Final Project

Luis Valderrama

12/11/2021

DATAANA 310 A

Follow link below to view final project R Code and write up in Github

github.com/lvalderr/Repository/Data Analysis Essentials Final Project

Contents

Assignment Instructions

Create a multivariate prediction model and perform data analyis of a dataset you choose. This can be either a linear regression, or a classification. This includes:

- 1 point - All code blocks run without error.
- 2 points - Create 2 charts exploring the data with respect to the prediction variable (label)
- 2 points - Create a hypothesis and perform a t.test to reject or fail to reject that hypothesis
- 1 point - Split the data into training set and testing set for each model and wrangle the data as desired
- 3 points - Create 2 prediction models of your chosen type (regression | classification), with at least one multivariate model including visualizing the results of each model
- 2 points - Compare the performance of your models
- 1 point - Include a written analysis of your prediction referencing using data to support your conclusions.

The above is what is required to achieve full credit for the assignment. You are welcome and encouraged to go above and beyond these requirements, just be sure these requirements are fully met first.

R Features

- You are welcome to use any feature covered in this class
- You are welcome to load any library that is already installed in the virtual environment, but you cannot install new packages. You can also reference installed packages by library name::function_name to avoid naming conflicts
- Use set.seed() as necessary to ensure reproducability as the instructor / TA will run the code to grade it
- Ensure your code runs to completion within 60 minutes from start to finish. You may save and load pre-trained models with your instructor's prior permission if you feel you need to exceed this time limit.

Dataset

- Your choice. Be sure the data lends itself to supervised learning, with a label and is appropriate for either regression or classification. Remove any personally identifiable data.
- Suggested data source UCI Machine Learning Repository and look for Data Types == Multivariate and Default Task in (Classification, Regression)
- The data would need to be uploaded to the virtual environment such that the instructor or TA can run the code without error.

```
# Load libraries
# Load any additional libraries. No install.package()
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.
3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflict
s() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()

# set.seed for reproducible results
set.seed(1222)
```

Load and explore data structure

The objective of this exercise is to build a model to predict house prices using the house price data set. The data set contains the prices and features of 21,613 houses.

Note: This data set is provided in the Final Project folder and its named kc_house_data.csv.

After high level review of the attributes and data types displayed below. Along with the label (`price`), there appear to be several features, most useful.

```
# High level view of the data
read_csv ("kc_house_data.csv")

## Rows: 21613 Columns: 21

## -- Column specification ------------------------------------------------
------
## Delimiter: ","
## dbl  (20): id, price, bedrooms, bathrooms, sqft_living, sqft_lot, floors,
```

```
wa...
## dttm  (1): date

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this m
essage.

## # A tibble: 21,613 x 21
##            id date                 price bedrooms bathrooms sqft_living s
qft_lot
##         <dbl> <dttm>               <dbl>    <dbl>     <dbl>       <dbl>
<dbl>
##  1 7129300520 2014-10-13 00:00:00  221900        3      1           1180
5650
##  2 6414100192 2014-12-09 00:00:00  538000        3      2.25        2570
7242
##  3 5631500400 2015-02-25 00:00:00  180000        2      1            770
10000
##  4 2487200875 2014-12-09 00:00:00  604000        4      3           1960
5000
##  5 1954400510 2015-02-18 00:00:00  510000        3      2           1680
8080
##  6 7237550310 2014-05-12 00:00:00 1230000        4      4.5         5420
101930
##  7 1321400060 2014-06-27 00:00:00  257500        3      2.25        1715
6819
##  8 2008000270 2015-01-15 00:00:00  291850        3      1.5         1060
9711
##  9 2414600126 2015-04-15 00:00:00  229500        3      1           1780
7470
## 10 3793500160 2015-03-12 00:00:00  323000        3      2.5         1890
6560
## # ... with 21,603 more rows, and 14 more variables: floors <dbl>,
## #   waterfront <dbl>, view <dbl>, condition <dbl>, grade <dbl>,
## #   sqft_above <dbl>, sqft_basement <dbl>, yr_built <dbl>, yr_renovated <d
bl>,
## #   zipcode <dbl>, lat <dbl>, long <dbl>, sqft_living15 <dbl>, sqft_lot15
<dbl>
```

Data processing

As a first step I read the data from the raw `.csv` file. The function below reads and prepares the data for exploration analysis and establishing a data.frame. Required data munging:

1. The `.csv` file is read, using coding for potential missing values in the raw file.
2. For consistency purposes, replace the _ character in the column names with `.`.

3. Remove four unnecessary columns (lat, long, sqft.living15, sqft.lot15) as they provide little to no value, and cases with potential missing data from the data frame using a dpyr verb pipeline. These verbs are chained by the %>% operator.

The established data frame is named house.price and consists of 17 attributes that are relevant for this analysis and the population size is 21,613 (which matches the original dataset).

```
# Create a function to read in the data
read.house = function(file = "kc_house_data.csv"){

## Read the raw csv file
house.price = read_csv (file, col_names = TRUE, na = c('?', '', NA))

# remove the '-' character from the column names
names(house.price) = str_replace_all(names(house.price), '_', '.')

# Remove unwanted columns: lat, long
# Remove the rows with missing values
    house.price <- house.price %>%
                   select (-lat, -long, -sqft.living15, -sqft.lot15) %>%
                   filter( complete.cases (.))
# Return the data frame
    house.price
}

# Call the read.house() function and
# Store the results on house.price
house.price = read.house ()

## Rows: 21613 Columns: 21

## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## dbl  (20): id, price, bedrooms, bathrooms, sqft_living, sqft_lot, floors,
wa...
## dttm  (1): date

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this m
essage.

# Explore the result
glimpse(house.price)

## Rows: 21,613
## Columns: 17
## $ id            <dbl> 7129300520, 6414100192, 5631500400, 2487200875, 1954
```

```
4005~
## $ date          <dttm> 2014-10-13, 2014-12-09, 2015-02-25, 2014-12-09, 201
5-02~
## $ price         <dbl> 221900, 538000, 180000, 604000, 510000, 1230000, 257
500,~
## $ bedrooms      <dbl> 3, 3, 2, 4, 3, 4, 3, 3, 3, 3, 3, 2, 3, 3, 5, 4, 3, 4
, 2,~
## $ bathrooms     <dbl> 1.00, 2.25, 1.00, 3.00, 2.00, 4.50, 2.25, 1.50, 1.00
, 2.~
## $ sqft.living   <dbl> 1180, 2570, 770, 1960, 1680, 5420, 1715, 1060, 1780,
189~
## $ sqft.lot      <dbl> 5650, 7242, 10000, 5000, 8080, 101930, 6819, 9711, 7
470,~
## $ floors        <dbl> 1.0, 2.0, 1.0, 1.0, 1.0, 1.0, 2.0, 1.0, 1.0, 2.0, 1.
0, 1~
## $ waterfront    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0,~
## $ view          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0
, 0,~
## $ condition     <dbl> 3, 3, 3, 5, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 4
, 4,~
## $ grade         <dbl> 7, 7, 6, 7, 8, 11, 7, 7, 7, 7, 8, 7, 7, 7, 7, 9, 7,
7, 7~
## $ sqft.above    <dbl> 1180, 2170, 770, 1050, 1680, 3890, 1715, 1060, 1050,
189~
## $ sqft.basement <dbl> 0, 400, 0, 910, 0, 1530, 0, 0, 730, 0, 1700, 300, 0,
0, ~
## $ yr.built      <dbl> 1955, 1951, 1933, 1965, 1987, 2001, 1995, 1963, 1960
, 20~
## $ yr.renovated  <dbl> 0, 1991, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0,~
## $ zipcode       <dbl> 98178, 98125, 98028, 98136, 98074, 98053, 98003, 981
98, ~
```

Data Transformation

The two relevant features, 'view' and 'waterfront' have various sub-classifications that may not be relevant to this analysis as we are not narrowing the data down into special details such as waterfront and/or view scale. Instead, I want to know if the house has view and/or waterfront feature Yes/No.

```
# Transforming the 'waterfront' attribute to: 0 = "no_waterfront", else "yes_
waterfront"
house.price <- house.price %>% mutate(waterfront_yes_no = if_else(waterfront
== 0, "no", "yes")) %>% mutate_if(is.character, as.factor)

# Transforming the 'view' attribute to: 0 = "no_view", else "yes_view"
house.price <- house.price %>% mutate(view_yes_no = if_else(view == 0, "no",
"yes")) %>% mutate_if(is.character, as.factor)
```

```
house.price <- house.price %>% mutate(view_TRUE_FALSE = if_else(view_yes_no =
= "yes", TRUE, FALSE) %>% as.factor())
# Review changes
house.price %>% glimpse()
```

```
## Rows: 21,613
## Columns: 20
## $ id                <dbl> 7129300520, 6414100192, 5631500400, 2487200875,
1954~
## $ date              <dttm> 2014-10-13, 2014-12-09, 2015-02-25, 2014-12-09,
201~
## $ price             <dbl> 221900, 538000, 180000, 604000, 510000, 1230000,
257~
## $ bedrooms          <dbl> 3, 3, 2, 4, 3, 4, 3, 3, 3, 3, 3, 2, 3, 3, 5, 4,
3, 4~
## $ bathrooms         <dbl> 1.00, 2.25, 1.00, 3.00, 2.00, 4.50, 2.25, 1.50,
1.00~
## $ sqft.living       <dbl> 1180, 2570, 770, 1960, 1680, 5420, 1715, 1060, 1
780,~
## $ sqft.lot          <dbl> 5650, 7242, 10000, 5000, 8080, 101930, 6819, 971
1, 7~
## $ floors            <dbl> 1.0, 2.0, 1.0, 1.0, 1.0, 1.0, 2.0, 1.0, 1.0, 2.0
, 1.~
## $ waterfront        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0~
## $ view              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3,
0, 0~
## $ condition         <dbl> 3, 3, 3, 5, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,
3, 4~
## $ grade             <dbl> 7, 7, 6, 7, 8, 11, 7, 7, 7, 7, 8, 7, 7, 7, 7, 9,
7, ~
## $ sqft.above        <dbl> 1180, 2170, 770, 1050, 1680, 3890, 1715, 1060, 1
050,~
## $ sqft.basement     <dbl> 0, 400, 0, 910, 0, 1530, 0, 0, 730, 0, 1700, 300
, 0,~
## $ yr.built          <dbl> 1955, 1951, 1933, 1965, 1987, 2001, 1995, 1963,
1960~
## $ yr.renovated      <dbl> 0, 1991, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0~
## $ zipcode           <dbl> 98178, 98125, 98028, 98136, 98074, 98053, 98003,
981~
## $ waterfront_yes_no <fct> no, no, no, no, no, no, no, no, no, no, no, no,
no, ~
## $ view_yes_no       <fct> no, no, no, no, no, no, no, no, no, no, no, no,
no, ~
## $ view_TRUE_FALSE   <fct> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,
FAL~
```

```
# Add log and square root columns in the form of column_name.log and column_n
ame.sqrt
# for the following: engine.size, city.mpg
# Update the existing data frame with the 4 new columns
house.price = house.price %>% mutate (price.log = log (price))
```

Explore prediction variable

As mentioned above I have identified the Label to be the 'Price' of houses. There are many attributes in this dataset to which can serve as prediction variables, and the prediction model can be very complex and/or there could be many potential prediction models. For my analysis, the prediction models are segmented based on:

1. House Geographical Feature = zip code
2. House Basic Features = bedrooms, bathrooms, sqft of living space, lot
3. House Age/Condition Features = grade, built year, renovated year
4. House Special Features = view, waterfront

Exploration Charts Group

The function code below plots the histogram and density estimate for the feature specified on each exploration chart to follow.

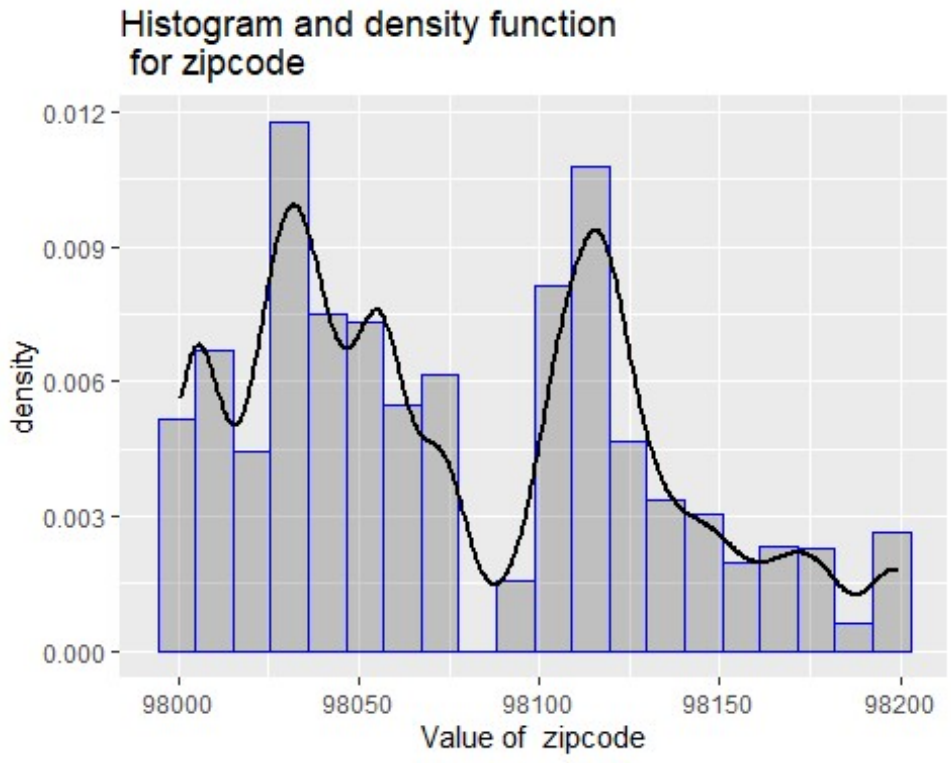
```
# Creating a function plot.hist to plot
# a histogram and a density plot
plot.hists = function(col, house.price, bins = 20){
  require(ggplot2)
  p1 = ggplot(house.price, aes_string(col)) +
       geom_histogram (aes(y = ..density..), bins = bins,
                       alpha = 0.3, color = 'blue') +
       geom_density (size = 1) +
       xlab(paste('Value of ', col)) +
       ggtitle(paste('Histogram and density function \n for', col))
       print(p1)
}

# Create a list of columns of interest
cols.geographical=c('price', 'zipcode')
cols.basicfeatures=c('price', 'bedrooms', 'bathrooms', 'sqft.living', 'sqft.l
ot', 'sqft.basement')
cols.agecondition=c('price', 'floors', 'grade', 'yr.built', 'yr.renovated')
cols.specialfeatures=c('price', 'waterfront', 'view')

# Creating a plotting function for points and a trend line
# Set y to price
plot.feature = function(col, house.price){
    p1 = ggplot(house.price, aes_string(x = col, y = 'price')) +
            geom_point () +
```

```
            geom_smooth(size = 1, color = 'red') +
            xlab(col) + ylab('Price') +
            ggtitle(paste('Relationship between ', col, ' and price'))


    # Print the plot
    p1 %>% print()
}
# Create a list of columns of interest
cols.geographical=c('zipcode')
cols.basicfeatures=c('bedrooms', 'bathrooms', 'sqft.living', 'sqft.lot', 'sqf
t.basement')
cols.agecondition=c('floors', 'grade', 'yr.built', 'yr.renovated')
cols.specialfeatures=c('waterfront', 'view')
```

