Housing Price Clusters and Prediction

King County, WA

By Group G_SM2R2 Consulting Group

Susiette, Michael, Marco, Ron, Ray

Table of Contents

Introduction:

The stakes are high in the home building industry. Time your project correctly and millions can be made; make a couple of bad business decisions and you'll be out of business. With improved data, decisions can be made based on facts and data. In the past, home builders relied on their gut and focus groups. SM2R2 was able to obtain a dataset containing house sale prices for the surrounding Seattle area from a contact at the local real estate board whom we've partnered with. It includes homes sold between May 2014 and May 2015.

Background:

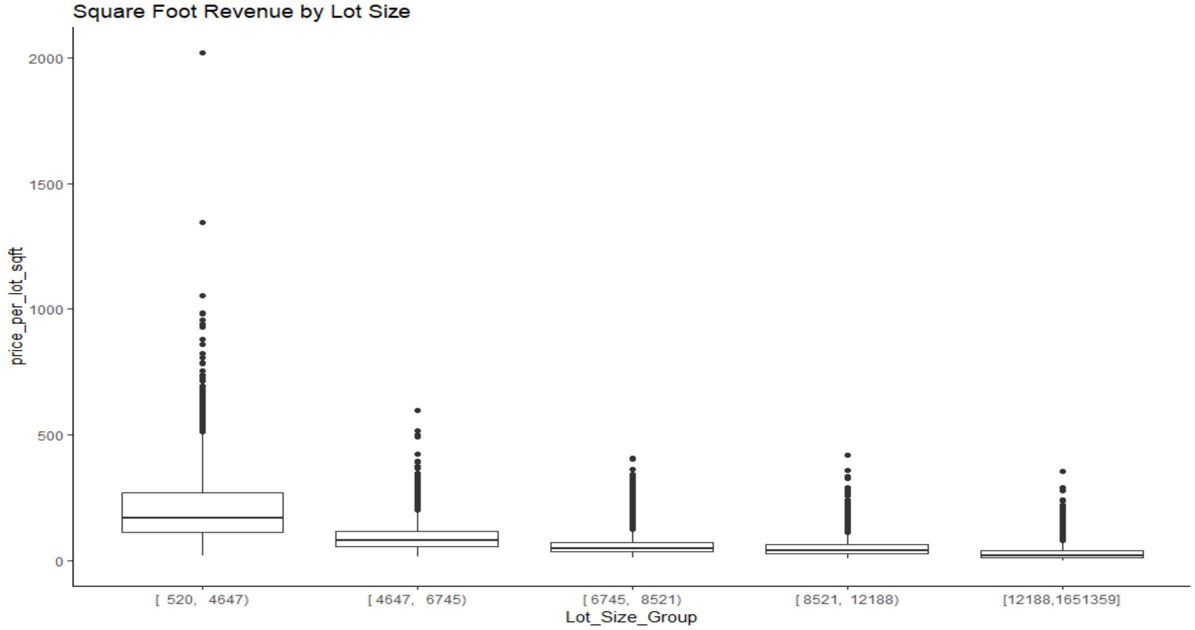
SM2R2 is a home builder in the greater Seattle area. In the next few weeks, the executive staff will be meeting with an urban planner and architects to discuss ideas for their next major build which is a 25-acre plot of land (zip code 90106) which has recently been rezoned as residential. This upcoming project will have major implications for the company as it will be their largest project to date; the stakes are extremely high. If homes are built that don't satisfy a fickle consumer they will sit on the market for an extended period and may eventually need to be discounted to sell, cutting into profit margins. In addition, the banks have provided financing, but only for a 1-year term. If the project cannot sell 90% of the homes within the year, the future of the company will be in jeopardy. The bank has requested to review the proposal including sales forecasts and estimates.

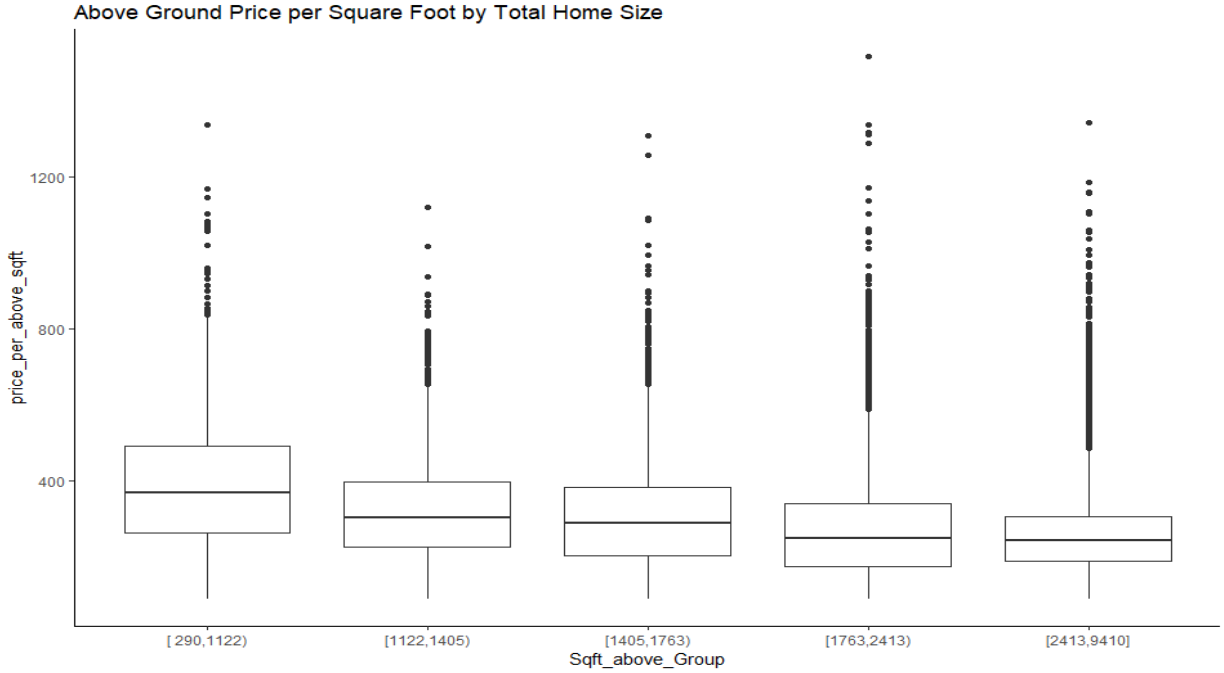
Objective:

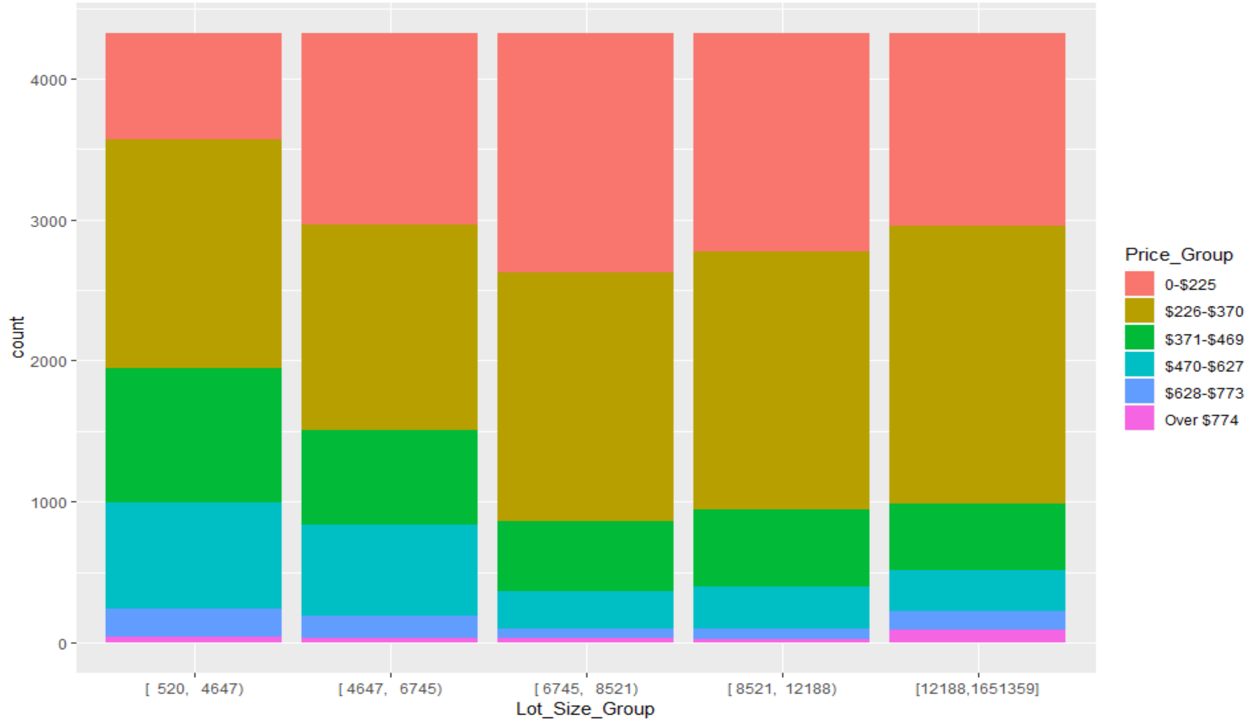
Determine what are the factors that will influence buyers the most. Within the market, different amenities appeal to different consumer groups. Some consumers are first time buyers, they prioritize space for growing families, others are empty nesters looking to downsize. The amenities which various buyers prefer and optimize the designs to home buyers which appeals to the largest number of buyers. While everyone wants their money to go further, if too many amenities are added, it will unnecessarily increase costs. The overriding objective of this endeavour is to determine the total sales revenue to confirm the viability of the project. SM2R2 will also optimize the revenue per square foot of each home, as well as, optimizing the lot size.

