysis of the Greater Saint Louis Housing Market

Paul M. Washburn
CSCI E-83 Fundamentals of Data Science

Submitted on May 12, 2017

Abstract

The Saint Louis, Missouri community is characterized by its schools, restaurants and municipalities. With ninety distinct municipalities, understanding the dominant factors at play in the housing market is of considerable economic interest. This inquiry utilizes several sources of data in an attempt to gain understanding of the mechanics influencing listing prices and price per square foot in the Saint Louis market, and translate this understanding into predictive algorithms capable of identifying opportunities in the market outside of the limited sample considered. Regressing listing price on various features yielded a strong multiple regression model for predicting the price of a given property, given certain attributes. To investigate PricePerSqFt a conditional inference tree was employed, yielding interesting insights that can be elaborated upon in future research.

Context & Objectives

This inquiry attempts to gain insight into the market pricing mechanics at work in the Saint Louis, Missouri housing market. The goal if this inquiry is to establish a base understanding of current market dynamics in order to guide future research into the topic, with the ultimate objective of deriving predictive models for both Price and PricePerSqFt. Data on residential properties that were for sale during the Spring of 2017 was gathered & combined with data on various features that describe the surrounding area. These models may then be used in production to identify opportunities and to help realtors advise their clients using scientifically valid, data-backed insights. This is the first investigation into the matter, and will be followed by several subsequent analyses to elaborate on the findings.

Data Acquisition

Home Listings

The home listing data used in this investigation was acquired, in its raw form, by simply searching for 15 pre-defined municipalities. Two of the most trusted real estate websites were used to search for listings in these cities; realtor.com and zillow.com. To ensure compliance with these companys' data access policies, the data was not scraped directly from their websites, but rather:

- Each municipality was searched for one-by-one in attempt at deep municipal exploration
- Each webpage in the search was saved via "View Source/CTRL+U" as an HTML document
- The HTML documents saved locally were then processed using BeautifulSoup

After manually obtaining over 50 HTML webpages from both sites, both sources were manually inspected for consistency as well as to identify the HTML tags that the website developers had used to render information on listing addresses and postal codes. The reason that only address and postal code were obtained using this method was to ensure data integrity. The raw webpages were scraped using Python's BeautifulSoup module, a tool which lowers considerably the barrier for engineers to scrape semantics from websites. The Python code below outlines the process of using BeautifulSoup to acquire data from the raw HTML files.

```
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
```

```
import bs4

## Files saved as "realtor_N.html"
txt = 'C:/Users/paulm/Desktop/Harvard/Realty Data/realtor_'
URL1,URL2,URL4,URL4 = [txt+str(i)+'.html' for i in range(1, 5)]
urls = [URL1,URL2,URL3,URL4]

## Declare empty DF to gather data
LISTING_DF = pd.DataFrame()

## Enumerate data to acquire and loop through files
attributes = ['listing-street-address','listing-city','listing-region','listing-postal']
namez = ['Address','City','State','Zip']
for i, l in enumerate(urls):
    soup = BeautifulSoup(open(l))
    ROWS = soup.find_all('span',{'class':attributes})
    new_listing = {}
    for j, row in enumerate(ROWS):
        new_listing[j%4+1] = row.text
        df = pd.DataFrame.from_dict(new_listing, 'index')
        if len(df) == 4:
            df = df.transpose()
            df.rename(columns={1:'Address',2:'City',3:'State',4:'Zip'}, inplace=True)
            ix = df.Address.astype(str)#+'_'+df.City.astype(str)+'_'+df.Zip.astype(str)
            df.set_index(ix, inplace=True)
            LISTING_DF = LISTING_DF.append(df)
        else:
            del df
    LISTING_DF.drop_duplicates(subset='Address', inplace=True)
    LISTING_DF.dropna(inplace=True)
```

As address/zip code pairs were obtained, Zillow's API was employed in an iterative process until a sufficient number of quality observations were obtained. The Python code below provides an example of one round of such API calls.

```
import re
from pyzillow.pyzillow import ZillowWrapper, GetDeepSearchResults

ZILLOW_ID = 'MY_ZID'
ZILLOW_API_KEY = 'MY_ZKEY'
zillow_data = ZillowWrapper(ZILLOW_API_KEY)

## Define function to acquire deep serach results from Zillow API
def get_zillow_price(address, zipcode, zillow_data=zillow_data):
    deep_search_response = zillow_data.get_deep_search_results(address, zipcode)
    result = GetDeepSearchResults(deep_search_response)
    return result

## For each row in listings, get data from Zillow
prices = []
for ix, row in LISTING_DF.iterrows():
    try:
        new_data = get_zillow_price(str(row['Address']), str(row['Zip']))
        print(str(row['Address']), str(row['Zip']))
```

```python
        LISTING_DF.loc[ix, 'ZillowID'] = new_data.zillow_id
        LISTING_DF.loc[ix, 'Price'] = new_data.last_sold_price
        LISTING_DF.loc[ix, 'Latitude'] = new_data.latitude
        LISTING_DF.loc[ix, 'Longitude'] = new_data.longitude
        LISTING_DF.loc[ix, 'TaxValue'] = new_data.tax_value
        LISTING_DF.loc[ix, 'PropertySize'] = new_data.property_size
        LISTING_DF.loc[ix, 'YearBuilt'] = new_data.year_built
        LISTING_DF.loc[ix, 'HomeSize'] = new_data.home_size
        LISTING_DF.loc[ix, 'Bedrooms'] = new_data.bedrooms
        LISTING_DF.loc[ix, 'Bathrooms'] = new_data.bathrooms
        LISTING_DF.loc[ix, 'LastSold'] = new_data.last_sold_date
        print(LISTING_DF.loc[ix])
    except TypeError:
        new_data = get_zillow_price(str(row['Address']), '63119')
        print(str(row['Address']), str(row['Zip']))
        LISTING_DF.loc[ix, 'ZillowID'] = new_data.zillow_id
        LISTING_DF.loc[ix, 'Price'] = new_data.last_sold_price
        LISTING_DF.loc[ix, 'Latitude'] = new_data.latitude
        LISTING_DF.loc[ix, 'Longitude'] = new_data.longitude
        LISTING_DF.loc[ix, 'TaxValue'] = new_data.tax_value
        LISTING_DF.loc[ix, 'PropertySize'] = new_data.property_size
        LISTING_DF.loc[ix, 'YearBuilt'] = new_data.year_built
        LISTING_DF.loc[ix, 'HomeSize'] = new_data.home_size
        LISTING_DF.loc[ix, 'Bedrooms'] = new_data.bedrooms
        LISTING_DF.loc[ix, 'Bathrooms'] = new_data.bathrooms
        LISTING_DF.loc[ix, 'LastSold'] = new_data.last_sold_date
        print(LISTING_DF.loc[ix])
    except:
        # A few bad apples passed through
        pass
```

Alcoholic Beverage Retailers, Restaurants & Bars

The Saint Louis community supports many brewers and liquor producers, and is also famous for its food, arts and social scene. To capture a proxy for this enthusiasm for social interaction, data was acquired on all retail liquor licenses. The dataset was sourced via the Missouri Alcohol Tobacco & Firearms and accessed via the data.mo.gov data portal. The data was pared down by selecting only the cities that are known to be in our dataset for home listings. Once acquired, the address, street, city, state and postal code were concatenated together in order to query ggmap's library for their geocodes. Geocodes will be necessary in computing Haversine distance for homes in the dataset, and are also necessary for visualizing both datasets on a map.

The R statistical computing language is used for the remaineder of this investigation.

```r
## The following code was used to clean and get geocodes for the alcohol
## license dataset. This code chunk does not run to avoid calling the ggmap
## API unnecessarily
library(ggmap)

## Read in alcohol licenses
new_path = "E:/Harvard Fundamentals of Data Science/Missouri_Primary_Alcohol_Licenses.csv"
alc_df = read.csv(new_path, header = TRUE)

## Reduce size of dataset by selecting cities/counties
```

```
CITIES_WE_HAVE = c("BALLWIN", "BRENTWOOD", "CHESTERFIELD", "CLAYTON", "CRESTWOOD",
    "CREVE COEUR", "ELLISVILLE", "FLETCHER", "GLENDALE", "KIRKWOOD", "LADUE",
    "MANCHESTER", "MARYLAND HEIGHTS", "OAKLAND", "RICHMOND HEIGHTS", "ROCK HILL",
    "SAINT LOUIS", "SHREWSBURY", "SUNSET HILLS", "TOWN AND COUNTRY", "UNIVERSITY CITY",
    "WARSON WOODS", "WEBSTER GROVES", "WILDWOOD", "WINCHESTER")
alc_df = alc_df[alc_df$CITY %in% CITIES_WE_HAVE, ]

msg = paste0("There are ", dim(alc_df)[1], " wine/beer/spirits retailers in the area. Generating geocode
print(msg)

## Put addresses into a format which ggmap can use to derive a geocode
addresses = paste0(alc_df$STREET.NUMBER, " ", alc_df$STREET, " ", alc_df$CITY,
    " ", alc_df$STATE, " ", alc_df$ZIPCODE, " ", "USA")

## Get geocodes from ggmap API -- less than their API max (2500)
gcodes = geocode(addresses, output = "latlon", source = "google")
alc_df$LON = unlist(gcodes[1])
alc_df$LAT = unlist(gcodes[2])
head(alc_df)  ## NOW PRINT TO FILE WITH COMPLETE DATA

## Write to csv for later use and redundancy
write.csv(alc_df, "E:/Harvard Fundamentals of Data Science/GEOCODED_STLCOUNTY_Primary_Alcohol_Licenses.c
write.csv(alc_df, "C:/Users/paulm/Desktop/Harvard/Data/GEOCODED_STLCOUNTY_Primary_Alcohol_Licenses.csv")
```

The pared down, tidy dataset for the Saint Louis area's alcohol license holders appears below. Now that geocodes have been accessed we can explore what impact, if any, these purveyors of spirits have on their surrounding housing markets.

```
## Read in alcohol licenses
new_path = "E:/Harvard Fundamentals of Data Science/GEOCODED_STLCOUNTY_Primary_Alcohol_Licenses.csv"
# new_path =
# 'C:/Users/paulm/Desktop/Harvard/Data/GEOCODED_STLCOUNTY_Primary_Alcohol_Licenses.csv'
alc_df = read.csv(new_path, header = TRUE)
rownames(alc_df) = alc_df$PRIMARY.LICENSE
alc_df[, c("PRIMARY.LICENSEE", "X")] = NULL
head(alc_df[, c("DBANAME", "CITY", "LAT", "LON", "PRIMARY.TYPE")])
```

	DBANAME	CITY	LAT	LON	PRIMARY.TY
8506	CECIL WHITTAKERS PIZZERIA	ELLISVILLE	38.59237	-90.55979	5BDW
9829	SPIRO'S	CHESTERFIELD	38.67815	-90.49920	RBD
9856	BELLERIVE COUNTRY CLUB	CREVE COEUR	38.65566	-90.48210	RBD
9926	CICERO'S	UNIVERSITY CITY	38.65647	-90.30834	RBD
9928	BALLWIN RECREATIONAL COMPLEX	BALLWIN	38.60530	-90.54107	RBD
9956	CREVE COEUR MEMORIAL BUILDING ASSN.	CREVE COEUR	38.67198	-90.44429	RBD

```
dim(alc_df)
```

```
## [1] 814  15
```

