SeattleRealEstate

NickFarnell

27/12/2020

Contents

INTRODUCTION **3**

Executive Summary . . . 3
Project Overview . . . 3

ANALYSIS **3**

Libraries . . . 3
Exploratory Data Analysis . . . 3
Import Data . . . 3
Missingness Map . . . 5
Weekly Sales . . . 6
Correlation of Variables . . . 8
Compare Bedrooms and Price (Part 1) . . . 9
Clean and add SQFT2 . . . 9
Compare Bedrooms . . . 10
Compare Bathroms . . . 11
Compare Zip Codes . . . 13
Compare Waterfront . . . 16
Compare Square Footage . . . 16
Mapping the Content . . . 19
Models of Property Characteristics . . . 21
Naive Model . . . 22
Number of Bedrooms . . . 23
Number of Bathrooms . . . 24
Waterfront . . . 25
Grade Model . . . 26
View Model . . . 27
Year Built Effect . . . 28

Zip Code Effect . . . 29
Square Foot Effect . . . 30
Zip Code + Square Foot Effect . . . 31
Zip Code + Square Footage + Year Built Effect . . . 32
Results of "House Effects" . . . 33
Regularization . . . 34
Regularized Square Footage . . . 35
Regularized Zip Code . . . 37
Regularized Square Footage + Regularized Zip Code . . . 39
Nearest Neighbours . . . 40
Tuning the KNN Model . . . 41
Multiple KNN . . . 44
KNN Results . . . 45
Regression . . . 46
Linear Regression . . . 46
Multiple Regression . . . 48
Regression Results . . . 49

CONCLUSION **50**

NEXT STEPS **51**

Super Luxury Homes . . . 51
3.5 Bathrooms . . . 51
Optimizing Categorical Square Footage . . . 51
Mercer Island . . . 51
Zip Code Border Map . . . 51
Seasonality . . . 51
Combining Other Factors . . . 52
Real Data . . . 52

INTRODUCTION

Executive Summary

Project Overview

This final project... from edx The project was guided by conent from the publication Introduction to Data Science

ANALYSIS

Libraries

The first step is to call the various libraries needed for this project. I ended up using a lot of different libraries to help with the analysis.

```
library(knitr)
library(kableExtra)
library(tidyverse)
library(tidymodels)
library(gridExtra)
library(caret)
library(data.table)
library(tidyr)
library(ggplot2)
library(dplyr)
library(reshape2)
library(ggpubr)
library(ggrepel)
library(corrplot)
library(Amelia)
library(RColorBrewer)
library(lubridate)
library(zipcodeR)
library(sf)
library(rnaturalearth)
library(rnaturalearthdata)
library(maps)
library(GGally)
library(tidyverse)
library(scales)
library(zipcodeR)
```

Exploratory Data Analysis

Import Data

In this project we will be looking at the King County dataset of house prices. This dataset is available ok kaggle.com here: https://www.kaggle.com/harlfoxem/housesalesprediction After downloading the data, it is important to explore what we have. Some highlights:

- 21 columns
- 21,613 rows
- prices ranging from $75,000 to $7,700,000 USD
- 90% of the homes are less than $887,000 USD

```
KingCounty <- read_csv('KingCounty/kc_house_data.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   id = col_character(),
##   date = col_datetime(format = "")
## )
```

```
## See spec(...) for full column specifications.
```

```
head(KingCounty)
```

```
## # A tibble: 6 x 21
##   id    date                price bedrooms bathrooms sqft_living sqft_lot
##   <chr> <dttm>              <dbl>    <dbl>     <dbl>       <dbl>    <dbl>
## 1 7129~ 2014-10-13 00:00:00 2.22e5        3      1           1180     5650
## 2 6414~ 2014-12-09 00:00:00 5.38e5        3      2.25        2570     7242
## 3 5631~ 2015-02-25 00:00:00 1.80e5        2      1            770    10000
## 4 2487~ 2014-12-09 00:00:00 6.04e5        4      3           1960     5000
## 5 1954~ 2015-02-18 00:00:00 5.10e5        3      2           1680     8080
## 6 7237~ 2014-05-12 00:00:00 1.23e6        4      4.5         5420   101930
## # ... with 14 more variables: floors <dbl>, waterfront <dbl>, view <dbl>,
## #   condition <dbl>, grade <dbl>, sqft_above <dbl>, sqft_basement <dbl>,
## #   yr_built <dbl>, yr_renovated <dbl>, zipcode <dbl>, lat <dbl>, long <dbl>,
## #   sqft_living15 <dbl>, sqft_lot15 <dbl>
```

```
names(KingCounty) #21 columns of info including...
```

```
##  [1] "id"            "date"          "price"         "bedrooms"
##  [5] "bathrooms"     "sqft_living"   "sqft_lot"      "floors"
##  [9] "waterfront"    "view"          "condition"     "grade"
## [13] "sqft_above"    "sqft_basement" "yr_built"      "yr_renovated"
## [17] "zipcode"       "lat"           "long"          "sqft_living15"
## [21] "sqft_lot15"
```

```
nrow(KingCounty) #21,613 observations
```

```
## [1] 21613
```

```
summary(KingCounty)
```

```
##       id                 date                          price
##  Length:21613       Min.   :2014-05-02 00:00:00   Min.   :  75000
##  Class :character   1st Qu.:2014-07-22 00:00:00   1st Qu.: 321950
```

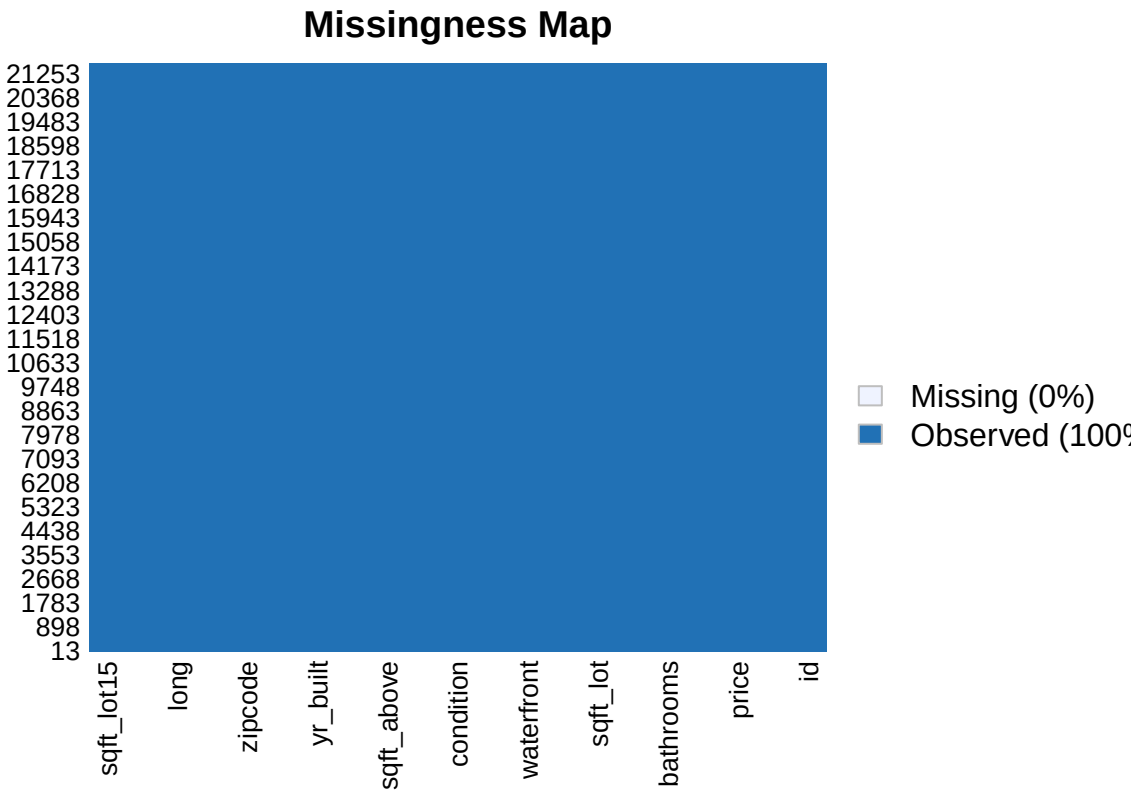
```
##  Mode  :character   Median :2014-10-16 00:00:00   Median : 450000
##                     Mean   :2014-10-29 04:38:01   Mean   : 540088
##                     3rd Qu.:2015-02-17 00:00:00   3rd Qu.: 645000
##                     Max.   :2015-05-27 00:00:00   Max.   :7700000
##     bedrooms        bathrooms      sqft_living       sqft_lot
##  Min.   : 0.000   Min.   :0.000   Min.   :  290   Min.   :    520
##  1st Qu.: 3.000   1st Qu.:1.750   1st Qu.: 1427   1st Qu.:   5040
##  Median : 3.000   Median :2.250   Median : 1910   Median :   7618
##  Mean   : 3.371   Mean   :2.115   Mean   : 2080   Mean   :  15107
##  3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2550   3rd Qu.:  10688
##  Max.   :33.000   Max.   :8.000   Max.   :13540   Max.   :1651359
##      floors        waterfront           view          condition
##  Min.   :1.000   Min.   :0.000000   Min.   :0.0000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:3.000
##  Median :1.500   Median :0.000000   Median :0.0000   Median :3.000
##  Mean   :1.494   Mean   :0.007542   Mean   :0.2343   Mean   :3.409
##  3rd Qu.:2.000   3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:4.000
##  Max.   :3.500   Max.   :1.000000   Max.   :4.0000   Max.   :5.000
##      grade          sqft_above   sqft_basement       yr_built
##  Min.   : 1.000   Min.   : 290   Min.   :   0.0   Min.   :1900
##  1st Qu.: 7.000   1st Qu.:1190   1st Qu.:   0.0   1st Qu.:1951
##  Median : 7.000   Median :1560   Median :   0.0   Median :1975
##  Mean   : 7.657   Mean   :1788   Mean   : 291.5   Mean   :1971
##  3rd Qu.: 8.000   3rd Qu.:2210   3rd Qu.: 560.0   3rd Qu.:1997
##  Max.   :13.000   Max.   :9410   Max.   :4820.0   Max.   :2015
##   yr_renovated       zipcode           lat             long
##  Min.   :   0.0   Min.   :98001   Min.   :47.16   Min.   :-122.5
##  1st Qu.:   0.0   1st Qu.:98033   1st Qu.:47.47   1st Qu.:-122.3
##  Median :   0.0   Median :98065   Median :47.57   Median :-122.2
##  Mean   :  84.4   Mean   :98078   Mean   :47.56   Mean   :-122.2
##  3rd Qu.:   0.0   3rd Qu.:98118   3rd Qu.:47.68   3rd Qu.:-122.1
##  Max.   :2015.0   Max.   :98199   Max.   :47.78   Max.   :-121.3
##  sqft_living15    sqft_lot15
##  Min.   : 399   Min.   :   651
##  1st Qu.:1490   1st Qu.:  5100
##  Median :1840   Median :  7620
##  Mean   :1987   Mean   : 12768
##  3rd Qu.:2360   3rd Qu.: 10083
##  Max.   :6210   Max.   :871200
```

```
#mid 2014 to mid 2015
#prices from $75,000 to $7,700,000 USD

quantile(KingCounty$price, c(.01, .05, .10, .50, .90, .95, .99))
```

```
##        1%        5%       10%       50%       90%       95%       99%
##  153500.4  210000.0  245000.0  450000.0  887000.0 1156480.0 1964400.0
```

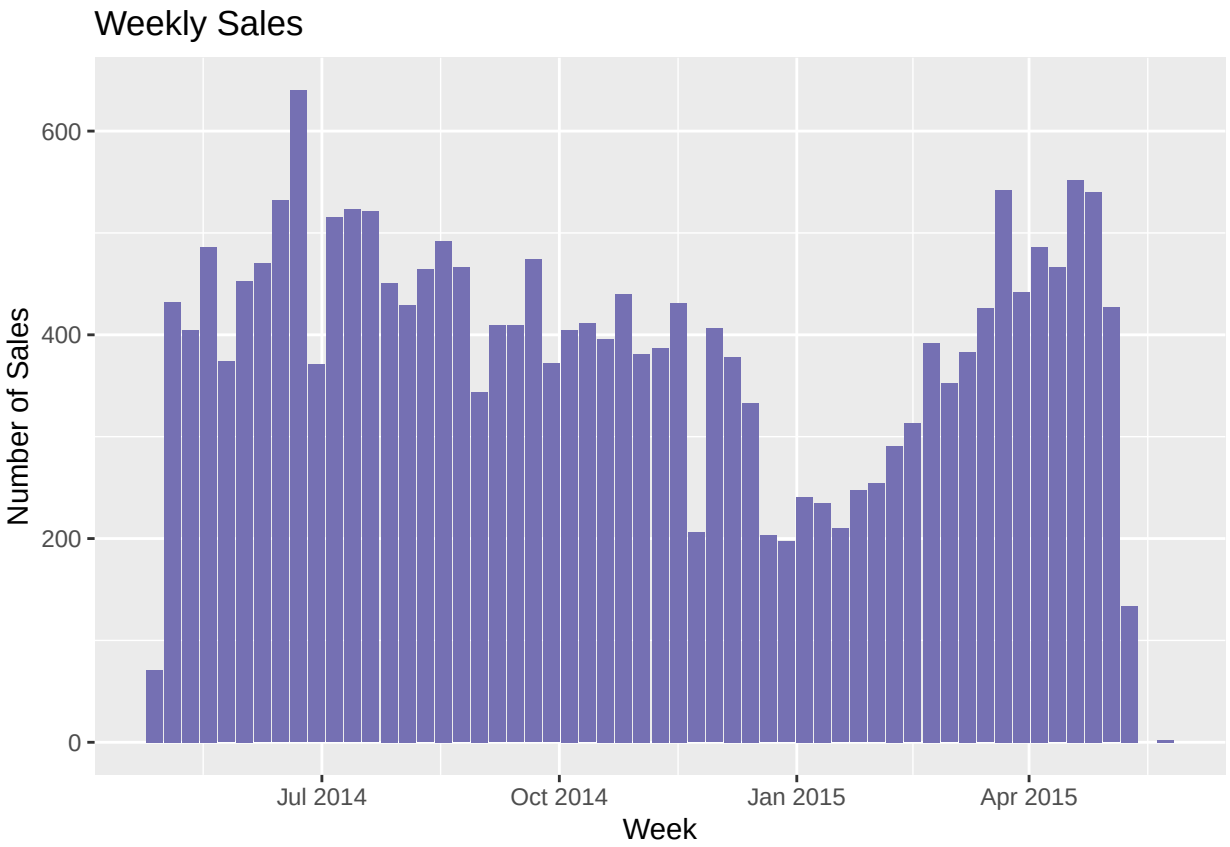
Missingness Map

To make sure that the data are complete, we can run a missingness map. This shows us that all of the 21,253 observations are complete.