Preliminary Analysis:

Of the 25 acres, approximately 15% must be reserved for municipal infrastructure. The remaining 21.25 acres can be used for housing development (~925,650 sq ft). Preliminary analysis of the dataset indicates the median lot size is 7,618 sq ft which would accommodate 121 average sized lots. However, the card below indicates a more profitable use of the available land would be to use a higher mix of smaller lot sizes that are under 5000 square feet. It also shows that some people paid exorbitant amounts for their lots on a per foot basis.







The above charts indicate the most expensive homes per square foot are smaller homes on smaller lots. It appears the public is willing to pay more per square foot of lot on a smaller home than on larger lots. The highest portion of low revenue is between 6745 sqft and 8521 sqft - our original median estimate.

Data Understanding:

The dataset provided contains 21,613 records contain sales data over the past year. While the data appeared complete and our analysis indicated there were no missing values, there were some odd data points which were data entry errors or other outliers.

Field Name	Description	Data Type / Values
id	Unique ID for each home sold	Number
date	Date of the home sale	Factor
price	Price of each home sold	Number
bedrooms	Number of bedrooms	Integer
bathrooms	Number of bathrooms, where .5 accounts for a room with a toilet but no shower	Number
sqft_living	Square footage of the apartments interior living space	Integer
sqft_lot	Square footage of the land space	Integer
floors	Number of floors	Number

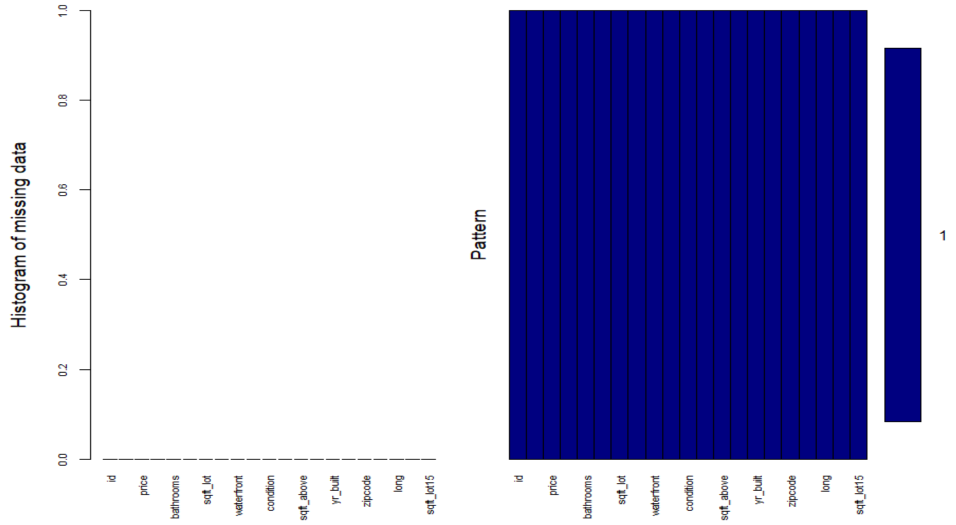
waterfront	A dummy variable for whether the apartment was overlooking the waterfront or not 1's represent a waterfront property, 0's represent a non-waterfront property	Integer
view	An index from 0 to 4 of how good the view of the property was, 0 - lowest, 4 - highest	Integer
condition	An index from 1 to 5 on the condition of the house, 1 - lowest, 4 - highest	Integer
grade	An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high-quality level of construction and design.	Integer
sqft_above	The square footage of the interior housing space that is above ground level	Integer
sqft_basement	The square footage of the interior housing space that is below ground level	Integer
yr_built	The year the house was initially built	Integer
yr_renovated	The year of the house's last renovation	Integer
zipcode	What zip code area the house is in	Integer
lat	Latitude	Number
long	Longitude	Number
sqft_living15	The square footage of interior housing living space for the nearest 15 neighbours	Integer
sqft_lot15	The square footage of the land lots of the nearest 15 neighbours	Integer

Load data into R:

```
housedf <- read.csv("C:\\....\\kc_house_data.csv")
zipdemog <- read.csv("C:\\....\\Zipcode.csv")
```

Determine if there are any missing values

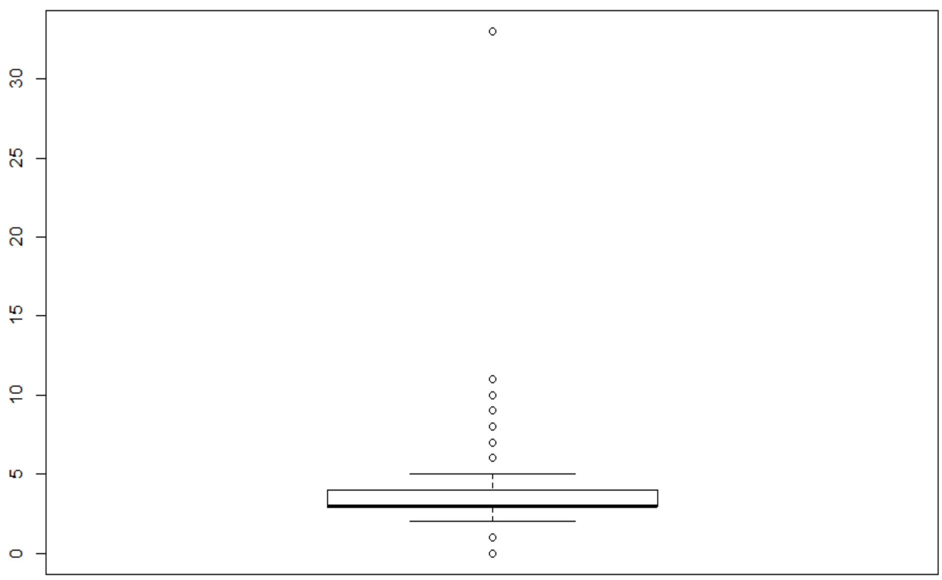
```
aggr_plot <- aggr(housedf, col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE,
labels=names(data), cex.axis=.7, gap=3, ylab=c("Histogram of missing data","Pattern"))
```



(No missing values were detected)

Change date to the correct format:
Before:
housedf$date
 [1] 20141013T000000 20141209T000000 20150225T000000 20141209T000000
20150218T000000
After:
housedf$date<-as.Date(housedf$date, "%Y%m%dT000000")
housedf$date
 [1] "2014-10-13" "2014-12-09" "2015-02-25" "2014-12-09" "2015-02-18" "2014-05-12" "2014-06-27"

Found an outlier in “Bedrooms”
boxplot(housedf$bedrooms)



make the house with 33 bedrooms into 3. (Was probably a typo.)
housedf[15871,4] <- 3

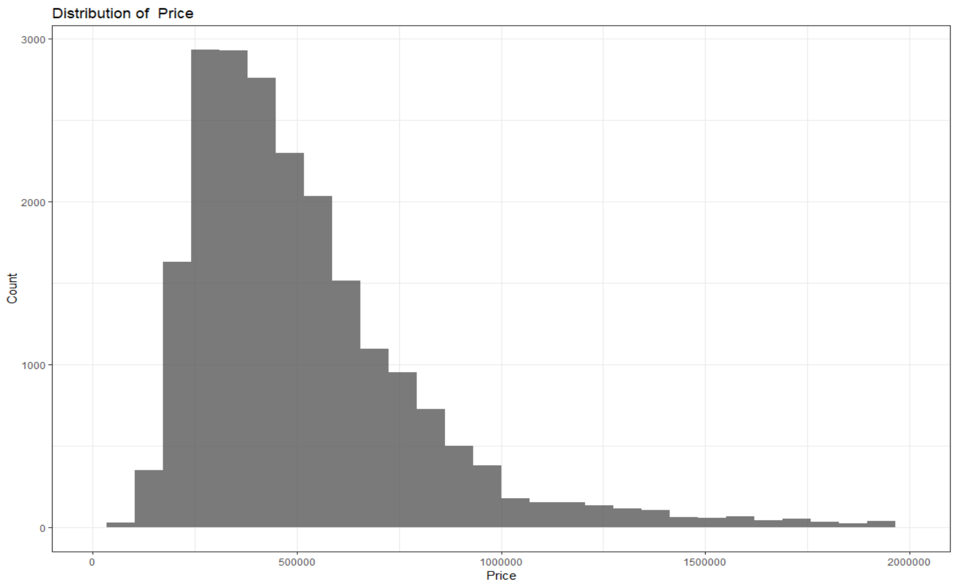
Use max date from built and reno to make relevant date appear on one column. (Assumption is that a renovated house has similar value to a new house.)

housedf <- transform(housedf, built_reno_date = pmax(yr_built, yr_renovated))

