Final Capstone

Abhishek Dendukuri

7/28/2018

I. Introduction

Housing prices have long been dictated and settled by real estate agents as their background knowledge about the property reliably led to sales. But as technology evolves and as new data is found, it was only time before companies like Zillow attacked the online housing market to track property prices for prospective homewoners.

The Zestimate model that Zillow devloped computs the estimated market value of a home using a formula that only analysts at the company know. Using certain factors and environmental variables, it generates a comparative market analysis - or CMA - that can determina valuation similar to one given by a real estate agent.

The Problem

Currently we have no way of assessing whether or not Zestimate is the most reliable way to predict home prices. But there are not a lot of companies that have had the resources or time to tackle this as long as Zillow has, so it is perhaps the only way we can get a good idea of a property's value whether or not it's correct.

What I'd like to investigate is if using basic machine learning methods can lay a foundation for estimating the price of a home. I believe that all the methods I utilize are going to be very inaccurate but I would like to understand which model proves to be the most accurate relative to the others.

The point of this investigation is to comprehend what exactly holds other companies back from creating an effective solution and revealing what can be added or removed to provide a more effective analysis.

II. Cleaning the Dataset

One of the benefits of having a dataset that Zillow provides is the incredible breadth of factors as well as the vast number of properties that the Zestmiate analyzes. It is important to have a number of data points in order to create a much more comprehensive formula that can utilize all this information to create a better model. Further analysis reveals that the entire dataset provided only supplies housing data for properties in the Los Angeles area. This is really helpful because we can visualize how the real estate market is for only one small area of the map - anything larger would cause the number of properties to explode, making it harder to organize the data and thus making it more challenging to divise a relatively reliable model.

However the dataset is not without its drawbacks. Namely, the breadth of factors comes into question to a point where some of the factors seem unnecessary to include because they are missing several data points. This will be explored to a greater extent in the data wrangling section. As mentioned earlier, the data is only available for the Los Angeles area, and while it is helpful that all the data is localized in one region, this can be seen as a drawback because as far as our analysis goes, we will not be able to compare how well the formula applies between states.

Setup

First we set the working directory and read in the information from our properties file into a dataframe. This spreadsheet proves to be massive in size so the data frame generation tends to take a few minutes before everything is properly setup.

The train spreadsheet is much smaller in size, but will prove to be a very significant part of the data organization and wrangling and will be further explored below.

```
setwd("/Users/JARVIS/RStudio/Capstone")

df <- read.csv("properties_2017.csv")
properties <- tbl_df(df)


df2 <- read.csv("train_2017.csv")
train <- tbl_df(df2)
```

Here we create a new dataframe that merges the previous two: because there are only so many parcel ids available in the train dataframe and since the merge() function performs an inner join, this will automatically clean out over 2 million parcels.

```
prop.train.join <- merge(properties, train, by="parcel_id")
nrow(properties)
```

```
## [1] 2985217
```

```
nrow(train)
```