Chart 1.1

The first exploration is to look at the distribution of the label and geographical feature. As it is commonly known, using geographical feature as a predictor for house pricing would be considered relevant and an important feature. However, the data set we are using in our analysis is limited as it presents the zip code as the primary geographical identifier, which is very granular and broad, and difficult to aggregate it into meaningful attributes such as states, county, city without having major data transformation. Therefore, I consider that using zip code as the sole feature would not result in meaningful values. Although, the zip code should be considered as a feature.

```
# Loop through each column and call the plot function
# 1.1 Price in relation to geographical feature
invisible(lapply(cols.geographical,plot.hists, house.price))
```




```
# Loop through each column and call the plot function
invisible(lapply(cols.geographical, plot.feature, house.price))

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

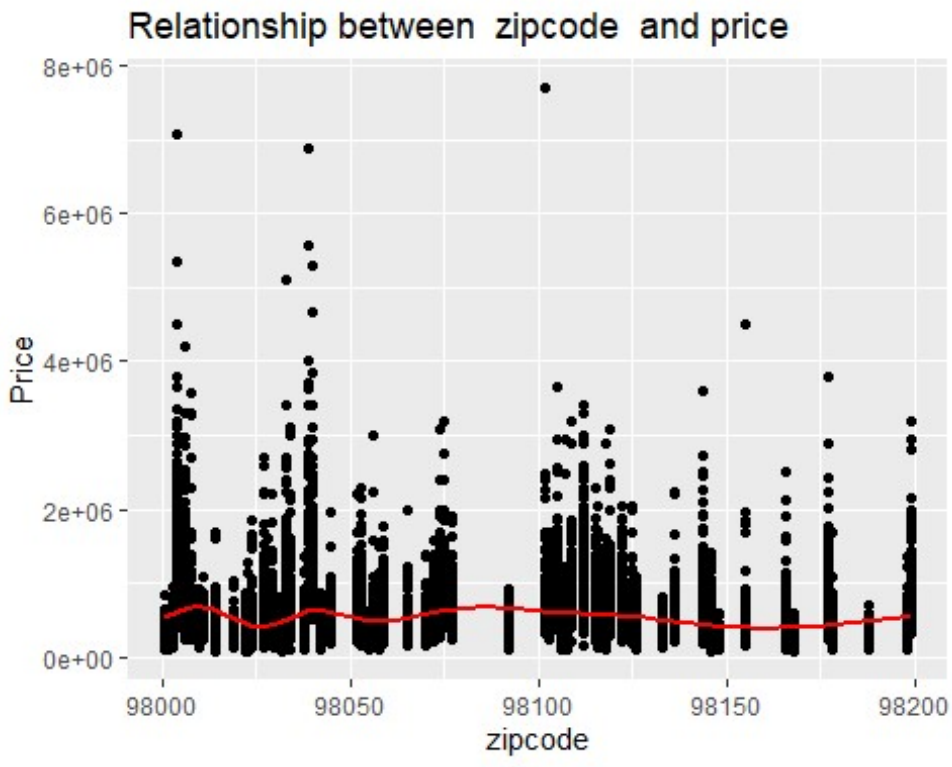
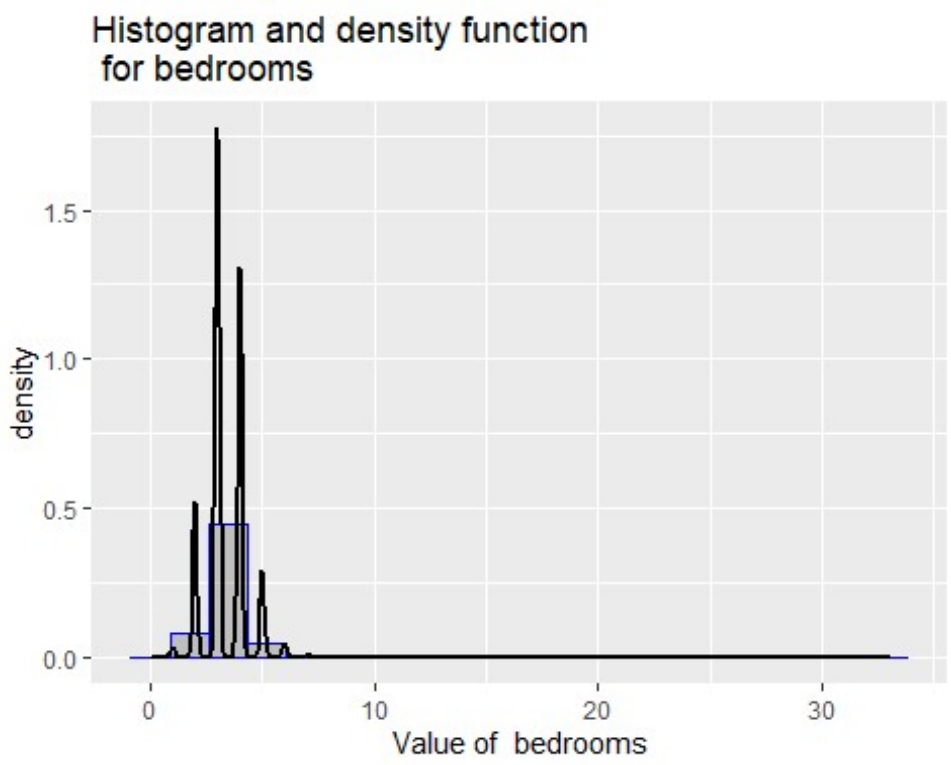
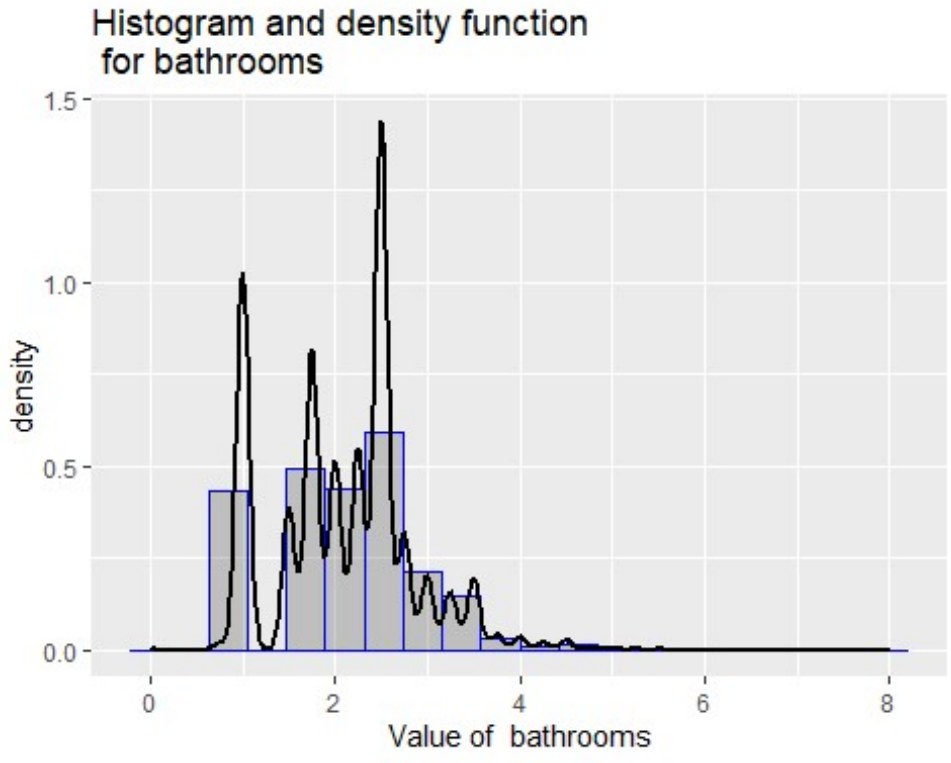


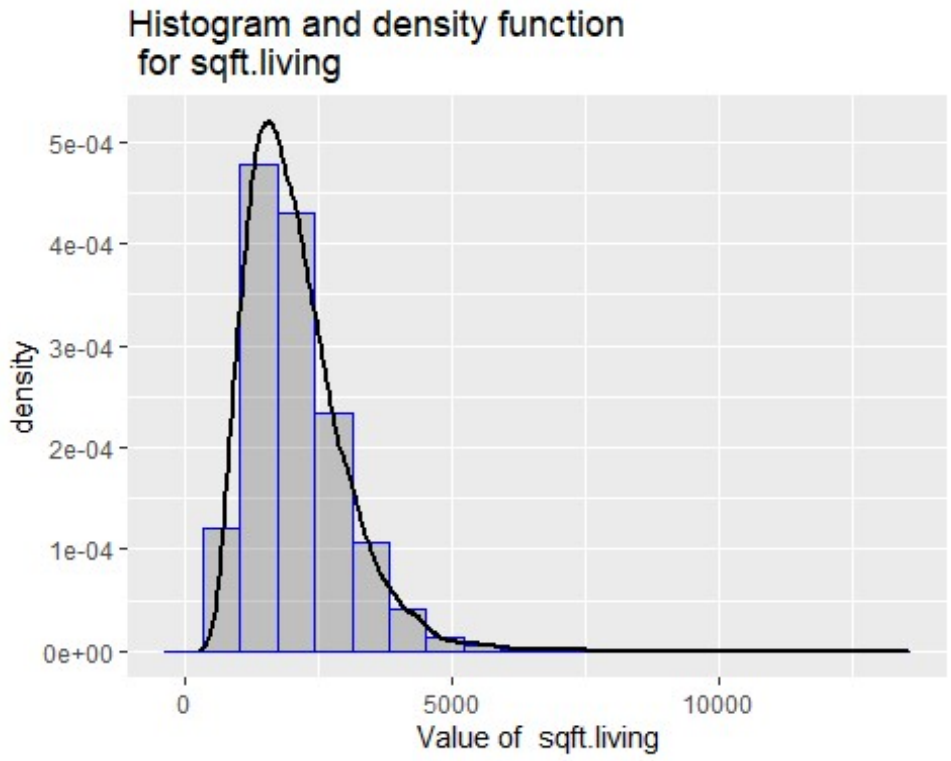
Chart 1.2

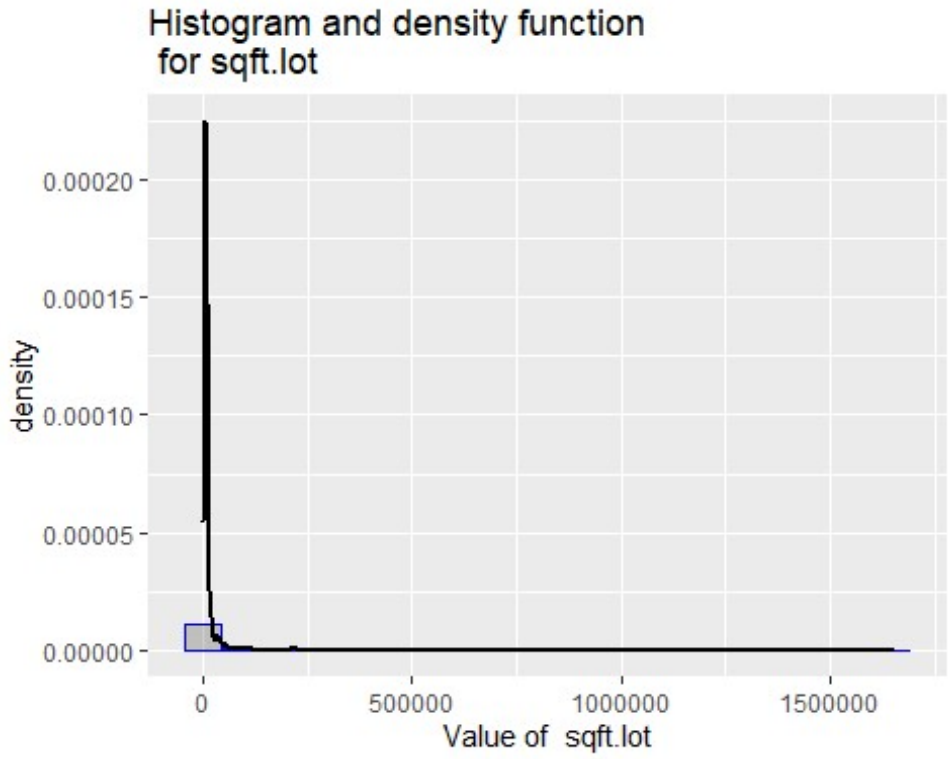