Remove extra columns no longer needed

housedf$id <- NULL

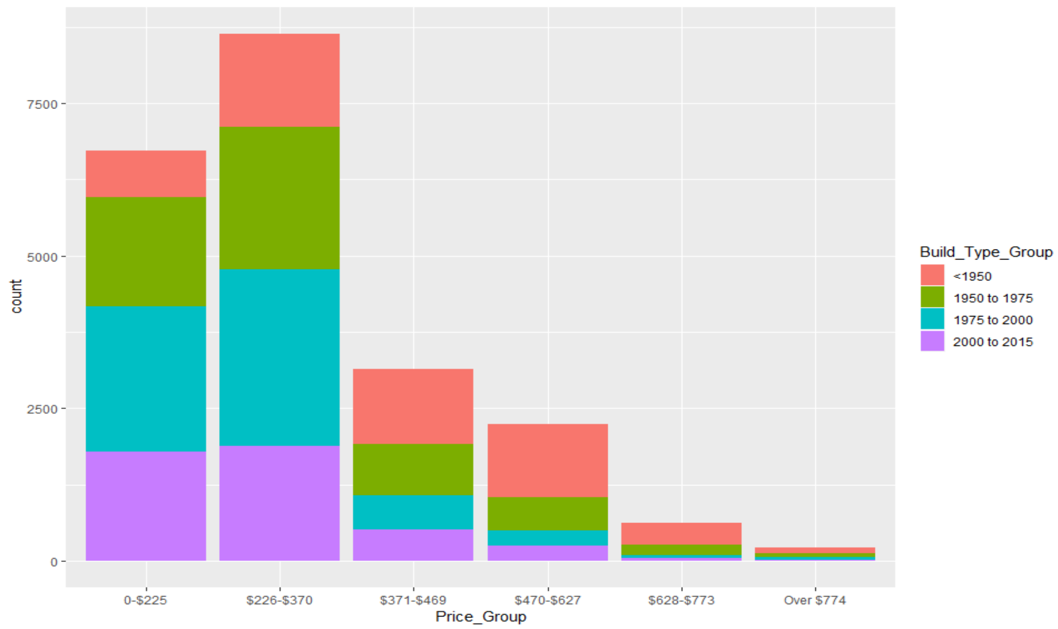
Now let's take a closer look at the "Price":

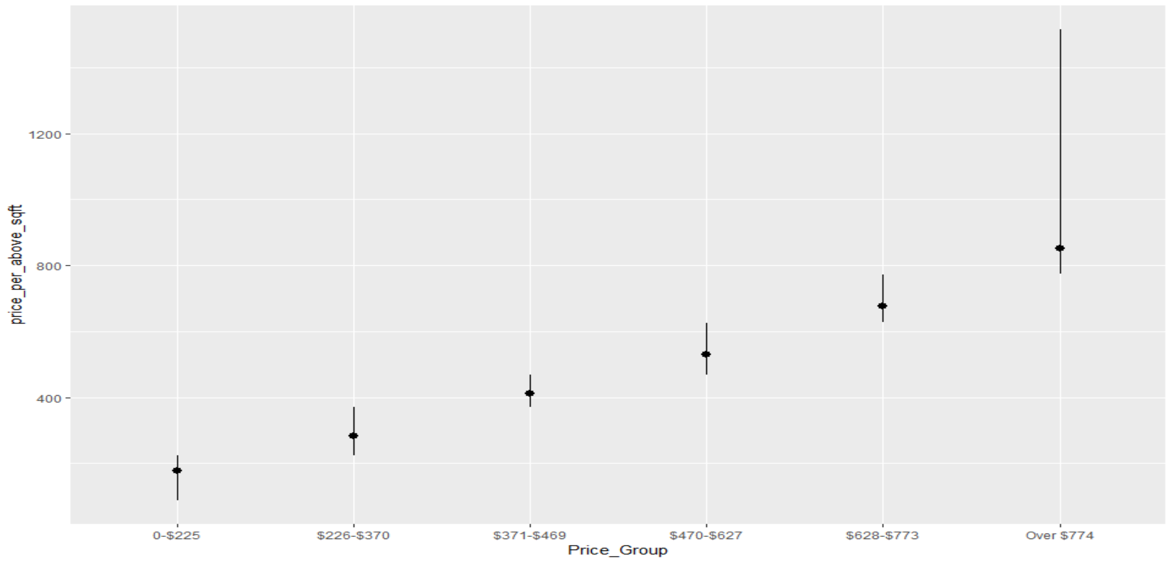


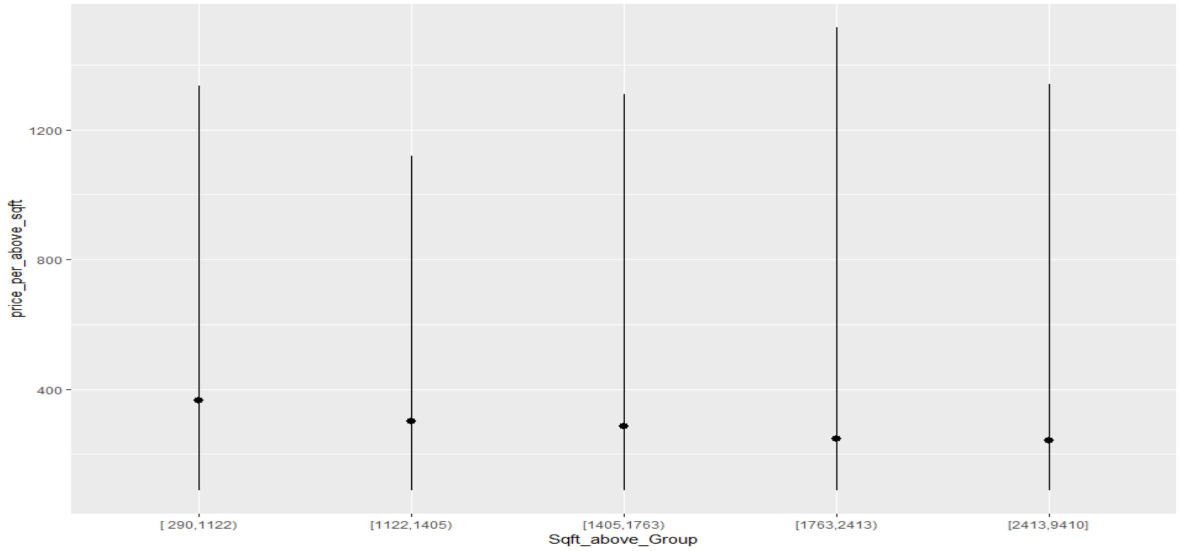
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
75000	321950	450000	540088	645000	7700000

As we can see, it shows a right-skewed distribution, and most of the houses are priced under $550,000. However, there is a very wide range of selling prices at the high end of the market.

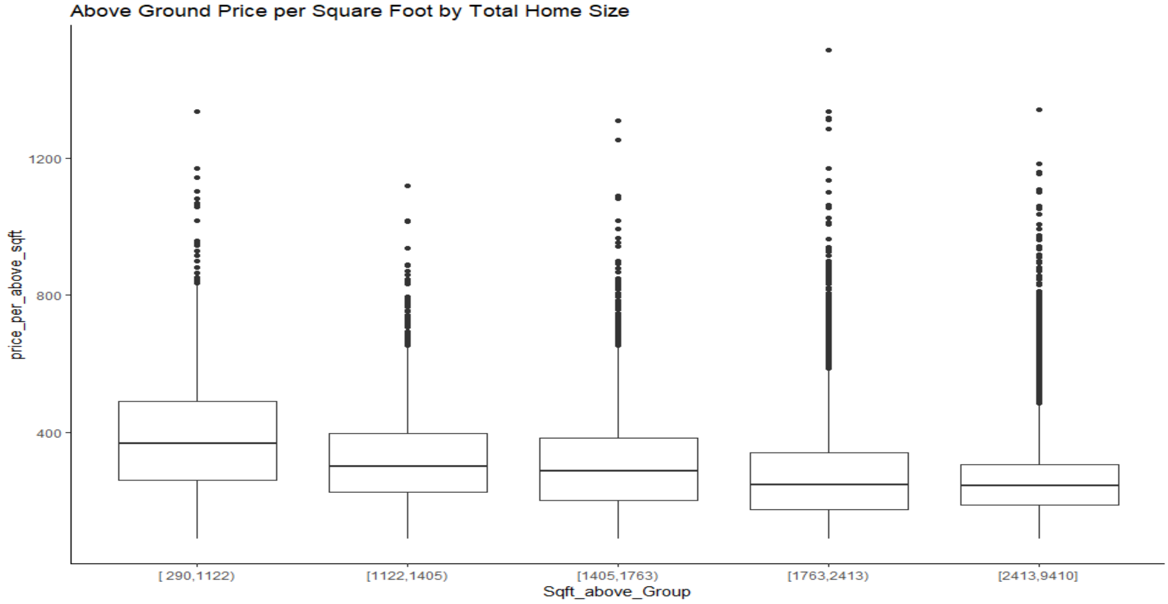
What about Price vs. other variables:

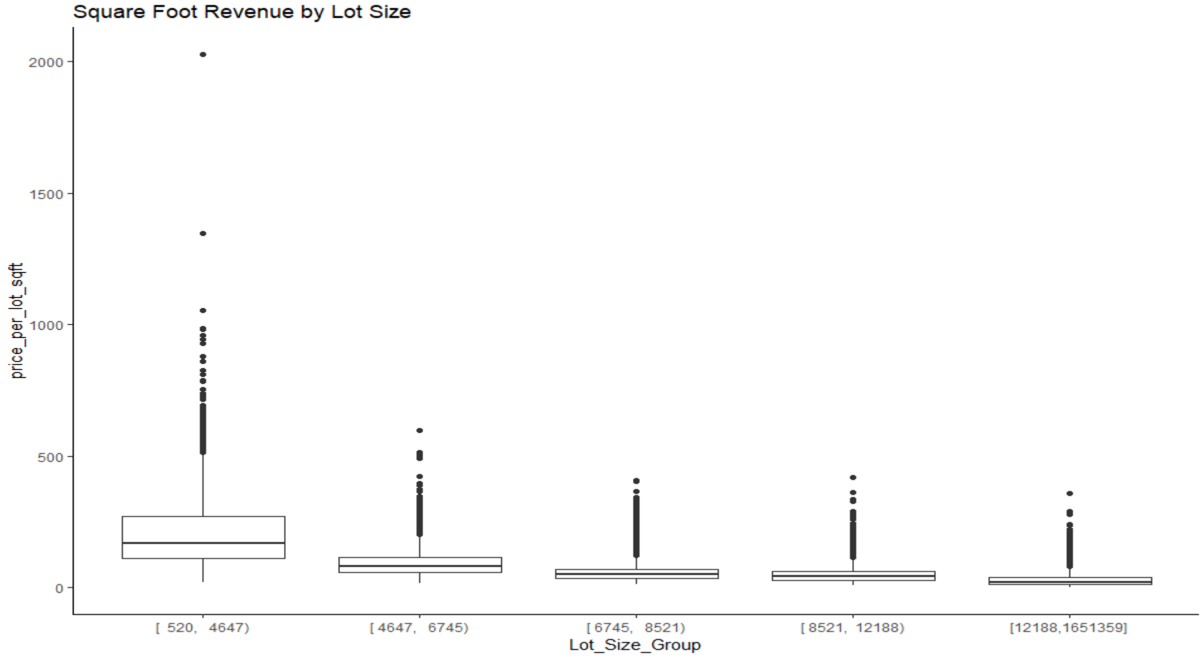




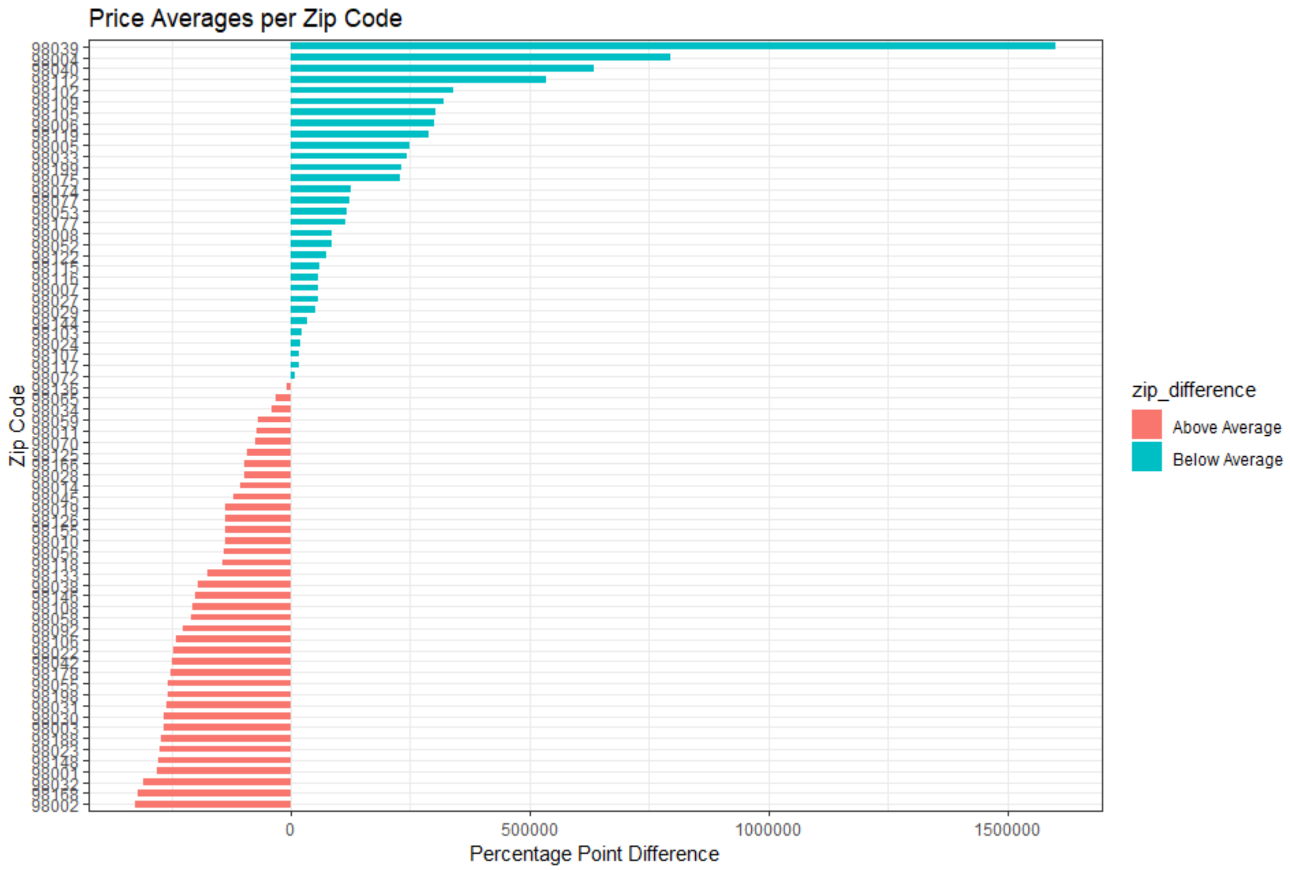


Above Ground Price per Square Foot by Total Home Size

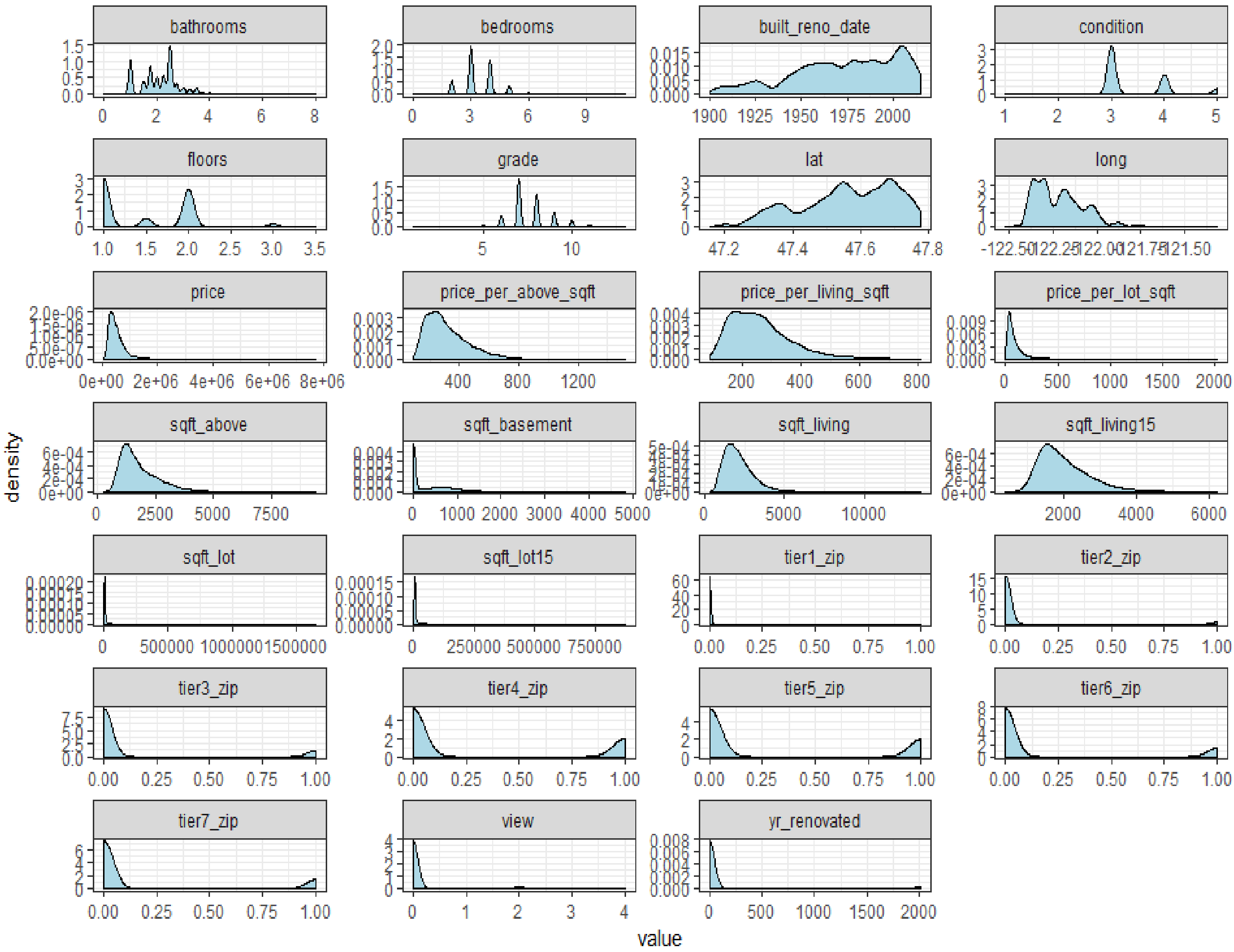




The relationship between Price and Zipcode:



The plot above represents the average price per house in the various zip codes



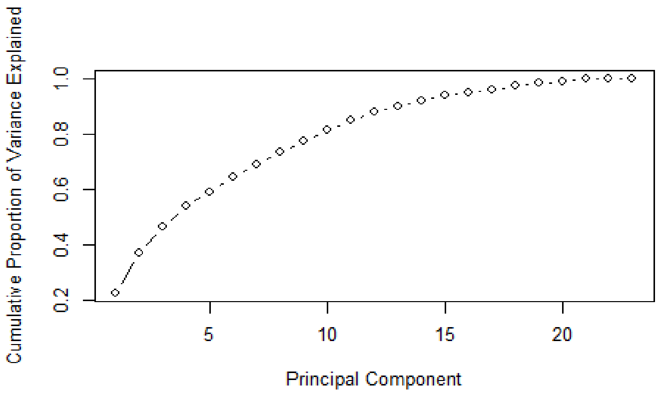
The plot above represents the various zip codes broken down into different tiers to prepare the data for modelling.