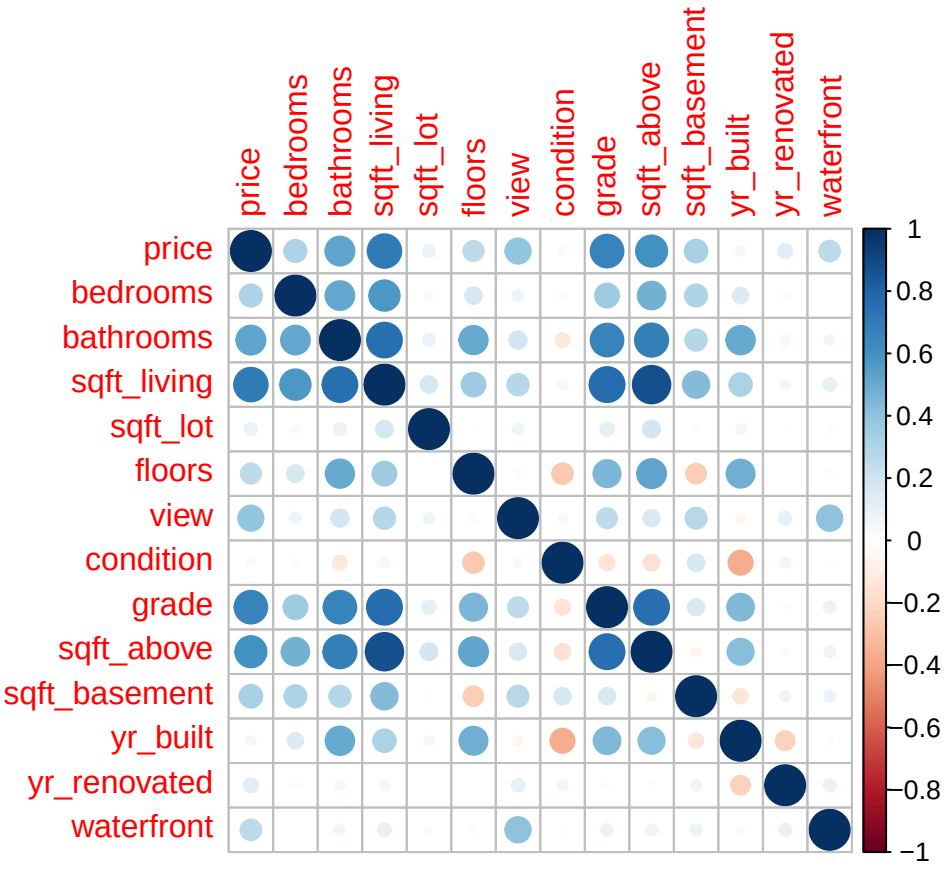
Weekly Sales

Reviewing weekly sales would be a good first step to see how activity is distributed throughout the year. Looking at the weekly sales, this is fairly expected to see more activity in the summer months and less activity near Christmas and through the winter. Since "Weekly Sales" was not an exisiting column in this data set I had to create a new column that grouped sales by day, then grouped sales by week.


Weekly Sales
Number of Sales
600
400
200
0
Jul 2014
Oct 2014
Jan 2015
Apr 2015
Week

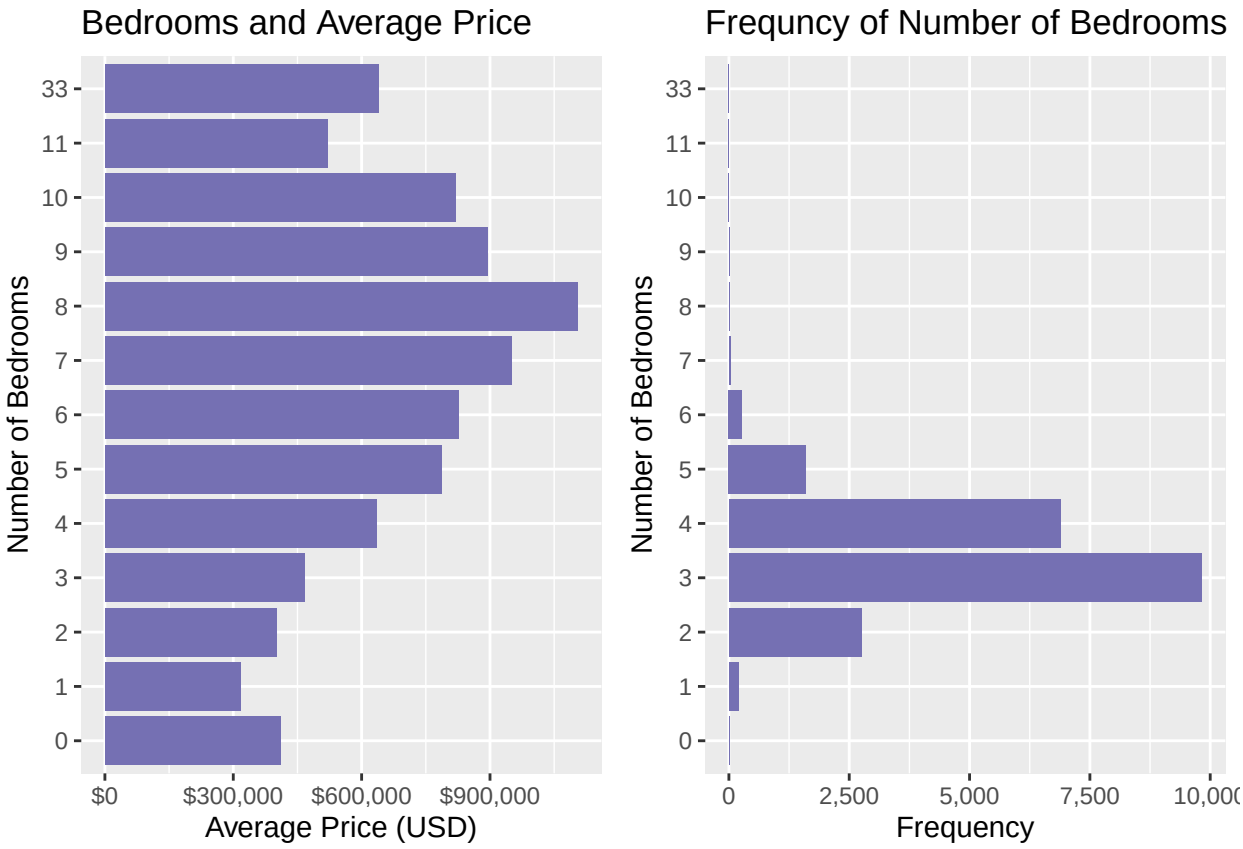
Correlation of Variables

Now that we know a little more about what information we have to work with, the next step will be to identify relationships between variables. Since we are trying to predict sale price we will mainly be focused on how other variables are correlated with price. This plot shows that price is highly correlated with bedrooms, bathrooms, sqft_living, view and grade. It is also correlated with sqft_above and sqft_basement but this is also covered by sqft_living.



Compare Bedrooms and Price (Part 1)

After identifying which variables to explore, we will start by looking at the number of bedrooms. Here we will explore both average price (how much does a home with n bedrooms sell for) as well as frequency (how many homes have n bedrooms). The most common layout is 3 bedrooms which is not surprising for North American homes. We can also see that the average price increases as the number of bedrooms increases, at least until 8 bedrooms. Since there are so few homes with more than 8 bedrooms, our average price has likely been skewed by few observations. One thing that is very odd is a 33 bedroom home that seems closer in price to a 3 bedroom home - after reviewing this specific property there is no way that a home of this size could have 33 bedrooms and it is likely a data entry error meant to be 3 bedrooms instead of 33.



Clean and add SQFT2

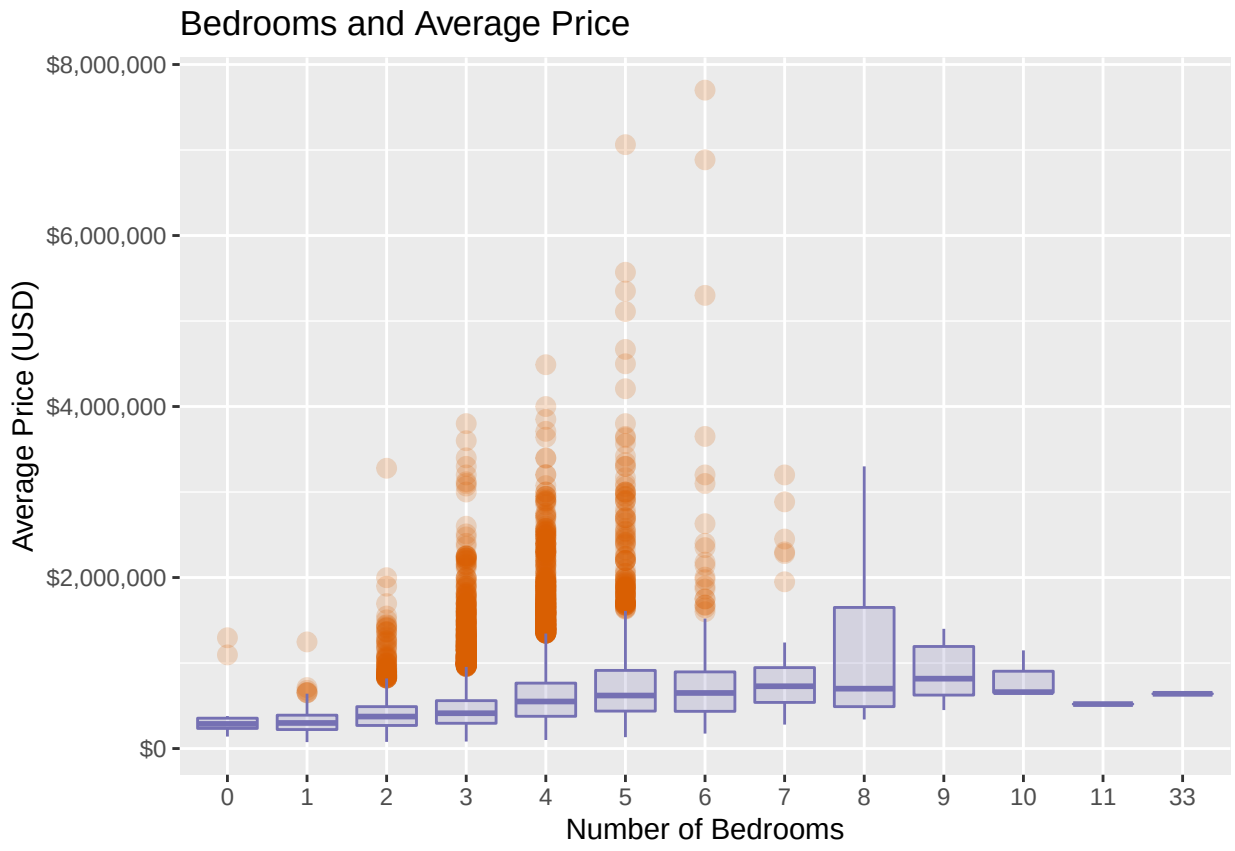
After uncovering the 33 bedroom error, we will locate that specific home and remove it from the data set. While we are in the process of cleaning, we will also create a categorical variable to represent the amount of living space within the home. This will be useful later when we are building models.