In the interest of feature engineering, the records in the liquor license dataset are aggregated up to the City level by a simple "count unique" function. This aggregated data is saved to a new namespace in order to preserve the individual locations for future geospatial analysis. It should be noted that no attempt was made to investigate the individual liquor license types with any type of rigor, and thus their hetergoneity will not be captured in any forthcoming models.

```
len_unique = function(x) length(unique(x))
alc_agg = aggregate(DBANAME ~ CITY, data = alc_df, FUN = len_unique)
names(alc_agg) = c("City", "RetailLiquorLicenses")
head(alc_agg)
```

City	RetailLiquorLicenses
BALLWIN	44
BRENTWOOD	25
CHESTERFIELD	140
CLAYTON	70
CRESTWOOD	24
CREVE COEUR	58

Saint Louis County Municipalities

Data on Saint Louis County's various municipalities characteristics was acquired from the Saint Louis County Municpality Wikipedia page via simple copy/paste into a .csv document. The datset contains information on Population, Total Area (square miles), and population density of each municipality in the listings dataset. The following code segment does not run to minimize the calls to Google's API for geocode generation; the pre-geocoded dataset is silently read in for later use.

```
## Do not re run this code
munis = read.csv("E:/Harvard Fundamentals of Data Science/STL County Municipality Data.csv",
    header = TRUE)
## Get cities in muni data
cities_togcode = paste0(munis$Municipality, " MO USA")

## Get geocodes from ggmap API -- less than their API max (2500)
gcodes = geocode(cities_togcode, output = "latlon", source = "google")
munis$LON = unlist(gcodes[1])
munis$LAT = unlist(gcodes[2])

## Coerce new data to numeric
munis$Population = as.numeric(gsub(",", "", munis$Population))
munis$Population.Density.sq.mi = as.numeric(gsub(",", "", munis$Population.Density.sq.mi))

## Convert name to uppercase for future merges
munis$Municipality = toupper(munis$Municipality)
```

Municipality	Population	Total.Area..mi2.	Population.Density.sq.mi	LON	LAT
AFFTON	20535	4.58	4480.1	-90.33317	38.55061
BALLWIN	31283	8.95	3494.6	-90.54623	38.59505
BELLA VILLA	687	0.10	5468.3	-90.28607	38.54310
BELLEFONTAINE NEIGHBORS	11271	4.40	2573.2	-90.22650	38.74033
BELLERIVE	254	0.40	713.0	-90.31400	38.71144
BEL-NOR	1598	0.60	2555.5	-90.31678	38.70200

Data Munging, Joining & Feature Engineering

Data Cleaning

The complete home listing dataset was cleaned by eliminating duplicate addresses, setting Address as the index, and purging observations that were missing values for Price, HomeSize, or YearBuilt. Originally the data that was filtered through the Zillow API contained 1777 observations; after the afforementioned process only 813 observations remain. This stringent purging of data reflects a bias towards reality, shying somewhat away from methods of data imputation.

```
# new_path = 'C:/Users/paulm/Desktop/Harvard/COMBINED FINAL DATASET.csv'
new_path = "E:/Harvard Fundamentals of Data Science/COMBINED FINAL DATASET.csv"
df = read.csv(new_path, header = TRUE)
obs_before = dim(df)[1]

## Remove poor quality data
out_1 = is.na(df$Price) == F
out_2 = is.na(df$HomeSize) == F
out_3 = is.na(df$YearBuilt) == F
df = df[out_1 & out_2 & out_3, ]
obs_after = dim(df)[1]
msg = paste0("Before there were ", obs_before, " records ... ", "After data cleanup there is ",
    obs_after, " records.")
print(msg)
```

```
## [1] "Before there were 1777 records ... After data cleanup there is 813 records."
```

```
## Drop duplicates & Set index to minimize change of duplicates
df = df[duplicated(df$Address) == FALSE, ]
rownames(df) = df$Address
head(df[, c("ZillowID", "Price", "Bedrooms", "Bathrooms", "LastSold", "HomeSize")])
```

	ZillowID	Price	Bedrooms	Bathrooms	LastSold	HomeSize
8925 Lawn Ave	2772452	247000	2	2	9/26/2012	1566
1927 Parkridge Ave	2771585	329000	3	2	6/17/2008	2255
1620 Redbird Cv	2772285	148000	2	1	6/19/2006	945
1251 Strassner Dr Apt 2303	87733595	242235	2	2	3/5/2008	1018
9082 W Swan Cir	2771807	158500	3	1	9/5/2006	1080
8627 Henrietta Ave	2785096	225000	2	2	7/8/2015	2016

Joining Listings with Municipal Statistics

First we need to understand which cities in the residential real estate dataeset are spelled the same way (and present) in the municipality statistics dataset. Below, the City column in the listing data is converted to an uppercase character string to match the municipal statistics dataset's Municipality column. Then a list of all unique cities contained in the listings data is cross-referenced with the municipal statistics data to ensure that each city in the listings dataset (a) exists in the municipal dataset, and (b) is spelled the same. It is demonstrated below that both Saint Louis and Fletcher are not present in the municipal statistics dataset. Knowing this, the data is merged as a left join to preserve the 813 observations in the listing data. It should be noted that alcohol license information for the cities of Saint Louis and Fletcher will not be considered due to their absence. A truncated header of the dataset is shown below.

```
## Convert listing cities to uppercase to match
df$City = toupper(df$City)
```

```r
listcity_in_municity = c(unique(df$City) %in% unique(munis$Municipality))
names(listcity_in_municity) = unique(df$City)

## ID which cities are not present in dataset
cities_absent_munidata = listcity_in_municity[listcity_in_municity == FALSE]
print(cities_absent_munidata)
```

```
## SAINT LOUIS    FLETCHER
##       FALSE       FALSE
```

```r
## Merge listings data with muni stats
df = merge(df, munis, by.x = "City", by.y = "Municipality", all.x = TRUE)

## Coerce new data to numeric
df$Population = as.numeric(gsub(",", "", df$Population))
df$Population.Density.sq.mi = as.numeric(gsub(",", "", df$Population.Density.sq.mi))

head(df[, c("ZillowID", "Price", "Bedrooms", "Population", "Total.Area..mi2.")])
```

ZillowID	Price	Bedrooms	Population	Total.Area..mi2.
2781273	275176	4	31283	8.95
55187544	568454	4	31283	8.95
2866646	95000	2	31283	8.95
2791381	250000	3	31283	8.95
2791301	154900	3	31283	8.95
2845274	202900	4	31283	8.95

```r
dim(df)
```

```
## [1] 813  21
```

Joining Listings with Retail Liquor License Summary

As above, the two City attributes are compared in order to ensure that data can be merged without misleading consumers of the analysis. It is shown that only cities in the 25 represented in the home listings dataset are not present in the retail alcohol license dataset. Encouragingly consistent with the previous merge, the datasets are merged on the City feature using a left merge. Following these operations, all columns that exist in the dataset are shown below.

```r
## Create list of unique cities in each source
list_cities = unique(df$City)
alc_cities = unique(alc_agg$City)

## Check that listing cities are in the alcohol data cities
listcity_in_alccity = c(list_cities %in% alc_cities)
names(listcity_in_alccity) = unique(df$City)

## ID which cities are not present in dataset
cities_absent_alcdata = listcity_in_alccity[listcity_in_alccity == FALSE]

print(paste0("Number of cities in listings data  ", length(listcity_in_alccity),
    " out of which ", length(cities_absent_alcdata), " are not in the liquor data."))
```

```
## [1] "Number of cities in listings data  25 out of which 2 are not in the liquor data."
```

```r
print(cities_absent_alcdata)
```

```
##    FLETCHER SAINT LOUIS
##       FALSE       FALSE
```

```r
## Merge datasets on City
df = merge(df, alc_agg, by = "City", all.x = TRUE)
print(head(df))
```

```
##      City                  Address                Address.1 State   Zip
## 1 BALLWIN     227 Oakbriar Farm Dr     227 Oakbriar Farm Dr    MO 63011
## 2 BALLWIN          968 Burgundy Ln          968 Burgundy Ln    MO 63021
## 3 BALLWIN   1584 Forest Springs Dr   1584 Forest Springs Dr    MO 63021
## 4 BALLWIN 16736 Highland Summit Dr 16736 Highland Summit Dr    MO 63021
## 5 BALLWIN     1039 Kiefer Ridge Dr     1039 Kiefer Ridge Dr    MO 63011
## 6 BALLWIN   725 Ridgeside Dr Apt H   725 Ridgeside Dr Apt H    MO 63011
##   ZillowID  Price Latitude Longitude TaxValue PropertySize YearBuilt
## 1  2829023 190000 38.58326 -90.57407   167100         6534      1987
## 2  2806547 294500 38.60405 -90.51114   359900        13068      1990
## 3 55187414 148000 38.55525 -90.52362   155700         3920      2000
## 4  2781273 275176 38.62333 -90.62510   365700        16552      1996
## 5 55187544 568454 38.55954 -90.56514   442600        14810      2004
## 6  2866646  95000 38.56350 -90.52083    77000         3964      1987
##   HomeSize Bedrooms Bathrooms LastSold Population Total.Area..mi2.
## 1     1562        3         2    42313      31283             8.95
## 2     2453        5         3    40596      31283             8.95
## 3     1378        2         4    42311      31283             8.95
## 4     3042        4         4    35156      31283             8.95
## 5     4122        4         4    38435      31283             8.95
## 6      868        2         2    39234      31283             8.95
##   Population.Density.sq.mi       LON      LAT RetailLiquorLicenses
## 1                   3494.6 -90.54623 38.59505                   44
## 2                   3494.6 -90.54623 38.59505                   44
## 3                   3494.6 -90.54623 38.59505                   44
## 4                   3494.6 -90.54623 38.59505                   44
## 5                   3494.6 -90.54623 38.59505                   44
## 6                   3494.6 -90.54623 38.59505                   44
```

Feature Extraction & Normalization

Now that we've enriched the listing data with municipal characteristics, we may now begin to extract basic features. This includes changing dates from strings to datetime objects to support mathematical operations, deriving ratios (e.g. PricePerSqFt), and preparing geographic information for use in various geospatial visualization packages.

```r
## Extract basic features
df$LastSold = as.Date(df$LastSold, "%m/%d/%Y")
df$PricePerSqFt = df$Price/df$HomeSize
df$YearsOld = as.numeric(format(Sys.Date(), "%Y")) - df$YearBuilt
df$YearsSinceLastSale = round(as.numeric(Sys.Date() - df$LastSold)/365, 2)
df$TaxValueToListPrice = df$TaxValue/df$Price
df$PricePerLotSqFt = df$Price/df$PropertySize
df$Coordinates = paste0("(", df$Latitude, ",", df$Longitude, ")")
```

```r
df$HomeSizeSquared = df$HomeSize^2
df$YearsSinceLastSale_5OrLess = df$YearsSinceLastSale <= 5
```