The second exploration is to look at the distribution of the label and basic features. The chart reveals promising house price features such as square footage of living space, lot size, and basement as well as number of bathrooms and bedrooms. These features are common in most if not all houses and it is known that they can influence the house price. I acknowledge that not all houses have basements and square footage may be difficult to classify.

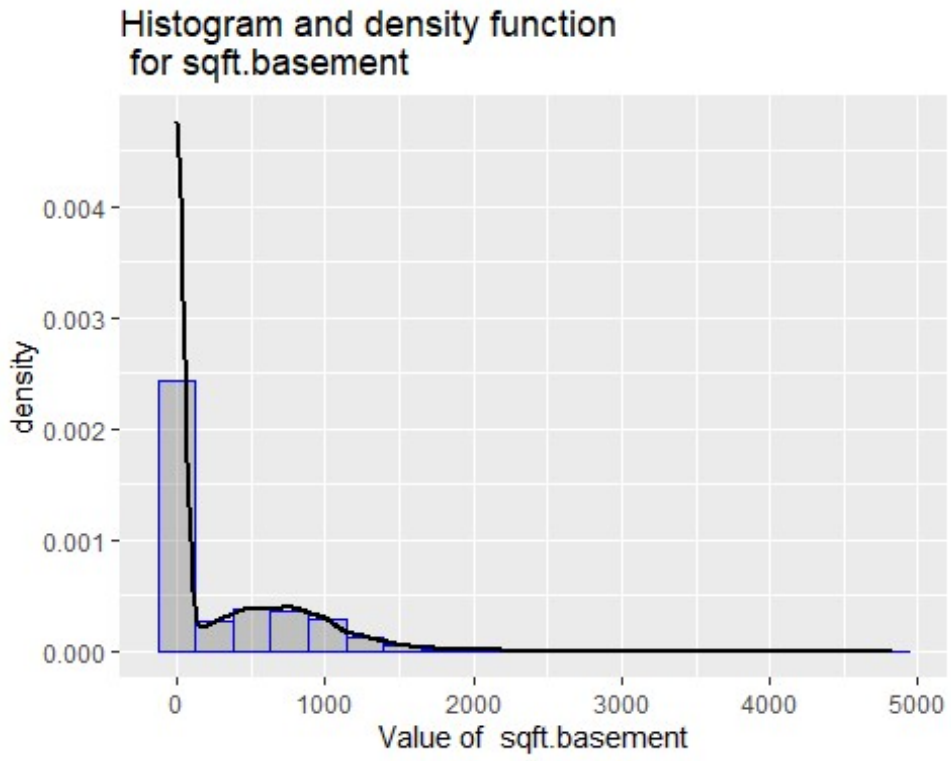
```
# Loop through each column and call the plot function
# 1.2 Price in relation to house basic features
invisible(lapply(cols.basicfeatures,plot.hists, house.price))
```


Histogram and density function
for bedrooms
density
1.5
1.0
0.5
0.0
0
10
20
30
Value of bedrooms

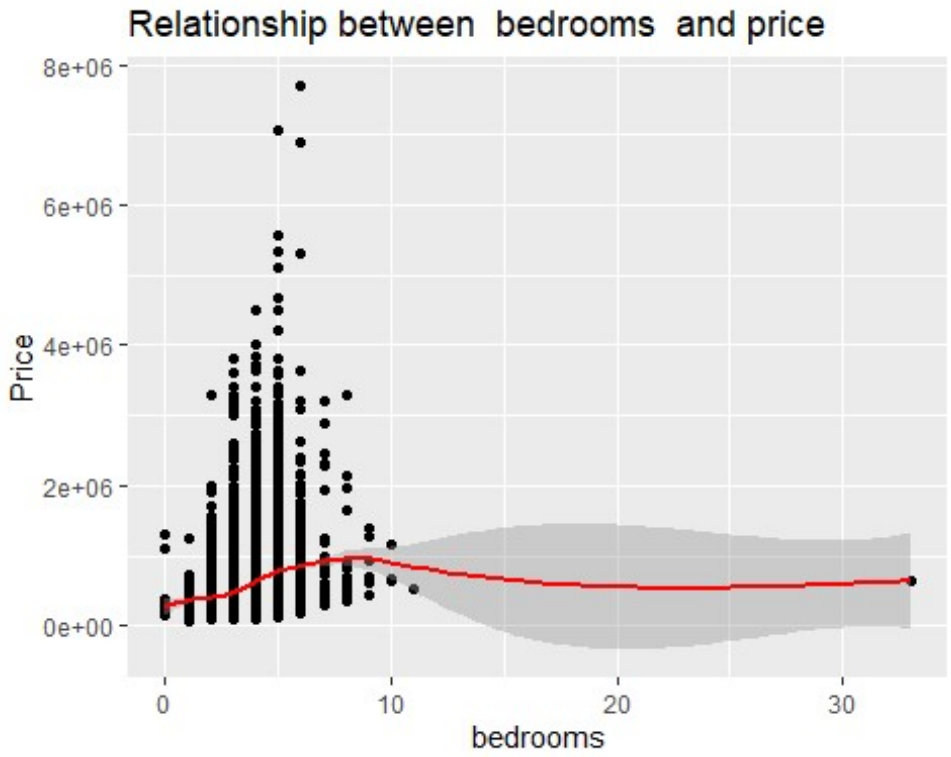

Histogram and density function
for bathrooms
density
1.5
1.0
0.5
0.0
0
2
4
6
8
Value of bathrooms


Histogram and density function
for sqft.living
density
5e-04
4e-04
3e-04
2e-04
1e-04
0e+00
0
5000
10000
Value of sqft.living

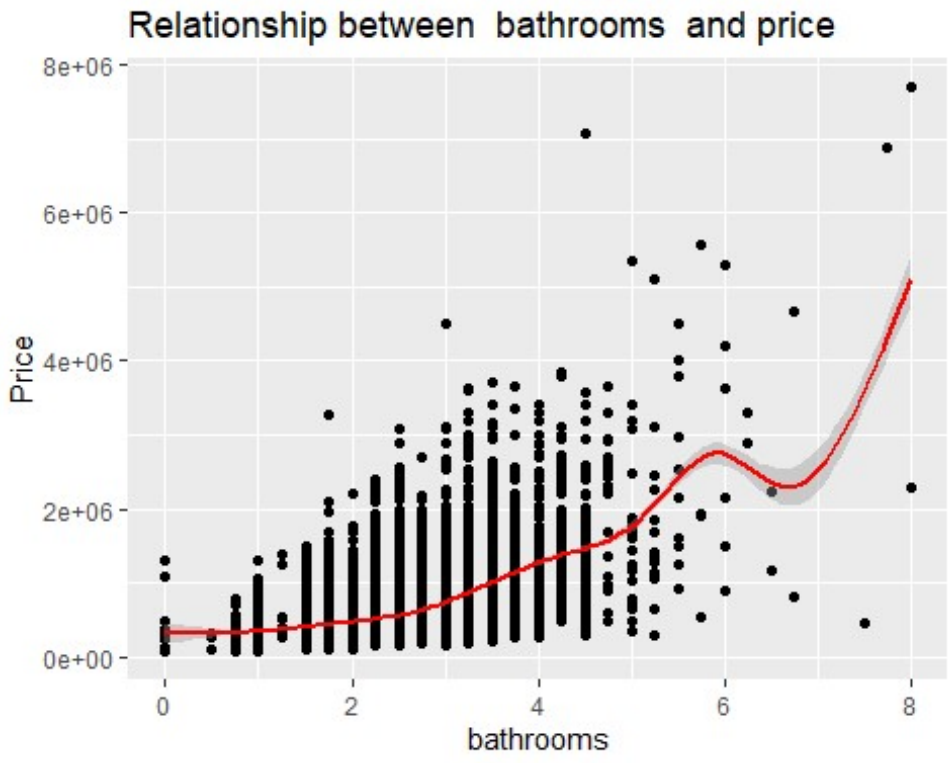

Histogram and density function
for sqft.lot
density
0.00020
0.00015
0.00010
0.00005
0.00000
0
500000
1000000
1500000
Value of sqft.lot



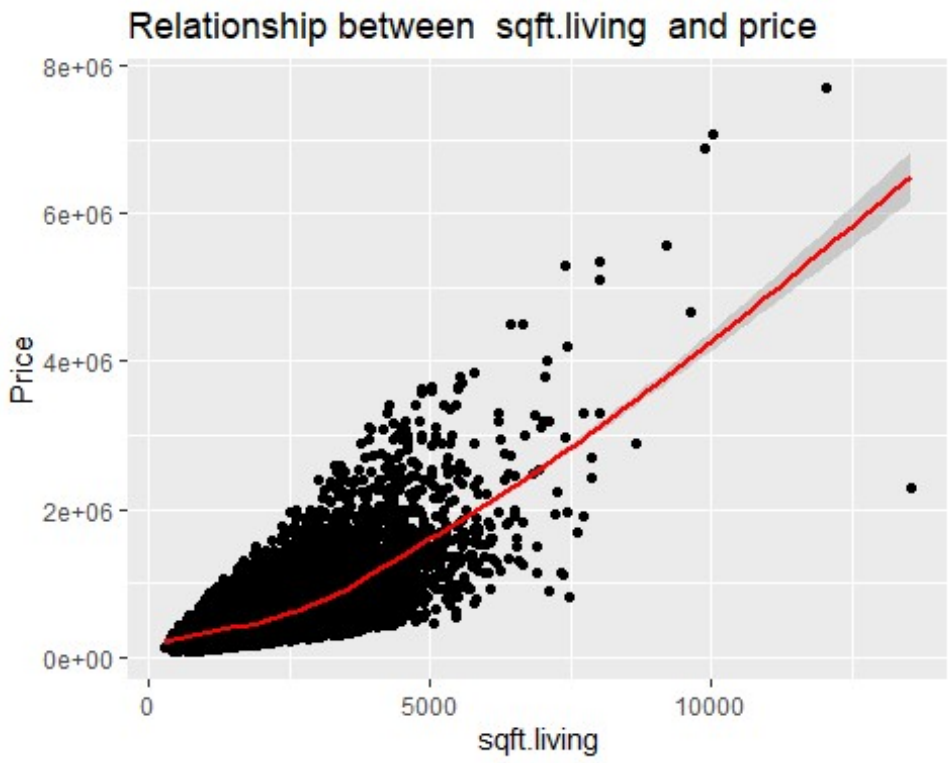