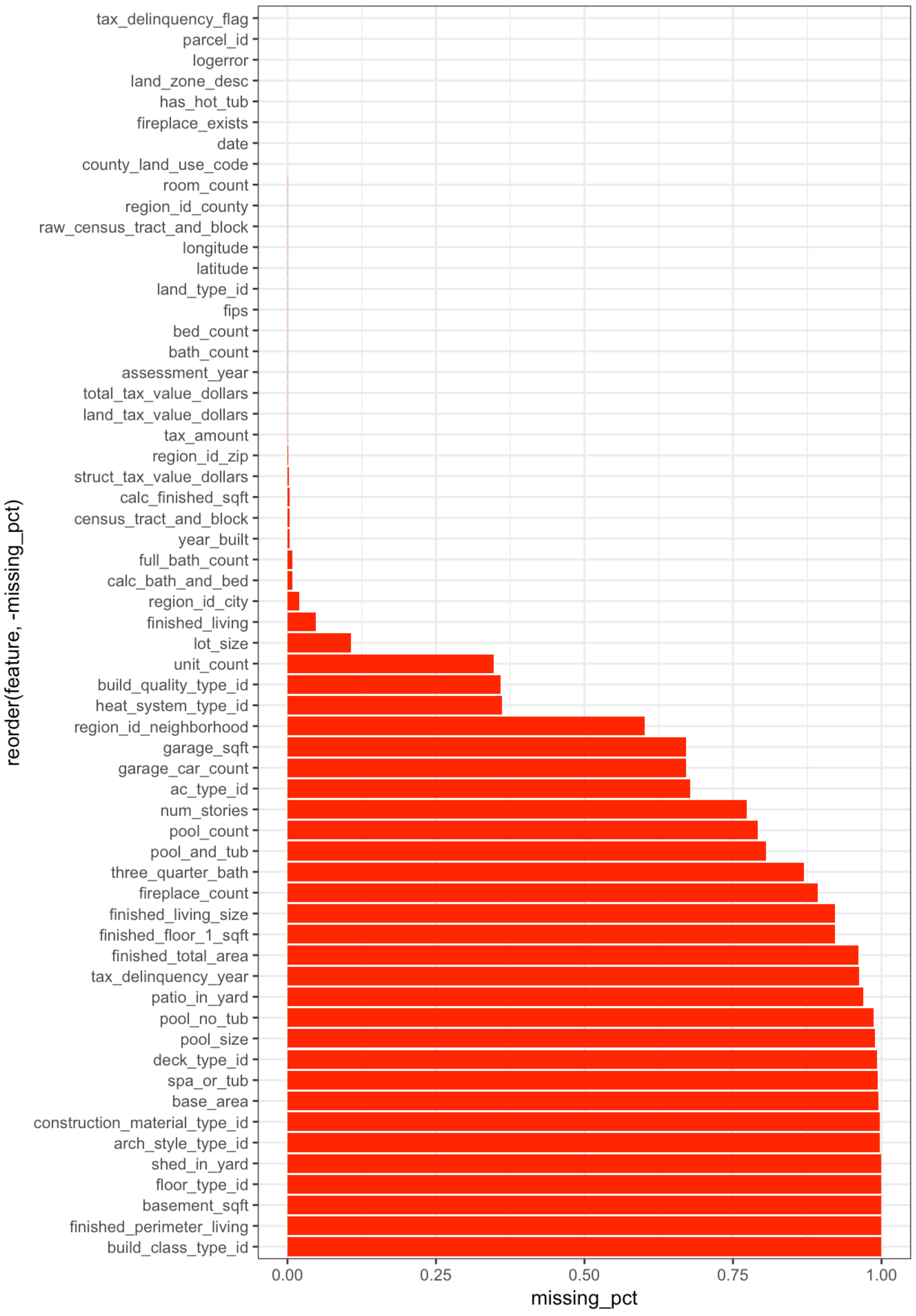
```
## [1] 77613
```

```
nrow(prop.train.join)
```

```
## [1] 77613
```

Missing Percentage Plot

In order to understand which information to remove, it is best to display what factors are missing the highest amount of values. From here we can create a cutoff that would represent our data in a tidier fashion


tax_delinquency_flag
parcel_id
logerror
land_zone_desc
has_hot_tub
fireplace_exists
date
county_land_use_code
room_count
region_id_county
raw_census_tract_and_block
longitude
latitude
land_type_id
fips
bed_count
bath_count
assessment_year
total_tax_value_dollars
land_tax_value_dollars
tax_amount
region_id_zip
struct_tax_value_dollars
calc_finished_sqft
census_tract_and_block
year_built
full_bath_count
calc_bath_and_bed
region_id_city
finished_living
lot_size
unit_count
build_quality_type_id
heat_system_type_id
region_id_neighborhood
garage_sqft
garage_car_count
ac_type_id
num_stories
pool_count
pool_and_tub
three_quarter_bath
fireplace_count
finished_living_size
finished_floor_1_sqft
finished_total_area
tax_delinquency_year
patio_in_yard
pool_no_tub
pool_size
deck_type_id
spa_or_tub
base_area
construction_material_type_id
arch_style_type_id
shed_in_yard
floor_type_id
basement_sqft
finished_perimeter_living
build_class_type_id
reorder(feature, -missing_pct)
0.00
0.25
0.50
0.75
1.00
missing_pct

As we talked about earlier, there are some factors that merited a second look because they were littered with empty data points. As this chart reveals, there are multiple factors that are missing more than 90% of the data points considering the number of ‘parcels’ that were initially given.