As an exercise in model-building creativity, the top and bottom 5% of a range of features were transformed into categorical variables. The columns are added "inplace" by looping through various numeric features. The column names are printed for verification that they were created.

```r
## Extract categories from continuous variables
cols_to_mark = list("Price", "HomeSize", "YearBuilt", "Population", "RetailLiquorLicenses",
    "PricePerSqFt")
for (vector in cols_to_mark) {
    cutoffs = quantile(df[, vector], c(0.05, 0.95), na.rm = TRUE)
    colname_low = paste0("Bottom5Percentile_", as.character(vector))
    colname_hi = paste0("Top5Percentile_", as.character(vector))
    ## Mark top and bottom 5%-iles
    df[, colname_low] = df[, vector] <= cutoffs[1]
    df[, colname_hi] = df[, vector] >= cutoffs[2]
}

## Print newest columns
head(df[, c(tail(seq_len(ncol(df))))], 2)
```

Bottom5Percentile_Population	Top5Percentile_Population	Bottom5Percentile_RetailLiquorLicenses	Top5Percentile_
FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE

By running simple linear regression on the number of bedrooms, treating the Bedrooms column as a factor rather than a number, it aligns with expectations that homes with 5, 6 or 7+ bedrooms command higher prices than others. To isolate their signal, dummy variables were derived for homes greater than or equal to 5 bedrooms, as well as for each of the significant groups in the Bedroom set. It is interesting to note that the effect between 5 bedroom and 6 bedroom listings is roughly equal, yet homes with 7 bedrooms seem to pack a bigger punch. This may reflect Saint Louis' strong Catholic tradition of large families, or may reflect the considerable income inequality in the Saint Louis region. Further investigation would be necessary to validate these hypotheses, especially within the context of the housing market.

```r
## Explore isolated impact of bedrooms on price
mod = lm(df$Price ~ factor(df$Bedrooms))
print(summary(mod))
```

```
##
## Call:
## lm(formula = df$Price ~ factor(df$Bedrooms))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -736000 -113984  -38484   53627 1872318
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             198656     101177   1.963 0.049939 *
## factor(df$Bedrooms)1    -66097     143086  -0.462 0.644253
## factor(df$Bedrooms)2    -52731     103045  -0.512 0.608984
## factor(df$Bedrooms)3     10329     102168   0.101 0.919498
## factor(df$Bedrooms)4    200022     102523   1.951 0.051405 .
```

```
## factor(df$Bedrooms)5   354027     104356   3.392 0.000726 ***
## factor(df$Bedrooms)6   357718     119715   2.988 0.002893 **
## factor(df$Bedrooms)7  1153345     202355   5.700 1.68e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 247800 on 805 degrees of freedom
## Multiple R-squared:  0.2664, Adjusted R-squared:   0.26
## F-statistic: 41.76 on 7 and 805 DF,  p-value: < 2.2e-16
```

```
## These features were extracted using what was learned in exploratory
## analysis
df$BedroomsFactor = factor(df$Bedrooms, levels = c(1, 2, 3, 4, 5, 6, 7))
df$FivePlusBedrooms = ifelse(df$Bedrooms >= 5, TRUE, FALSE)
df$FiveBedrooms = ifelse(df$Bedrooms == 5, TRUE, FALSE)
df$SixBedrooms = ifelse(df$Bedrooms == 6, TRUE, FALSE)
df$SevenBedrooms = ifelse(df$Bedrooms == 7, TRUE, FALSE)
df$ThreeOrMoreBaths = ifelse(df$Bathrooms >= 3, TRUE, FALSE)
```

The simple regression method of feature detection that was employed above on Bedrooms is also utilized to identify which municipalities currently command higher residential housing prices. This method indicates that the cities of Ladue, Richmond Heights, and Town and Country command higher prices. Maryland Heights lags behind, consistent with the fact that Maryland Heights is a highly industrial area compared with other, more residential municipalities in the area. This limited dataset is not sufficient to verify or deny any claims regarding any one municipality's residential real estate market. Information on other factors, as well as a larger sample of panel data would be a natural extension of this work.

```
## Explore isolated impact of cities on price
mod = lm(df$Price ~ factor(df$City))
print(summary(mod))
```

```
##
## Call:
## lm(formula = df$Price ~ factor(df$City))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -721563 -132622  -46499   63365 1663865
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       246623      21881  11.271  < 2e-16 ***
## factor(df$City)BRENTWOOD          -75573      70660  -1.070  0.28516
## factor(df$City)CHESTERFIELD         8378     146780   0.057  0.95450
## factor(df$City)CLAYTON            176547      97503   1.811  0.07057 .
## factor(df$City)CRESTWOOD          -66885      91533  -0.731  0.46517
## factor(df$City)CREVE COEUR        205116      50846   4.034 6.02e-05 ***
## factor(df$City)ELLISVILLE          64473      53098   1.214  0.22502
## factor(df$City)FLETCHER          -101123     252340  -0.401  0.68872
## factor(df$City)GLENDALE            91378      97503   0.937  0.34895
## factor(df$City)KIRKWOOD            -9370      47795  -0.196  0.84463
## factor(df$City)LADUE              569941      41746  13.653  < 2e-16 ***
## factor(df$City)MANCHESTER         -67343     114534  -0.588  0.55672
## factor(df$City)MARYLAND HEIGHTS  -115024      44141  -2.606  0.00934 **
## factor(df$City)OAKLAND            -10622     252340  -0.042  0.96643
## factor(df$City)RICHMOND HEIGHTS   177512      64778   2.740  0.00628 **
```

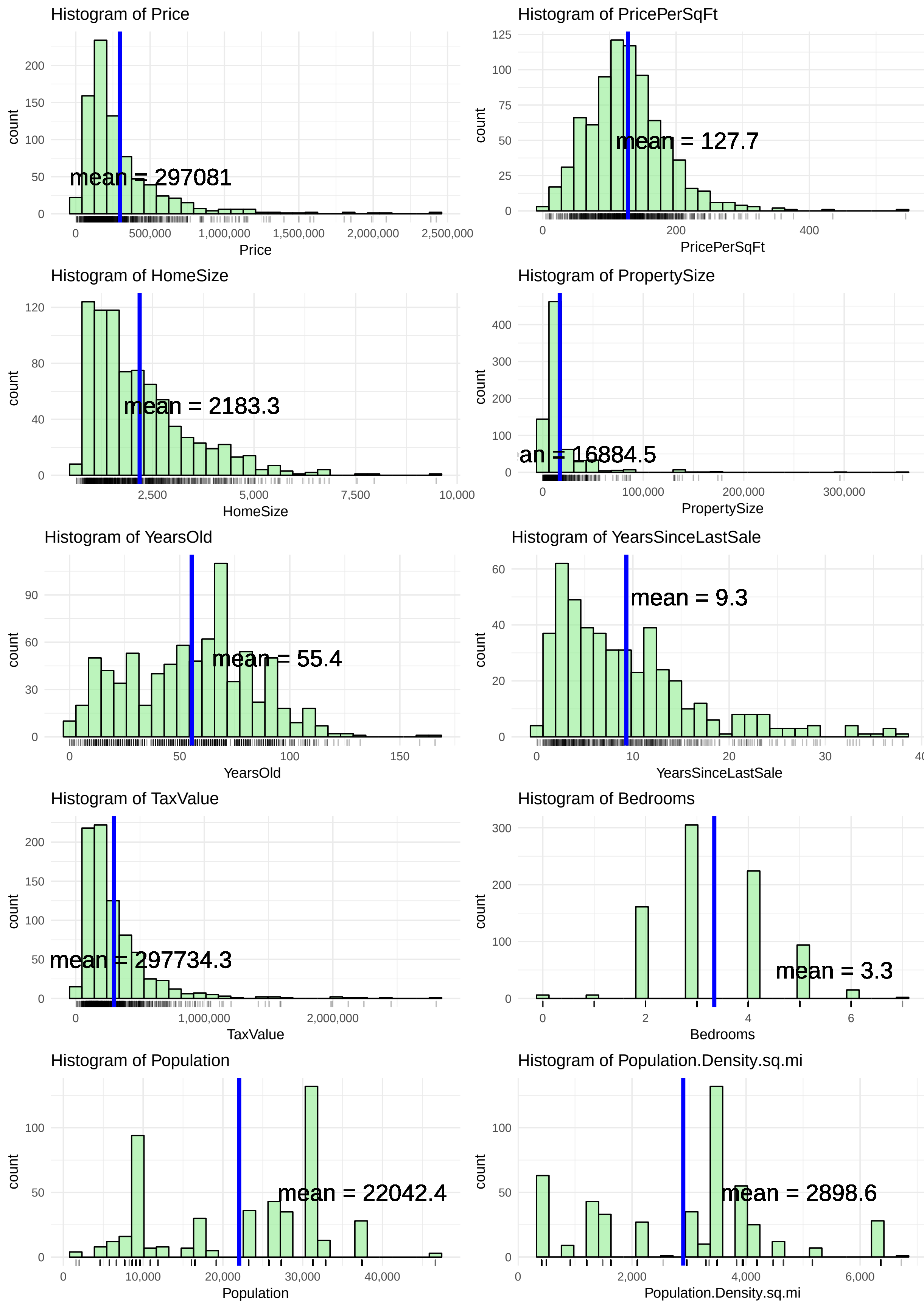
```
## factor(df$City)ROCK HILL          -55085      91533  -0.602  0.54748
## factor(df$City)SAINT LOUIS          16127      25878   0.623  0.53335
## factor(df$City)SHREWSBURY           37034     114534   0.323  0.74652
## factor(df$City)SUNSET HILLS        -89623     179101  -0.500  0.61693
## factor(df$City)TOWN AND COUNTRY   228092      97503   2.339  0.01957 *
## factor(df$City)UNIVERSITY CITY     -6808      52305  -0.130  0.89647
## factor(df$City)WARSON WOODS       -46673     179101  -0.261  0.79447
## factor(df$City)WEBSTER GROVES      18818      47268   0.398  0.69066
## factor(df$City)WILDWOOD            13230      73076   0.181  0.85638
## factor(df$City)WINCHESTER         -36623     252340  -0.145  0.88464
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 251400 on 788 degrees of freedom
## Multiple R-squared:  0.2611, Adjusted R-squared:  0.2386
## F-statistic:  11.6 on 24 and 788 DF,  p-value: < 2.2e-16
```

```r
df$IsLadue = ifelse(df$City == "LADUE", TRUE, FALSE)
df$IsTownAndCountry = ifelse(df$City == "TOWN AND COUNTRY", TRUE, FALSE)
df$IsRichmondHeights = ifelse(df$City == "RICHMOND HEIGHTS", TRUE, FALSE)
df$IsMarylandHeights = ifelse(df$City == "MARYLAND HEIGHTS", TRUE, FALSE)
```