```
# Loop through each column and call the plot function
invisible(lapply(cols.basicfeatures, plot.feature, house.price))

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



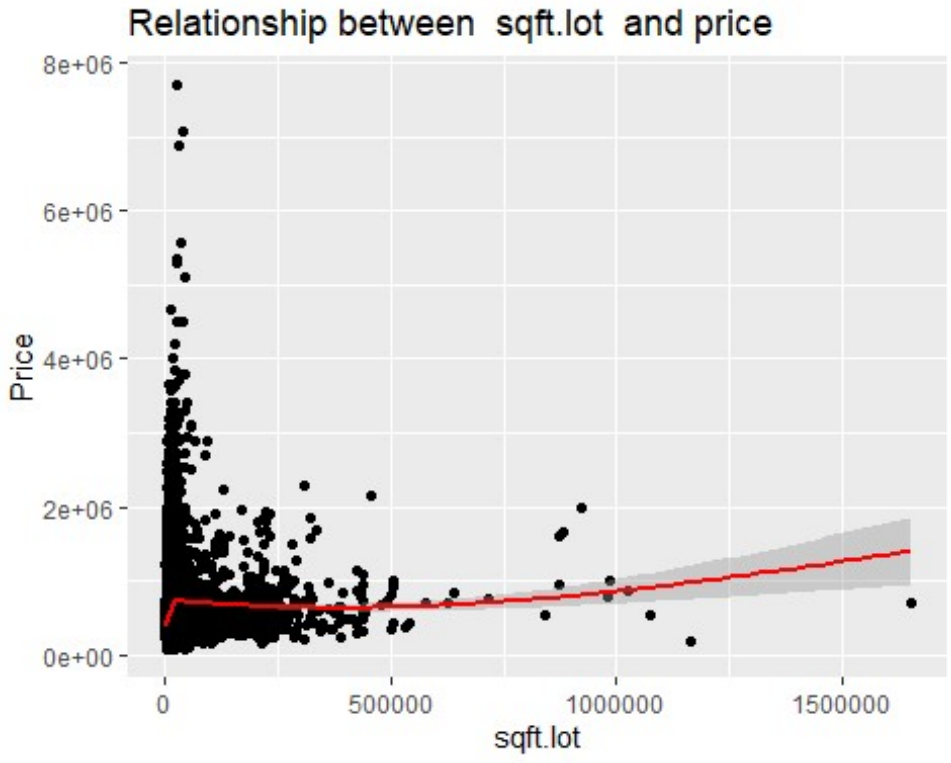

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

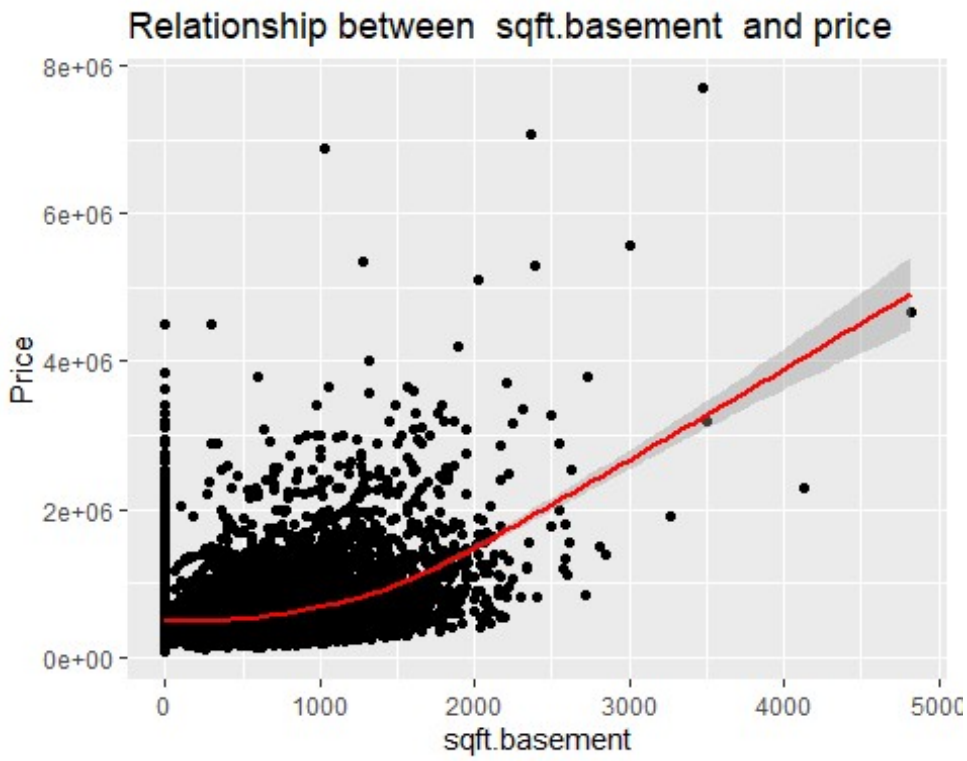
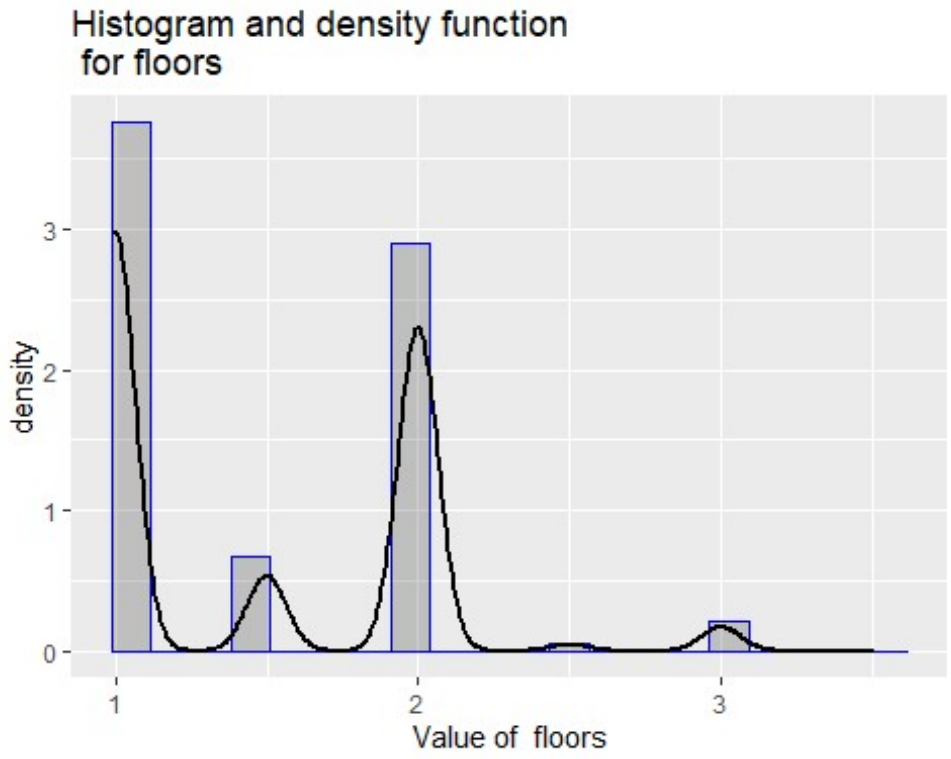
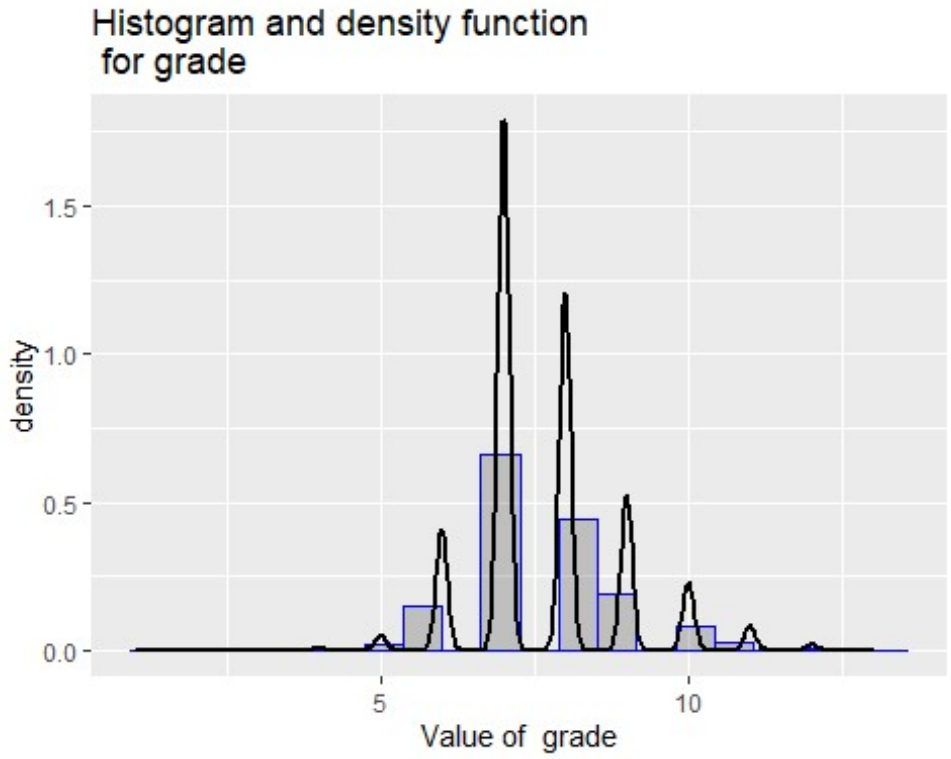


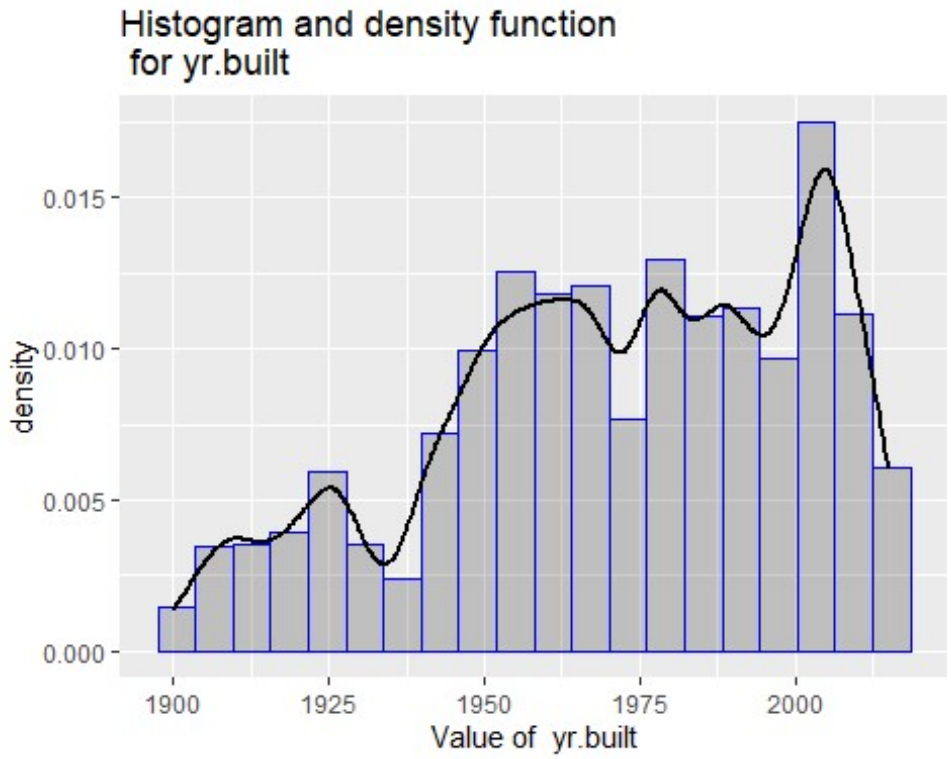
Chart 1.3

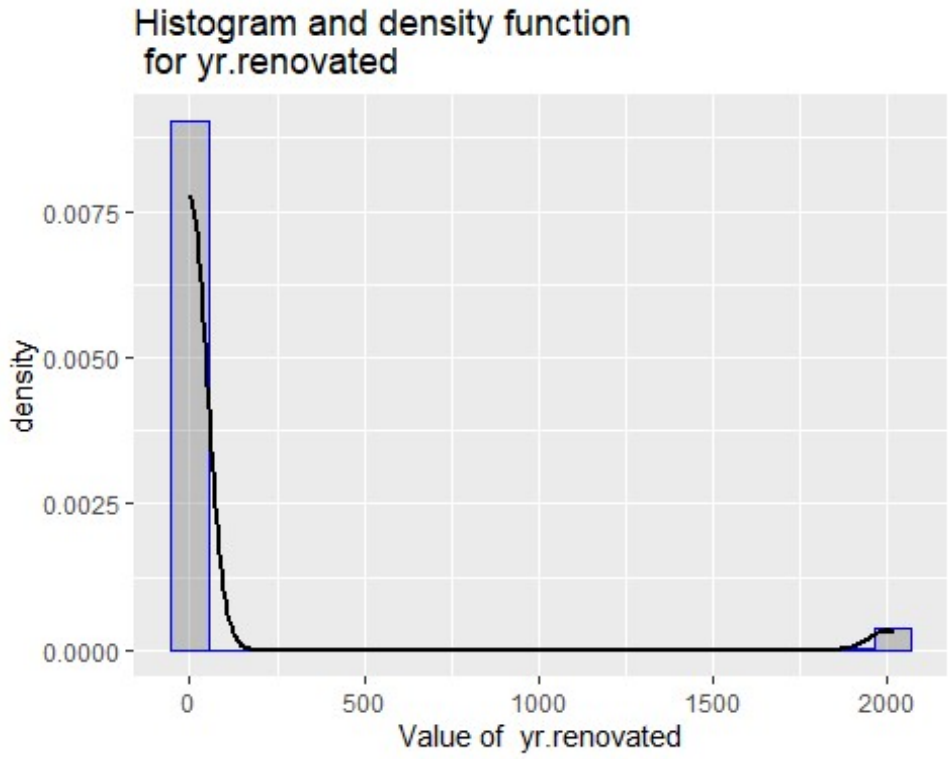
The third exploration is to look at the distribution of the label and age/condition features. Another set of features influencing the house price are year the house was built and/or renovated as well as built grade.

```
# Loop through each column and call the plot function
# 1.3 Price in relation to house age and condition
invisible(lapply(cols.agecondition,plot.hists, house.price))
```


Histogram and density function
for floors
density
0
1
2
3
1
2
3
Value of floors

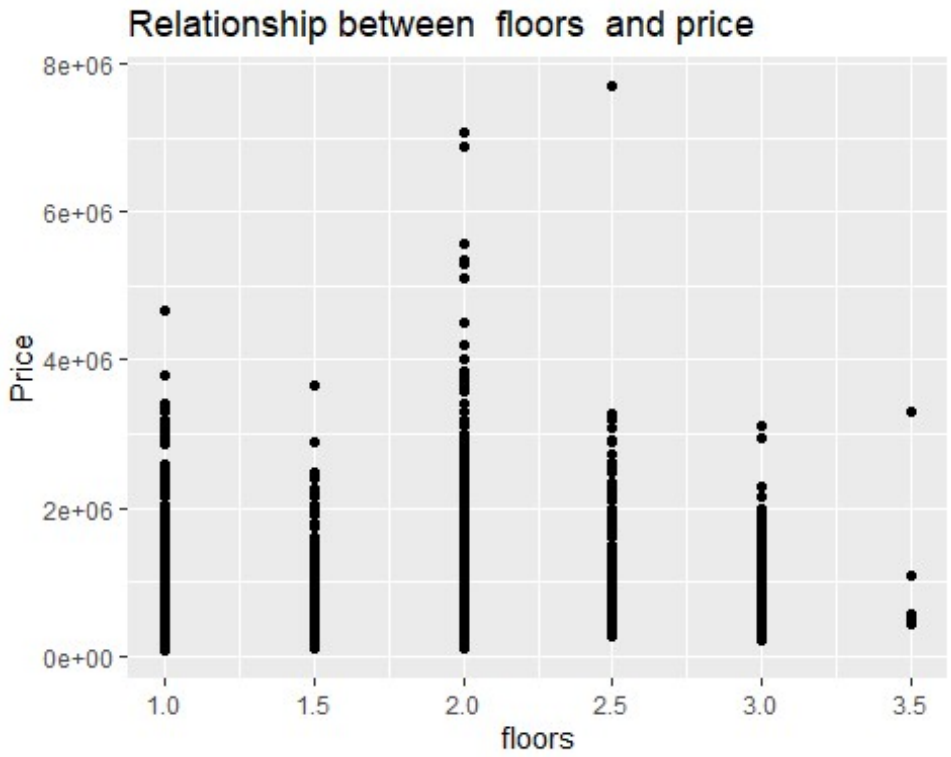

Histogram and density function
for grade
density
0.0
0.5
1.0
1.5
5
10
Value of grade