Using this information we can now limit our data to the most consistently available factors using the following chunk of code:

```
# Create array that stores the missing percentage of each feature
percentMiss <- sapply(prop.train.join, function(x) {sum(is.na(x)) / length(x)})

# Remove values that have a percent miss rate of greater than 3.5%
removeMiss <- which(percentMiss > 0.035)
prop.train.join <- subset(prop.train.join, select=-removeMiss)
prop.train.join <- prop.train.join %>% select(-has_hot_tub, -land_zone_desc, -fireplace_
exists, -tax_delinquency_flag)


# Next omit the remainder of the NA values that persisted in the remaining columns
prop.train.join <- na.omit(prop.train.join)
```

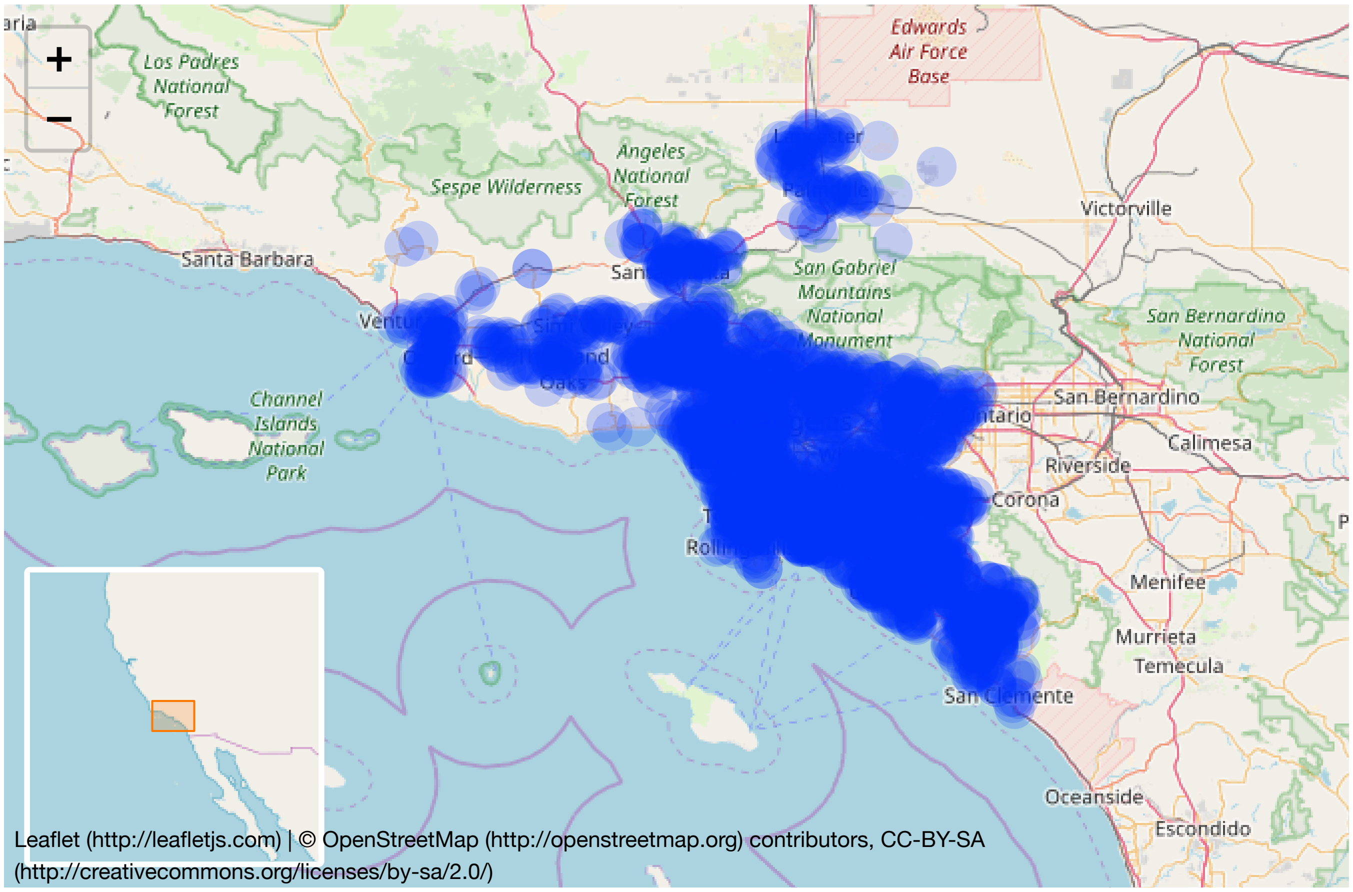
```
# Output
head(prop.train.join)
```

```
##   parcel_id bath_count bed_count calc_bath_and_bed calc_finished_sqft fips
## 1  10711855          2         3                 2               2107 6037
## 2  10711877          2         4                 2               1882 6037
## 3  10711888          2         4                 2               1882 6037
## 4  10711910          2         3                 2               1477 6037
## 5  10711923          2         4                 2               1918 6037
## 6  10711945          2         3                 2               2031 6037
##   full_bath_count latitude  longitude county_land_use_code land_type_id
## 1               2 34222559 -118617387                 0101          261
## 2               2 34220261 -118616409                 0101          261
## 3               2 34222491 -118616854                 0100          261
## 4               2 34221864 -118615739                 0101          261
## 5               2 34220619 -118615253                 0101          261
## 6               2 34220664 -118614105                 