Before normalizing the features, it is imperative to first understand something of the nature of each feature's underlying statistical distribution. These visualizations will help inform the method of normalization and data pre-processing that will be necessary for maximizing the probability of predictive success.

```r
library(gridExtra)

## Declare histogram function for re-use
check_histogram = function(col) {
    xlab = col
    COL_MU = mean(df[, col], na.rm = T)
    mu_lab = paste0("mean = ", as.character(round(COL_MU, 1)))
    ggplot(data = df, aes(x = df[, col])) + geom_histogram(fill = "lightgreen",
        colour = "black", alpha = 0.6) + labs(title = paste0("Histogram of ",
        col), x = col) + geom_rug(alpha = 0.25) + geom_vline(xintercept = COL_MU,
        colour = "blue", size = 1.5) + geom_text(aes(x = COL_MU + (COL_MU *
        0.7), y = 50, label = mu_lab), size = 6.4) + theme_minimal() + scale_x_continuous(labels = comma
        scale_y_continuous(labels = comma)
}

## Visualize histograms
hist1 = check_histogram("Price")
hist2 = check_histogram("PricePerSqFt")
hist3 = check_histogram("HomeSize")
hist4 = check_histogram("PropertySize")
hist5 = check_histogram("YearsOld")
hist6 = check_histogram("YearsSinceLastSale")
hist7 = check_histogram("TaxValue")
hist8 = check_histogram("Bedrooms")
hist9 = check_histogram("Population")
hist10 = check_histogram("Population.Density.sq.mi")


grid.arrange(hist1, hist2, hist3, hist4, hist5, hist6, hist7, hist8, hist9,
```

```
    hist10, ncol = 2)
```


Histogram of Price
mean = 297081
count
Price
0
500,000
1,000,000
1,500,000
2,000,000
2,500,000
Histogram of PricePerSqFt
mean = 127.7
count
PricePerSqFt
Histogram of HomeSize
mean = 2183.3
count
HomeSize
Histogram of PropertySize
an = 16884.5
count
PropertySize
Histogram of YearsOld
mean = 55.4
count
YearsOld
Histogram of YearsSinceLastSale
mean = 9.3
count
YearsSinceLastSale
Histogram of TaxValue
mean = 297734.3
count
TaxValue
Histogram of Bedrooms
mean = 3.3
count
Bedrooms
Histogram of Population
mean = 22042.4
count
Population
Histogram of Population.Density.sq.mi
mean = 2898.6
count
Population.Density.sq.mi

These histograms indicate that Price, HomeSize, TaxValue, and PropertySize are all skewed right, resembling more a log-normal distribution than a normal distribution. To adjust for this phenomenon, the natural log of each of the afforementioned features is taken prior to normalization using a z-score method.

Also gleaned from these visualizations is that PricePerSqFt and Bedrooms appear to be normally distributed, meaning it is safe to use a z-score normalization method without any pre-processing. Bedrooms can also be treated as an ordered factor, and both manifestations of the same feature were considered.

The empirical distributions for YearsOld, YearsSinceLastSale, Population, and Population.Density.sq.mi are not easily classifiable distributions, and thus will be scaled using a min-max method rather than over-fitting an empirically defined distribution function. Since there is some uncertainty here, YearsOld is normalized via two methods. Finally, YearsSinceLastSale resembles a Poisson distribution; thus, R's rpois() function was leveraged to scale this variable. The choice to evaluate this method was experimental.

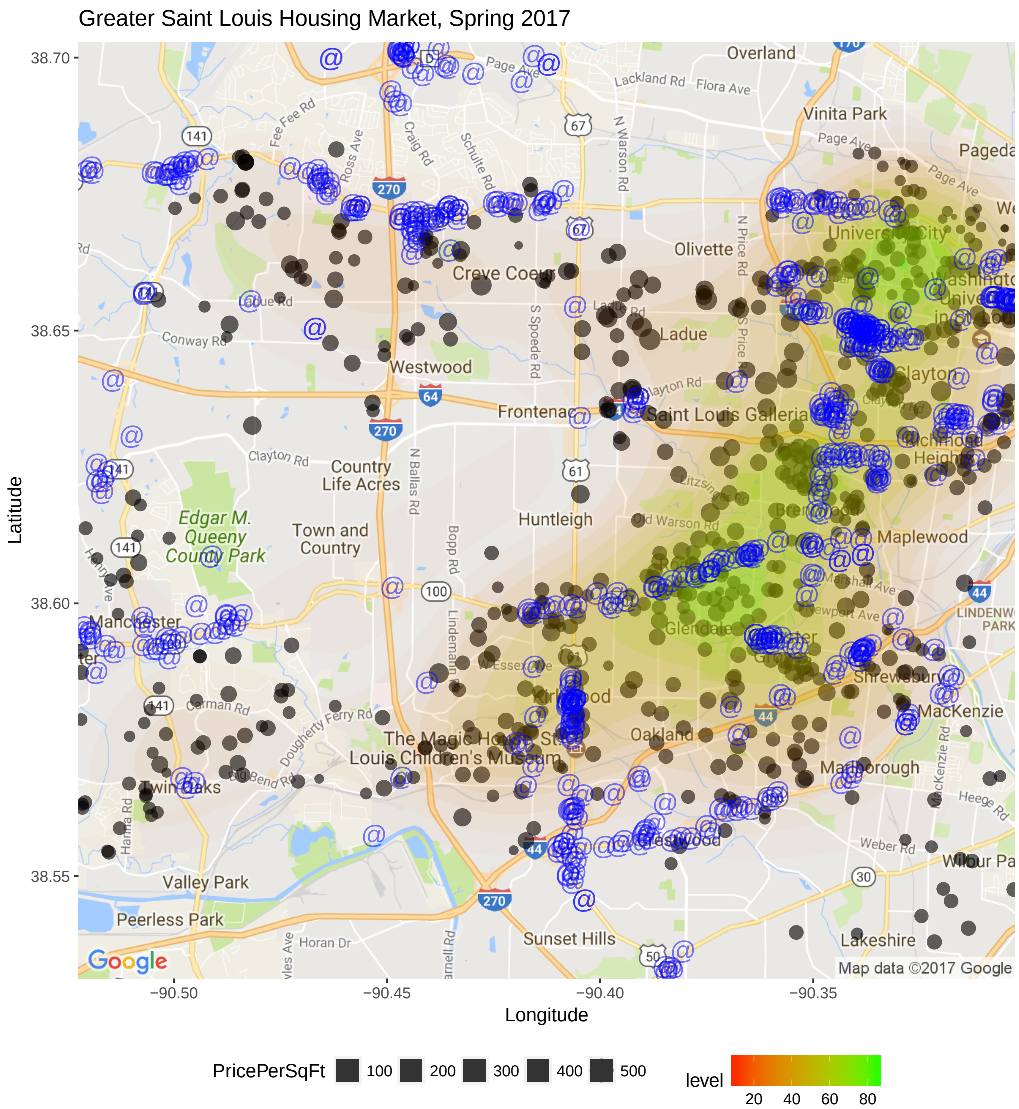
```
## Extract features based on distributions
df$LnPrice = log(df$Price)
df$LnPropertySize = log(df$PropertySize)
df$LnHomeSize = log(df$HomeSize)
df$LnTaxValue = log(df$TaxValue)

## Declare z-score scaler for re-use
scale_z = function(vector) {
    mu = mean(vector, na.rm = TRUE)
    sd = sd(vector, na.rm = TRUE)
    z = (vector - mu)/sd
    return(z)
}

## Normalize using z-score
df$BedroomsNormalized = scale_z(df$Bedrooms)
df$LnPriceNormalized = scale_z(df$LnPrice)
df$PricePerSqFtNormalized = scale_z(df$PricePerSqFt)
df$PriceNormalized = scale_z(df$Price)
df$LnPropertySizeNormalized = scale_z(df$LnPropertySize)
df$LnHomeSizeNormalized = scale_z(df$LnHomeSize)
df$YearsOldNormalized = scale_z(df$YearsOld)
df$LnTaxValueNormalized = scale_z(df$LnTaxValue)

## Declare function for minmax normalization
scale_minmax = function(vector) {
    min = min(vector, na.rm = TRUE)
    max = max(vector, na.rm = TRUE)
    rng = max - min
    scl = (vector - min)/(rng - min)
    return(scl)
}

## Normalize using minmax
df$YearsSinceLastSaleMinMaxScaled = scale_minmax(df$YearsSinceLastSale)
df$HomeSizeMinMaxScaled = scale_minmax(df$HomeSize)
df$HomeSizeSquaredMinMaxScaled = scale_minmax(df$HomeSizeSquared)
df$PopulationMinMaxScaled = scale_minmax(df$Population)
df$PopulationDensityMinMaxScaled = scale_minmax(df$Population.Density.sq.mi)

## Test Poisson normalization - pure inquiry
```

```r
mu_yrs = mean(ceiling(df$YearsSinceLastSale), na.rm = T)
df$YearsSinceLastSalePoissonNormalized = rpois(df$YearsSinceLastSale, lambda = mu_yrs)
```

Exploratory Analysis

Visualizing the community on geographic coordinates is often helpful for context, and may lead to hypotheses to test down the line. R's ggplot2 (Grammar of Graphics) and ggmap packages were used to plot the home listing dataset as well as the liquor license dataset in the same cartesian space. The listing data, represented by the black points, is scaled in size by PricePerSqFt, while the 2d density map represents price levels for the area. Quick analysis of this visualization confirms that the affluent areas of town (Clayton, Ladue, Webster Groves, et. al.) tend to have higher price levels.