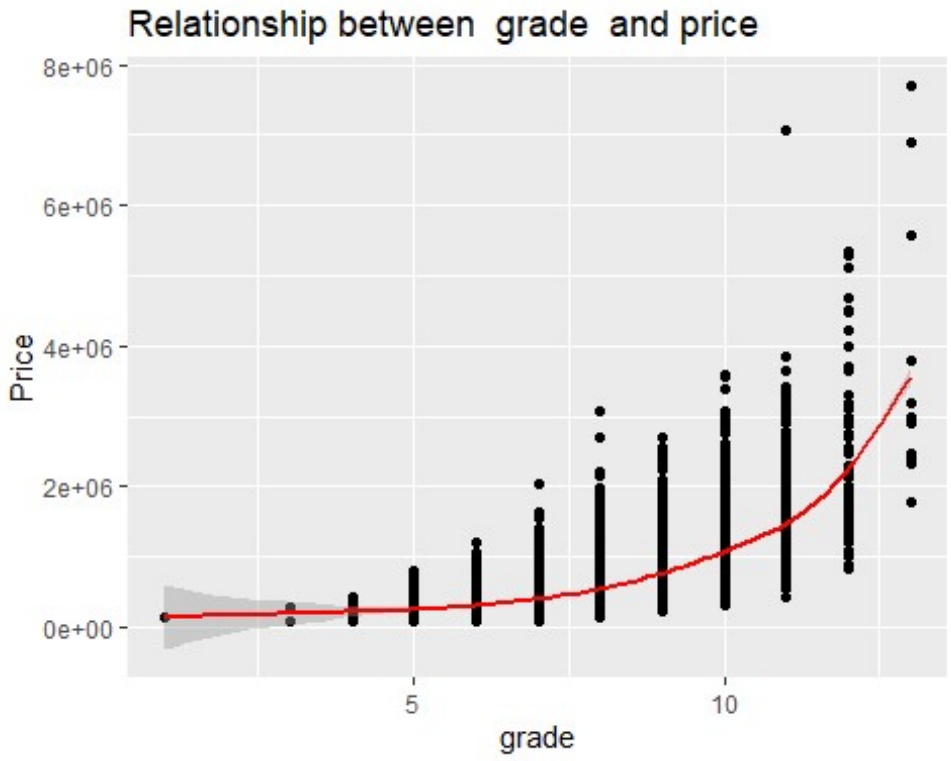



```
# Loop through each column and call the plot function
invisible(lapply(cols.agecondition, plot.feature, house.price))

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

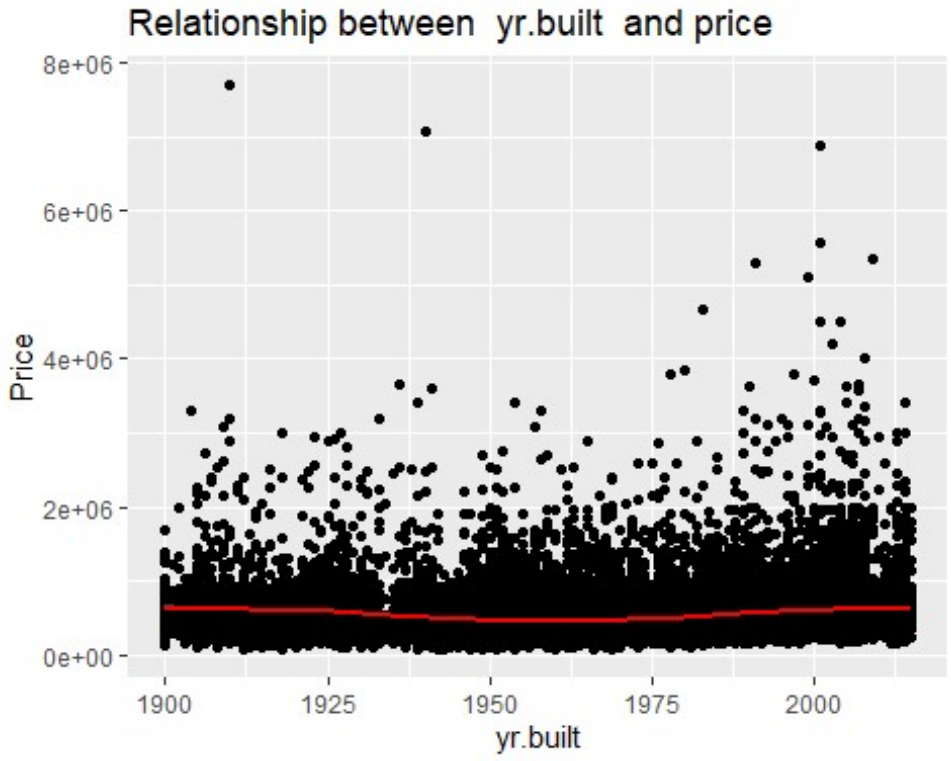
```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

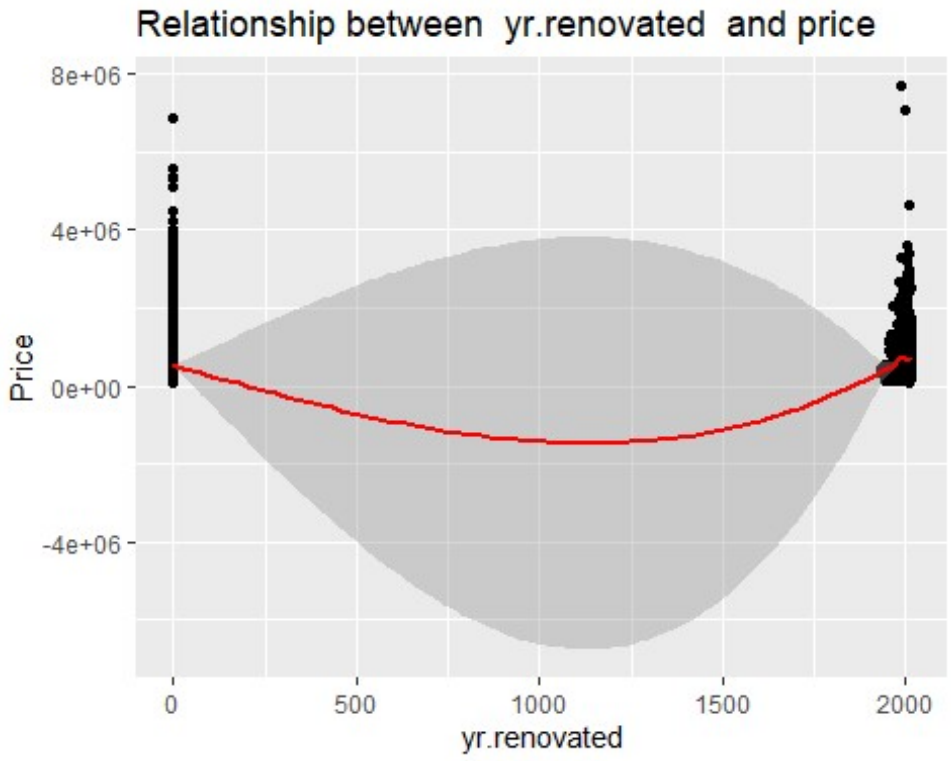
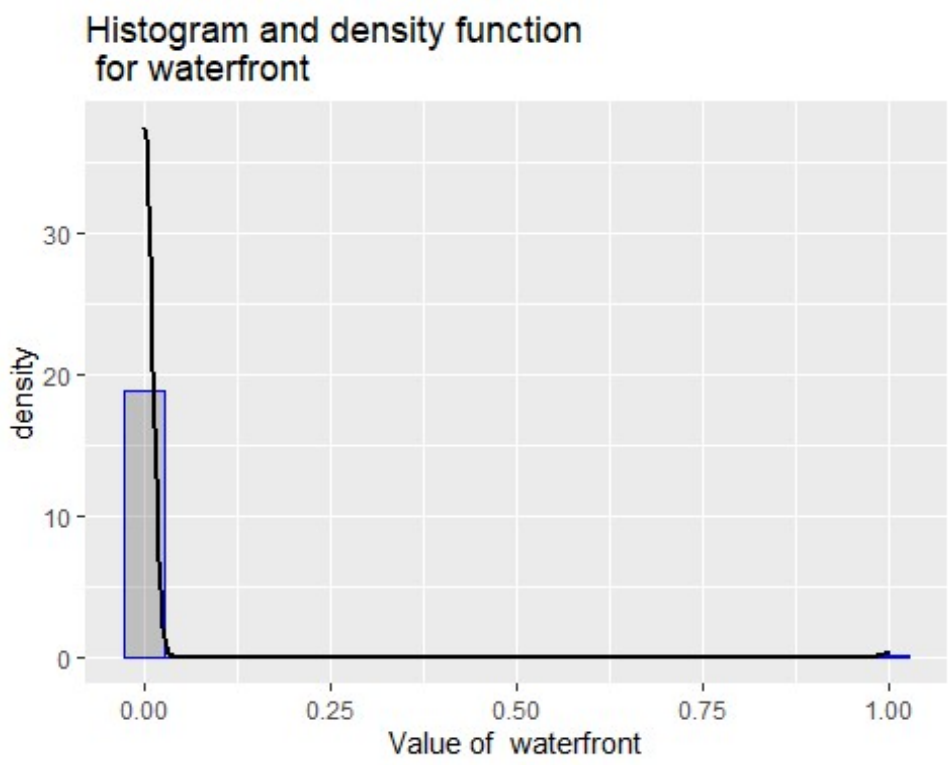
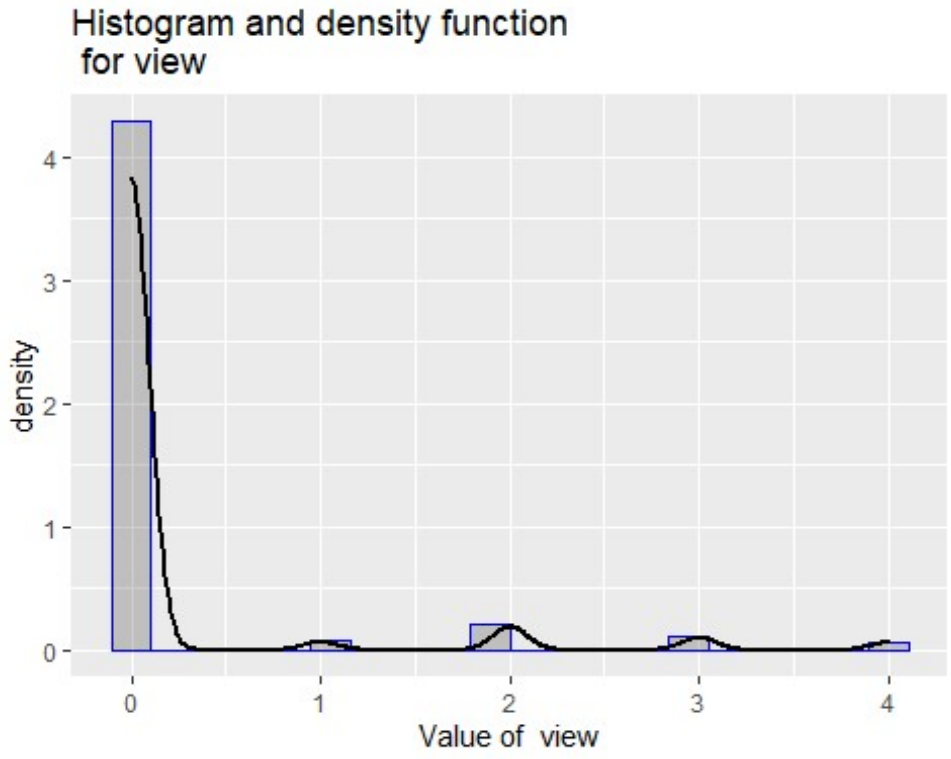


Chart 1.4

The fourth exploration is to look at the distribution of the label and special features. This chart quantifies and confirms that special features such as having a house with a view, or waterfront may be unique, desirable and will influence the house price. These special features may be in some cases almost impossible to change while the structure of the house can be reasonably renovated or upgraded to the buyers' specifications. This data set provides various levels of waterfront and views which can add complexity to my analysis. To reduce the complexity of the analysis I have limited the view/waterfront features to a simple yes/no presented in the violin plots below. The special features present good potential for a price prediction alternative.

```
# Loop through each column and call the plot function
# 1.4 Price in relation to house special features
invisible(lapply(cols.specialfeatures,plot.hists, house.price))
```