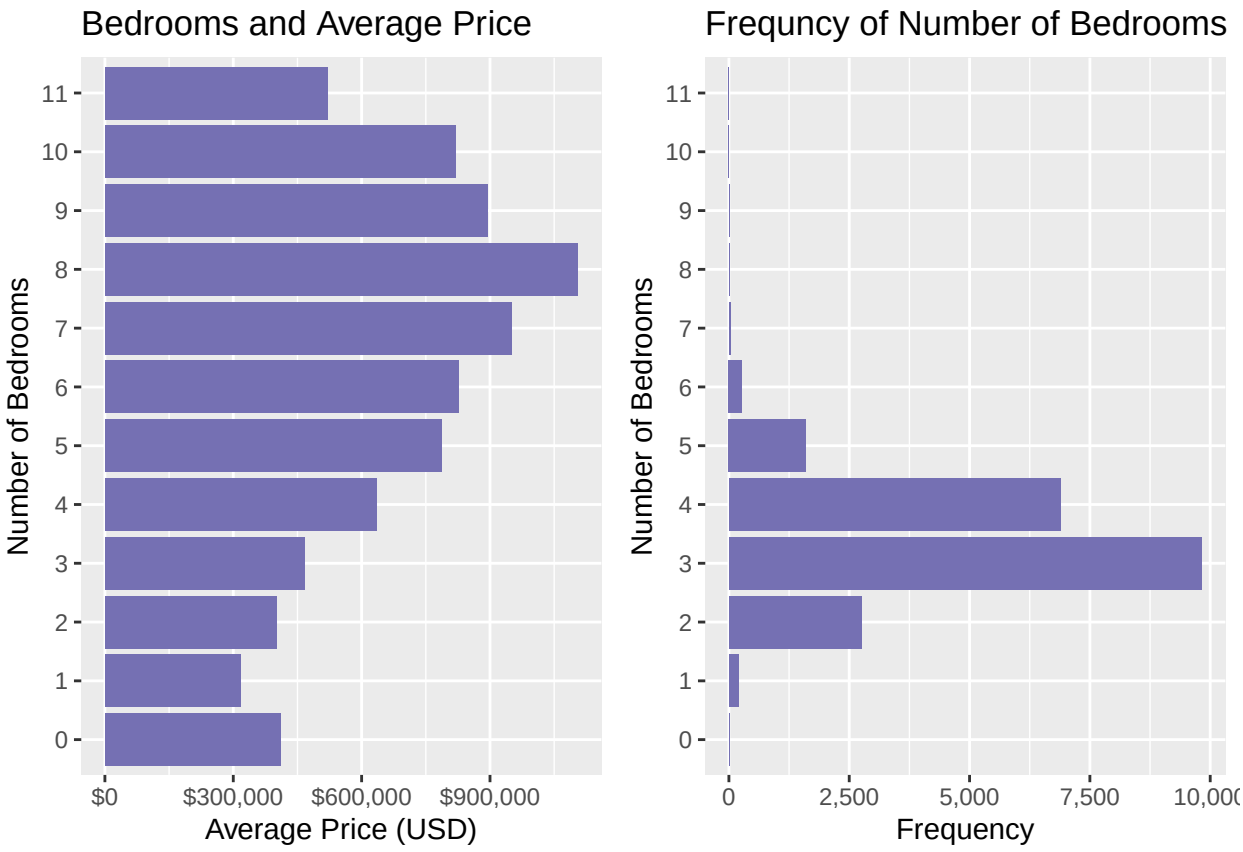
```
KingCountyClean <- subset(KingCounty, id!="2402100895") #here we will remove one row, the house that do

#we will also add a categorical variable to sqft to make things easier to compare later
# change to categorical variable (sqft)
KingCountyClean$sqft2<-cut(KingCountyClean$sqft_living, seq(0,15000,250), right=FALSE, labels=c(1:60))

SqftSummary <- KingCountyClean %>%
  group_by(sqft2) %>%
```

```
  summarize (num = n(),
             priceav = mean(price)) %>%
  arrange(desc(priceav))
```

Compare Bedrooms

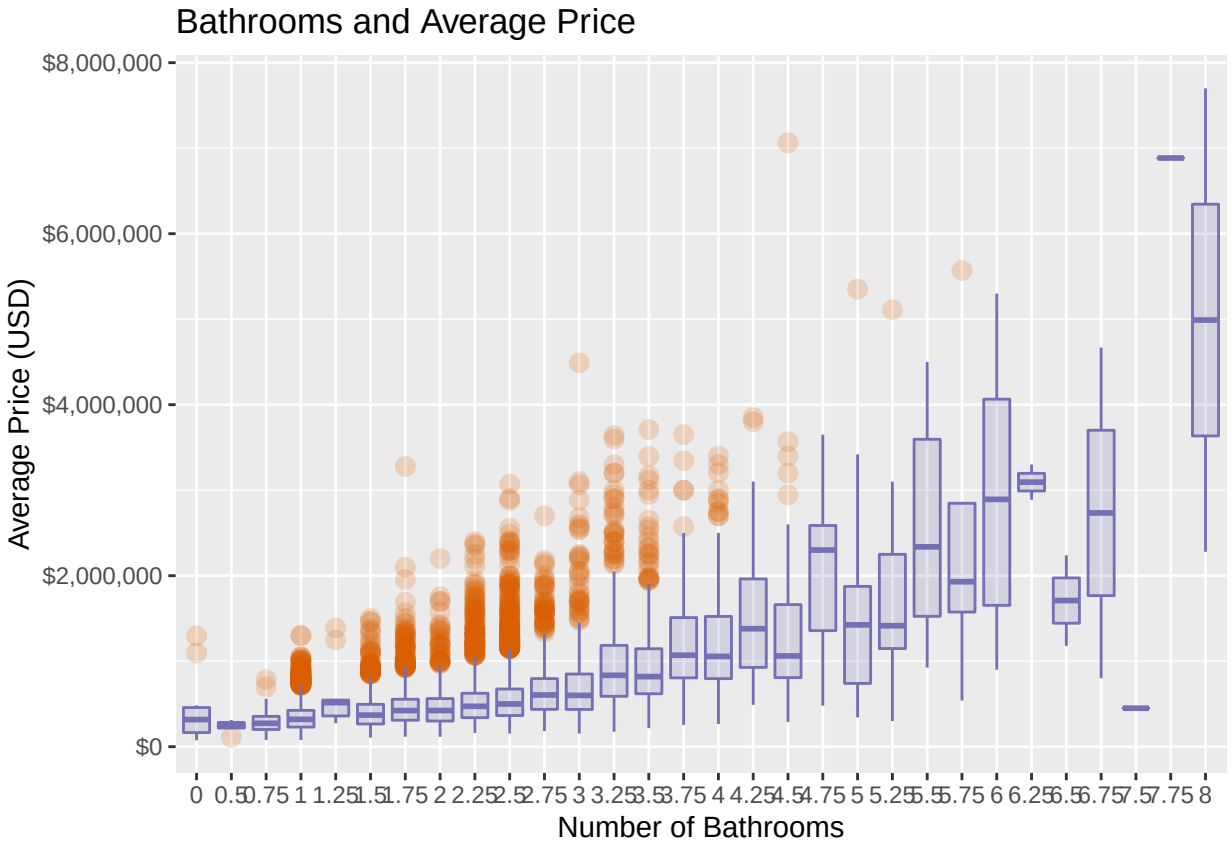
By far it is most common for homes to have 3 or 4 bedrooms. Generally the average price of a home increases betwen 1 and 8 bedrooms, but anything higher than 5 is really tough to call since there are very few homes with that number of bedrooms. After a certain point, the number of bedrooms does not really seem to be that useful, or in this case, does not seem to add value. The box plot below gives some more insight into this. The following plot confirms that 3 and 4 bedroom homes are most common and that average price generally increases with the number of bedrooms.

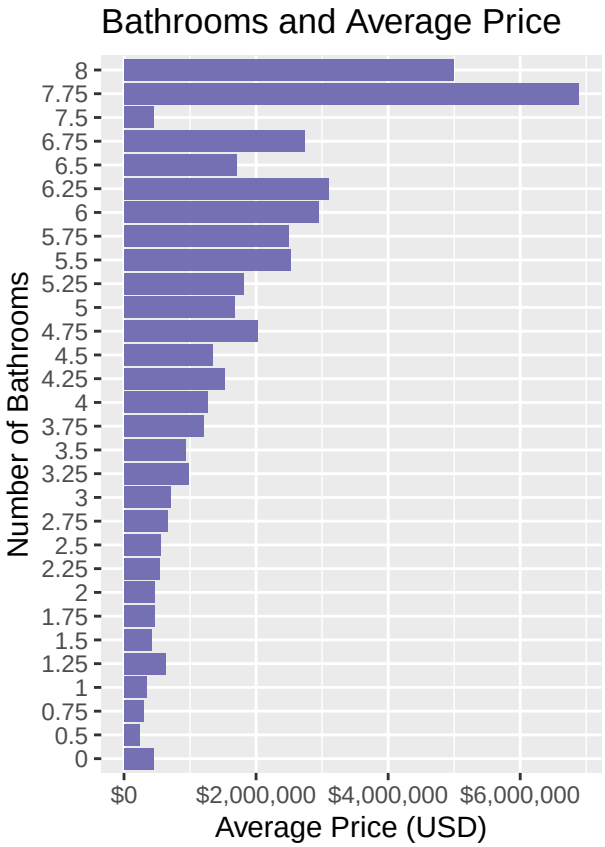


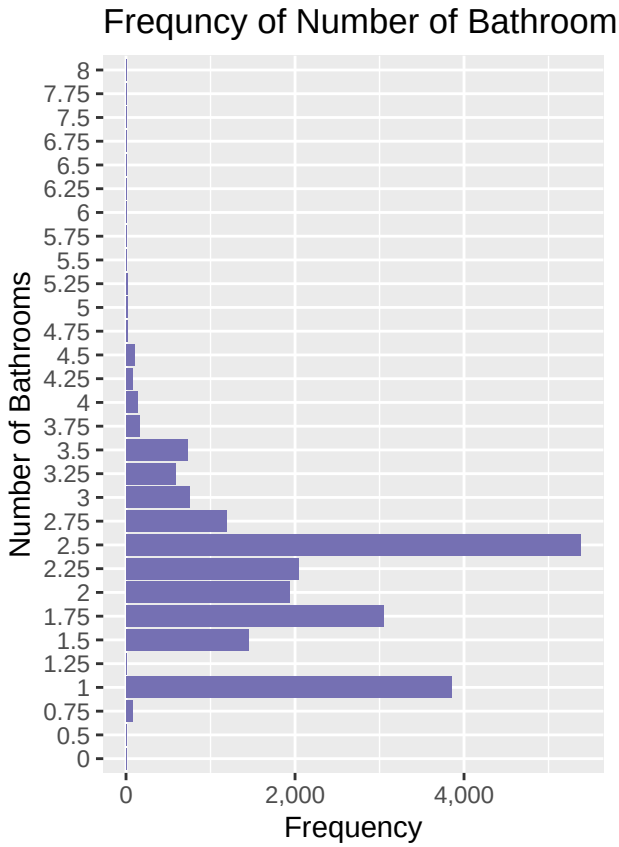


Compare Bathroms

We will complete a similar analysis for the number of bathrooms. By far most homes have 2.5 bathrooms which means 2 full bathrooms (with a shower/tub) and one half bathroom (a bathroom without a shower). From the plots below we can see that prices generally increase with the number of bathrooms, but there are very few observations of homes with more than 4 bathrooms.

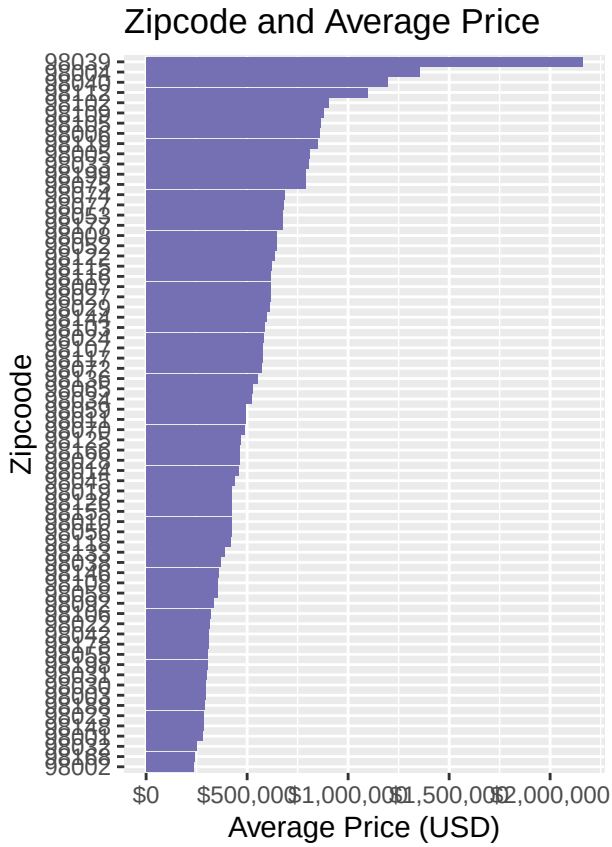

Bathrooms and Average Price
$8,000,000
$6,000,000
$4,000,000
$2,000,000
$0
Average Price (USD)
0 0.5 0.75 1 1.25 1.5 1.75 2 2.25 2.5 2.75 3 3.25 3.5 3.75 4 4.25 4.5 4.75 5 5.25 5.5 5.75 6 6.25 6.5 6.75 7.5 7.75 8
Number of Bathrooms