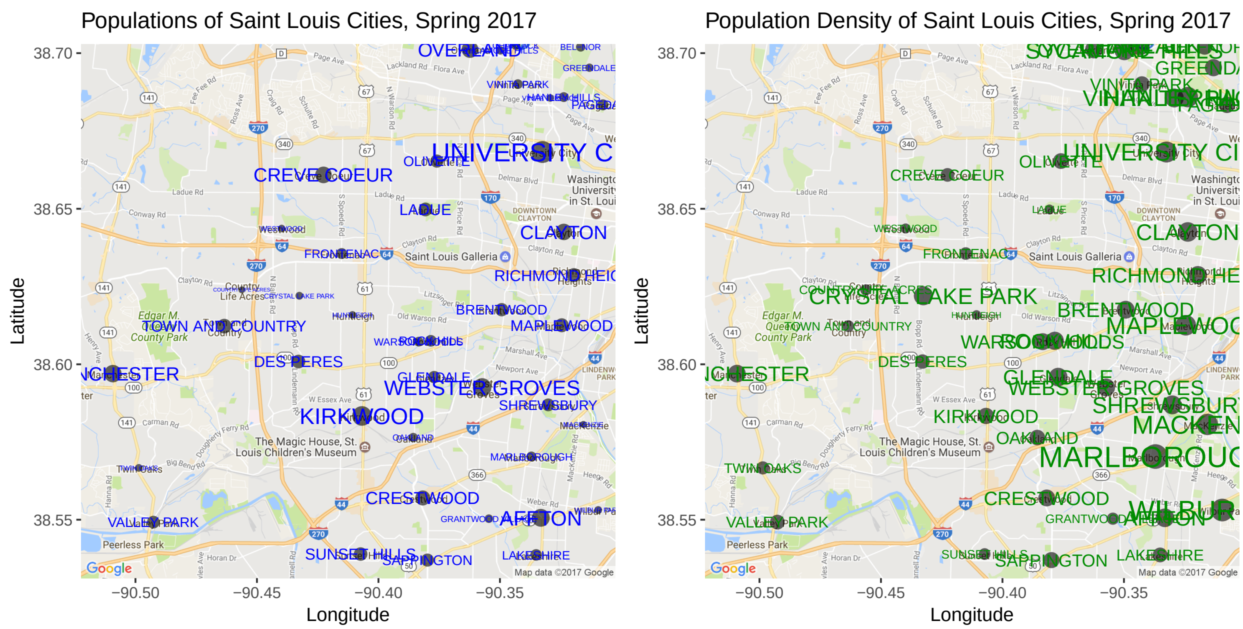
```r
## Display visually on map using ggmap
library(ggmap)
mu_lat = mean(df$Latitude, na.rm = T)
mu_lon = mean(df$Longitude, na.rm = T)

mp = get_map(c(mu_lon, mu_lat), zoom = 12, source = "google", maptype = "roadmap")
ggmap(mp) + geom_point(data = df, alpha = 0.6, aes(x = Longitude, y = Latitude,
    size = PricePerSqFt)) + stat_density2d(data = df, aes(x = Longitude, y = Latitude,
    size = Price, fill = ..level.., alpha = ..level..), bins = 12, geom = "polygon") +
    scale_fill_gradient(low = "red", high = "green") + scale_alpha(range = c(0,
    0.2), guide = FALSE) + geom_point(data = alc_df, aes(x = LON, y = LAT),
    shape = "@", size = 5, colour = "blue", alpha = 0.5) + labs(title = "Greater Saint Louis Housing Ma
    x = "Longitude", y = "Latitude") + theme(legend.position = "bottom")
```



The geospatial visualizations below are intended to help one gain a feel for the population and population density in the area. For visual clarity, the size of the text on the maps below is proportional to the value that is being displayed.

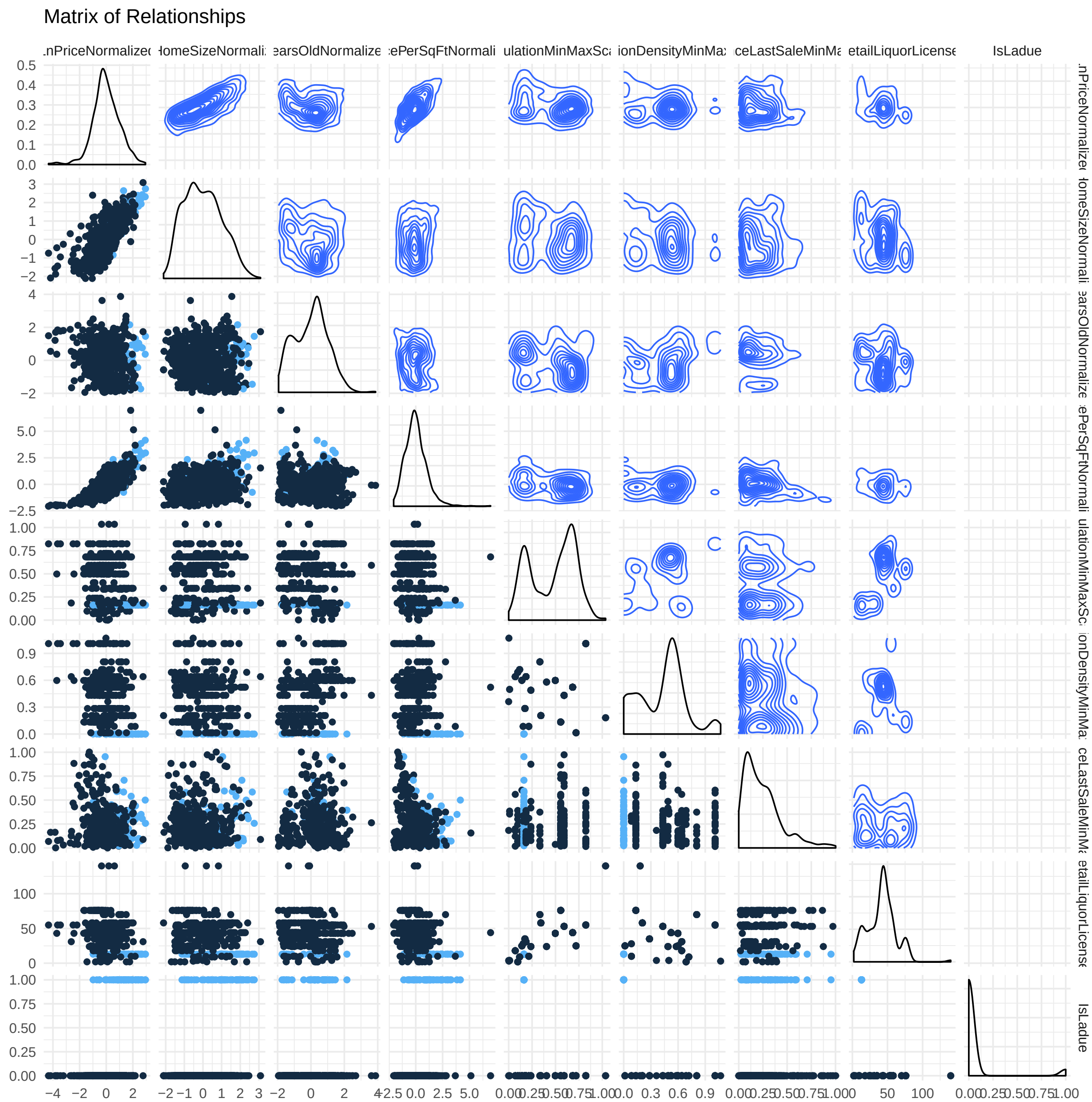
```
## Display visually on map using ggmap
library(gridExtra)
map1 = ggmap(mp) + geom_point(data = munis, alpha = 0.6, aes(x = LON, y = LAT,
    size = Population)) + scale_fill_gradient(low = "white", high = "red") +
    scale_alpha(range = c(0, 0.3), guide = FALSE) + labs(title = "Populations of Saint Louis Cities, Spr
    x = "Longitude", y = "Latitude") + theme(legend.position = "none") + geom_text(data = munis,
    aes(x = LON, y = LAT, size = Population, label = Municipality), colour = "blue1")

map2 = ggmap(mp) + geom_point(data = munis, alpha = 0.6, aes(x = LON, y = LAT,
    size = Population.Density.sq.mi)) + scale_fill_gradient(low = "white", high = "red") +
    scale_alpha(range = c(0, 0.3), guide = FALSE) + labs(title = "Population Density of Saint Louis Citi
    x = "Longitude", y = "Latitude") + theme(legend.position = "none") + geom_text(data = munis,
    aes(x = LON, y = LAT, size = Population.Density.sq.mi, label = Municipality),
    colour = "green4")
```

```r
grid.arrange(map1, map2, ncol = 2)
```



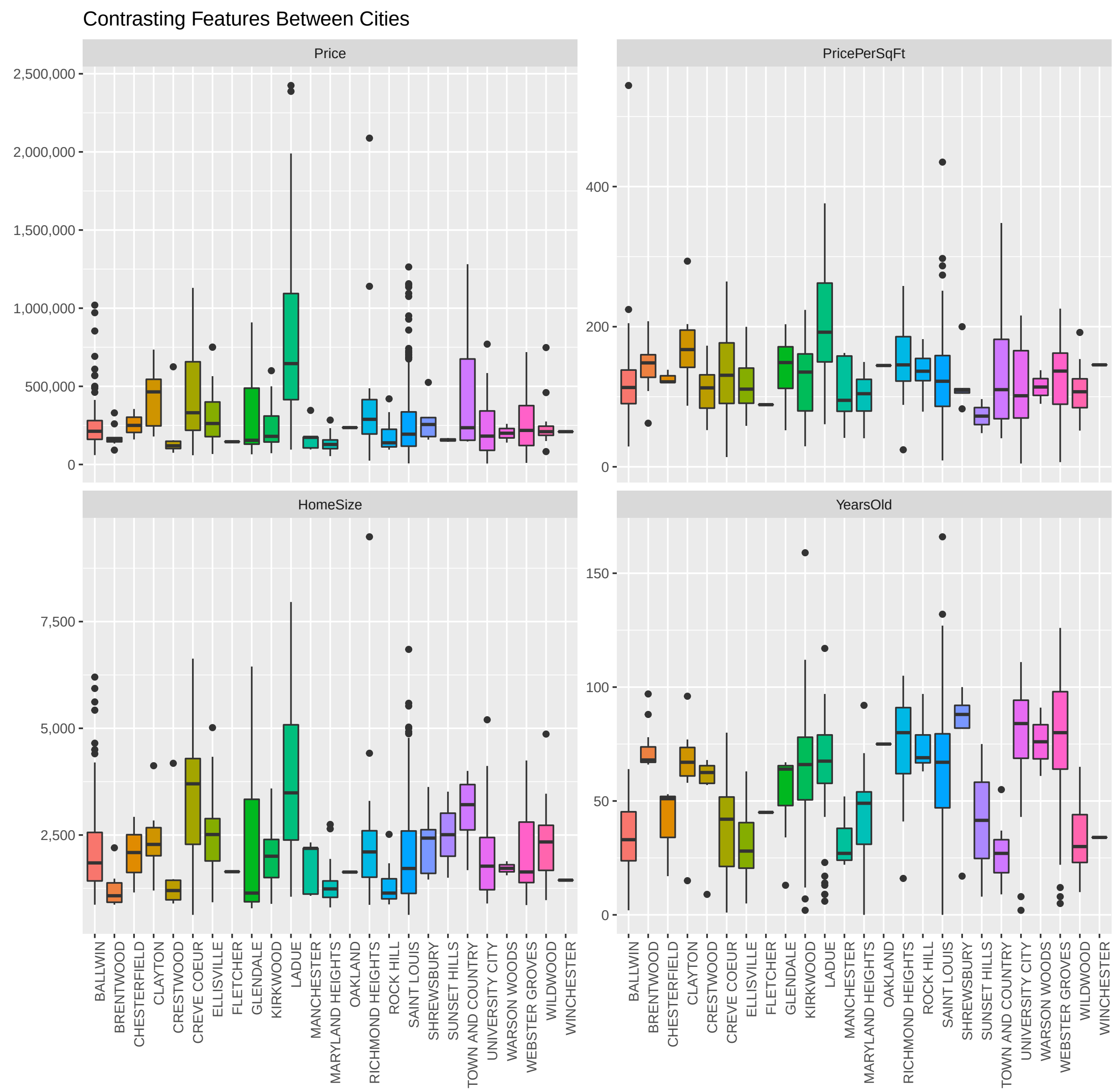
To gain an overall sense of the relationships present in the dataset, it is helpful to visualize a scatterplot/correlation matrix. The scatterplot matrix below shows tight correlations between HomeSize and Price. Relatively strong relationships exist between PricePerSqFt & Price and PropertySize & Price. It is important to note that HomeSize and PropertySize are highly correlated, thus invalidating their simultaneous use in predicting Price. Going forward HomeSize will be considered dominant to PropertySize following the intuition that the size of a home is often a dominant the home buying decision. There appears to be a linear relation between the normalized Price and YearsSinceLastSale variables, however there does appear to be considerable noise around the relationship. Interestingly, there appears to be a strong relationship between PricePerSqFt and Price. It is also shown that properties in Ladue, as marked by IsLadue, tend to command higher Price and PricePerSqFt. The variables for population, population density, and retail liquor licenses do not appear to be significant, likely reflecting the lack of granularity of information that is inherent with merging aggregated data with disaggregated observations.

```r
library(GGally)
suspected_PC = c("LnPriceNormalized", "LnHomeSizeNormalized", "YearsOldNormalized",
    "PricePerSqFtNormalized", "PopulationMinMaxScaled", "PopulationDensityMinMaxScaled",
    "YearsSinceLastSaleMinMaxScaled", "RetailLiquorLicenses", "IsLadue")
df_num = df[, suspected_PC]
df_num$IsLadue = as.numeric(df_num$IsLadue)
ggpairs(df_num, title = "Matrix of Relationships", upper = list(continuous = "density",
    combo = "box_no_facet"), lower = list(combo = "facetdensity"), aes(color = IsLadue)) +
    theme_minimal()
```