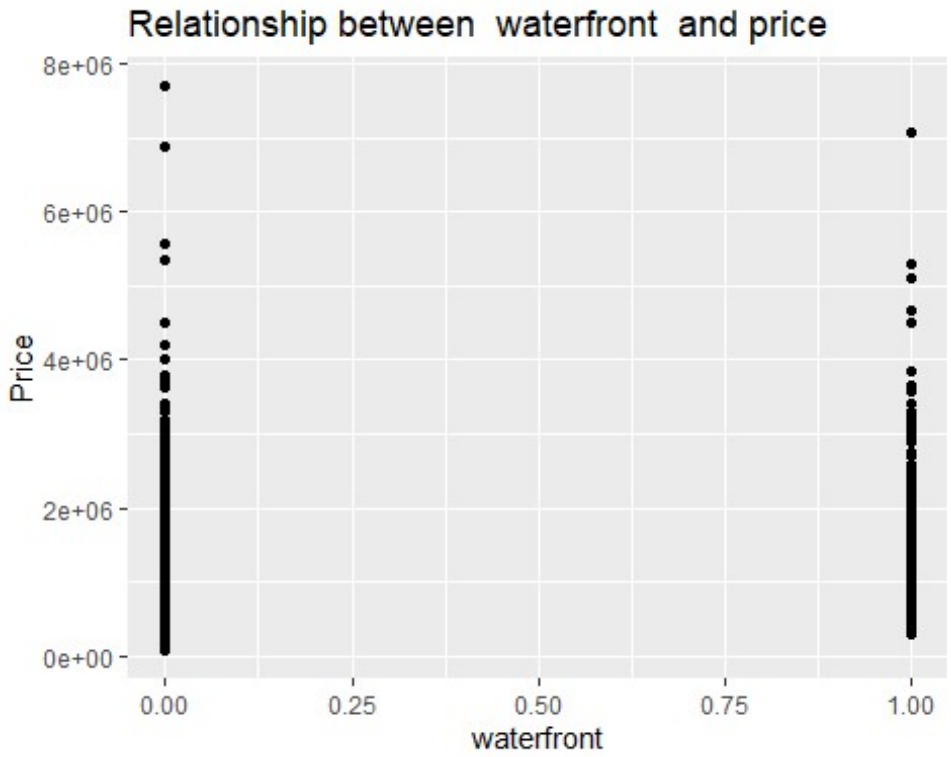




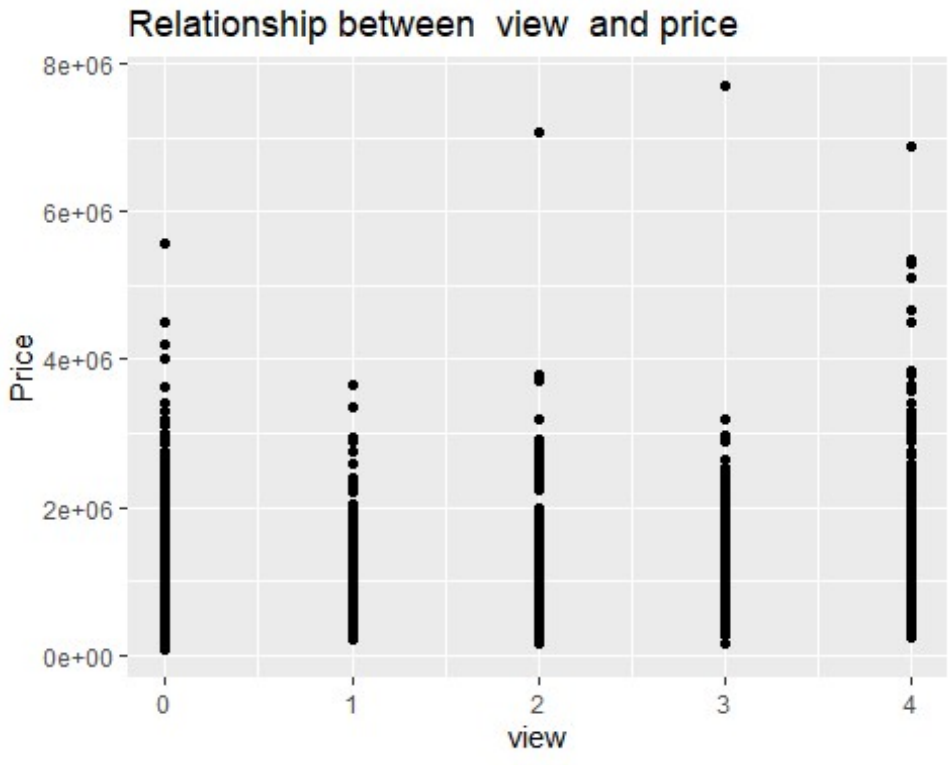

```
# Loop through each column and call the plot function
invisible(lapply(cols.specialfeatures, plot.feature, house.price))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```



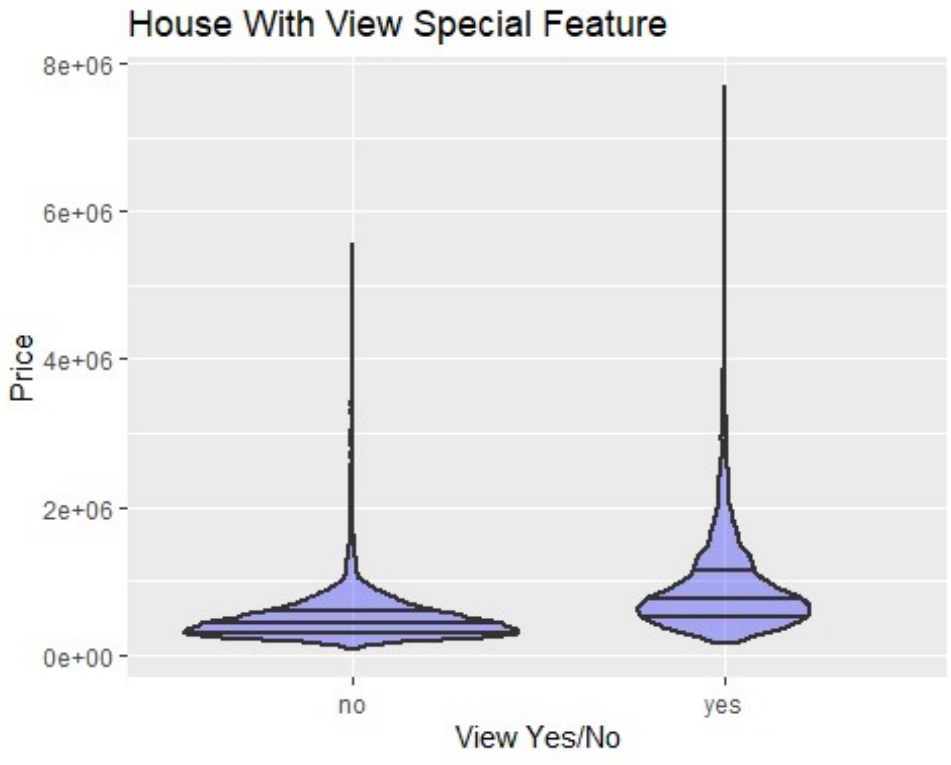

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```




```
# Houses with a view Yes or No
ggplot(house.price, aes(view_yes_no, price)) +
  geom_violin(draw_quantiles = c(0.25, 0.5, 0.75),
              fill = 'blue', alpha = 0.3, size = 1.0) +
  xlab('View Yes/No') +
  ylab('Price') +
  ggtitle('House With View Special Feature')
```

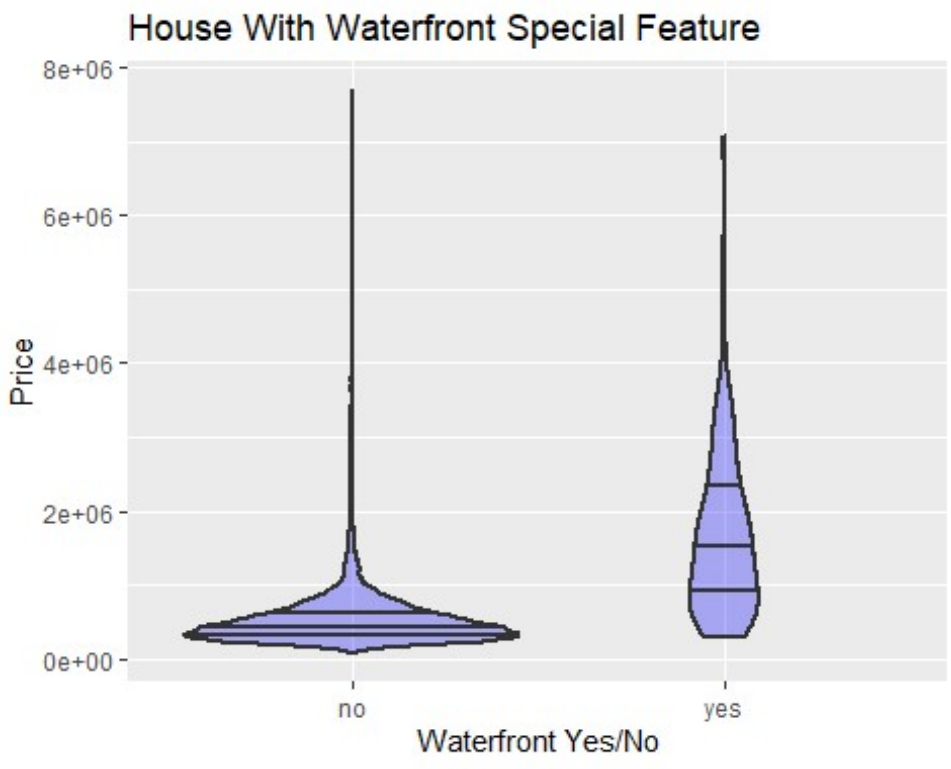
```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
# Houses with waterfront Yes or No
ggplot(house.price, aes(waterfront_yes_no, price)) +
  geom_violin(draw_quantiles = c(0.25, 0.5, 0.75),
              fill = 'blue', alpha = 0.3, size = 1.0) +
  xlab('Waterfront Yes/No') +
  ylab('Price') +
  ggtitle('House With Waterfront Special Feature')
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



Hypothesis Testing

It is common knowledge that the basic features examined previously such as square footage of living space, lot, basement, as well as number of bedrooms and bathrooms in addition to the geographical location of the house influence the house price. As previously mentioned, special features such as having a view and/or the house being in a waterfront location will have an influence in the price. The Hypothesis tests performed below are:

1. Hypothesis Test Series 1 - Waterfront Houses Yes/No Influence the Price
2. Hypothesis Test Series 2 - Houses with a View Yes/No Influence the Price
3. Hypothesis Test Series 3 - Waterfront Properties Yes/No with a View Yes/No Influence the Price

Note: I acknowledge that the sub-dataset used for these tests include 163 houses that have both special features, a view, and waterfront lot. This is not considered material and it is appropriate to proceed with the tests.

Hypothesis Test Series 1 - Waterfront Houses Yes/No Influence the Price

```
# Creating two sub-data sets for testing
# Data set 1 is all waterfront properties
df_waterfront_yes <- house.price %>% filter(waterfront_yes_no == "yes")
df_waterfront_yes %>% glimpse()
```