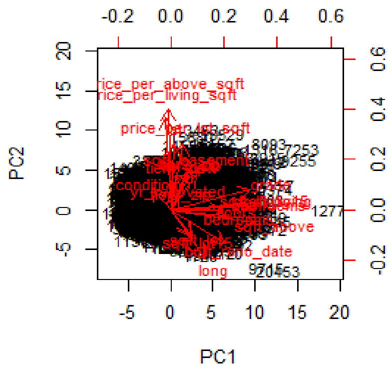
DATA PREPARATION

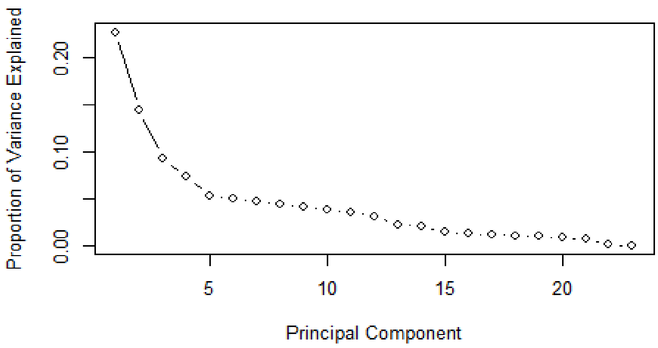
Principal Component Analysis (PCA)

In order to reduce redundancy in the data set, SM2R2 utilized Principal Component Analysis to modify the dataset prior to modelling. The principal components were scaled and centred prior to processing to provide equal weighting to all variables. The dataset was reduced to the first 10 principal components as these accounted for 81% of the original dataset as shown

```
> summary(prin_comp)
Importance of components:
                          PC1    PC2     PC3     PC4     PC5    PC6     PC7     PC8     PC9    PC10    PC11    PC12    PC13
Standard deviation     2.2868 1.8221 1.46632 1.30143 1.11142 1.0735 1.04224 1.00926 0.96693 0.93467 0.89609 0.85344 0.71979
Proportion of Variance 0.2274 0.1444 0.09348 0.07364 0.05371 0.0501 0.04723 0.04429 0.04065 0.03798 0.03491 0.03167 0.02253
Cumulative Proportion  0.2274 0.3717 0.46521 0.53885 0.59255 0.6427 0.68988 0.73417 0.77482 0.81280 0.84771 0.87938 0.90191
                          PC14    PC15    PC16    PC17    PC18    PC19    PC20    PC21    PC22      PC23
Standard deviation     0.68794 0.59379 0.53793 0.51282 0.50199 0.47979 0.44306 0.40390 0.19042 2.252e-14
Proportion of Variance 0.02058 0.01533 0.01258 0.01143 0.01096 0.01001 0.00853 0.00709 0.00158 0.000e+00
Cumulative Proportion  0.92249 0.93782 0.95040 0.96183 0.97279 0.98280 0.99133 0.99842 1.00000 1.000e+00
```







Cluster Evaluation

We have clustered the data using k-means in order to gain a better understanding of the different areas of Seattle. This will provide us with insights as to what type of dwellings we should build and for whom. It will also help us focus our marketing strategy. The clustering is as follows:

K-Means Cluster Analysis

set.seed(123)
fit <- kmeans(scaled_housedf_num, 5)

get cluster means

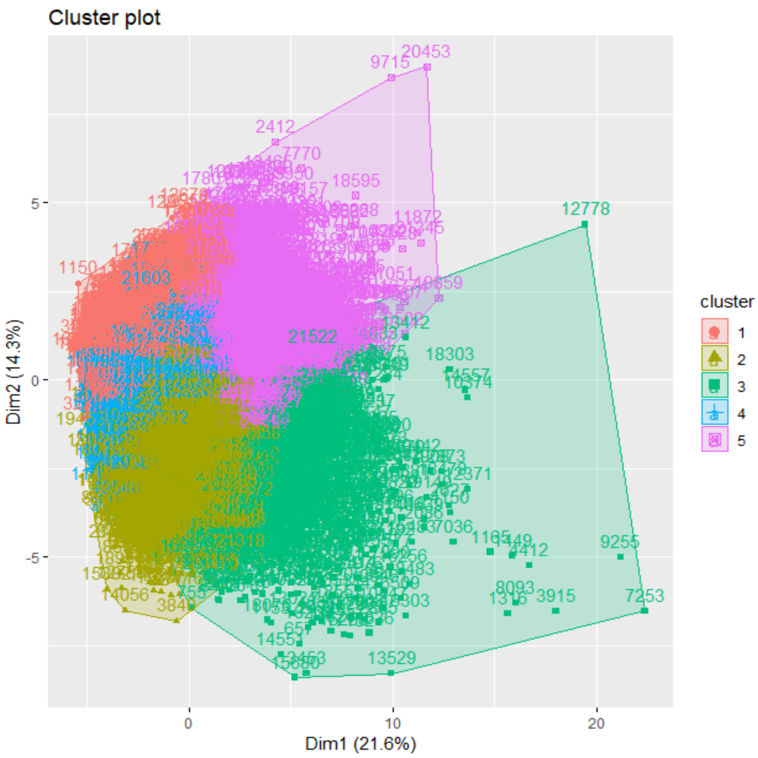
```
clust_means <- round(aggregate(scaled_housedf_num,by=list(fit$cluster),FUN=mean),2)
```

Append cluster assignment

```
housedf_num1<- data.frame(scaled_housedf_num, fit$cluster)
fviz_cluster(fit, data <- scaled_housedf_num)
str(clust_means)

housedf_clus <- cbind(housedf, fit$cluster)
colnames(housedf_clus)[35] <- "cluster"

cluster_means_summary <- round(t(housedf_clus %>%
  keep(is.numeric)%>%
  group_by(cluster) %>%
  summarise_all(mean)),3)
Cluster_means_summary
```



cluster	0.000	1.000
price	510842.193	619579.968
bedrooms	3.395	3.299
bathrooms	2.093	2.175
sqft_living	2068.587	2110.649
sqft_lot	15269.045	14666.434
floors	1.443	1.634
view	0.254	0.181
condition	3.423	3.372
grade	7.582	7.862
sqft_above	1776.710	1820.138
sqft_basement	291.876	290.511
yr_renovated	83.975	85.565
lat	47.529	47.645
long	-122.217	-122.205
sqft_living15	1979.742	2005.064
sqft_lot15	12827.616	12607.656
built_reno_date	1973.723	1972.470
price_per_living_sqft	246.378	312.479
price_per_lot_sqft	73.090	132.119
price_per_above_sqft	294.003	371.701
tier1_zip	0.003	0.000
tier2_zip	0.055	0.000
tier3_zip	0.152	0.000
tier5_zip	0.360	0.000
tier6_zip	0.206	0.000
tier7_zip	0.224	0.000
fit$cluster	2.775	2.911

The clusters seem to indicate that the location and zip code tiers play an important factor in the clustering.

Here is how the clusters look when looking at the different features and how they are linked to the zip code tiers we created and corresponding demographic data:

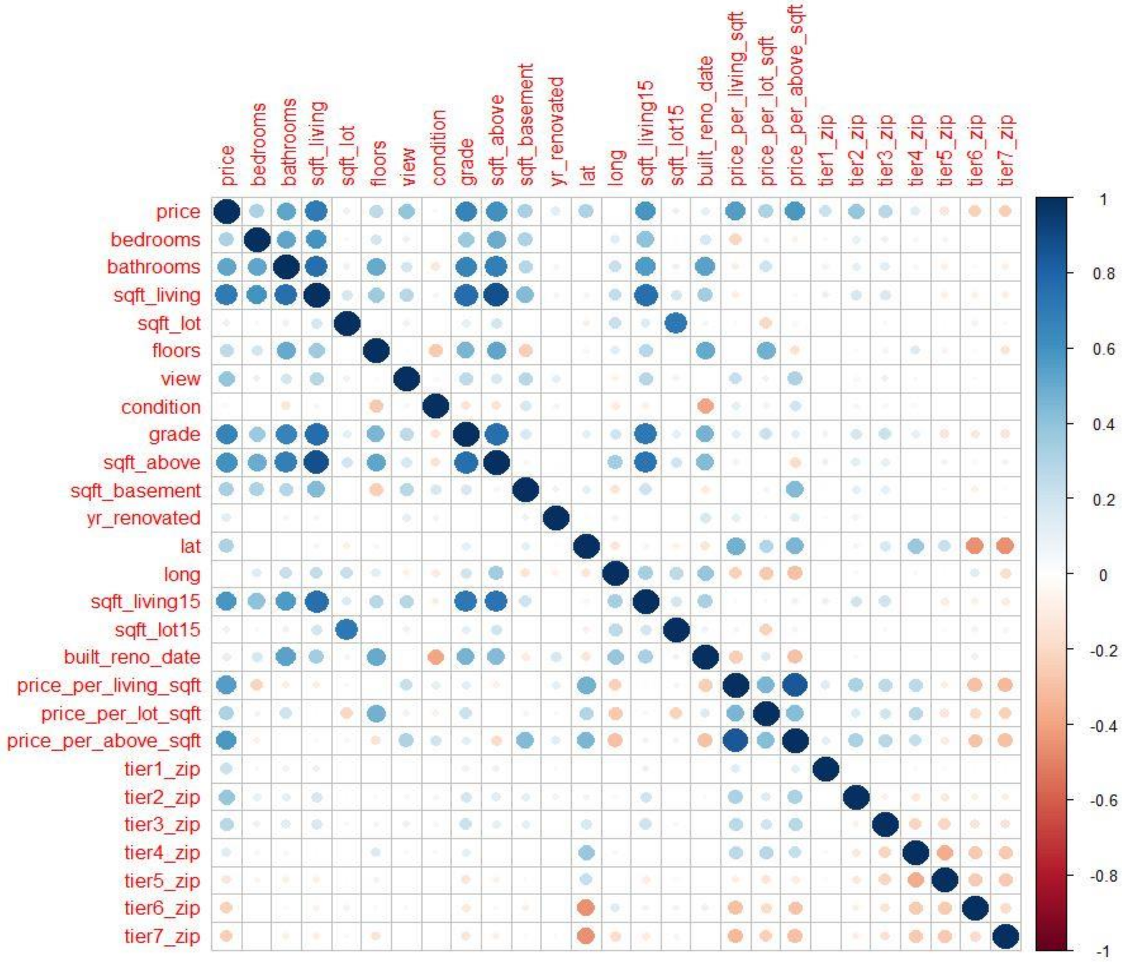
Feature/Cluster	1	2	3	4	5
price	$296,205.45	$554,715.98	$1,393,435.93	$391,566.00	$699,859.29
bedrooms	3.285	3.04	3.954	3.122	3.932
bathrooms	1.9	1.885	2.856	1.761	2.781
sqft_living	1735.82	1674.23	3282.689	1640.556	3061.729
sqft_lot	12672.191	6878.762	12593.266	11664.531	32854.303
floors	1.326	1.477	1.634	1.281	1.904
view	0.102	0.135	1.348	0.154	0.227
condition	3.42	3.495	3.608	3.485	3.145
grade	7.105	7.385	9.147	7.017	8.852
sqft_above	1516.159	1362.461	2505.785	1368.765	2848.274
sqft_basement	219.661	311.769	776.903	271.791	213.456
sqft_living15	1726.321	1699.67	2800.597	1653.409	2738.623
sqft_lot15	10933.949	6440.409	11269.194	11021.608	25319.555
built_reno_date	1973.732	1961.86	1973.137	1964.736	1995.873
price_per_living_sqft	178.195	347.302	428.839	252.79	230.607
price_per_lot_sqft	40.621	154.373	164.054	63.005	72.596
price_per_above_sqft	206.8	423.834	570.086	301.061	250.097
tier1_zip	0	0	0.033	0	0
tier2_zip	0	0.016	0.508	0.001	0
tier3_zip	0	0.233	0.238	0.001	0.175
tier4_zip	0.003	0.722	0.098	0	0.392
tier5_zip	0	0.005	0.062	0.998	0.275
tier6_zip	0.461	0.007	0.012	0	0.108
tier7_zip	0.535	0.018	0.049	0	0.049

Feature/Zipcode Tier	Tier7	Tier6	Tier5	Tier4	Tier3	Tier2	Tier1
Population	30443.29	30316.22	23868.33	27819	26139	23907.33	2971
Median.Age	36.48571	37.45556	39.5	38.0375	35.93333	41.66667	45.5
Households	11347	10967.889	9693.278	11984	11553.222	10856	1062
Persons.Per.Household	2.625	2.736667	2.503889	2.385625	2.182222	2.206667	2.8
Income.Per.Household	62045.36	71584.56	82781.44	101446.44	97685.22	117365.33	182604
Average.House.Value	287978.6	314777.8	400455.6	539125	619133.3	882633.3	1654600
price	340091.9	337851.4	458552	619580	829095.5	1222685.6	2160606.6
bedrooms	3.370308	3.342259	3.293373	3.298985	3.58434	3.804147	4.06
bathrooms	1.958227	2.044937	2.000923	2.174652	2.388172	2.541187	3.2
sqft_living	1887.993	1914.75	1954.494	2110.649	2496.409	2846.139	3800.9
sqft_lot	11066.65	21728.19	16788.86	14666.43	10705.45	10784.7	17403.56
floors	1.316399	1.456448	1.434786	1.63401	1.581424	1.572005	1.56
view	0.2085803	0.1655894	0.2394094	0.1813177	0.4127447	0.4124424	0.44
condition	3.422806	3.375192	3.392336	3.371753	3.498542	3.596774	3.48
grade	7.305109	7.318867	7.385129	7.861517	8.364848	8.700461	9.56
sqft_above	1626.324	1710.496	1690.921	1820.138	2067.463	2309.197	3290.9
sqft_basement	261.6692	204.2536	263.5727	290.5108	428.9467	536.9424	510
sqft_living15	1829.729	1850.533	1872.329	2005.064	2373.117	2625.192	3132.2
sqft_lot15	9496.581	17232.95	14084.224	12607.656	9605.843	10353.409	17291.1
built_reno_date	1972.835	1979.074	1972.715	1972.47	1971.684	1969.132	1981.3

Our property sits in the middle of zip code 90106 which grouped in Tier 6 of our zip code tiers. From the cluster work do we see that cluster 1 has the largest portion of its cluster in Tier 6, which tells us that we are looking at the lower end of the spectrum in terms of home prices. Another cluster that has a significant amount of its cluster in Tier 6 is cluster 5, which has a much higher home price average. This tells us that there is potential for these properties to appeal to a different demographic with a higher income level.
When building the homes we will keep this in mind to ensure the properties are not too small and low quality that they would not appeal to those in cluster 5.
Our targets are definitely on the younger side, the households have the second highest average persons per household and second lowest average income per household.
Sacrificing some of the quality in the finishes for more bedrooms/bathrooms/living square footage may be enough to ensure the proprieties appeal our target audience. We can also offer an upgrade for finishes for those in cluster 5 that have higher incomes and are looking to sacrifice prime location for more square footage.
We can also target these 2 clusters specifically with our marketing efforts to ensure maximum value out of the marketing budget.

Checking correlation between all variables

```
CorrelationResults = cor(housedf_num)
corrplot(CorrelationResults)
```



The above correlation chart only shows a strong correlation between elements that are related and may cause the model to be overfitted.

MODELLING

Models:

1. **XGB Linear:**

model <- train(price~., data=train,trControl=train_control,method="xgbLinear")
print(model)

```
eXtreme Gradient Boosting

15129 samples
    9 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12105, 12102, 12105, 12102, 12102
Resampling results across tuning parameters:

  lambda  alpha  nrounds  RMSE      Rsquared   MAE
  0e+00   0e+00   50      91017.35  0.9407117  51479.28
  0e+00   0e+00  100      90447.30  0.9414051  51098.77
  0e+00   0e+00  150      90381.07  0.9414961  51153.95
...
  1e-01   1e-01   50      90626.94  0.9413589  51059.65
  1e-01   1e-01  100      90150.95  0.9418891  50619.19
  1e-01   1e-01  150      90070.80  0.9419777  50678.59
Tuning parameter 'eta' was held constant at a value of 0.3
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nrounds = 150, lambda =
 0.1, alpha = 0 and eta = 0.3.
> mean(model$results$Rsquared)
[1] 0.9412856
> mean(model$results$RsquaredSD)
[1] 0.007308234
> mean(model$results$RMSE)
[1] 90560.63
```

2. **Random Forest**:

model2 <- train(price~., data=train,trControl=train_control, method="rf")
print(model2)

```
Random Forest

15129 samples
    9 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12102, 12103, 12103, 12104, 12104
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
  2     118638.71  0.9079688  59845.11
  5      95989.43  0.9365774  52117.04
  9      92662.14  0.9396975  52040.61

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 9.
There were 15 warnings (use warnings() to see them)
> mean(model2$results$Rsquared)
[1] 0.9280813
```

```
> mean(model2$results$RsquaredSD)
[1] 0.01412225
> mean(model2$results$RMSE)
[1] 102430.1
```

3. GENERALIZED LINEAR REGRESSION

```
set.seed(123)
model3 <- train(price~., data=train,trControl=train_control, method="glm")
print(model3)
Generalized Linear Model

15129 samples
   9 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12103, 12105, 12103, 12102, 12103
Resampling results:

  RMSE      Rsquared  MAE
  156927.3  0.822703  88225.11
```

4. XGB Tree

```
model4 <- train(price~., data=train,trControl=train_control, method="xgbTree")
print(model4)
eXtreme Gradient Boosting

15129 samples
   9 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12103, 12102, 12104, 12104, 12103
Resampling results across tuning parameters:

  eta  max_depth  colsample_bytree  subsample  nrounds  RMSE
  0.3  1          0.6               0.50        50      157330.83
  0.3  1          0.6               0.50       100      152041.97
…..
  0.4  3          0.8               1.00       100       93880.25
  0.4  3          0.8               1.00       150       91688.01
  Rsquared   MAE
  0.8213293  89496.64
  0.8326911  85291.64
  …
  0.9281506  59227.97
  0.9368500  54724.02
  0.9397781  53008.92

Tuning parameter 'gamma' was held constant at a value of 0

Tuning parameter 'min_child_weight' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nrounds = 150, max_depth =
 3, eta = 0.4, gamma = 0, colsample_bytree = 0.8, min_child_weight =
 1 and subsample = 1.
> mean(model4$results$Rsquared)
[1] 0.8912003
> mean(model4$results$RsquaredSD)
```

```
[1] 0.01030337
> mean(model3$results$RMSE)
[1] 156927.3
```

Model Performance based on Test Dataset (70 train / 30 test)			
Model	**R-squared**	**R-squared SD**	**RMSE**
XGBLinear	**0.9412856**	**0.007308234**	**90,561**
Random Forest	**0.9280813**	**0.014122250**	**102,430**
XGBTree	**0.9020742**	**0.007925838**	**115,223**
GLM	**0.8236274**	**-**	**156,529**

Model Evaluation

XGBLinear, Random Forest, XGBTree, and GLM models were compared to determine if one was more effective than the other. While all models performed well based on a 70/30 train test split. The XGB Linear model had a lower RMSE error rate and a slightly better R-squared indicator. After the initial model was developed, additional features were added to the original dataset. The simple addition of square foot analysis improved the models' performance by over 10%. While the RMSE was still higher than desired, it was due to home sales at the higher end of the marketplace which skewed the results. If this becomes an issue, the data could be reduced to include only the target market for the project (ex under $1M).
In terms of performance, both the XGBLinear and XGBTree were quick to provide results in under a minute. The Random Forest model, on the other hand, took over 30 minutes to run, and the results were slightly inferior.