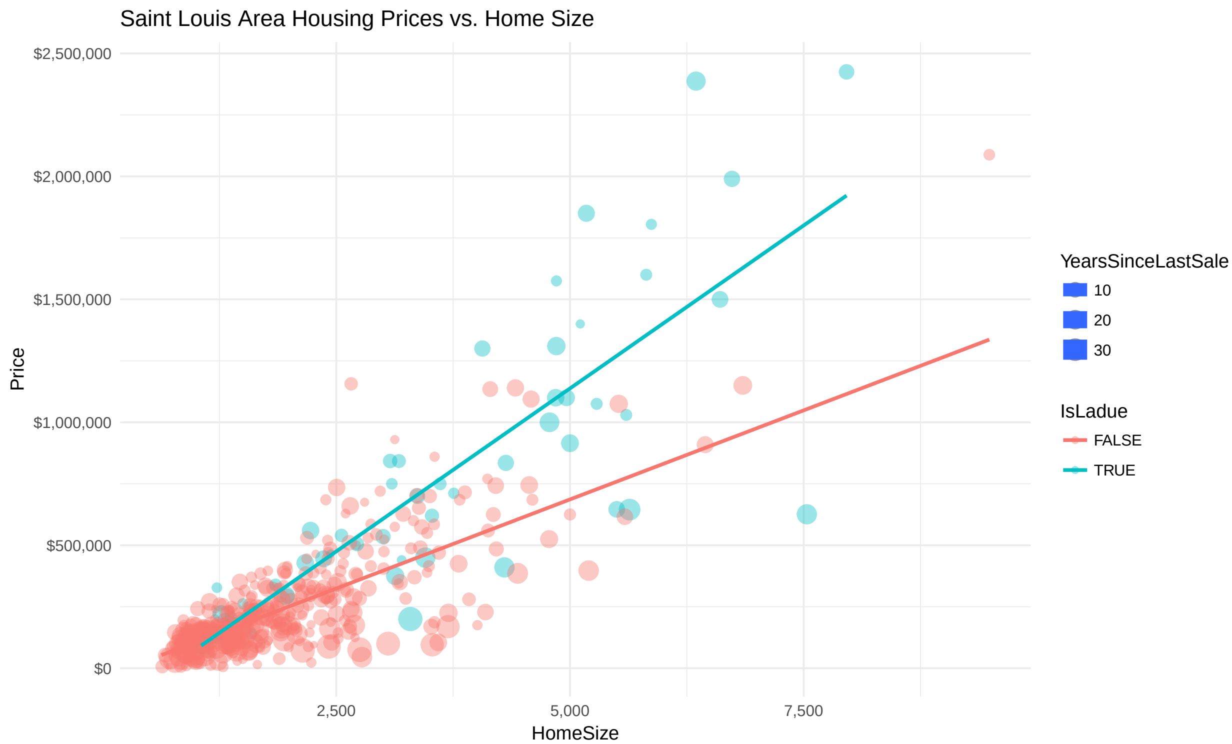
To visualize the differences between cities across several columns at once, the compiled dataset is reconfigured using R's reshape2 package. The data was melted into this format in order to take advantage of ggplot2's facet_wrap() function, which enables faceted visualizations to share their common axes, reducing visual clutter. The city of Ladue stands out again with large values for HomeSize, Price, and PricePerSqFt.

```
## Get data in easier to visualize format
library(reshape2)
cols = c("Price", "PricePerSqFt", "HomeSize", "YearsOld")
df_melt = melt(df, "City", cols)

## Create boxplot function to compare variables between cities
ggplot(data = df_melt, aes(x = City, y = value)) + geom_boxplot(aes(fill = City)) +
    theme(legend.position = "none", axis.text.x = element_text(angle = 90, hjust = 1)) +
    scale_y_continuous(labels = comma) + facet_wrap(~variable, ncol = 2, scales = "free_y") +
    labs(x = "", y = NULL, title = "Contrasting Features Between Cities")
```



Exploring the relationship previously uncovered between certain municipalities and list price, below visualizes the relationship between HomeSize and Price, using IsLadue (a binary variable indicating the listing's city is Ladue) as a dummy variable. The pre-normalized features are used to maintain interpretability and context.

```
g = ggplot(data = df, aes(x = HomeSize, y = Price, size = YearsSinceLastSale,
    group = IsLadue, label = City))
regr_plot = g + geom_point(aes(colour = IsLadue), alpha = 0.4) + geom_smooth(aes(colour = IsLadue),
    method = "lm", se = F) + scale_y_continuous(labels = dollar) + scale_x_continuous(labels = comma) +
    theme_minimal() + labs(title = "Saint Louis Area Housing Prices vs. Home Size")
regr_plot
```



The final dataset to be used in machine learning is shown below.

```r
print(head(df))
```

```
##      City                  Address                Address.1 State   Zip
## 1 BALLWIN     227 Oakbriar Farm Dr     227 Oakbriar Farm Dr    MO 63011
## 2 BALLWIN          968 Burgundy Ln          968 Burgundy Ln    MO 63021
## 3 BALLWIN   1584 Forest Springs Dr   1584 Forest Springs Dr    MO 63021
## 4 BALLWIN 16736 Highland Summit Dr 16736 Highland Summit Dr    MO 63021
## 5 BALLWIN     1039 Kiefer Ridge Dr     1039 Kiefer Ridge Dr    MO 63011
## 6 BALLWIN    725 Ridgeside Dr Apt H    725 Ridgeside Dr Apt H    MO 63011
##   ZillowID  Price Latitude Longitude TaxValue PropertySize YearBuilt
## 1  2829023 190000 38.58326 -90.57407   167100         6534      1987
## 2  2806547 294500 38.60405 -90.51114   359900        13068      1990
## 3 55187414 148000 38.55525 -90.52362   155700         3920      2000
## 4  2781273 275176 38.62333 -90.62510   365700        16552      1996
## 5 55187544 568454 38.55954 -90.56514   442600        14810      2004
## 6  2866646  95000 38.56350 -90.52083    77000         3964      1987
##   HomeSize Bedrooms Bathrooms LastSold Population Total.Area..mi2.
## 1     1562        3         2     <NA>      31283             8.95
## 2     2453        5         3     <NA>      31283             8.95
## 3     1378        2         4     <NA>      31283             8.95
## 4     3042        4         4     <NA>      31283             8.95
## 5     4122        4         4     <NA>      31283             8.95
## 6      868        2         2     <NA>      31283             8.95
##   Population.Density.sq.mi       LON      LAT RetailLiquorLicenses
## 1                   3494.6 -90.54623 38.59505                   44
## 2                   3494.6 -90.54623 38.59505                   44
## 3                   3494.6 -90.54623 38.59505                   44
## 4                   3494.6 -90.54623 38.59505                   44
```

```
## 5                  3494.6 -90.54623 38.59505                  44
## 6                  3494.6 -90.54623 38.59505                  44
##   PricePerSqFt YearsOld YearsSinceLastSale TaxValueToListPrice
## 1    121.63892       30                 NA           0.8794737
## 2    120.05707       27                 NA           1.2220713
## 3    107.40203       17                 NA           1.0520270
## 4     90.45891       21                 NA           1.3289676
## 5    137.90733       13                 NA           0.7786030
## 6    109.44700       30                 NA           0.8105263
##   PricePerLotSqFt            Coordinates HomeSizeSquared
## 1        29.07867 (38.583256,-90.574074)         2439844
## 2        22.53597 (38.604051,-90.511138)         6017209
## 3        37.75510  (38.555251,-90.52362)         1898884
## 4        16.62494 (38.623328,-90.625099)         9253764
## 5        38.38312 (38.559545,-90.565137)        16990884
## 6        23.96569 (38.563499,-90.520828)          753424
##   YearsSinceLastSale_5OrLess Bottom5Percentile_Price Top5Percentile_Price
## 1                         NA                   FALSE                FALSE
## 2                         NA                   FALSE                FALSE
## 3                         NA                   FALSE                FALSE
## 4                         NA                   FALSE                FALSE
## 5                         NA                   FALSE                FALSE
## 6                         NA                   FALSE                FALSE
##   Bottom5Percentile_HomeSize Top5Percentile_HomeSize
## 1                      FALSE                   FALSE
## 2                      FALSE                   FALSE
## 3                      FALSE                   FALSE
## 4                      FALSE                   FALSE
## 5                      FALSE                   FALSE
## 6                       TRUE                   FALSE
##   Bottom5Percentile_YearBuilt Top5Percentile_YearBuilt
## 1                       FALSE                    FALSE
## 2                       FALSE                    FALSE
## 3                       FALSE                    FALSE
## 4                       FALSE                    FALSE
## 5                       FALSE                    FALSE
## 6                       FALSE                    FALSE
##   Bottom5Percentile_Population Top5Percentile_Population
## 1                        FALSE                     FALSE
## 2                        FALSE                     FALSE
## 3                        FALSE                     FALSE
## 4                        FALSE                     FALSE
## 5                        FALSE                     FALSE
## 6                        FALSE                     FALSE
##   Bottom5Percentile_RetailLiquorLicenses
## 1                                  FALSE
## 2                                  FALSE
## 3                                  FALSE
## 4                                  FALSE
## 5                                  FALSE
## 6                                  FALSE
##   Top5Percentile_RetailLiquorLicenses Bottom5Percentile_PricePerSqFt
## 1                               FALSE                          FALSE
## 2                               FALSE                          FALSE
```

```
## 3                              FALSE                          FALSE
## 4                              FALSE                          FALSE
## 5                              FALSE                          FALSE
## 6                              FALSE                          FALSE
##   Top5Percentile_PricePerSqFt BedroomsFactor FivePlusBedrooms FiveBedrooms
## 1                       FALSE              3            FALSE        FALSE
## 2                       FALSE              5             TRUE         TRUE
## 3                       FALSE              2            FALSE        FALSE
## 4                       FALSE              4            FALSE        FALSE
## 5                       FALSE              4            FALSE        FALSE
## 6                       FALSE              2            FALSE        FALSE
##   SixBedrooms SevenBedrooms ThreeOrMoreBaths IsLadue IsTownAndCountry
## 1       FALSE         FALSE            FALSE   FALSE            FALSE
## 2       FALSE         FALSE             TRUE   FALSE            FALSE
## 3       FALSE         FALSE             TRUE   FALSE            FALSE
## 4       FALSE         FALSE             TRUE   FALSE            FALSE
## 5       FALSE         FALSE             TRUE   FALSE            FALSE
## 6       FALSE         FALSE            FALSE   FALSE            FALSE
##   IsRichmondHeights IsMarylandHeights  LnPrice LnPropertySize LnHomeSize
## 1             FALSE             FALSE 12.15478       8.784775   7.353722
## 2             FALSE             FALSE 12.59303       9.477922   7.805067
## 3             FALSE             FALSE 11.90497       8.273847   7.228388
## 4             FALSE             FALSE 12.52517       9.714262   8.020270
## 5             FALSE             FALSE 13.25068       9.603058   8.324094
## 6             FALSE             FALSE 11.46163       8.285009   6.766192
##   LnTaxValue BedroomsNormalized LnPriceNormalized PricePerSqFtNormalized
## 1   12.02635         -0.3177216        -0.1404369             -0.1014588
## 2   12.79358          1.5677359         0.3909986             -0.1278966
## 3   11.95569         -1.2604504        -0.4433630             -0.3394031
## 4   12.80957          0.6250071         0.3087005             -0.6225773
## 5   13.00042          0.6250071         1.1884659              0.1704386
## 6   11.25156         -1.2604504        -0.9809592             -0.3052251
##   PriceNormalized LnPropertySizeNormalized LnHomeSizeNormalized
## 1    -0.371678999               -0.4462561           -0.3730219
## 2    -0.008958659                0.2207531            0.4924855
## 3    -0.517461336               -0.9379170           -0.6133646
## 4    -0.076032418                0.4481813            0.9051638
## 5     0.941937876                0.3411704            1.4877812
## 6    -0.701424762               -0.9271760           -1.4996820
##   YearsOldNormalized LnTaxValueNormalized YearsSinceLastSaleMinMaxScaled
## 1         -0.8899630           -0.3741781                             NA
## 2         -0.9949964            0.6362040                             NA
## 3         -1.3451076           -0.4672332                             NA
## 4         -1.2050631            0.6572577                             NA
## 5         -1.4851521            0.9085951                             NA
## 6         -0.8899630           -1.3945072                             NA
##   HomeSizeMinMaxScaled HomeSizeSquaredMinMaxScaled PopulationMinMaxScaled
## 1           0.11329929                 0.022905426              0.6828837
## 2           0.22161439                 0.063014733              0.6828837
## 3           0.09093119                 0.016840199              0.6828837
## 4           0.29321663                 0.099302889              0.6828837
## 5           0.42450766                 0.186051243              0.6828837
## 6           0.02893265                 0.003997336              0.6828837
##   PopulationDensityMinMaxScaled YearsSinceLastSalePoissonNormalized
```

```
## 1                    0.5225785                                  6
## 2                    0.5225785                                 12
## 3                    0.5225785                                  8
## 4                    0.5225785                                 11
## 5                    0.5225785                                 10
## 6                    0.5225785                                 10
```