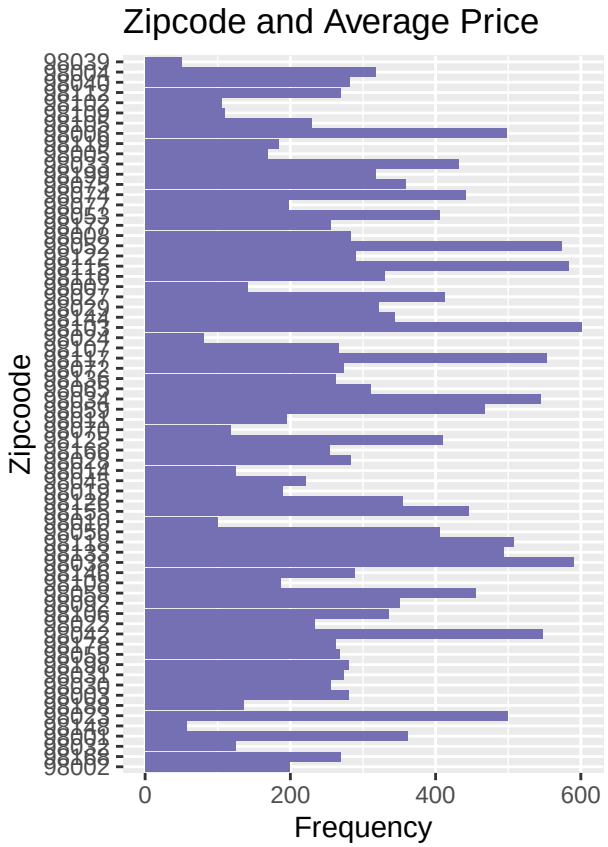


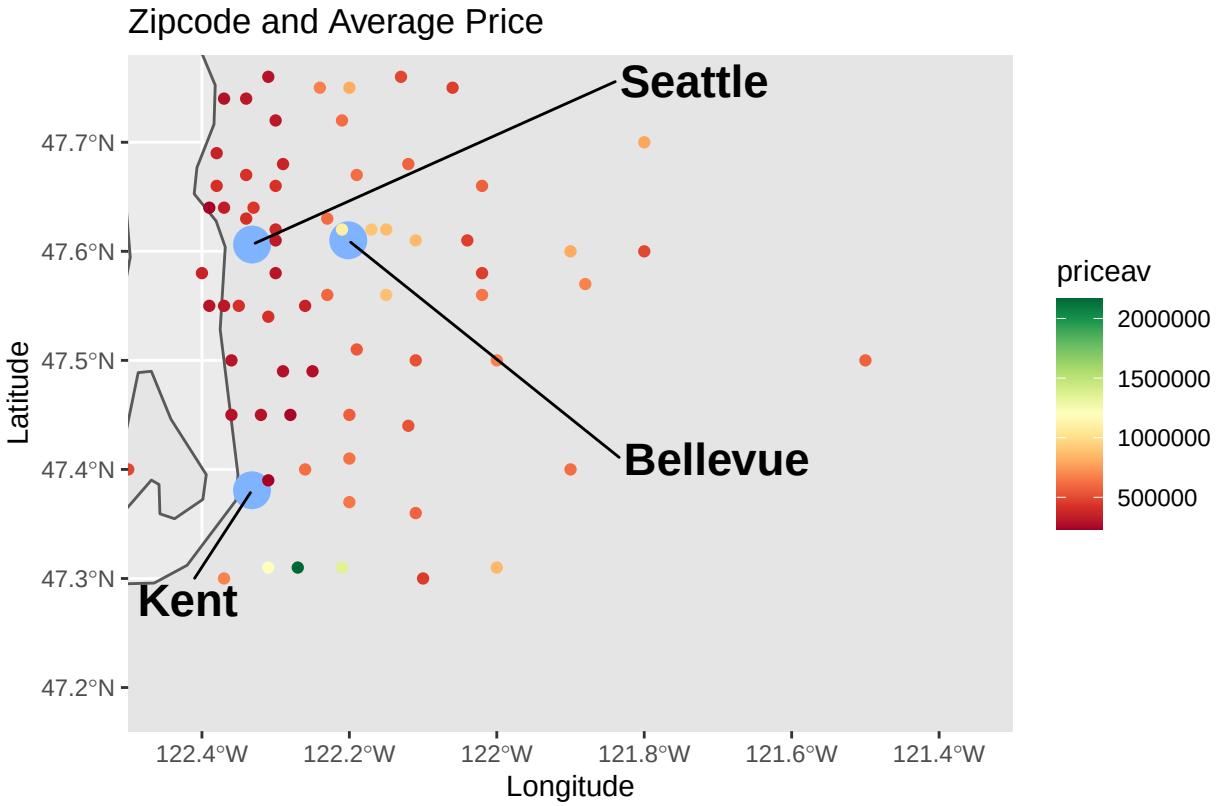


Compare Zip Codes

The location is likely the most important variable when predicting home price. We will review zip codes to look for any specific patterns. There are 70 different zip codes represented in this data set and a wide variety of average prices from $234,284.00 all the way up to $2,160,606.60. The most expensive zip code in this list is 98039, which is not that surprising when you realize that Bill Gates has a 66,000 square foot mansion located there - https://en.wikipedia.org/wiki/Bill_Gates%27s_house. The most expensive zip code also has very few properties that sold in this time period, but other than that there are no obvious patterns between the number of properties sold per zip code and price. The following plot maps average values based on zip code.

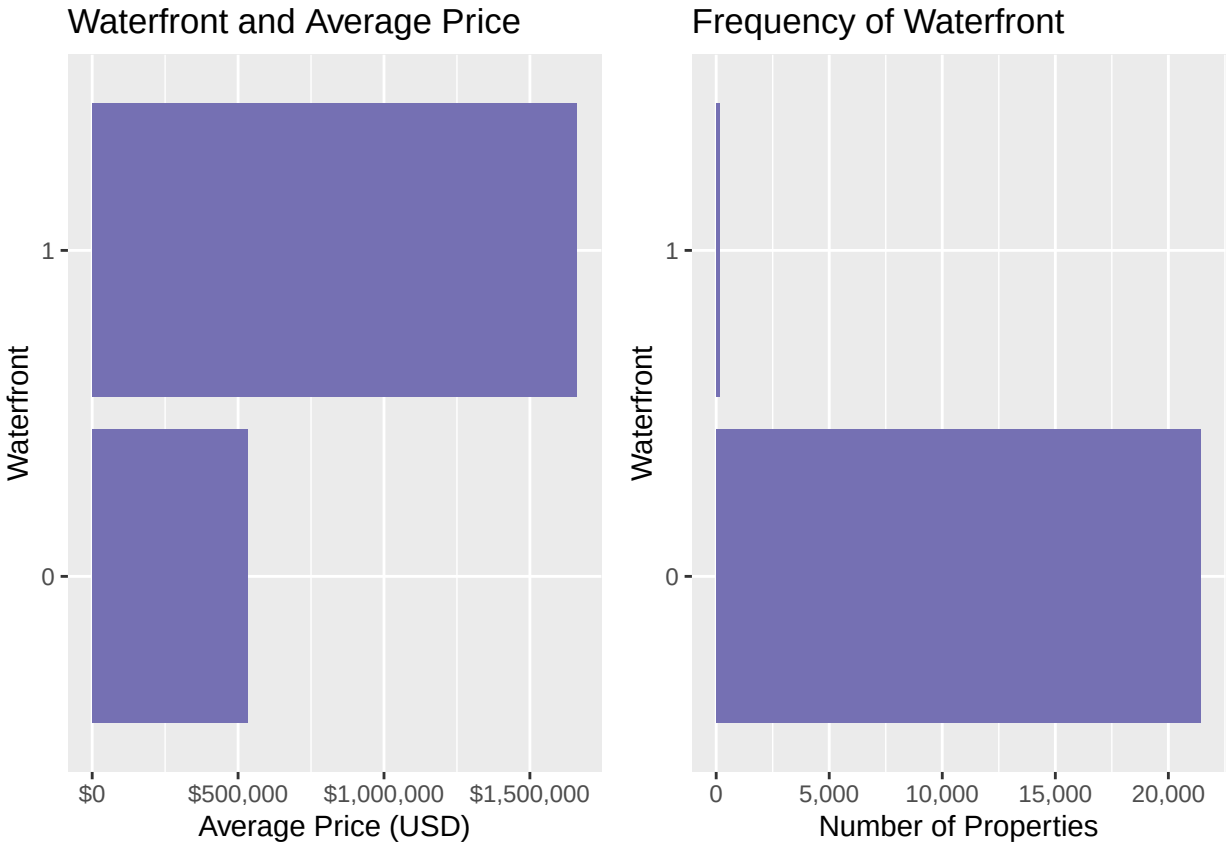

Zipcode and Average Price
Zipcoode
$0
$500,000
$1,000,000
$1,500,000
$2,000,000
Average Price (USD)


Zipcode and Average Price
Zipcoode
0
200
400
600
Frequency


Zipcode and Average Price
Seattle
Bellevue
Kent
47.7°N
47.6°N
47.5°N
47.4°N
47.3°N
47.2°N
Latitude
122.4°W
122.2°W
122°W
121.8°W
121.6°W
121.4°W
Longitude
priceav
2000000
1500000
1000000
500000

Compare Waterfront

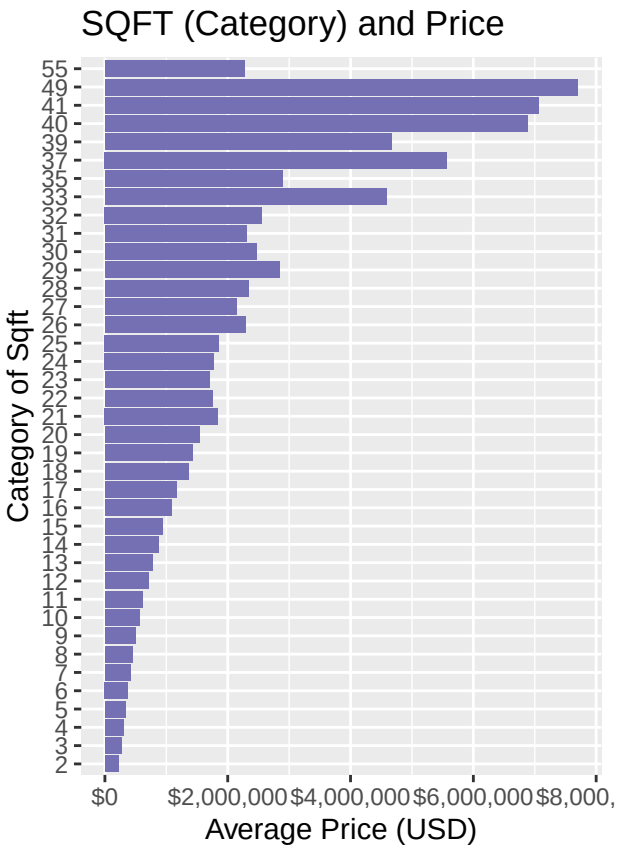
There are very few waterfront properties - only 163 properties (of more than 21,000 in the data set) are waterfront. These rare waterfront homes are also very expensive with an average price of $1,661,876.00 compared to $531,559.00 for non-waterfront.

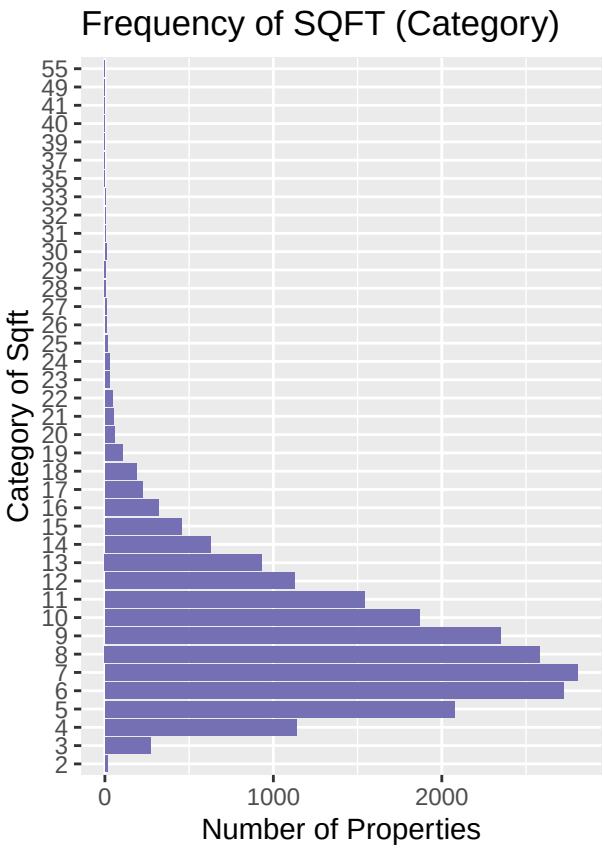


Compare Square Footage

Other than location, livible square footage is likely the best predictor of sale price. Larger homes are more expensive than smaller homes. The first scatter plot shows this relationship and confirms an obvious trend of price increasing with size. The following plots show the average prices and frequency of house size based on the categorial variable "sqft2" that we created earlier. This plot also confirms that we generally see an increase in price for larger homes, and that the most common house size is around 2,000 square feet of livible space.