Conclusion

These preliminary results demonstrate that this data can be used to predict house sale prices in Seattle to determine the project's viability. Future models enhancements could include: economic indicators such as interest and employment rates which impact pricing would further enhance the model. Once the meeting with the planners and architects conclude the team at SM2R2 will be able to predict the expected sales revenue of this project to ensure it’s worth pursuing. With updates to the dataset, we anticipate following the trends in the house design and construction that were delineated in our Cluster Analysis on page 15.

Appendix (Codes):

```
---
  title: "Housing price clusters and prediction"
author: "SM2R2 Consulting Group"
date: "31 December 2018"
output:
  word_document:
  keep_md: yes
---

library(tidyverse)
library(sqldf)
library(gridExtra) #for plotting
library(boot) #For diognastic plots
library(car) # for avplots
library(ggrepel) #For plotting
library(scales) #for changing decimals to be displayed as percentages
library(naniar) #For missing values plot
library(stringr) #For strings
library(timeDate)
library(lubridate)
library(dplyr)
library(tidyr)
library(data.table)
library(dbscan)
library(data.table)
library(zoo)
library(factoextra)
library(clue)
library(cluster)
library(tsne)
library(fpc)
library(ClustOfVar)
library(PCAmixdata)
library(klaR)
library(ggfortify)
library(maps)
library(ggplot2)
library(stringr)
library(DT)
library(leaflet)
library(corrplot)
library(psych)
library(randomForest)
library(hydroGOF)
library(e1071)
library(gbm)
library(caret)
library(VIM)
#housedf <- read.csv("C:/Users/774712616/Desktop/Data Course Semester 2/final lab/housedf.csv")
housedf <- read.csv("C:/Users/MKAlbini/Desktop/York Data Class/2 trimester/housedf.csv")
zipdemog <- read.csv("C:/Users/MKAlbini/Desktop/York Data Class/2 trimester/zipdemog.csv")
#zipdemog <- read.csv("C:/Users/774712616/Desktop/Data Course Semester 2/final lab/zipdemog.csv")


str(housedf)
colnames(housedf)

# Determine if there are any missing values
aggr_plot <- aggr(housedf, col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE,
labels=names(data), cex.axis=.7, gap=3, ylab=c("Histogram of missing data","Pattern"))
# no missing values were detected
```

```
# fix date field
housedf$date<-as.Date(housedf$date, "%Y%m%dT000000")
str(housedf)

#fix formats
housedf$zipcode <- as.factor(housedf$zipcode)
housedf$waterfront <- as.factor(housedf$waterfront)

# make the house with 33 bedrooms into 3. Was probably a typo.
housedf[15871,4] <- 3
summary(housedf)

# use max date from built and reno to make relevant date appear on one column.
#Assumption is that a renovated house has similar value to the new house.

housedf <- transform(housedf, built_reno_date = pmax(yr_built, yr_renovated))

#remove extra columns no longer needed
housedf$id <- NULL

#calculate square foot values
housedf$price_per_living_sqft <- housedf$price / housedf$sqft_living
housedf$price_per_lot_sqft <- housedf$price / housedf$sqft_lot
housedf$price_per_above_sqft <- housedf$price / housedf$sqft_above
housedf$FinishedBasement <- ifelse(housedf$sqft_basement>0,"Yes","No")

# determine breaks for binning
library(rpart)
temp1 <- rpart(housedf$price~housedf$price_per_above_sqft)
plot(temp1)
text(temp1)

temp2 <- rpart(housedf$price_per_lot_sqft~housedf$sqft_lot)
plot(temp2)
text(temp2)

library(Hmisc)
housedf <- housedf%>%
  mutate (
    Price_Group = cut(housedf$price_per_above_sqft, breaks = c(0, 225, 370, 469, 627, 773, 5000) ,
labels = c("0-$225","$226-$370","$371-$469", "$470-$627","$628-$773", "Over $774")),
    Lot_Size_Group = cut2(housedf$sqft_lot,g=5),
    Sqft_above_Group = cut2(housedf$sqft_above,g=5),
    Build_Type_Group = cut(housedf$yr_built,breaks = c(0,1950,1975,2000,2015),labels =
c("<1950","1950 to 1975","1975 to 2000","2000 to 2015"))
  )

housedf$GradeGroup <- as.factor(housedf$grade)
housedf$yr_built <- as.Date(housedf$yr_built)

housedf$Home_Age <- Sys.Date()- housedf$yr_built
housedf$Home_Age <- housedf$Home_Age / 365

str(housedf)
summary(housedf)

# Plotting for data analysis
ggplot(data = housedf) + stat_count(mapping = aes(x = Price_Group, fill = Build_Type_Group))

ggplot(data = housedf) + stat_summary(mapping = aes(x = Price_Group, y = price_per_above_sqft),
                                      fun.ymin = min,
```

```
                              fun.ymax = max,
                              fun.y = median)

ggplot(data = housedf) + stat_summary(mapping = aes(x = Sqft_above_Group, y =
price_per_above_sqft),
                              fun.ymin = min,
                              fun.ymax = max,
                              fun.y = median)

ggplot(data = housedf) +
  geom_bar(mapping = aes(x = Lot_Size_Group, fill = Price_Group))

ggplot(data = housedf) +
  geom_bar(mapping = aes(x = Lot_Size_Group, fill = GradeGroup), position = "dodge")

ggplot (data = housedf, mapping = aes(x = Sqft_above_Group, y = price_per_above_sqft)) +
geom_boxplot() + labs(title = "Above Ground Price per Square Foot by Total Home Size") +
theme_classic()
ggplot (data = housedf, mapping = aes(x = Lot_Size_Group, y = price_per_lot_sqft)) +
geom_boxplot() + labs(title = "Square Foot Revenue by Lot Size") + theme_classic()


# Price averages with Zipcodes
housedf %>%
  group_by(zipcode) %>%
  summarise(count = n(), zip_ave = mean(price)) %>%
  mutate(zip_diff = zip_ave - mean(zip_ave)) %>%
  mutate(zip_difference = ifelse(zip_diff > 0, 'Below Average' , 'Above Average')) %>%
  arrange((zip_diff)) %>%
  #filter(count > 20) %>%
  ggplot(aes( x = factor(zipcode, levels =zipcode), y = zip_diff)) +
  geom_bar(stat = 'identity', aes(fill=zip_difference), width = .6) +
  coord_flip() +
  labs(x = 'Zip Code', y = 'Percentage Point Difference', title = 'Price Averages per Zip Code') +
  theme_bw()

# Break up the zip codes into tiers for models
housedf$tier1_zip <-ifelse(housedf$zipcode %in% c(98039),1,0)
housedf$tier2_zip <-ifelse(housedf$zipcode %in% c(98004,98040,98112),1,0)
housedf$tier3_zip <-ifelse(housedf$zipcode %in% c(98102,98109,98105,98006,98119, 98005,
98033, 98199, 98075),1,0)
housedf$tier4_zip <-ifelse(housedf$zipcode %in%
c(98074,98077,98053,9817798008,98052,98122,98115,98116,98007,98027,98029, 98144,
98103,98024,98107,98117,98072),1,0)
housedf$tier5_zip <-ifelse(housedf$zipcode %in%
c(98136,98065,98034,98059,98011,98070,98125,98166,98028,98014,98045,98019,98126,98155,98
010,98056,98118,98133),1,0)
housedf$tier6_zip <-ifelse(housedf$zipcode %in%
c(98038,98146,98108,98058,98092,98106,98022,98042,98178),1,0)
housedf$tier7_zip <- 1 - housedf$tier1_zip - housedf$tier2_zip - housedf$tier3_zip - housedf$tier4_zip
- housedf$tier5_zip - housedf$tier6_zip

str(housedf)
# Explore Numeric columns

housedf %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
  facet_wrap(~ key, scales = "free", ncol = 4) +
  geom_density(fill= 'lightblue') +
  theme_bw()
```

```
# Explore non-numeric columns

housedf %>%
  keep(is.factor) %>%
  gather() %>%
  ggplot(aes(value)) +
  facet_wrap(~ key, scales = "free") +
  geom_bar(fill = 'blue') +
  theme_bw()

# get numeric values
housedf_num <- housedf[,sapply(housedf, is.numeric)]

str(housedf_num)
#scale the variables
scaled_housedf_num <- as.data.frame(scale(housedf_num))
str(scaled_housedf_num)

d <- dist(scaled_housedf_num,method = "euclidean") #distance matrix
h_clust <- hclust(d, method = "ward.D") #clustering
plot(h_clust) #dendrogram

rect.hclust(h_clust,k=5)

# Determine number of clusters
set.seed(123)
wss <- (nrow(scaled_housedf_num)-1)*sum(apply(scaled_housedf_num,2,var))
for (i in 1:ncol(scaled_housedf_num)) wss[i] <- sum(kmeans(scaled_housedf_num,
                                             centers=i)$withinss)
plot(1:ncol(scaled_housedf_num), wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")



##### Checking correlation between all variables
CorrelationResults = cor(housedf_num)

corrplot(CorrelationResults)