Machine Learning

Using the features generated up until this point, and guided by the insights gained from the exploratory analysis conducted, several machine learning models were tested on the data for their utility in predicting Price and PricePerSqFt, among other features.

The first step to ensure integrity is to create a training dataset and a testing dataset by splitting the original dataset. A random seed is also set for reproducibility.

```
## Set seed for reproducibility
set.seed(7)

## Specify 70% training, 30% testing
pct_train = 0.7
ix = NROW(df)
cutoff_ix = round(pct_train * ix, 0)

## Generate random sorting column & sort to ensure we get as close to random
## sample as possible
df$SORTME = rnorm(1:ix, 500, 10000)
df = df[order(df$SORTME), ]
df$SORTME = NULL

train_df = df[1:cutoff_ix, ]
test_df = df[cutoff_ix:ix, ]
```

A natural place to start is with linear regression. It is somewhat obvious to use square footage as a predictor of price, however this model attempts to go a bit deeper by introducing two more variables. While the IsLadue variable is significant in determining HomeSize (a source of collinearity), the IsLadue variable was used in the model because the coefficient of variation of HomeSize ~ IsLadue is only about 7.4%. YearsSinceLastSale is tested as a predictor of Price in an attempt to capture the recent phenomenon of speculators remodeling their properties with the intent to sell at a profit. This variable is a proxy for the "HGTV Effect". It is shown that all variables are significant at the 1% level. Further, each predictor was scrutinized for collinearity with the others, and only those that pass the test were kept in the model. The training model appears to be strong, showing an adjusted coefficient of determination of 62.19%. When analyzing the residuals, however, it appears that there is bias in the model due to the non-normal distributions present in the dataset.

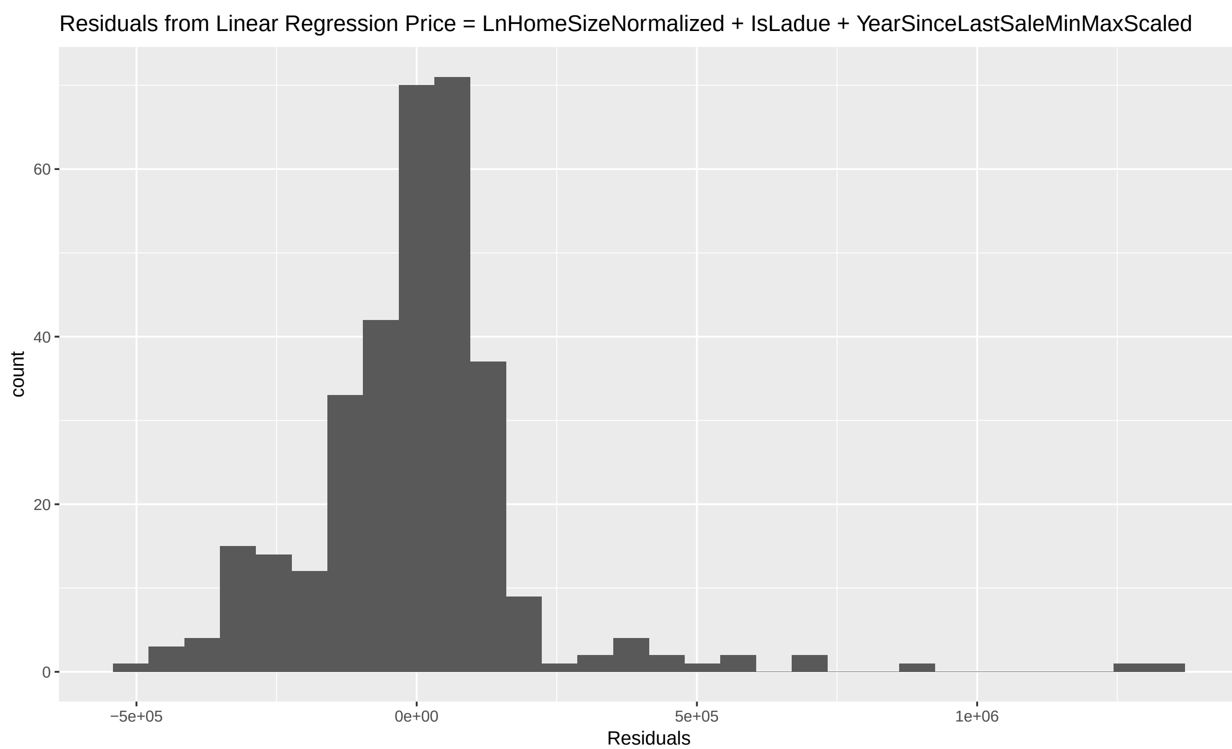
```
mod = lm(Price ~ LnHomeSizeNormalized + IsLadue + YearsSinceLastSaleMinMaxScaled,
    data = train_df)
summary(mod)
```

```
##
## Call:
## lm(formula = Price ~ LnHomeSizeNormalized + IsLadue + YearsSinceLastSaleMinMaxScaled,
##     data = train_df)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -505779  -96312    9772   69637 1343074
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     336460      18206  18.480  < 2e-16 ***
## LnHomeSizeNormalized            191543      12071  15.868  < 2e-16 ***
## IsLadueTRUE                     371640      41181   9.024  < 2e-16 ***
## YearsSinceLastSaleMinMaxScaled  -214675      56466  -3.802 0.000172 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 203100 on 324 degrees of freedom
##   (241 observations deleted due to missingness)
## Multiple R-squared:  0.627,  Adjusted R-squared:  0.6235
## F-statistic: 181.5 on 3 and 324 DF,  p-value: < 2.2e-16
```

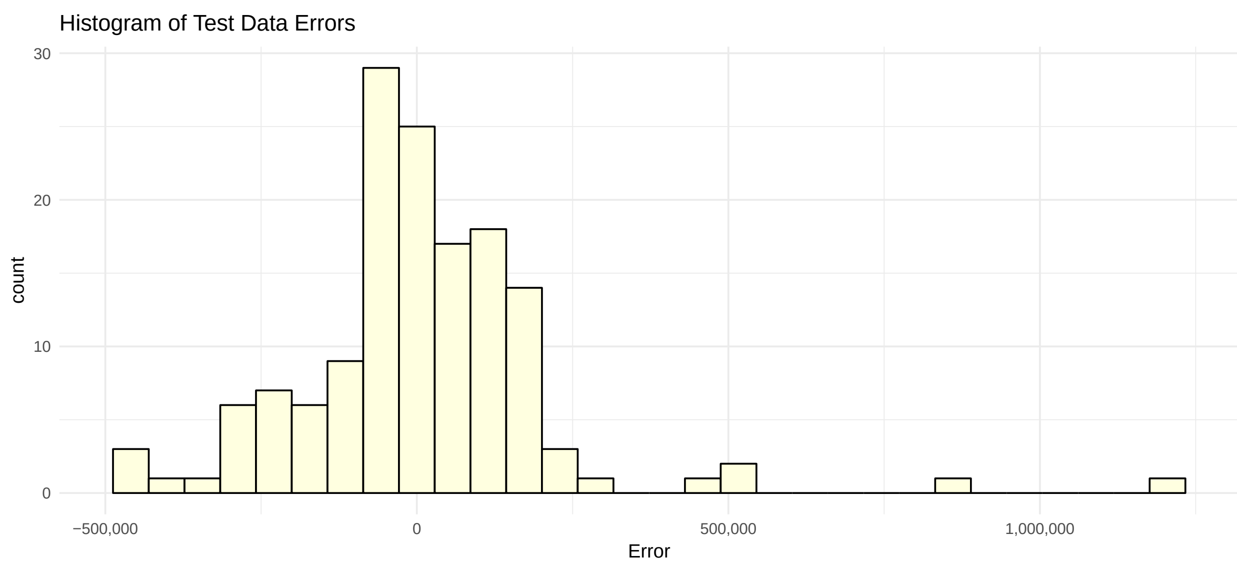
We see that the model performs well on the training data, achieving an adjusted coefficient of determination of 62.35%. Further, the p-value on each predictor is highly significant. Below is a crude visualization of the empirical distribution of the errors, showing that the residuals are biased considerably. This indicates that the model is not an ideal representation of the phenomena, and further investigation is needed. The source of this bias is likely However this model is still a "good" model in the sense that it is useful as a lense for viewing the market.