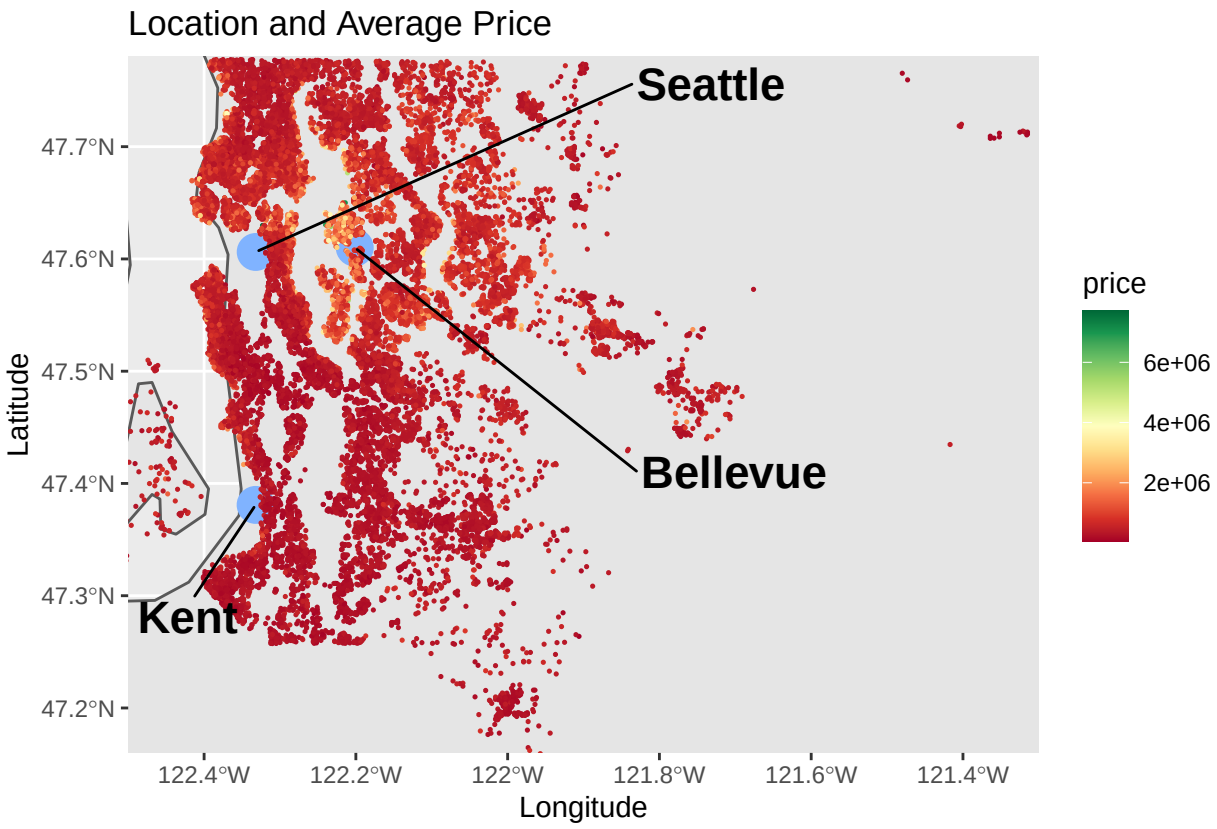

Size vs Price
$8,000,000
$6,000,000
$4,000,000
$2,000,000
$0
Price (USD)
0
2,500
5,000
7,500
10,000
12,500
House Size (square footage)


SQFT (Category) and Price
Category of Sqft
55
49
41
40
39
37
35
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
$0
$2,000,000
$4,000,000
$6,000,000
$8,000,
Average Price (USD)

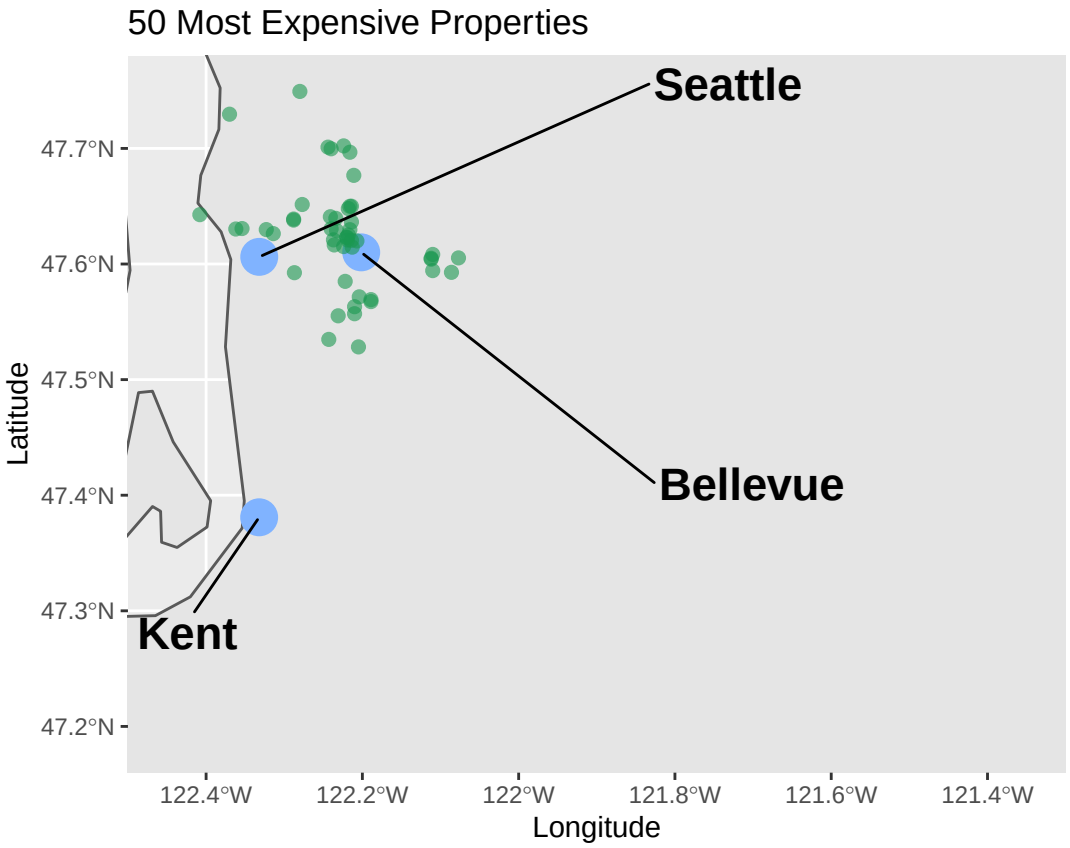

Frequency of SQFT (Category)
Category of Sqft
55
49
41
40
39
37
35
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
0
1000
2000
Number of Properties

Mapping the Content

Plotting this content on a map is a great way to see a lot of information at once. This plot shows the location (based on lat/long of each home) with the price represented by colour (red = low, yellow = mid, green = high). Note that there are very few homes at the top of the price range. There are also pockets of expensive homes.



Since so many of the properties are on the lower end of the price scale, and so few are at the top we will graph only the most expensive 50 properties to get a better understanding of where they are.



Models of Property Characteristics

Now that we have explored the data and have a better understanding of what is included, we can start to build our models. We will start by building our RMSE function which is how we will evaluate our prediction models.

```
# Validation set will be 10% of real estate data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = KingCountyClean$price, times = 1, p = 0.1, list = FALSE)
trainNF <- KingCountyClean[-test_index,]
testNF <- KingCountyClean[test_index,]


# Make sure city and zipcode in test set are also in train set
validation <- testNF %>%
  semi_join(trainNF, by = "zipcode") %>%
  semi_join(trainNF, by = "bathrooms")
# Add rows removed from validation set back into train set
removed <- anti_join(testNF, validation)
trainNF <- rbind(trainNF, removed)
```

The next step is to create the rmse function, this is how we will evaluate our models

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Naive Model

The first model that we will build is simply guessing that a home will be exactly the average. Of course this is not a usable model, but it is a good place to start. If our future models are not peforming any better than the average, this would be a good indication of a problem.

```
mu_hat <- mean(trainNF$price)
naive_rmse <- RMSE(testNF$price, mu_hat)

rmse_results <- tibble(Method = "Just the Average", RMSE = naive_rmse)
#rmse_results %>% knitr::kable()

kable(rmse_results, booktabs = T) %>%
  row_spec(1, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261

Number of Bedrooms

As we have seen from our exploratory data analysis, one of the key factors of the price of a home is how many bedrooms it has. In general, more bedrooms means a higher selling price. If we factor in the number of bedrooms in our analysis we should be able to improve the accuracy of our "Naive Model". This model gives us a RMSE of $346,804.20 which is an improvement over our naive model, but still not great.

```
#modelling bedroom effects
mu <- mean(trainNF$price)
beds_avgs <- trainNF %>%
  group_by(bedrooms) %>%
  summarize(b_beds = mean(price - mu))

predicted_ratings_beds <- mu + testNF %>%
  left_join(beds_avgs, by="bedrooms") %>%
  pull(b_beds)
BedsEffect <- RMSE(predicted_ratings_beds, testNF$price)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Bed Effect",
                                 RMSE = BedsEffect ))
#rmse_results %>% knitr::kable()
kable(rmse_results, booktabs = T) %>%
  row_spec(2, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261.0
Bed Effect	346804.2

Number of Bathrooms

From our previous analysis we have also observed that homes with a higher number of bathrooms generally sell for a higher price. When using only the number of bathrooms to predict the sale price of a house, our RMSE is $318,502.60 which is an improvement over our "Bedroom" model, but still not accurate enough to be useful.

```
#modelling bath effects
mu <- mean(trainNF$price)
bath_avgs <- trainNF %>%
  group_by(bathrooms) %>%
  summarize(b_bath = mean(price - mu))

predicted_ratings_bath <- mu + testNF %>%
  left_join(bath_avgs, by="bathrooms") %>%
  pull(b_bath)
BathEffect <- RMSE(predicted_ratings_bath, testNF$price)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Bath Effect",
                                 RMSE = BathEffect ))
#rmse_results %>% knitr::kable()

kable(rmse_results, booktabs = T) %>%
  row_spec(3, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261.0
Bed Effect	346804.2
Bath Effect	318502.6

Waterfront

In our previous exploration, we have seen that waterfront properties are much rarer than non-waterfront and substantially more expensive. If we use only waterfront as a predictor of sale price, we get an RMSE of $339,4460.00. This model is better than our naive model, but is not an improvement over using bedrooms or bathrooms as a predictor of price.

```
#modelling waterfront effects
mu <- mean(trainNF$price)
bath_avgs <- trainNF %>%
  group_by(waterfront) %>%
  summarize(b_waterfront = mean(price - mu))

predicted_ratings_bath <- mu + testNF %>%
  left_join(bath_avgs, by="waterfront") %>%
  pull(b_waterfront)
WaterfrontEffect <- RMSE(predicted_ratings_bath, testNF$price)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Waterfront Effect",
                                 RMSE = WaterfrontEffect ))
#rmse_results %>% knitr::kable()
kable(rmse_results, booktabs = T) %>%
  row_spec(4, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261.0
Bed Effect	346804.2
Bath Effect	318502.6
Waterfront Effect	339446.0

Grade Model

In this dataset, "Grade" is a representation of the overall quality of the home. We would expect that a better quality home would sell for a higher price than a lower quality home, all else being equal. When we use only this characteristic, our RMSE value is $258,363.90 which is by far our most accurate predictor so far.

```
#modelling grade effects
mu <- mean(trainNF$price)
grade_avgs <- trainNF %>%
  group_by(grade) %>%
  summarize(b_grade = mean(price - mu))

predicted_ratings_grade <- mu + testNF %>%
  left_join(grade_avgs, by="grade") %>%
  pull(b_grade)
GradeEffect <- RMSE(predicted_ratings_grade, testNF$price)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Grade Effect",
                                 RMSE = GradeEffect ))
#rmse_results %>% knitr::kable()
kable(rmse_results, booktabs = T) %>%
  row_spec(5, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261.0
Bed Effect	346804.2
Bath Effect	318502.6
Waterfront Effect	339446.0
Grade Effect	258363.9

View Model

Parts of Seattle have views of oceans and mountains which of course is something that will increase the value of a property. The better the view, the higher the price, all else being equal. To see how much of an impact this has, we can use the "View" characteristic to build a model which results in an RMSE of $319,624.60. This is a better predictor than our naive model, bedroom model, and waterfront model.