housedf_num %>%
  gather(key,value,-price) %>%
  ggplot(aes(x=value,y=price)) +
  geom_jitter(color = 'light blue',alpha = .6) +
  geom_smooth(method = 'gam', color= 'dark blue', fill = 'grey', alpha = .2) +
  facet_wrap(~key, scales = 'free') +
  theme_bw()

# K-Means Cluster Analysis
set.seed(123)
fit <- kmeans(scaled_housedf_num, 5) # 5 cluster solution. 8 or 13 better?
# get cluster means
clust_means <- round(aggregate(scaled_housedf_num,by=list(fit$cluster),FUN=mean),2)
# append cluster assignment
housedf_num1<- data.frame(scaled_housedf_num, fit$cluster)

fviz_cluster(fit, data <- scaled_housedf_num)

str(clust_means)

housedf_clus <- cbind(housedf, fit$cluster)
colnames(housedf_clus)[35] <- "cluster"
```

```
cluster_means_summary <- round(t(housedf_clus %>%
  keep(is.numeric)%>%
  group_by(cluster) %>%
  summarise_all(mean)),3)

cluster_means_summary

# The clusters seem to indicate that the location and zip code tiers play an important factor in the clustering.
#lets get more info specific to the zip code tiers to use for our marketing strategy for each zone.

## Break up the zip codes into tiers for demographics comparison
housedf2 <- read.csv("C:/Users/MKAlbini/Desktop/York Data Class/2 trimester/housedf.csv")

# fix date field
housedf2$date<-as.Date(housedf$date, "%Y%m%dT000000")
str(housedf)

#fix formats
housedf2$zipcode <- as.factor(housedf$zipcode)
housedf2$waterfront <- as.factor(housedf$waterfront)

# make the house with 33 bedrooms into 3. Was probably a typo.
housedf2[15871,4] <- 3
summary(housedf2)

# use max date from built and reno to make relevant date appear on one column.
#Assumption is that a renovated house has similar value to a new house.

housedf2 <- transform(housedf2, built_reno_date = pmax(yr_built, yr_renovated))

#remove extra columns no longer needed
housedf2$id <- NULL

housedf2$zip_tier <- ifelse(housedf2$zipcode %in% c(98039), "Tier1",
                    ifelse(housedf2$zipcode %in% c(98004,98040,98112),"Tier2",
                        ifelse(housedf2$zipcode %in% c(98102,98109,98105,98006,98119, 98005,
98033, 98199, 98075),"Tier3",
                            ifelse(housedf2$zipcode %in%
c(98074,98077,98053,98177,98008,98052,98122,98115,98116,98007,98027,98029, 98144,
98103,98024,98107,98117,98072),"Tier4",
                                ifelse(housedf2$zipcode %in%
c(98136,98065,98034,98059,98011,98070,98125,98166,98028,98014,98045,98019,98126,98155,98
010,98056,98118,98133),"Tier5",
                                    ifelse(housedf2$zipcode %in%
c(98038,98146,98108,98058,98092,98106,98022,98042,98178), "Tier6","Tier7"))))))

housedf2 <- as.data.frame(aggregate(housedf2[,2:21], list(housedf2$zip_tier), mean))
housedf2 <- as.data.frame(t(housedf2))
housedf2

# Break up zipcode demographicsfile zipcodes into tiers to see how the price tiers allign to the demographics for
the area.
housedf_zips <- as.data.frame(unique(housedf[["zipcode"]]))
colnames(housedf_zips)[1] <- "zipcode"
zipdemog$zipcode <- as.factor(zipdemog$zipcode)
str(zipdemog)
str(housedf_zips)

zipdemog_housedf <- merge(housedf_zips, zipdemog, by.x = "zipcode", by.y = "zipcode")

zipdemog_housedf <- zipdemog_housedf[c(1,4,8,13,14,15,21)] #just took the ones I thought would
help for now.
```

```
zipdemog_housedf$zip_tier <- ifelse(zipdemog_housedf$zipcode %in% c(98039), "Tier1",
                              ifelse(zipdemog_housedf$zipcode %in% c(98004,98040,98112),"Tier2",
                                 ifelse(zipdemog_housedf$zipcode %in%
c(98102,98109,98105,98006,98119, 98005, 98033, 98199, 98075),"Tier3",
                                    ifelse(zipdemog_housedf$zipcode %in%
c(98074,98077,98053,98177,98008,98052,98122,98115,98116,98007,98027,98029, 98144,
98103,98024,98107,98117,98072),"Tier4",
                                       ifelse(zipdemog_housedf$zipcode %in%
c(98136,98065,98034,98059,98011,98070,98125,98166,98028,98014,98045,98019,98126,98155,98
010,98056,98118,98133),"Tier5",
                                          ifelse(zipdemog_housedf$zipcode %in%
c(98038,98146,98108,98058,98092,98106,98022,98042,98178), "Tier6","Tier7"))))))

zip_tiers_means <- as.data.frame(aggregate(zipdemog_housedf[, 2:7],
list(zipdemog_housedf$zip_tier), mean))
zip_tiers_means <- as.data.frame(t(zip_tiers_means))

#Use the Compare_housedf table to gain more insights to the zip tiers created to use with clustering data
compare_housedf <- rbind(zip_tiers_means,housedf2)
compare_housedf <- compare_housedf[-c(8,15,23),]
compare_housedf

#PCA
housedf_clus_num <- housedf_clus[,sapply(housedf_clus, is.numeric)]

prin_comp <- prcomp(housedf_clus_num[2:28], scale = TRUE)
names(prin_comp)
prin_comp$centre
prin_comp$scale
prin_comp$rotation
prin_comp$sdev

dim(prin_comp$x)

prin_comp$sdev^2 / sum(prin_comp$sdev^2)
plot(prin_comp)

par(mfrow=c(2,2))

plot(prin_comp$x[,2], housedf_clus_num$price)
plot(prin_comp$x[,3], housedf_clus_num$price)
plot(prin_comp$x[,4], housedf_clus_num$price)
plot(prin_comp$x[,5], housedf_clus_num$price)
plot(prin_comp$x[,6], housedf_clus_num$price)
plot(prin_comp$x[,7], housedf_clus_num$price)

dim(prin_comp$x)

biplot(prin_comp, scale = 0)

summary(prin_comp)

std_dev <- prin_comp$sdev
pr_var <- std_dev^2
pr_var[1:10]

#proportion of variance explained
prop_varex <- pr_var/sum(pr_var)
prop_varex[1:15]

#scree plot
plot(prop_varex, xlab = "Principal Component",
```

```
    ylab = "Proportion of Variance Explained",type = "b")

#cumulative scree plot
plot(cumsum(prop_varex), xlab = "Principal Component",
    ylab = "Cumulative Proportion of Variance Explained",type = "b")

#add a data set with principal components
housedf_princomp <- data.frame(price = housedf_clus_num$price, prin_comp$x)

#we are interested in first 10 PCAs?
housedf_princomp1 <- housedf_princomp[,1:10]

housedf_princomp_dt = sort(sample(nrow(housedf_princomp1), nrow(housedf_princomp1)*.7))
train<-housedf_princomp1[housedf_princomp_dt,]
test<-housedf_princomp1[-housedf_princomp_dt,]

# define training control
train_control <- trainControl(method="cv", number=5)
# train the model
model <- train(price~., data=train,trControl=train_control, method="xgbLinear")
# summarize results
print(model)
mean(model$results$Rsquared)
mean(model$results$RsquaredSD)
mean(model$results$RMSE)
summary(model)

pricepredicted = predict(model, test)
test_pred <- cbind(test, pricepredicted)
rmse(test_pred$price, test_pred$pricepredicted)
R2(test_pred$price, test_pred$pricepredicted, formula = "corr")

#Lets see if RandomForest is better.
model2 <- train(price~., data=train,trControl=train_control, method="rf")
# summarize results
print(model2)
mean(model2$results$Rsquared)
mean(model2$results$RsquaredSD)
mean(model2$results$RMSE)
summary(model2)

pricepredicted2 = predict(model2, test)
test_pred <- cbind(test, pricepredicted2)
rmse(test_pred$price, test_pred$pricepredicted2)
R2(test_pred$price, test_pred$pricepredicted2, formula = "corr")

# Let's see if GENERALIZED LINEAR REGRESSION is better
set.seed(123)
# define training control & train model
train_control3 <- trainControl(method="cv", number=5)
model3 <- train(price~., data=train,trControl=train_control, method="glm")
# summarize results
print(model3)

pricepredicted3 = predict(model3, test)
test_pred <- cbind(test, pricepredicted3)
rmse(test_pred$price, test_pred$pricepredicted3)
R2(test_pred$price, test_pred$pricepredicted3, formula = "corr")

# train the model by using XGB Tree
model4 <- train(price~., data=train,trControl=train_control, method="xgbTree")
# summarize results
print(model4)
```

```
mean(model4$results$Rsquared)
mean(model4$results$RsquaredSD)
mean(model3$results$RMSE)
summary(model4)

pricepredicted4 = predict(model4, test)
test_pred <- cbind(test, pricepredicted4)
rmse(test_pred$price, test_pred$pricepredicted4)
R2(test_pred$price, test_pred$pricepredicted4, formula = "corr")

# XGBLinear is the more accurate for prediction.

##################################################
```