0101          261
##   raw_census_tract_and_block region_id_city region_id_county region_id_zip
## 1                   60371132          12447             3101         96339
## 2                   60371132          12447             3101         96339
## 3                   60371132          12447             3101         96339
## 4                   60371132          12447             3101         96339
## 5                   60371132          12447             3101         96339
## 6                   60371132          12447             3101         96339
##   room_count year_built struct_tax_value_dollars total_tax_value_dollars
## 1          0       1972                   249655                  624139
## 2          0       1972                   253000                  660000
## 3          0       1972                   257591                  542923
## 4          0       1960                    57968                   78031
## 5          0       1960                   167869                  415459
## 6          0       1958                    85298                  424414
##   assessment_year land_tax_value_dollars tax_amount census_tract_and_block
## 1            2016                 374484    7659.36           6.037113e+13
## 2            2016                 407000    8123.91           6.037113e+13
## 3            2016                 285332    6673.24           6.037113e+13
## 4            2016                  20063    1116.46           6.037113e+13
## 5            2016                 247590    5239.85           6.037113e+13
## 6            2016                 339116    5376.97           6.037113e+13
##       logerror       date
## 1 -0.007357289 2017-07-07
## 2  0.021066284 2017-08-29
## 3  0.077174457 2017-04-04
## 4 -0.041237636 2017-03-17
## 5 -0.009496421 2017-03-24
## 6  0.001270756 2017-01-30
```