```
#modelling view effects
mu <- mean(trainNF$price)
view_avgs <- trainNF %>%
  group_by(view) %>%
  summarize(b_view = mean(price - mu))

predicted_ratings_view <- mu + testNF %>%
  left_join(view_avgs, by="view") %>%
  pull(b_view)
ViewEffect <- RMSE(predicted_ratings_view, testNF$price)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="View Effect",
                                 RMSE = ViewEffect ))
#rmse_results %>% knitr::kable()

kable(rmse_results, booktabs = T) %>%
  row_spec(6, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261.0
Bed Effect	346804.2
Bath Effect	318502.6
Waterfront Effect	339446.0
Grade Effect	258363.9
View Effect	319624.6

Year Built Effect

The age of a property might have an impact on the price. This is not necessarily a straight line, since very old historic properties could be in higher demand than properties built in the 1980's. When using "Year Built" as a predictor of price, our RMSE is $357,461.10. This is not a very good model.

```
#modelling age effects
mu <- mean(trainNF$price)
yr_built_avgs <- trainNF %>%
  group_by(yr_built) %>%
  summarize(b_yr = mean(price - mu))

predicted_yrbuilt <- mu + testNF %>%
  left_join(yr_built_avgs, by="yr_built") %>%
  pull(b_yr)
YearEffect <- RMSE(predicted_yrbuilt, testNF$price)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Year Effect",
                                 RMSE = YearEffect ))
#rmse_results %>% knitr::kable()
kable(rmse_results, booktabs = T) %>%
  row_spec(7, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261.0
Bed Effect	346804.2
Bath Effect	318502.6
Waterfront Effect	339446.0
Grade Effect	258363.9
View Effect	319624.6
Year Effect	357462.1

Zip Code Effect

Certain neighbourhoods will demand a premium price. The easiest way to capture this in a model is by using zip code to differentiate between properties. Using only zip code as a predictor, our RMSE is $284,712.00. Compared to other results so far, this is a reasonable, but not great number. The value will likely come when we combine zip code effect with other effects.

```
#modelling zipcode effects
mu <- mean(trainNF$price)
zip_avgs <- trainNF %>%
  group_by(zipcode) %>%
  summarize(b_zip = mean(price - mu))

predicted_ratings_zip <- mu + testNF %>%
  left_join(zip_avgs, by="zipcode") %>%
  pull(b_zip)
ZipEffect <- RMSE(predicted_ratings_zip, testNF$price)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Zip Effect",
                                 RMSE = ZipEffect ))
#rmse_results %>% knitr::kable()

kable(rmse_results, booktabs = T) %>%
  row_spec(8, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261.0
Bed Effect	346804.2
Bath Effect	318502.6
Waterfront Effect	339446.0
Grade Effect	258363.9
View Effect	319624.6
Year Effect	357462.1
Zip Effect	284712.0

Square Foot Effect

For this model we are using the usable living space in the home as a predictor of the sale price. Just like in previous models, we are grouping similar properties together to make these predictions, for example whether there is 1, 2, 3, or 7 bedrooms. Categorical variables like "number of bedrooms" or whether or not the property is waterfront are straight forward to build into these models since there is a relatively small number of possibilities. For square footage this is more complicated since rather than a categorical variable, this is a continuous variable ranging from 290sqft to 13,540sqft. Rather than using strict square footage we have created a categorical variable called sqft2 which creates groupings for every 250sqft from 0 to 15,000. When using this "sqft2" variable as a predictor of price, our RMSE is $251,256.20. This is a reasonable predictor but will likely improve when we combine it with other variables.

```
#modelling categorical sqft effects
mu <- mean(trainNF$price)
sqftcat_avgs <- trainNF %>%
  group_by(sqft2) %>%
  summarize(b_sqftcat = mean(price - mu))

predicted_ratings_sqftcat <- mu + testNF %>%
  left_join(sqftcat_avgs, by="sqft2") %>%
  pull(b_sqftcat)
sqftcatEffect <- RMSE(predicted_ratings_sqftcat, testNF$price)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="SQFT Cat Effect",
                                 RMSE = sqftcatEffect ))
# rmse_results %>% knitr::kable()

kable(rmse_results, booktabs = T) %>%
  row_spec(9, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261.0
Bed Effect	346804.2
Bath Effect	318502.6
Waterfront Effect	339446.0
Grade Effect	258363.9
View Effect	319624.6
Year Effect	357462.1
Zip Effect	284712.0
SQFT Cat Effect	251256.2

Zip Code + Square Foot Effect

Now that we have a fairly good understanding of how each of the variables above individually act as predictors, we can start to combine some models. The first combination will be zip code and square footage. When we combine these models we substanially improve our prediction, with an RMSE of $215,332.10. This is our best model so far.

```
#combine zip and sqft
predicted_ratings_zipsqftcat <- testNF %>%
  left_join(zip_avgs, by="zipcode") %>%
  left_join(sqftcat_avgs, by="sqft2") %>%
  mutate(pred = mu + b_zip + b_sqftcat) %>%
  pull(pred)
ZipSQFTcatEffect <- RMSE(predicted_ratings_zipsqftcat, testNF$price)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Zip Effect + SQFT Cat Effect",
                                 RMSE = ZipSQFTcatEffect ))
# rmse_results %>% knitr::kable()
kable(rmse_results, booktabs = T) %>%
  row_spec(10, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261.0
Bed Effect	346804.2
Bath Effect	318502.6
Waterfront Effect	339446.0
Grade Effect	258363.9
View Effect	319624.6
Year Effect	357462.1
Zip Effect	284712.0
SQFT Cat Effect	251256.2
Zip Effect + SQFT Cat Effect	215332.1

Zip Code + Square Footage + Year Built Effect

Our next promising model would be to combine zip code, square footage, and the year built. This model does not actually improve our prediction and is worse than zip code + square footage. The RMSE of this model is $234,218.70.

```
#combine zip and sqft and year
yr_built_avgs <- trainNF %>%
  group_by(yr_built) %>%
  summarize(b_yr = mean(price - mu))

predicted_yrbuilt <- mu + testNF %>%
  left_join(yr_built_avgs, by="yr_built") %>%
  pull(b_yr)

predicted_ratings_zipsqftcatyear <- testNF %>%
  left_join(zip_avgs, by="zipcode") %>%
  left_join(sqftcat_avgs, by="sqft2") %>%
  left_join(yr_built_avgs, by="yr_built") %>%
  mutate(pred = mu + b_zip + b_sqftcat + b_yr) %>%
  pull(pred)
ZipSQFTcatYearEffect <- RMSE(predicted_ratings_zipsqftcatyear, testNF$price)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Zip Effect + SQFT Cat Effect + Year Effect",
                                 RMSE = ZipSQFTcatYearEffect ))
# rmse_results %>% knitr::kable()
kable(rmse_results, booktabs = T) %>%
  row_spec(11, color = 'white', background = 'black')
```

Method	RMSE
Just the Average	362261.0
Bed Effect	346804.2
Bath Effect	318502.6
Waterfront Effect	339446.0
Grade Effect	258363.9
View Effect	319624.6
Year Effect	357462.1
Zip Effect	284712.0
SQFT Cat Effect	251256.2
Zip Effect + SQFT Cat Effect	215332.1
Zip Effect + SQFT Cat Effect + Year Effect	234218.7

Results of "House Effects"

After reviewing the different characteristics of the properties and building a model that takes these into consideration, we can see the following table. There were some surprising results, specifically that the age of the home does not improve our prediction model. The following table ranks the models in order of effectiveness, with the "Zip Effect + SQFT Cat Effect + Year Effect" performing the best (highlighted). Other combinations were also tested but not shown here.

```
kable(rmse_results %>% arrange(RMSE), booktabs = T) %>%
  row_spec(1, background = '#E7298A')
```

Method	RMSE
Zip Effect + SQFT Cat Effect	215332.1
Zip Effect + SQFT Cat Effect + Year Effect	234218.7
SQFT Cat Effect	251256.2
Grade Effect	258363.9
Zip Effect	284712.0
Bath Effect	318502.6
View Effect	319624.6
Waterfront Effect	339446.0
Bed Effect	346804.2
Year Effect	357462.1
Just the Average	362261.0

Regularization

There are some things that happen very infrequently that might be throwing off our predictions in the models so far. We will start by looking at the worst predictions - the ones that had the largest error and try to find some patterns. The best predictions were only off by about $22,000 but the worst predictions were off by more than $7,000,000! With regularization we can reduce the impact of some rare occurances in our models so far. The code below will highlight our worst and best predictions of the categorical sqft model.

```
#worst predictions
LeastAccurateOld <- trainNF %>%
  left_join(sqftcat_avgs, by = "sqft2") %>%
  arrange(desc(abs(b_sqftcat))) %>%
  select(price, bedrooms, bathrooms, sqft_living, b_sqftcat) %>%
  slice(1:10)
LeastAccurateOld
```

```
## # A tibble: 10 x 5
##      price bedrooms bathrooms sqft_living b_sqftcat
##      <dbl>    <dbl>     <dbl>       <dbl>     <dbl>
##  1 7700000        6      8          12050  7159324.
##  2 7062500        5      4.5        10040  6521824.
##  3 6885000        6      7.75        9890  6344324.
##  4 5570000        5      5.75        9200  5029324.
##  5 4668000        5      6.75        9640  4127324.
##  6 5110800        5      5.25        8010  3664724.
##  7 3300000        5      6.25        8020  3664724.
##  8 4000000        4      5.5         7080  2687324.
##  9 1940000        4      5.75        7220  2687324.
## 10 3800000        5      5.5         7050  2687324.
```

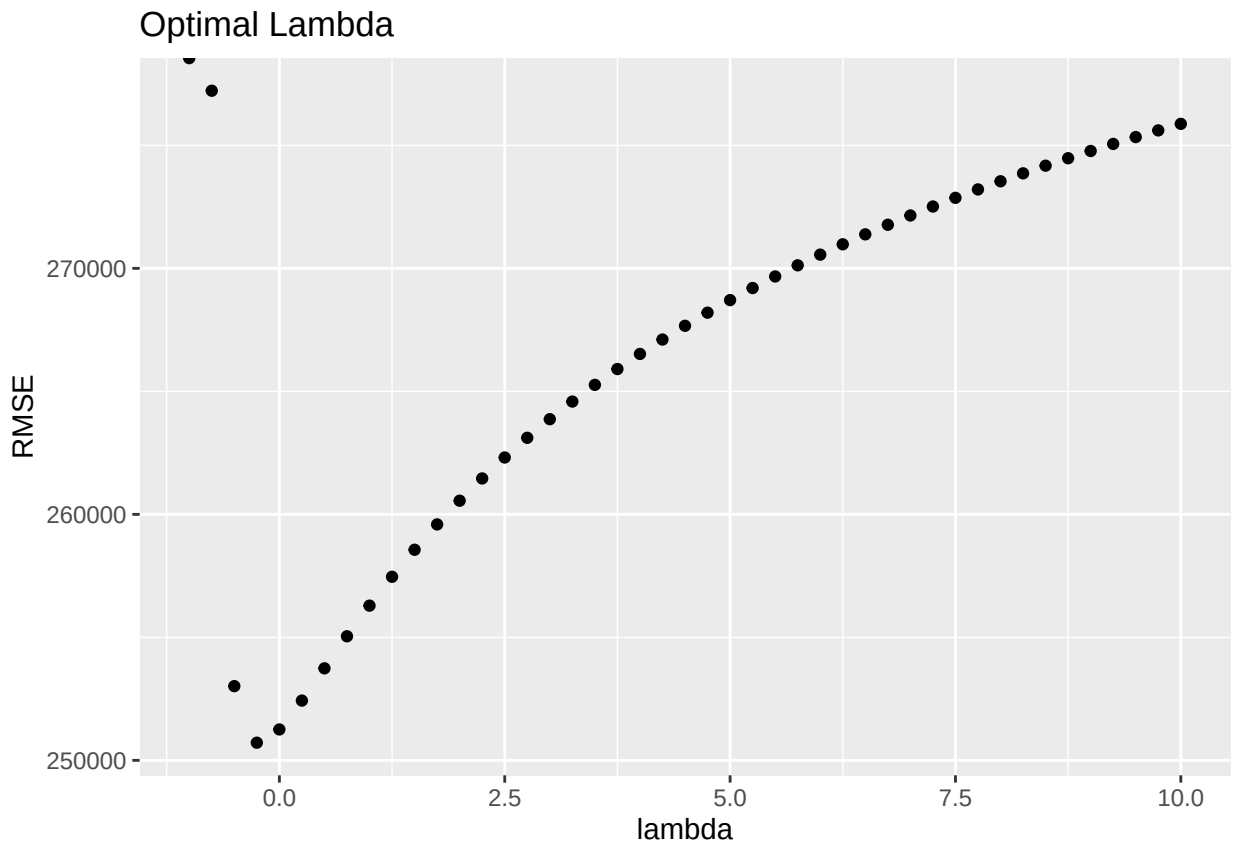
```
#best predictions
MostAccurateOld <- trainNF %>%
  left_join(sqftcat_avgs, by = "sqft2") %>%
  arrange(abs(b_sqftcat)) %>%
  select(price, bedrooms, bathrooms, sqft_living, b_sqftcat) %>%
  slice(1:10)
MostAccurateOld
```

```
## # A tibble: 10 x 5
##     price bedrooms bathrooms sqft_living b_sqftcat
##     <dbl>    <dbl>     <dbl>       <dbl>     <dbl>
##  1 285000        5      2.5          2270    22467.
##  2 329000        3      2.25         2450    22467.
##  3 937000        3      1.75         2450    22467.
##  4 580500        3      2.5          2320    22467.
##  5 687500        4      1.75         2330    22467.
##  6 696000        3      2.5          2300    22467.
##  7 640000        4      2            2360    22467.
##  8 785000        4      2.5          2290    22467.
##  9 292500        4      2.5          2250    22467.
## 10 301000        3      2.5          2420    22467.
```