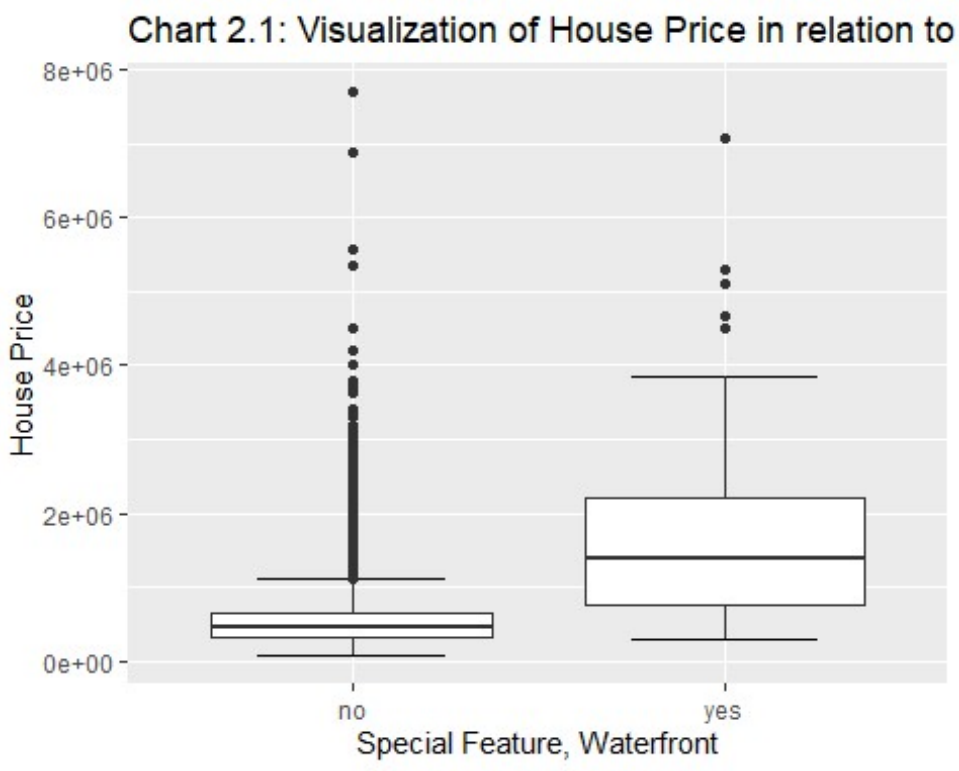
```
## Rows: 163
## Columns: 21
## $ id                <dbl> 822039084, 8096000060, 2025069065, 2123039032, 3
2250~
## $ date              <dttm> 2015-03-11, 2015-04-13, 2014-09-29, 2014-10-27,
201~
## $ price             <dbl> 1350000, 655000, 2400000, 369900, 3080000, 70500
0, 2~
## $ bedrooms          <dbl> 3, 2, 4, 1, 4, 3, 3, 3, 2, 4, 4, 5, 2, 3, 6, 5,
2, 4~
## $ bathrooms         <dbl> 2.50, 1.75, 2.50, 0.75, 5.00, 3.00, 2.50, 2.50,
1.00~
## $ sqft.living       <dbl> 2753, 1450, 3650, 760, 4550, 1970, 5403, 3930, 1
150,~
## $ sqft.lot          <dbl> 65005, 15798, 8354, 10079, 18641, 20978, 24069,
5586~
## $ floors            <dbl> 1.0, 2.0, 1.0, 1.0, 1.0, 2.0, 2.0, 1.0, 1.0, 1.0
, 1.~
## $ waterfront        <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1~
## $ view              <dbl> 2, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4,
4, 4~
## $ condition         <dbl> 5, 3, 3, 5, 3, 4, 4, 4, 4, 4, 3, 3, 4, 3, 4, 5,
3, 3~
## $ grade             <dbl> 9, 7, 9, 5, 10, 9, 12, 8, 6, 8, 7, 12, 5, 9, 12,
8, ~
## $ sqft.above        <dbl> 2165, 1230, 1830, 760, 2600, 1770, 5403, 2330, 1
150,~
## $ sqft.basement     <dbl> 588, 220, 1820, 0, 1950, 200, 0, 1600, 0, 1560,
1050~
## $ yr.built          <dbl> 1953, 1915, 2000, 1936, 2002, 1980, 1976, 1957,
1908~
## $ yr.renovated      <dbl> 0, 1978, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1990,
0, 0~
## $ zipcode           <dbl> 98070, 98166, 98074, 98070, 98074, 98070, 98166,
980~
## $ waterfront_yes_no <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes
, ye~
## $ view_yes_no       <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes
, ye~
## $ view_TRUE_FALSE   <fct> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE~
## $ price.log         <dbl> 14.11562, 13.39239, 14.69098, 12.82099, 14.94044
, 13~
```

```
# Data set 2 is all homes not in waterfront properties
df_waterfront_no <- house.price %>% filter(waterfront_yes_no == "no")
df_waterfront_no %>% glimpse()
```

```
## Rows: 21,450
## Columns: 21
## $ id                <dbl> 7129300520, 6414100192, 5631500400, 2487200875,
1954~
## $ date              <dttm> 2014-10-13, 2014-12-09, 2015-02-25, 2014-12-09,
201~
## $ price             <dbl> 221900, 538000, 180000, 604000, 510000, 1230000,
257~
## $ bedrooms          <dbl> 3, 3, 2, 4, 3, 4, 3, 3, 3, 3, 3, 2, 3, 3, 5, 4,
3, 4~
## $ bathrooms         <dbl> 1.00, 2.25, 1.00, 3.00, 2.00, 4.50, 2.25, 1.50,
1.00~
## $ sqft.living       <dbl> 1180, 2570, 770, 1960, 1680, 5420, 1715, 1060, 1
780,~
## $ sqft.lot          <dbl> 5650, 7242, 10000, 5000, 8080, 101930, 6819, 971
1, 7~
## $ floors            <dbl> 1.0, 2.0, 1.0, 1.0, 1.0, 1.0, 2.0, 1.0, 1.0, 2.0
, 1.~
## $ waterfront        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0~
## $ view              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3,
0, 0~
## $ condition         <dbl> 3, 3, 3, 5, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,
3, 4~
## $ grade             <dbl> 7, 7, 6, 7, 8, 11, 7, 7, 7, 7, 8, 7, 7, 7, 7, 9,
7, ~
## $ sqft.above        <dbl> 1180, 2170, 770, 1050, 1680, 3890, 1715, 1060, 1
050,~
## $ sqft.basement     <dbl> 0, 400, 0, 910, 0, 1530, 0, 0, 730, 0, 1700, 300
, 0,~
## $ yr.built          <dbl> 1955, 1951, 1933, 1965, 1987, 2001, 1995, 1963,
1960~
## $ yr.renovated      <dbl> 0, 1991, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0~
## $ zipcode           <dbl> 98178, 98125, 98028, 98136, 98074, 98053, 98003,
981~
## $ waterfront_yes_no <fct> no, no, no, no, no, no, no, no, no, no, no, no,
no, ~
## $ view_yes_no       <fct> no, no, no, no, no, no, no, no, no, no, no, no,
no, ~
## $ view_TRUE_FALSE   <fct> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,
FAL~
## $ price.log         <dbl> 12.30998, 13.19561, 12.10071, 13.31133, 13.14217
, 14~

# Calculating Mean and SD for waterfront houses
df_waterfront_yes_mean <- df_waterfront_yes$price %>% mean() %>% round(1)
df_waterfront_yes_sd <- df_waterfront_yes$price %>% sd() %>% round(1)

# Calculating Mean and SD for NO waterfront houses
```

```r
df_waterfront_no_mean <- df_waterfront_no$price %>% mean() %>% round(1)
df_waterfront_no_sd <- df_waterfront_no$price %>% sd() %>% round(1)

# Print results of mean and SD
cat(str_c("\nWaterfront Properties: mean = ", df_waterfront_yes_mean," and st
adard deviation = ", df_waterfront_yes_sd))

##
## Waterfront Properties: mean = 1662524.2 and stadard deviation = 1120388.2

cat(str_c("\nNot In Waterfront Properties: mean = ", df_waterfront_no_mean,"
and stadard deviation = ", df_waterfront_no_sd))

##
## Not In Waterfront Properties: mean = 531653.4 and stadard deviation = 3418
39.8

# Performing a two sample, one-sided t-test
# Hypothesis Test: waterfront houses are priced lower than houses NOT in wate
rfront properties
# NULL Hypothesis: waterfront houses are priced higher than houses NOT in wat
erfront properties
# Confidence Level: 95%
cat("\n")

cat("\n**** Waterfront !< Not in Waterfront ****")

##
## **** Waterfront !< Not in Waterfront ****

t.test(df_waterfront_yes$price, df_waterfront_no$price, conf.level = 0.95, al
ternative = "less")

##
##  Welch Two Sample t-test
##
## data:  df_waterfront_yes$price and df_waterfront_no$price
## t = 12.882, df = 162.23, p-value = 1
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##     -Inf 1276096
## sample estimates:
## mean of x mean of y
## 1662524.2  531653.4

# Performing a two sample, two-sided t-test
# Hypothesis Test: waterfront houses and houses NOT in waterfront properties
are priced similarly
# NULL Hypothesis: waterfront houses and houses NOT in waterfront properties
are NOT priced similarly
# Confidence Level: 95%
cat("\n**** Waterfront != Not in Waterfront ****")
```

```
##
## **** Waterfront != Not in Waterfront ****

t.test(df_waterfront_yes$price, df_waterfront_no$price, conf.level = 0.95, al
ternative = "two.sided")

##
##  Welch Two Sample t-test
##
## data:  df_waterfront_yes$price and df_waterfront_no$price
## t = 12.882, df = 162.23, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   957519 1304223
## sample estimates:
## mean of x mean of y
## 1662524.2  531653.4

# Performing a two sample, one-sided t-test
# Hypothesis Test: waterfront houses are priced higher than houses NOT in wat
erfront properties
# NULL Hypothesis: waterfront houses are priced lower than houses NOT in wate
rfront properties
cat("\n**** Waterfront !> Not in Waterfront ****")

##
## **** Waterfront !> Not in Waterfront ****

t.test(df_waterfront_yes$price, df_waterfront_no$price, conf.level = 0.95, al
ternative = "greater")

##
##  Welch Two Sample t-test
##
## data:  df_waterfront_yes$price and df_waterfront_no$price
## t = 12.882, df = 162.23, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  985645.3      Inf
## sample estimates:
## mean of x mean of y
## 1662524.2  531653.4

# Visualization
ggplot(house.price, aes(x=waterfront_yes_no, y=price)) +
  stat_boxplot(geom = "errorbar", width = 0.5) +
  geom_boxplot() +
  ggtitle("Chart 2.1: Visualization of House Price in relation to Waterfront
as Special Feature") +
  xlab ("Special Feature, Waterfront") +
  ylab("House Price")
```



Summary of Hypothesis Test Series 1:

1. Waterfront houses = 163 rows

2. Houses NOT in waterfront = 21,450

3. t-value = 12.882, and df = 162.23

4. Mean of waterfront houses = 1,662,524 with Standard Deviation = 1,120,388

5. Mean of houses NOT in waterfront = 531,653 with Standard Deviation = 341,840

6. The mean variance between the waterfront versus not in waterfront properties is significant and supports the hypothesis that waterfront properties appear to be priced higher than NOT in waterfront properties.

7. Based on p-value:

    7.a. One sided test of waterfront house price < NOT in waterfront house (p-value = 1): The t.test results cannot statistically prove that houses located in waterfront properties are priced lower than houses NOT located in waterfront properties. Thus, accepting the NULL hypothesis that houses in waterfront properties are priced higher than houses NOT in waterfront properties.

    7.b. Two-sided test of Waterfront house price = NOT in waterfront house (p-value <2.2e-16): The t.test results cannot statistically reject the hypothesis that the houses located in waterfront properties are priced higher than houses NOT located in

waterfront properties. Thus, rejecting the NULL hypothesis that these houses are priced similarly.

7. c. One sided test of waterfront house price > NOT in waterfront house (p-value <2.2e-16): The t.test value cannot statistically reject that waterfront houses are priced higher than houses NOT in waterfront properties. Thus, accepting the NULL hypothesis that houses NOT in waterfront properties are priced lower than houses in waterfront properties.

Hypothesis Test Series 2 - Houses with a View Yes/No Influence the Price

```
# Creating two sub-data sets for testing
# Data set 1 is all properties with a view
df_view_yes <- house.price %>% filter(view_yes_no == "yes")
df_view_yes %>% glimpse()
```

```
## Rows: 2,124
## Columns: 21
## $ id                <dbl> 9297300055, 2524049179, 822039084, 7922800400, 1
5160~
## $ date              <dttm> 2015-01-24, 2014-08-26, 2015-03-11, 2014-08-27,
201~
## $ price             <dbl> 650000, 2000000, 1350000, 951000, 650000, 437500
, 32~
## $ bedrooms          <dbl> 4, 3, 3, 5, 3, 3, 3, 4, 3, 3, 4, 3, 5, 5, 2, 2,
6, 2~
## $ bathrooms         <dbl> 3.00, 2.75, 2.50, 3.25, 2.25, 2.50, 1.75, 2.50,
2.50~
## $ sqft.living       <dbl> 2950, 3050, 2753, 3250, 2150, 2320, 2080, 3230,
2400~
## $ sqft.lot          <dbl> 5000, 44867, 65005, 14342, 21235, 36847, 5969, 1
6171~
## $ floors            <dbl> 2.0, 1.0, 1.0, 2.0, 1.0, 2.0, 1.0, 2.0, 1.0, 1.0
, 2.~
## $ waterfront        <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0~
## $ view              <dbl> 3, 4, 2, 4, 3, 2, 2, 3, 2, 2, 3, 2, 2, 3, 3, 4,
2, 2~
## $ condition         <dbl> 3, 3, 5, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3,
4, 4~
## $ grade             <dbl> 9, 9, 9, 8, 8, 9, 7, 9, 8, 8, 12, 7, 10, 9, 8, 7
, 9,~
## $ sqft.above        <dbl> 1980, 2330, 2165, 3250, 1590, 2320, 1080, 2520,
1560~
## $ sqft.basement     <dbl> 970, 720, 588, 0, 560, 0, 1000, 710, 840, 480, 1
640,~
## $ yr.built          <dbl> 1979, 1968, 1953, 1968, 1959, 1992, 1971, 2001,
1964~
```

```
## $ yr.renovated      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1991, 0,
1978~
## $ zipcode           <dbl> 98126, 98040, 98070, 98008, 98166, 98045, 98108,
980~
## $ waterfront_yes_no <fct> no, no, yes, no, no, no, no, no, no, no, no, no,
no,~
## $ view_yes_no       <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes
, ye~
## $ view_TRUE_FALSE   <fct> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE~
## $ price.log         <dbl> 13.38473, 14.50866, 14.11562, 13.76527, 13.38473
, 12~
```

```
# Data set 2 is all homes without a view
df_view_no <- house.price %>% filter(view_yes_no == "no")
df_view_no %>% glimpse()
```