III. Exploring the Data

Now that the data has been properly cleaned and sorted for our use, let us take a look at where the properties lie on the map.

```
## Assuming "long" and "lat" are longitude and latitude, respectively
```


Los Padres National Forest
Edwards Air Force Base
Angeles National Forest
Sespe Wilderness
Victorville
Santa Barbara
San Gabriel Mountains National Monument
San Bernardino National Forest
Channel Islands National Park
San Bernardino
Calimesa
Riverside
Corona
Menifee
Murrieta
Temecula
San Clemente
Oceanside
Escondido
Leaflet (http://leafletjs.com) | © OpenStreetMap (http://openstreetmap.org) contributors, CC-BY-SA (http://creativecommons.org/licenses/by-sa/2.0/)

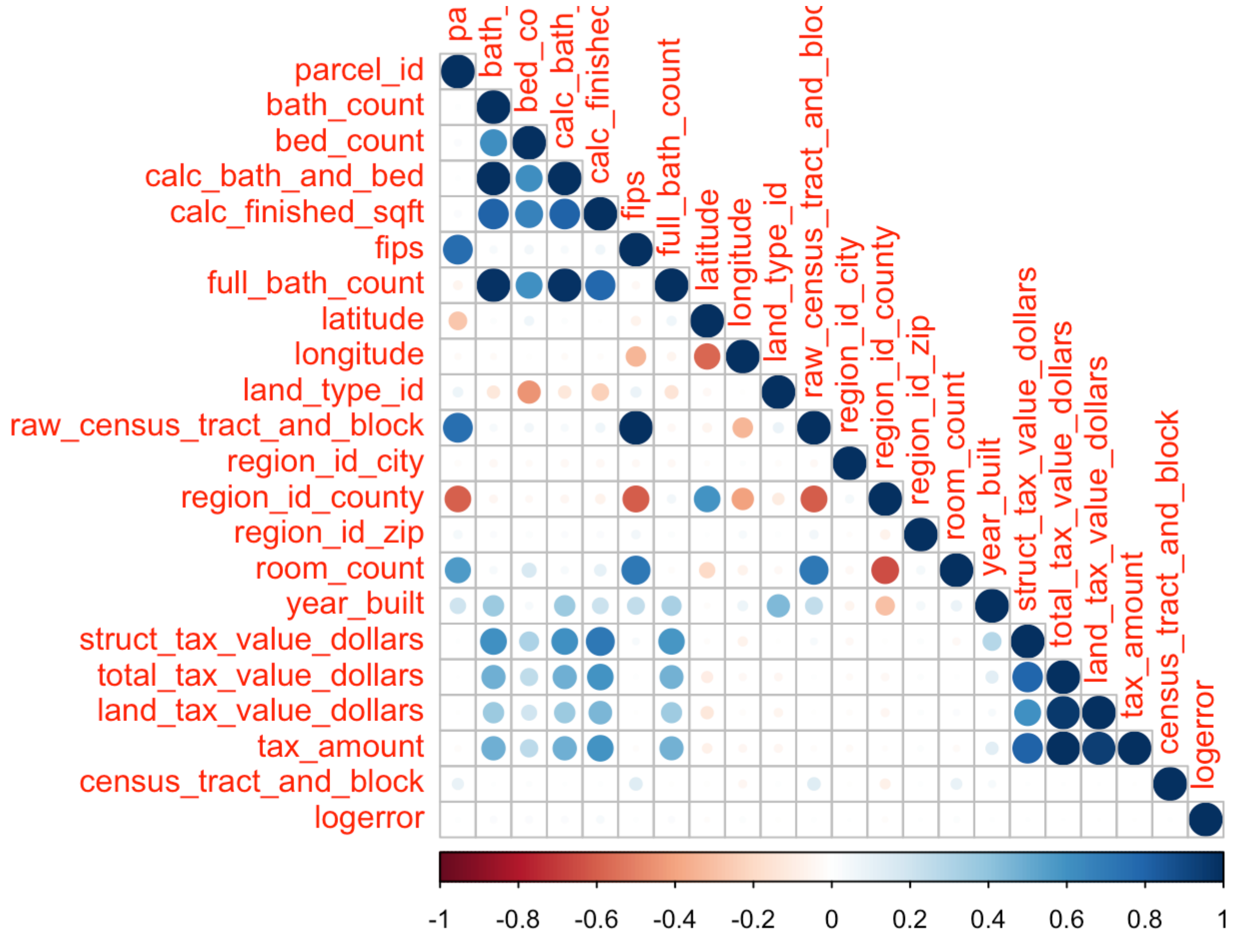
As seen here and discussed earlier, an extremely large portion of the data is centered in the heart of Los Angeles with other data points in north eastern outskirts of Los Angeles in areas like Santa Clarita and Palmdale. The latitude and longitude variables are extremely useful in visualizing information and we will see later how the implementation of these features can be further explored.Until then, let’s understand how the remaining variables correlate to each other.

After perusing the generated data sheet created from prop.train.join, I’ve deduced that *total_tax_value_dollars* is the total price of the property and is comprised of two parts: *struct_tax_value_dollars* and *land_tax_value_dollars*