Regularized Square Footage

We will try to find the optimal penalty term (lambda) for the size of the house. Essentially we are trying to determine how much to reduce the impact of infrequent occurances. Using a variety of different potential numbers for lambda, we can plot the results to see number which minimizes the RMSE. We can see that the optimal number for lambda is 0.25 and when we use this in our model of categorical square footage, it results in an RMSE of $250,717.10 which is a slight improvement over our first sqft model.

```
readRDS(file = "lambdas_qplot_sqft2.rds")
```




```
lambda_sqft2 <- -0.25

mu <- mean(trainNF$price)
sqft2_avgs_reg <- trainNF %>%
  group_by(sqft2) %>%
  summarize(b_sqft2 = sum(price - mu)/(n()+lambda_sqft2), n_i = n())

predicted_price_regsqft2 <- testNF %>%
  left_join(sqft2_avgs_reg, by = "sqft2") %>%
  mutate(pred = mu + b_sqft2) %>%
  pull(pred)
RegularizedSqft2 <- RMSE(predicted_price_regsqft2, testNF$price)

rmse_results_reg <- tibble(Method="Regularized SQFT2 Effect",
                                 RMSE = RegularizedSqft2 )
```

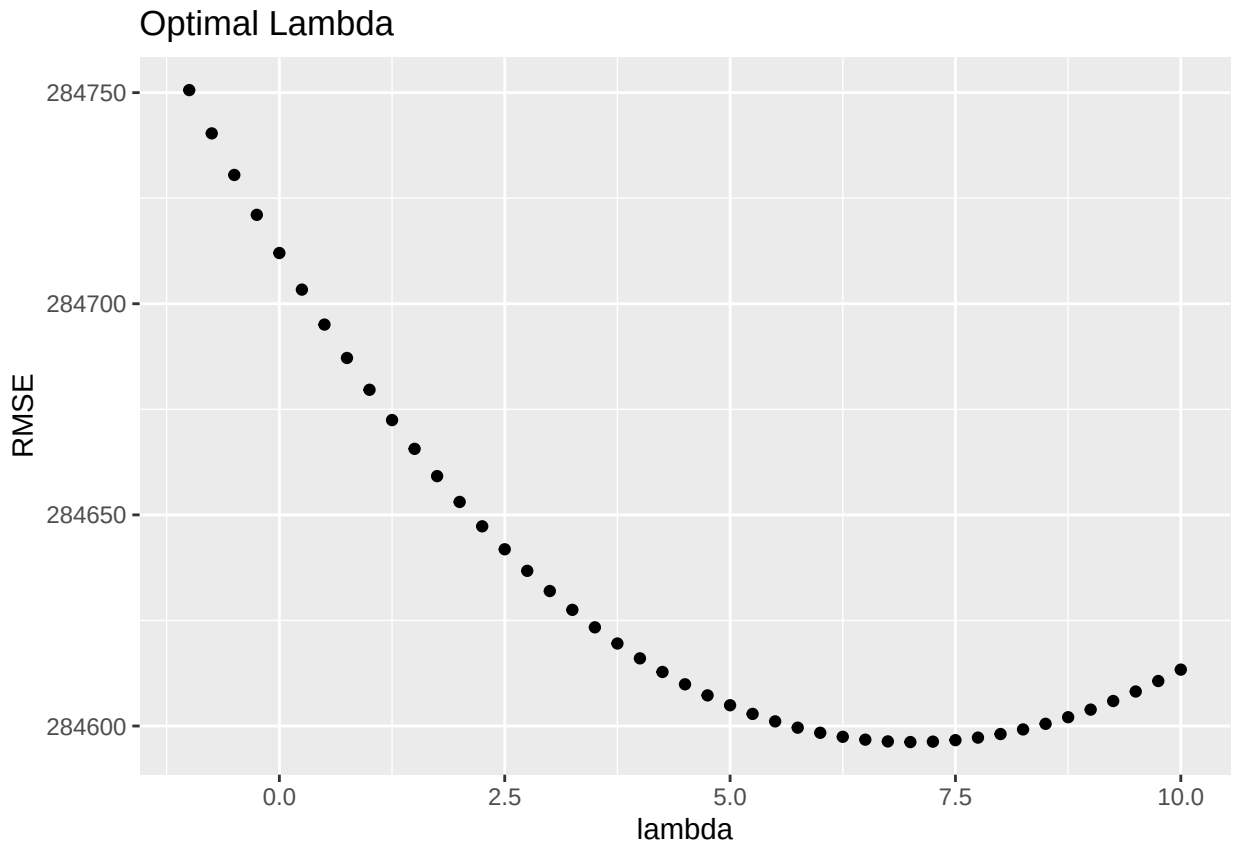
```
#rmse_results_reg %>% knitr::kable()

kable(rmse_results_reg, booktabs = T) %>%
  row_spec(1, color = 'white', background = 'black')
```

Method	RMSE
Regularized SQFT2 Effect	250717.1

Regularized Zip Code

Using the same concept as above we will apply the same process to zip code. When we regularize the zip code model, it slightly improves our result with an RMSE of $284,596.20

```
readRDS(file = "lambdas_qplot_zip.rds")
```




```
lambda_zip <- 7

mu <- mean(trainNF$price)
zip_avgs_reg <- trainNF %>%
  group_by(zipcode) %>%
  summarize(b_zip = sum(price - mu)/(n()+lambda_zip), n_i = n())


predicted_price_regzip <- testNF %>%
  left_join(zip_avgs_reg, by = "zipcode") %>%
  mutate(pred = mu + b_zip) %>%
  pull(pred)
RegularizedZip <- RMSE(predicted_price_regzip, testNF$price)
rmse_results_reg <- bind_rows(rmse_results_reg,
                          tibble(Method="Regularized Zip Effect",
                                 RMSE = RegularizedZip ))
#rmse_results %>% knitr::kable()
```

```
kable(rmse_results_reg, booktabs = T) %>%
  row_spec(2, color = 'white', background = 'black')
```

Method	RMSE
Regularized SQFT2 Effect	250717.1
Regularized Zip Effect	284596.2

Regularized Square Footage + Regularized Zip Code

Now that we have built regularized models of square footage and zip code we can combine the two. When we use the regularized models our RMSE is $213,943.30 which is a slight improvement over our non-regularized model.

```
### regularized zip + regularized sqft
mu <- mean(trainNF$price)

sqft2_avgs_reg2 <- trainNF %>%
  group_by(sqft2) %>%
  summarize(b_sqft2 = sum(price - mu)/(n()+lambda_sqft2))

zip_avgs_reg2 <- trainNF %>%
  left_join(sqft2_avgs_reg2, by="sqft2") %>%
  group_by(zipcode) %>%
  summarize(b_zip = sum(price - mu)/(n()+lambda_zip), n_i = n())


predicted_price_regsqftzip <- testNF %>%
  left_join(sqft2_avgs_reg2, by = "sqft2") %>%
  left_join(zip_avgs_reg2, by = "zipcode") %>%
  mutate(pred = mu + b_sqft2 + b_zip) %>%
  pull(pred)
RegularizedSQFTZip <- RMSE(predicted_price_regsqftzip, testNF$price)
rmse_results_reg <- bind_rows(rmse_results_reg,
                          tibble(Method="Regularized SQFT + Zip",
                                 RMSE = RegularizedSQFTZip ))
# rmse_results %>% knitr::kable()

kable(rmse_results_reg %>% arrange(RMSE), booktabs = T) %>%
  row_spec(1, background = '#E7298A')
```

Method	RMSE
Regularized SQFT + Zip	213943.3
Regularized SQFT2 Effect	250717.1
Regularized Zip Effect	284596.2

I also reviewed regularized grade, and regularized grade plus regularized sqft plus regularized zip, but it did not improve the model

Nearest Neighbours

The next approach to building a price prediction model will be to use "K Nearest Neighbours". A good way to introduce this concept is to ask a question; "How much should a 2,000 square foot house cost?" If we use the closest 5 points to 2,000 square feet, this model would predict an average price of $541,800.

```
readRDS(file = "neighbours_plot.rds")
```


```
#next we can use neighbouring pints to see what the price should be
nearest_neighbours <- trainNF %>%
  mutate(diff = abs(2000 - sqft_living)) %>%
  arrange(diff) %>%
  select(price, bedrooms, bathrooms, sqft_living, zipcode)
  head(5)
```

```
## [1] 5
```

```
nearest_neighbours
```

```
## # A tibble: 19,448 x 5
##     price bedrooms bathrooms sqft_living zipcode
##     <dbl>    <dbl>     <dbl>       <dbl>   <dbl>
##  1 577500        3      2.5         2000   98034
##  2 420000        3      2.25        2000   98008
```

```
##  3 715000        4      1           2000   98103
##  4 456500        4      3.5         2000   98177
##  5 540000        3      2.25        2000   98024
##  6 546000        3      1.75        2000   98117
##  7 450000        4      1           2000   98118
##  8 549000        3      1.75        2000   98117
##  9 229950        5      2.75        2000   98023
## 10 300000        5      2.75        2000   98055
## # ... with 19,438 more rows
```

```
#now we have the 5 closest values, we can take their average
prediction <- nearest_neighbours %>%
  summarise(predicted = mean(price))
prediction
```

```
## # A tibble: 1 x 1
##   predicted
##       <dbl>
## 1   540676.
```

Tuning the KNN Model

Next we will refine the KNN model. This model performs fairly well even with only one predictor, having an RMSE of $256,134.10. This KNN model does not perform as well as the earlier SQFT model.

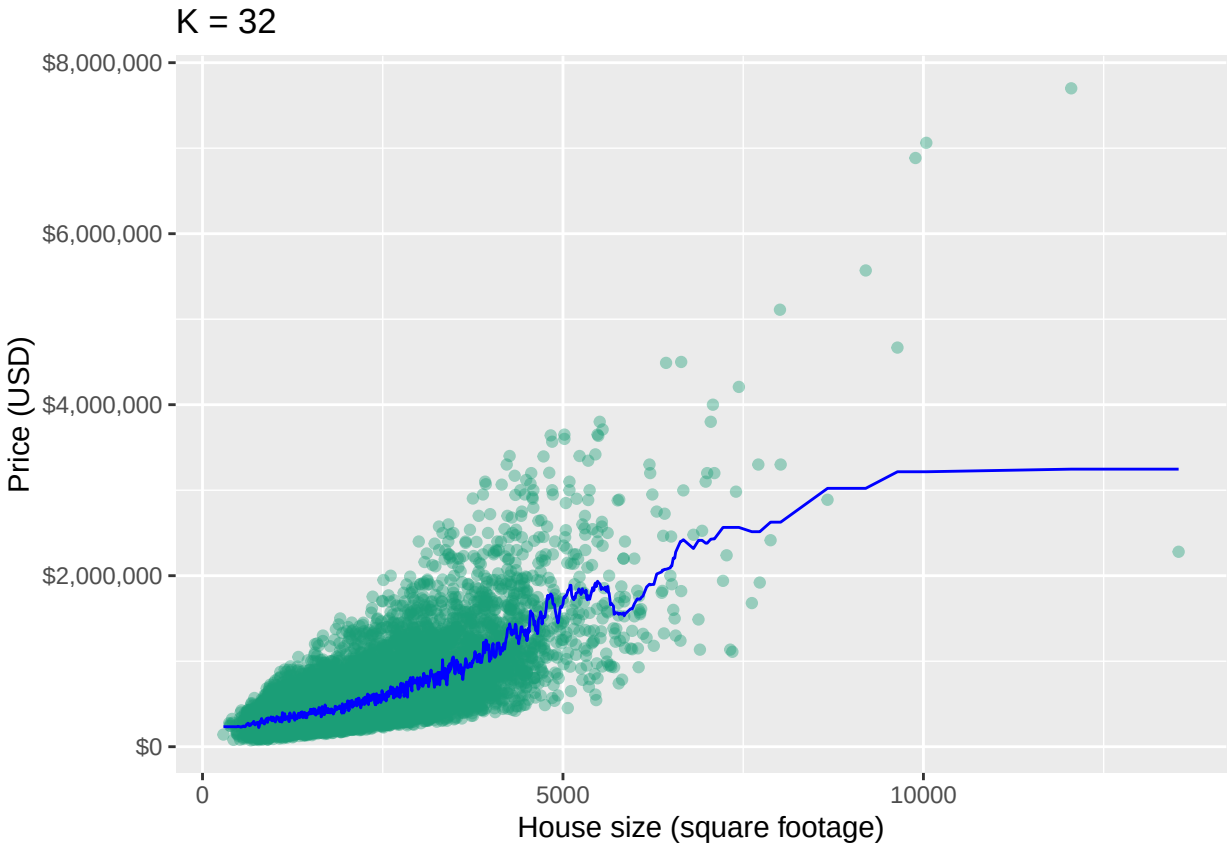
```
#including standardization even though we dont really need to, one variable
king_recipe <- recipe(price ~ sqft_living, data = trainNF) %>%
  step_scale(all_predictors()) %>%
  step_center(all_predictors())

king_spec <- nearest_neighbor(weight_func = "rectangular", neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("regression")

king_vfold <- vfold_cv(trainNF, v = 5, strata = price)

king_wkflw <- workflow() %>%
  add_recipe(king_recipe) %>%
  add_model(king_spec)
king_wkflw

#we are telling the model to use regression

gridvals <- tibble(neighbors = seq(1,200))

king_results <- king_wkflw %>%
  tune_grid(resamples = king_vfold, grid = gridvals) %>%
  collect_metrics()

# show all the results
king_results

# show only the row of minimum RMSPE
```

```
king_min <- king_results %>%
  filter(.metric == 'rmse') %>%
  filter(mean == min(mean))
king_min

###evaluate the test set
set.seed(1234)
kmin <- king_min %>% pull(neighbors)
king_spec <- nearest_neighbor(weight_func = "rectangular", neighbors = kmin) %>%
  set_engine("kknn") %>%
  set_mode("regression")

king_fit <- workflow() %>%
  add_recipe(king_recipe) %>%
  add_model(king_spec) %>%
  fit(data = trainNF)

king_summary <- king_fit %>%
  predict(testNF) %>%
  bind_cols(testNF) %>%
  metrics(truth = price, estimate = .pred)

#king_summary %>% pull(.estimate$rmse)
king_summary_df <- as.data.frame(king_summary)
king_knn <- king_summary_df$.estimate[1]
```

```
king_knn <- 256134.1 #I ran this model outside of the RMD file since it was too resource intensive to r


rmse_results_knn <- tibble(Method="KNN",
                                  RMSE = king_knn )
#rmse_results %>% knitr::kable()

kable(rmse_results_knn, booktabs = T) %>%
  row_spec(1, color = 'white', background = 'black')
```