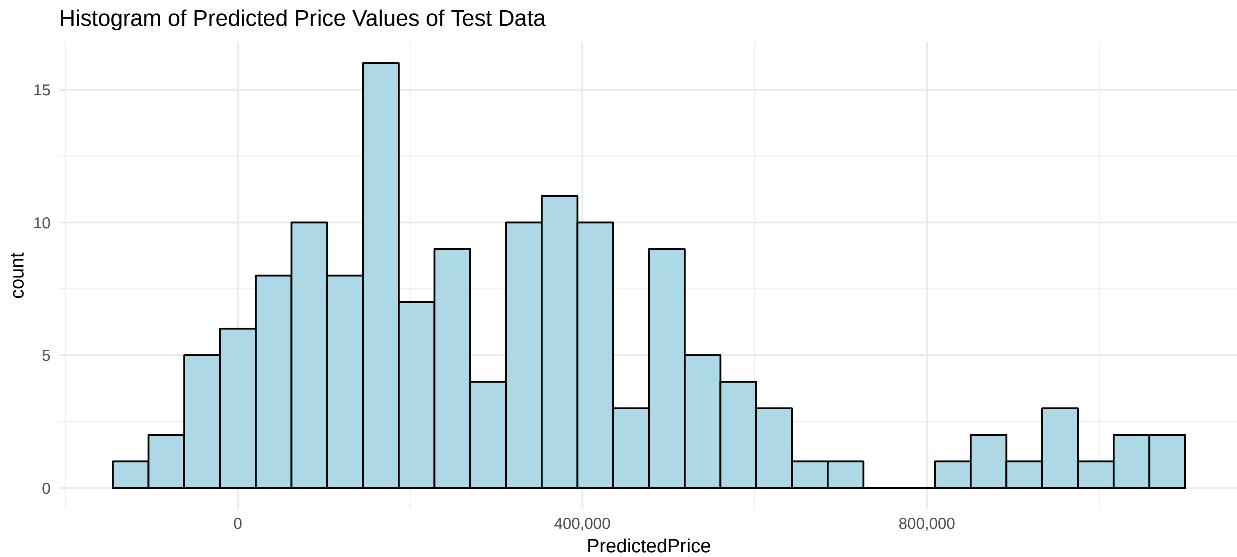
```
qplot(mod$residuals, geom = "histogram", main = "Residuals from Linear Regression Price = LnHomeSizeNor
    xlab = "Residuals")
```



The testing dataset is used below to validate the model derived above. The same bias seems to appear in this dataset as in the training data, though less pronounced. The linear model also predicted many values that are negative, meaning the model is certainly not realistic as a pricing tool. This is likely due to the lack of signal coming from the large number of homes that occupy the lower end of the Price spectrum. It is likely that the noise identified in YearsSinceLastSale is causing miscalibration in the model, and in the future this variable will be dropped to test for sensitivity. At a high-level, it does appear that the model is robust in teh sense that the coefficient of determination is similar to the training model, coming in at 63.6%.

```
keep_cols = c("LnHomeSizeNormalized", "IsLadue", "YearsSinceLastSaleMinMaxScaled")
test_df = test_df[complete.cases(test_df[, keep_cols]) == T, ]
Y = test_df$Price
test_df = test_df[, keep_cols]
test_df$PredictedPrice = predict(mod, test_df)
test_df$ActualPrice = Y
test_df$Error = test_df$ActualPrice - test_df$PredictedPrice

one = ggplot(test_df, aes(x = Error)) + geom_histogram(colour = "black", fill = "lightyellow") +
    labs(title = "Histogram of Test Data Errors") + theme_minimal() + scale_x_continuous(labels = comma
two = ggplot(test_df, aes(x = PredictedPrice)) + geom_histogram(colour = "black",
    fill = "lightblue") + labs(title = "Histogram of Predicted Price Values of Test Data") +
    theme_minimal() + scale_x_continuous(labels = comma)
grid.arrange(one, two, ncol = 1)
```



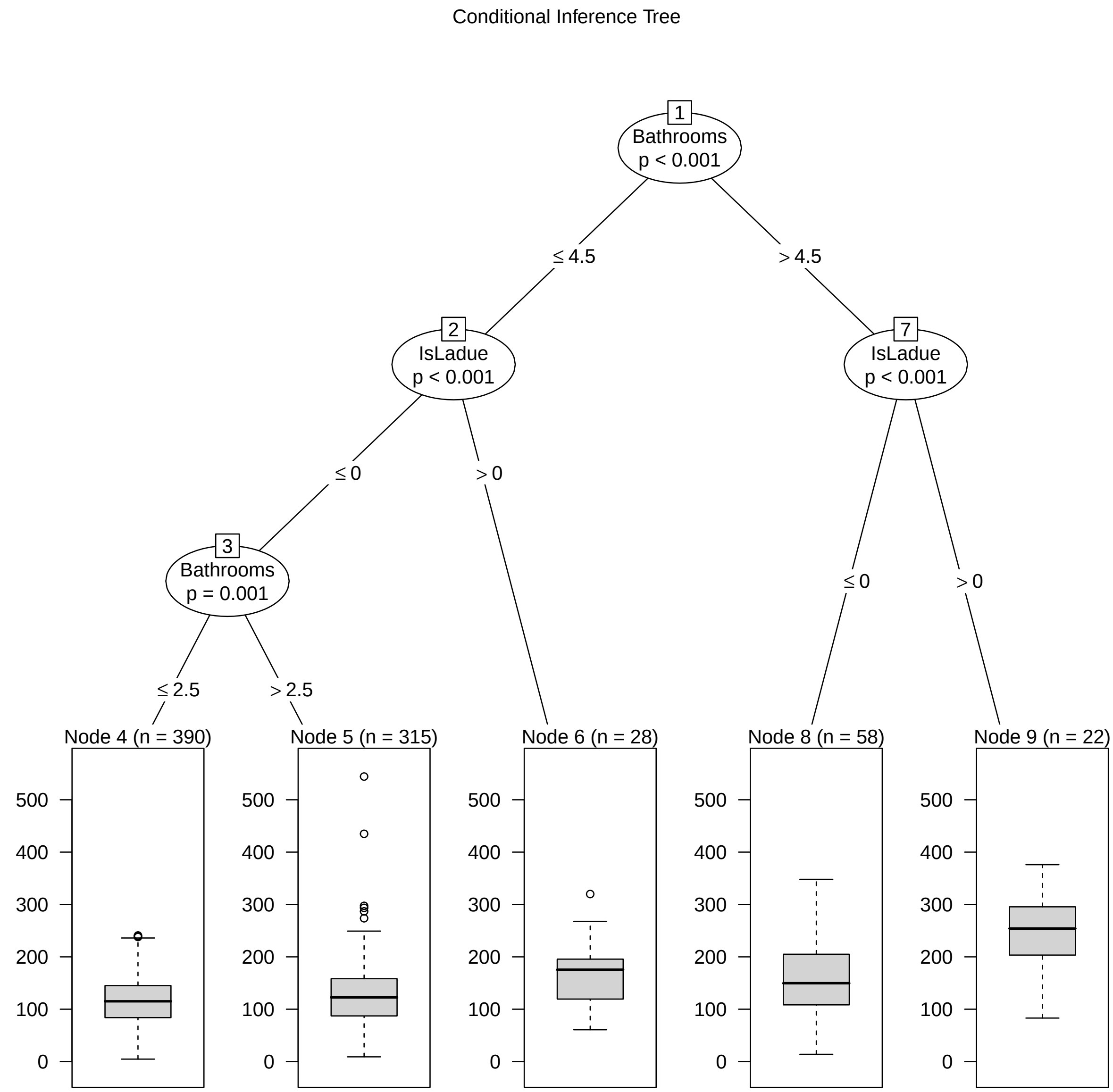


```r
r_sq = cor(test_df$ActualPrice, test_df$PredictedPrice)^2
print(paste0("R-squared for the testing model is ", round(r_sq, 4)))
```

```
## [1] "R-squared for the testing model is 0.6285"
```

To explore the available heuristics in the context of predicting PricePerSqFt, a conditional inference tree was explored. YearSinceLastSale was dropped for this model due to its inconsistent signal. It is interesting to note that Bathrooms emerges as a key differentiator, along with the IsLadue indicator. We see that homes in Ladue with more than 4.5 bathrooms command a much higher PricePerSqFt than others. It is interesting that the Bathrooms variable is more important than HomeSize, PropertySize, YearsOld, and Bedrooms. This model was explored but not employed due to its lack of interpretability and the limited sample used to train. In future elaborations, this model will be fed more data to better represent each leaf in the tree.

```r
library(party)
FN = PricePerSqFt ~ HomeSize + IsLadue + PropertySize + YearsOld + Bedrooms +
    Bathrooms
tree = ctree(FN, data = df)
plot(tree, main = "Conditional Inference Tree")
```



Conclusions & Further Research

The pricing model derived using multiple regression appears to have utility and promise, and elaborating on both the dataset and exploring further the models available will likely prove fruitful. Given the limited sample size of this investigation it is safe to assume that any conclusions reached in this inquiry are incomplete. However the mechanics underlying Price seem to make intuitive sense, giving plenty of inspiration for further research. Further inquiry should include the creation of a database of all records investigated, panel data on the same properties over time, and if possible should include the actual price at which a given property wound up selling.

References

- Municipalities of Saint Louis, Missouri, Wikipedia https://en.wikipedia.org/wiki/Municipalities_of_St._Louis_County,_Missouri Accessed April 2017
- Realtor.com, https://realtor.com
- Zillow.com, https://zillow.com
- Zillow API https://www.zillow.com/howto/api/APIOverview.htm
- Leonard Richardson, BeautifulSoup, https://www.crummy.com/software/BeautifulSoup/
- BeautifulSoup via Stanford http://web.stanford.edu/~zlotnick/TextAsData/Web_Scraping_with_Beautiful_Soup.html
- Hadley Wickham & David Kahle, ggmap https://cran.r-project.org/web/packages/ggmap/ggmap.pdf
- Hadley Wickham, ggplot2 (Grammar of Graphics) https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf
- pandas http://pandas.pydata.org/
- Gretchen R. Crowe, Top 10 Catholic Cities, USA, OSV Newsweekly https://www.osv.com/osvnewsweekly/byissue/article/tabid/735/artmid/13636/articleid/9926/top-10-catholic-cities-usa.aspx 5/22/2013
- Hadley Wickham, reshape2, https://cran.r-project.org/web/packages/reshape2/reshape2.pdf