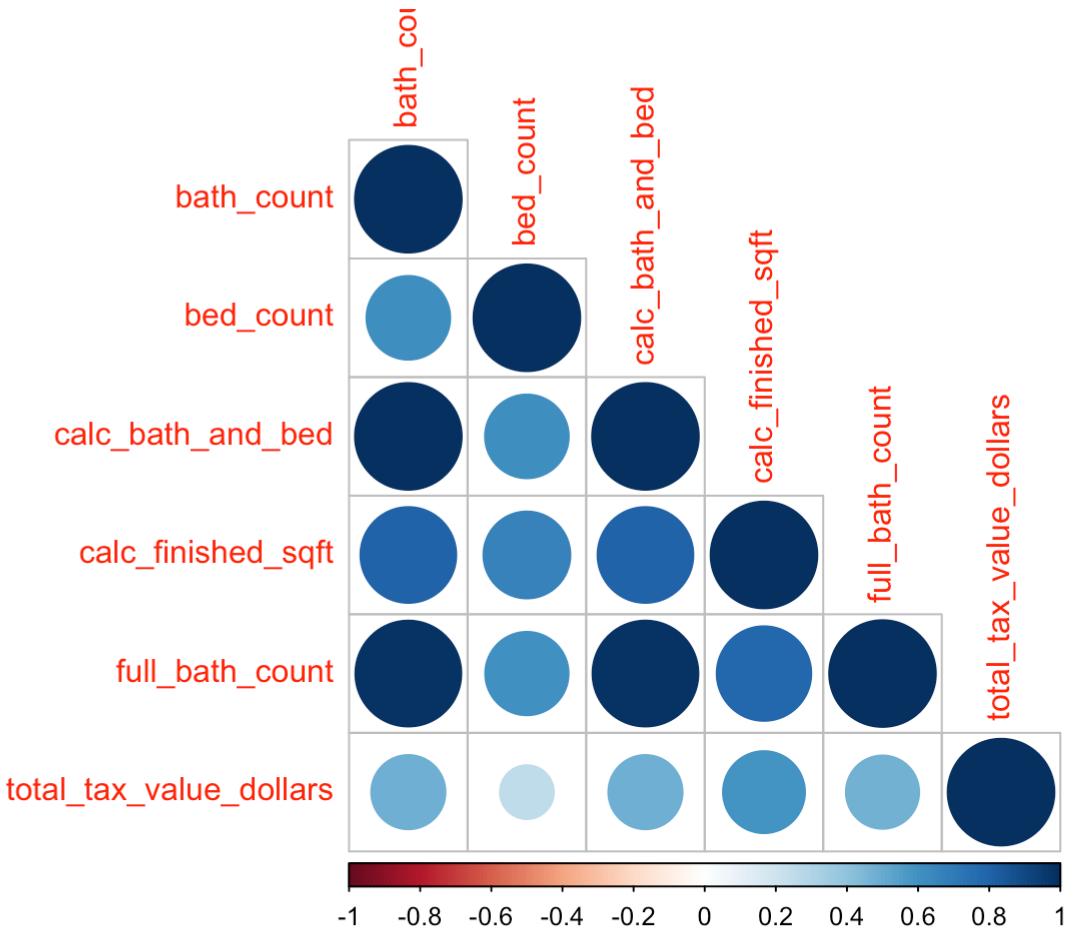
```
prop.train.join %>%
  select(struct_tax_value_dollars, land_tax_value_dollars, total_tax_value_dollars) %>%
  head()
```

```
##   struct_tax_value_dollars land_tax_value_dollars total_tax_value_dollars
## 1                   249655                 374484                  624139
## 2                   253000                 407000                  660000
## 3                   257591                 285332                  542923
## 4                    57968                  20063                   78031
## 5                   167869                 247590                  415459
## 6                    85298                 339116                  424414
```

Knowing this information, we can create a correlation plot that directly shows how each variable correlates to each other