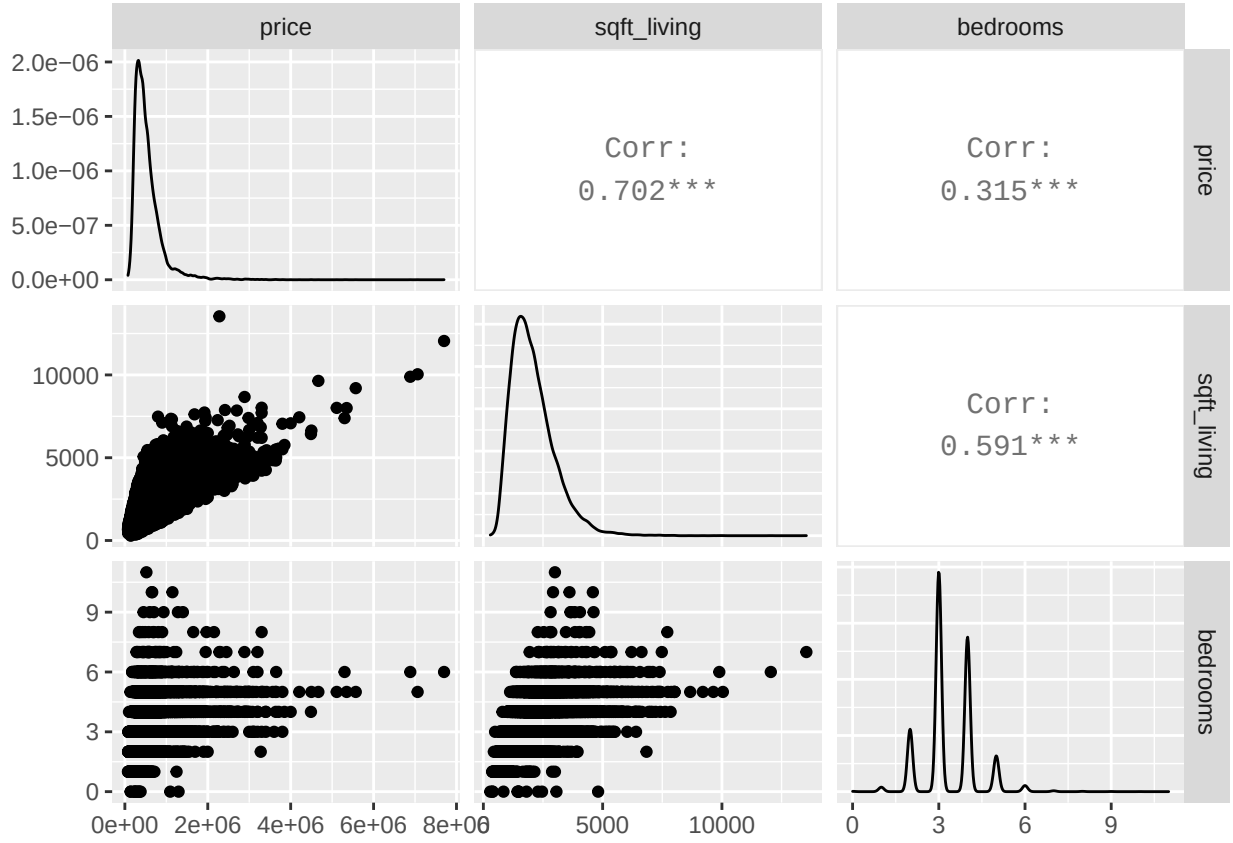
Method	RMSE
KNN	256134.1

We can plot the results of this model to see what it looks like across a variety of house sizes.


K = 32
$8,000,000
$6,000,000
$4,000,000
$2,000,000
$0
Price (USD)
0
5000
10000
House size (square footage)

Multiple KNN

The KNN model will also allow us to use multiple variables. The first step is to take another look at what predictors are correlated with price and we can create a plot to visualize this.



After reviewing the predictors in the previous plot, we can build a new KNN model with both sqft and bedrooms. This model will allow us to use actual square feet (sqft_living) as a continuous variable rather than our sqft2 categorical variable.

```
#build a new model and recipe
king_recipe <- recipe(price ~ sqft_living + bedrooms, data = trainNF) %>%
  step_scale(all_predictors()) %>%
  step_center(all_predictors())

king_spec <- nearest_neighbor(weight_func = "rectangular", neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("regression")

#5 fold cross validation
gridvals <- tibble(neighbors = seq(1,200))
king_k <- workflow() %>%
  add_recipe(king_recipe) %>%
  add_model(king_spec) %>%
  tune_grid(king_vfold, grid = gridvals) %>%
  collect_metrics() %>%
  filter(.metric == 'rmse') %>%
  filter(mean == min(mean)) %>%
```

```
  pull(neighbors)
king_k
#the smallest rmspe occurs when k=16


#retrain the model
king_spec <- nearest_neighbor(weight_func = "rectangular", neighbors = king_k) %>%
  set_engine("kknn") %>%
  set_mode("regression")

knn_mult_fit <- workflow() %>%
  add_recipe(king_recipe) %>%
  add_model(king_spec) %>%
  fit(data = trainNF)

knn_mult_preds <- knn_mult_fit %>%
  predict(testNF) %>%
  bind_cols(testNF)

knn_mult_mets <- metrics(knn_mult_preds, truth = price, estimate = .pred)
knn_mult_mets

knn_mult_mets_df <- as.data.frame(knn_mult_mets)
king_knn_mult <- knn_mult_mets_df$.estimate[1]
```

```
king_knn_mult <- 256143.1 #I ran this model outside of the RMD file since it was too resource intensive

rmse_results_knn <- bind_rows(rmse_results_knn,
                          tibble(Method="KNN Multiple",
                                 RMSE = king_knn_mult ))
#rmse_results %>% knitr::kable()

kable(rmse_results_knn, booktabs = T) %>%
  row_spec(1, color = 'white', background = 'black')
```

Method	RMSE
KNN	256134.1
KNN Multiple	256143.1

KNN Results

The KNN models worked fairly well but did not improve our predictions from the earlier models. The best result came from the first KNN model with an RMSE of $256,134.10.

Method	RMSE
KNN	256134.1
KNN Multiple	256143.1

Regression

Linear Regression

The third category of models that we will explore will be regression. First we will use a linear regression model using square footage to predict price. The model performs reasonably well with an RMSE of $263,507.00 but is not an improvement on our other models.

```
set.seed(1234)
sacramento_split <- initial_split(KingCountyClean, prop = 0.6, strata = price)
sacramento_train <- training(sacramento_split)
sacramento_test <- testing(sacramento_split)


lm_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

lm_recipe <- recipe(price ~ sqft_living, data = trainNF)

lm_fit <- workflow() %>%
  add_recipe(lm_recipe) %>%
  add_model(lm_spec) %>%
  fit(data = trainNF)
lm_fit


lm_test_results <- lm_fit %>%
  predict(testNF) %>%
  bind_cols(testNF) %>%
  metrics(truth = price, estimate = .pred)
lm_test_results  #rmse = $263,507


coeffs <- tidy(pull_workflow_fit(lm_fit))
coeffs
```

We can visualize the results of our linear model by plotting this.

```
## `geom_smooth()` using formula 'y ~ x'
```

Method	RMSE
Linear Regression (SQFT)	263507

Multiple Regression

Continuing with regression, we can now build a model that has multiple predictors. Here we are using square footage, number of bedrooms, and grade of the house. Adding these two variables improves our predictions over our first regression model with an RMSE of $250,179.00 but still does not perform as well as our other models.

```
lm_recipe <- recipe(price ~ sqft_living + bedrooms + grade, data = trainNF)

lm_fit <- workflow() %>%
  add_recipe(lm_recipe) %>%
  add_model(lm_spec) %>%
  fit(data = trainNF)
lm_fit

lm_mult_test_results <- lm_fit %>%
  predict(testNF) %>%
  bind_cols(testNF) %>%
  metrics(truth = price, estimate = .pred)
lm_mult_test_results #rmse = $263,025

coeffs <- tidy(pull_workflow_fit(lm_fit))
coeffs
```

Method	RMSE
Linear Regression (SQFT)	263507
Multi Linear Regression (SQFT+Bedrooms+Grade)	250179

Regression Results

In this case regression was a useful model but did not outperform our earlier models.

Method	RMSE
Multi Linear Regression (SQFT+Bedrooms+Grade)	250179
Linear Regression (SQFT)	263507

CONCLUSION

After working through a variety of different models, the most accurate was combining regularized zip code and regularized categorical square footage. This model resulted in an RMSE of $213,943.30. In terms of the models themselves, this is a reasonable predictor since there was such a wide range of property types and prices. In practical terms however, this model would need to be more accurate in order to be used in real world scenarios since an error of more than $200,000 is quite expensive for most people. In the next section I will identify things to look at in future versions of this model.

Method	RMSE
Regularized SQFT + Zip	213943.3
Zip Effect + SQFT Cat Effect	215332.1
Zip Effect + SQFT Cat Effect + Year Effect	234218.7
Multi Linear Regression (SQFT+Bedrooms+Grade)	250179.0
Regularized SQFT2 Effect	250717.1
SQFT Cat Effect	251256.2
KNN	256134.1
KNN Multiple	256143.1
Grade Effect	258363.9
Linear Regression (SQFT)	263507.0
Regularized Zip Effect	284596.2
Zip Effect	284712.0
Bath Effect	318502.6
View Effect	319624.6
Waterfront Effect	339446.0
Bed Effect	346804.2
Year Effect	357462.1
Just the Average	362261.0

This project was more challanging that I anticipated but it was very enjoyable to work through.

NEXT STEPS

In the next version of this project there are several things that I would like to explore.

Super Luxury Homes

Some of the homes in this data set are super luxury homes, and none of our models do a good job at predicting their sales price. If we remove these super luxury homes from our models we can dramatically improve our accuracy. For example, if we exclude the biggest 0.5% homes (larger than 5,583 square feet) and exclude the 0.1% of most bathrooms (only keeping homes with 5.5 or fewer bathrooms) our "Regularized SQFT + Regularized Zip Code" model improves to $193,631.50 from $213,943.30.

3.5 Bathrooms

To evaluate the accuracy of each model, we reviewed the worst predictions in each case. For our best performing model "Regularized SQFT + Regularized Zip Code" it did a bad job of predicting prices for homes with a large number of bathrooms. There was also a surprising trend where homes with 3.5 bathrooms were valued higher than homes with 4 bathrooms. This is something worth exploring in future versions of this project.

Optimizing Categorical Square Footage

To use square footage as a predictor in the Regularized SQFT + Regularized Zip Code" model, I needed to create a categorical variable that represented the livable square footage. I used bins of 250 square feet at a time, but this number could likely be optimized to a different value.

Mercer Island

Our best performing model overall had a hard time with the Mercer Island zip code. On average, the RMSE within this zip code was nearly $500,000. This could be due to the fact that there are very expensive homes here meaning that the error will be higher, but it is worth exploring more.

Zip Code Border Map

To better visualize the average prices I would like to accurately plot zip code boundries and their average price. I was not able to create a map that had zip code boundaries so instead I used points and the lat/long of each zip code. In future versions I would like to be able to plot based on geographic area, filling the boundary with colour based on price.

Seasonality

It is common to hear of a "hot spring real estate market". I would like to explore seasonality over the course of several years to identify the busiest and slowest times of year for sales. Since there is a supply/demand relationship with housing, is there an ideal time of year to sell in order to get the best price?

Combining Other Factors

In future models I would like to include other data sets that were not part of the real estate sample. Some variables that I would like to experiment with are:

- Interest Rate (do lower rates result in higher prices)
- Income (do zip codes with higher income have higher prices)
- Affordability (what are the most affordable areas - price/income)
- Crime Rate (what impact does crime have on home prices)
- School Quality (what impact does elementary school quality have on home prices)

Real Data

Ultimately I would like to apply this model to real data in real time rather than an existing historical data set. This would likely require a lot of scraping and cleaning but I think it would be worth the effort.