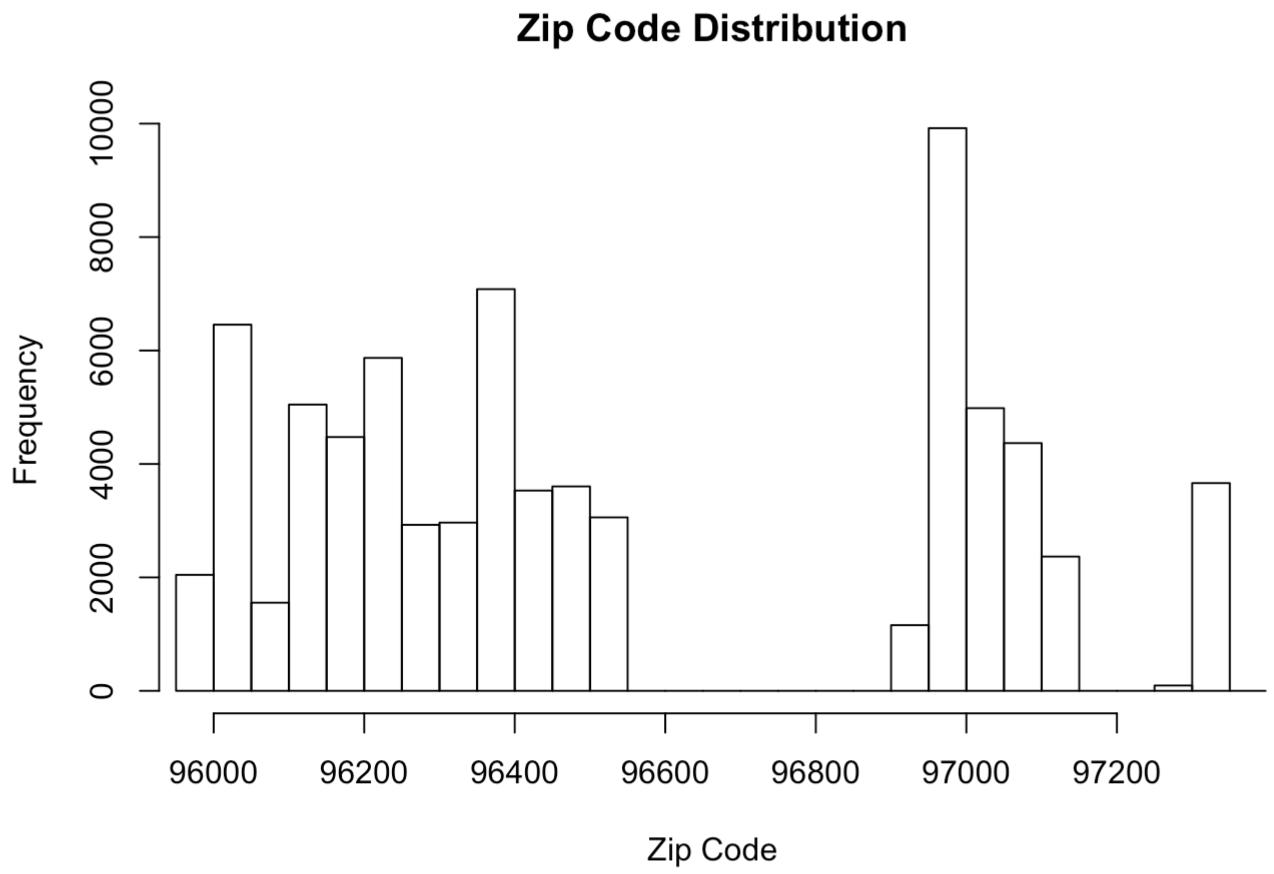
Though this initial correlation plot is not very elegant, it has however revealed that there are certain sets variables that reveal a relatively strong correlation with *total_tax_value_dollars*. We can narrow the features down with much smaller correlation plots



But if you notice in the first correlation plot, features such as *city, zipcode,* and *yearbuilt* showed little to correlation with *total_tax_value_dollars* which is strange because those seem to be primary features that need to be considered in the price along with bath and bed count.

Since the data…tbc

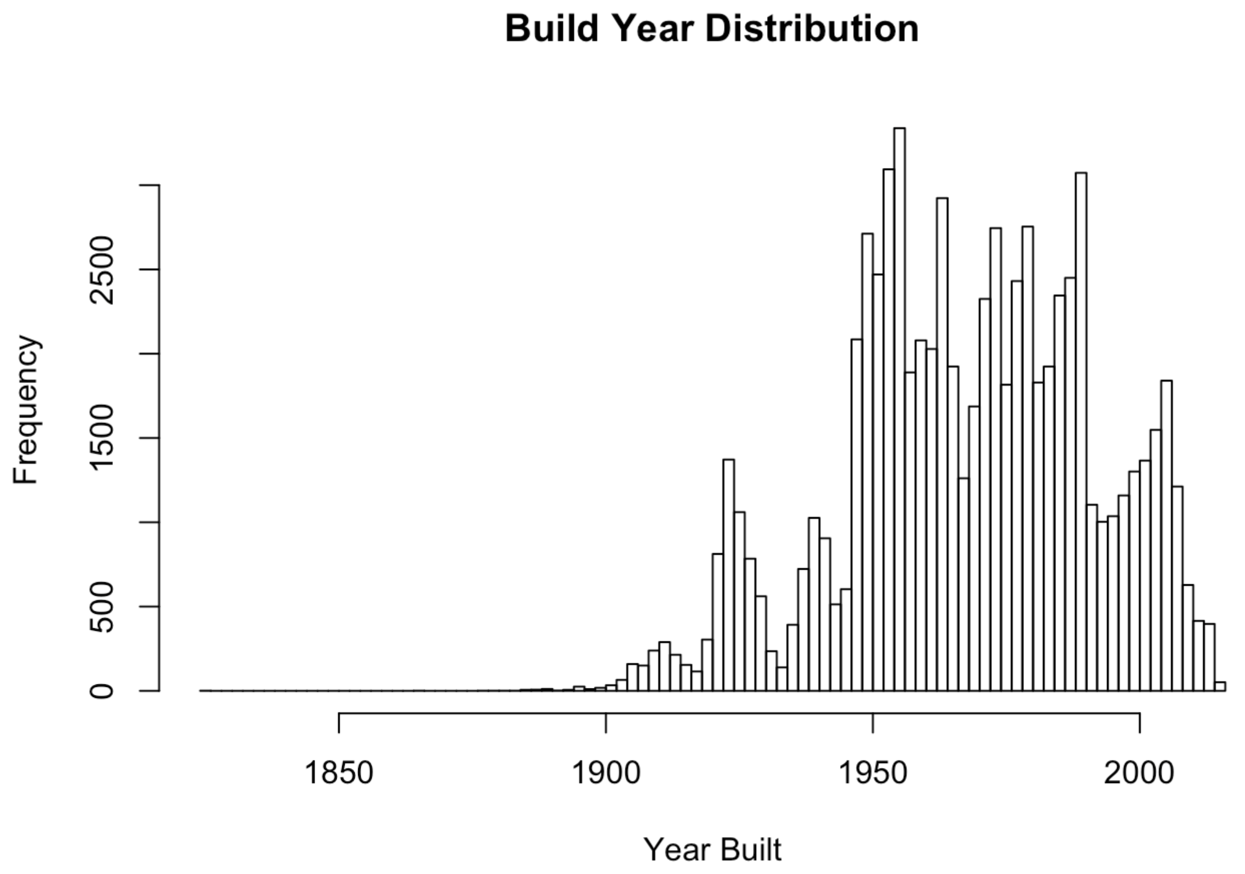
```
# Zip Code Distribution Histogram
minZip <- min(prop.train.join$region_id_zip, na.rm = TRUE)
maxZip <- max(prop.train.join$region_id_zip, na.rm = TRUE)
zipDist <- hist(prop.train.join$region_id_zip,
                main =  "Zip Code Distribution",
                breaks = "FD",
                xlab = "Zip Code",
                xlim = range(minZip, 97344))
```



```
summary(zipDist)
```

```
##           Length Class  Mode     
## breaks   6076   -none- numeric  
## counts   6075   -none- numeric  
## density  6075   -none- numeric  
## mids     6075   -none- numeric  
## xname       1   -none- character
## equidist    1   -none- logical  
```

```
# Build Year Distribution Histogram
minYr <- min(prop.train.join$year_built, na.rm = TRUE)
maxYr <- max(prop.train.join$year_built, na.rm = TRUE)
yrDist <- hist(prop.train.join$year_built,
                main =  "Build Year Distribution",
                breaks = "FD",
                xlab = "Year Built",
                xlim = range(minYr, maxYr))
```



```
summary(yrDist)
```

```
##          Length Class  Mode
## breaks   97     -none- numeric
## counts   96     -none- numeric
## density  96     -none- numeric
## mids     96     -none- numeric
## xname     1     -none- character
## equidist  1     -none- logical
```