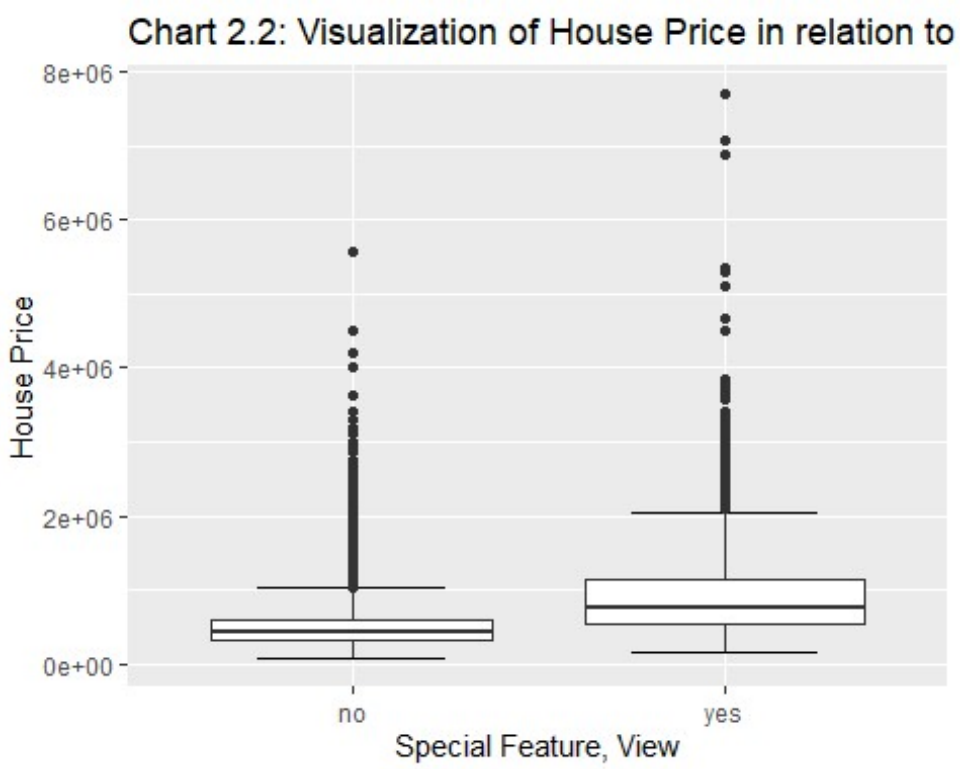
```
## Rows: 19,489
## Columns: 21
## $ id                <dbl> 7129300520, 6414100192, 5631500400, 2487200875,
1954~
## $ date              <dttm> 2014-10-13, 2014-12-09, 2015-02-25, 2014-12-09,
201~
## $ price             <dbl> 221900, 538000, 180000, 604000, 510000, 1230000,
257~
## $ bedrooms          <dbl> 3, 3, 2, 4, 3, 4, 3, 3, 3, 3, 3, 2, 3, 3, 5, 3,
4, 2~
## $ bathrooms         <dbl> 1.00, 2.25, 1.00, 3.00, 2.00, 4.50, 2.25, 1.50,
1.00~
## $ sqft.living       <dbl> 1180, 2570, 770, 1960, 1680, 5420, 1715, 1060, 1
780,~
## $ sqft.lot          <dbl> 5650, 7242, 10000, 5000, 8080, 101930, 6819, 971
1, 7~
## $ floors            <dbl> 1.0, 2.0, 1.0, 1.0, 1.0, 1.0, 2.0, 1.0, 1.0, 2.0
, 1.~
## $ waterfront        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0~
## $ view              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0~
## $ condition         <dbl> 3, 3, 3, 5, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,
4, 4~
## $ grade             <dbl> 7, 7, 6, 7, 8, 11, 7, 7, 7, 7, 8, 7, 7, 7, 7, 7,
7, ~
## $ sqft.above        <dbl> 1180, 2170, 770, 1050, 1680, 3890, 1715, 1060, 1
050,~
## $ sqft.basement     <dbl> 0, 400, 0, 910, 0, 1530, 0, 0, 730, 0, 1700, 300
, 0,~
## $ yr.built          <dbl> 1955, 1951, 1933, 1965, 1987, 2001, 1995, 1963,
1960~
## $ yr.renovated      <dbl> 0, 1991, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0~
## $ zipcode           <dbl> 98178, 98125, 98028, 98136, 98074, 98053, 98003,
981~
## $ waterfront_yes_no <fct> no, no, no, no, no, no, no, no, no, no, no, no,
no, ~
## $ view_yes_no       <fct> no, no, no, no, no, no, no, no, no, no, no, no,
no, ~
## $ view_TRUE_FALSE   <fct> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,
FAL~
## $ price.log         <dbl> 12.30998, 13.19561, 12.10071, 13.31133, 13.14217
, 14~

# Calculating Mean and SD for houses with a view
df_view_yes_mean <- df_view_yes$price %>% mean() %>% round(1)
df_view_yes_sd <- df_view_yes$price %>% sd() %>% round(1)

# Calculating Mean and SD for houses without a view
df_view_no_mean <- df_view_no$price %>% mean() %>% round(1)
df_view_no_sd <- df_view_no$price %>% sd() %>% round(1)

# Print results of mean and SD
cat(str_c("\n Properties with a View: mean = ", df_view_yes_mean," and stadar
d deviation = ", df_view_yes_sd))

##
##  Properties with a View: mean = 939859.4 and stadard deviation = 662440

cat(str_c("\n Properties Without A View: mean = ", df_view_no_mean," and stad
ard deviation = ", df_view_no_sd))

##
##  Properties Without A View: mean = 496623.5 and stadard deviation = 287316
.5

# Performing a two sample, one-sided t-test
# Hypothesis Test: houses with a view are priced lower than houses without a
view
# NULL Hypothesis: houses with a view are priced higher than houses without a
view
# Confidence Level: 95%
cat("\n")

cat("\n**** View !< No View ****")

##
## **** View !< No View ****

t.test(df_view_yes$price, df_view_no$price, conf.level = 0.95, alternative =
"less")

##
##  Welch Two Sample t-test
```

```
##
## data:  df_view_yes$price and df_view_no$price
## t = 30.525, df = 2210.8, p-value = 1
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##      -Inf 467129.7
## sample estimates:
## mean of x mean of y
##  939859.4  496623.5

# Performing a two sample, two-sided t-test
# Hypothesis Test: houses with a view and houses without a view are priced si
milarly
# NULL Hypothesis: houses with a view and houses without a view are NOT price
d similarly
# Confidence Level: 95%
cat("\n**** View != No View ****")

##
## **** View != No View ****

t.test(df_view_yes$price, df_view_no$price, conf.level = 0.95, alternative =
"two.sided")

##
##  Welch Two Sample t-test
##
## data:  df_view_yes$price and df_view_no$price
## t = 30.525, df = 2210.8, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  414761.0 471710.8
## sample estimates:
## mean of x mean of y
##  939859.4  496623.5

# Performing a two sample, one-sided t-test
# Hypothesis Test: houses with a view are priced higher than houses without a
view
# NULL Hypothesis: houses with a view are priced lower than houses without a
view
cat("\n**** View !> No View ****")

##
## **** View !> No View ****

t.test(df_view_yes$price, df_view_no$price, conf.level = 0.95, alternative =
"greater")

##
##  Welch Two Sample t-test
##
```

```
## data:  df_view_yes$price and df_view_no$price
## t = 30.525, df = 2210.8, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  419342.1      Inf
## sample estimates:
## mean of x mean of y
##  939859.4  496623.5
```

```
# Visualization
ggplot(house.price, aes(x=view_yes_no, y=price)) +
  stat_boxplot(geom = "errorbar", width = 0.5) +
  geom_boxplot() +
  ggtitle("Chart 2.2: Visualization of House Price in relation to View as Spe
cial Feature") +
  xlab ("Special Feature, View") +
  ylab("House Price")
```



Summary of Hypothesis Test Series 2:

1. Houses with a view = 2,124 rows

2. Houses without a view = 19,489

3. t-value = 30.525, df = 2210.8

4. Mean of houses with a view = 939,859 and Standard Deviation = 662,440

5. Mean of houses without a view = 496,624 and Standard Deviation = 287,317

6. The mean variance between houses with a view versus without a view is significant and supports the hypothesis that houses with a view appear to be priced higher than houses without a view.

7. Based on p-value:

7.a. One sided test of prices of houses with a view < houses without a view (p-value = 1): The t.test results cannot statistically prove that houses with a view are priced lower than houses without a view. Thus, accepting the NULL hypothesis that houses with a view are priced higher than houses without a view.

7.b. Two-sided test of prices of houses with a view = houses without a view (p-value <2.2e-16): The t.test results cannot statistically reject the hypothesis that the houses with a view are priced higher than houses without a view. Thus, rejecting the NULL hypothesis that these houses are priced similarly.

7.c. One sided test of prices of houses with a view > houses without a view (p-value <2.2e-16): The t.test results cannot statistically reject that houses with a view are priced higher than houses without a view. Thus, accepting the NULL hypothesis that houses without a view are priced lower than houses with a view.

Hypothesis Test Series 3 - Waterfront Properties or View Influence the Price

```
# Using existing sub-data sets for testing
df_waterfront_yes %>% glimpse()
```

```
## Rows: 163
## Columns: 21
## $ id                <dbl> 822039084, 8096000060, 2025069065, 2123039032, 3
2250~
## $ date              <dttm> 2015-03-11, 2015-04-13, 2014-09-29, 2014-10-27,
201~
## $ price             <dbl> 1350000, 655000, 2400000, 369900, 3080000, 70500
0, 2~
## $ bedrooms          <dbl> 3, 2, 4, 1, 4, 3, 3, 3, 2, 4, 4, 5, 2, 3, 6, 5,
2, 4~
## $ bathrooms         <dbl> 2.50, 1.75, 2.50, 0.75, 5.00, 3.00, 2.50, 2.50,
1.00~
## $ sqft.living       <dbl> 2753, 1450, 3650, 760, 4550, 1970, 5403, 3930, 1
150,~
## $ sqft.lot          <dbl> 65005, 15798, 8354, 10079, 18641, 20978, 24069,
5586~
## $ floors            <dbl> 1.0, 2.0, 1.0, 1.0, 1.0, 2.0, 2.0, 1.0, 1.0, 1.0
, 1.~
## $ waterfront        <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1~
```

```
## $ view              <dbl> 2, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4,
4, 4~
## $ condition         <dbl> 5, 3, 3, 5, 3, 4, 4, 4, 4, 4, 3, 3, 4, 3, 4, 5,
3, 3~
## $ grade             <dbl> 9, 7, 9, 5, 10, 9, 12, 8, 6, 8, 7, 12, 5, 9, 12,
8, ~
## $ sqft.above        <dbl> 2165, 1230, 1830, 760, 2600, 1770, 5403, 2330, 1
150,~
## $ sqft.basement     <dbl> 588, 220, 1820, 0, 1950, 200, 0, 1600, 0, 1560,
1050~
## $ yr.built          <dbl> 1953, 1915, 2000, 1936, 2002, 1980, 1976, 1957,
1908~
## $ yr.renovated      <dbl> 0, 1978, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1990,
0, 0~
## $ zipcode           <dbl> 98070, 98166, 98074, 98070, 98074, 98070, 98166,
980~
## $ waterfront_yes_no <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes
, ye~
## $ view_yes_no       <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes
, ye~
## $ view_TRUE_FALSE   <fct> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE~
## $ price.log         <dbl> 14.11562, 13.39239, 14.69098, 12.82099, 14.94044
, 13~

df_view_yes %>% glimpse()

## Rows: 2,124
## Columns: 21
## $ id                <dbl> 9297300055, 2524049179, 822039084, 7922800400, 1
5160~
## $ date              <dttm> 2015-01-24, 2014-08-26, 2015-03-11, 2014-08-27,
201~
## $ price             <dbl> 650000, 2000000, 1350000, 951000, 650000, 437500
, 32~
## $ bedrooms          <dbl> 4, 3, 3, 5, 3, 3, 3, 4, 3, 3, 4, 3, 5, 5, 2, 2,
6, 2~
## $ bathrooms         <dbl> 3.00, 2.75, 2.50, 3.25, 2.25, 2.50, 1.75, 2.50,
2.50~
## $ sqft.living       <dbl> 2950, 3050, 2753, 3250, 2150, 2320, 2080, 3230,
2400~
## $ sqft.lot          <dbl> 5000, 44867, 65005, 14342, 21235, 36847, 5969, 1
6171~
## $ floors            <dbl> 2.0, 1.0, 1.0, 2.0, 1.0, 2.0, 1.0, 2.0, 1.0, 1.0
, 2.~
## $ waterfront        <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0~
## $ view              <dbl> 3, 4, 2, 4, 3, 2, 2, 3, 2, 2, 3, 2, 2, 3, 3, 4,
2, 2~
## $ condition         <dbl> 3, 3, 5, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3,
```

```
4, 4~
## $ grade             <dbl> 9, 9, 9, 8, 8, 9, 7, 9, 8, 8, 12, 7, 10, 9, 8, 7
, 9,~
## $ sqft.above        <dbl> 1980, 2330, 2165, 3250, 1590, 2320, 1080, 2520,
1560~
## $ sqft.basement     <dbl> 970, 720, 588, 0, 560, 0, 1000, 710, 840, 480, 1
640,~
## $ yr.built          <dbl> 1979, 1968, 1953, 1968, 1959, 1992, 1971, 2001,
1964~
## $ yr.renovated      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1991, 0,
1978~
## $ zipcode           <dbl> 98126, 98040, 98070, 98008, 98166, 98045, 98108,
980~
## $ waterfront_yes_no <fct> no, no, yes, no, no, no, no, no, no, no, no, no,
no,~
## $ view_yes_no       <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes
, ye~
## $ view_TRUE_FALSE   <fct> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE~
## $ price.log         <dbl> 13.38473, 14.50866, 14.11562, 13.76527, 13.38473
, 12~
```

```
# Houses with both special features.
house.price.waterfrontandview <- df_waterfront_yes %>% filter(view_yes_no ==
"yes")
house.price.waterfrontandview
```

```
## # A tibble: 163 x 21
##            id date                price bedrooms bathrooms sqft.living s
qft.lot
##         <dbl> <dttm>              <dbl>    <dbl>     <dbl>       <dbl>
<dbl>
##  1  822039084 2015-03-11 00:00:00 1350000        3      2.5         2753
65005
##  2 8096000060 2015-04-13 00:00:00  655000        2      1.75        1450
15798
##  3 2025069065 2014-09-29 00:00:00 2400000        4      2.5         3650
8354
##  4 2123039032 2014-10-27 00:00:00  369900        1      0.75         760
10079
##  5 3225069065 2014-06-24 00:00:00 3080000        4      5           4550
18641
##  6 2122039094 2014-11-26 00:00:00  705000        3      3           1970
20978
##  7  622049114 2015-02-18 00:00:00 2130000        3      2.5         5403
24069
##  8 3760500116 2014-11-20 00:00:00 3070000        3      2.5         3930
55867
##  9 7567600045 2014-08-27 00:00:00  825000        2      1           1150
12775
```

```
## 10 4055701200 2015-04-21 00:00:00 1960000        4      2.75        3120
7898
## # ... with 153 more rows, and 14 more variables: floors <dbl>,
## #   waterfront <dbl>, view <dbl>, condition <dbl>, grade <dbl>,
## #   sqft.above <dbl>, sqft.basement <dbl>, yr.built <dbl>, yr.renovated <d
bl>,
## #   zipcode <dbl>, waterfront_yes_no <fct>, view_yes_no <fct>,
## #   view_TRUE_FALSE <fct>, price.log <dbl>
```

```
# Print results of mean and SD
cat(str_c("\n Waterfront Properties: mean = ", df_waterfront_yes_mean," and s
tadard deviation = ", df_waterfront_yes_sd))
```

```
##
##  Waterfront Properties: mean = 1662524.2 and stadard deviation = 1120388.2
```

```
cat(str_c("\n Properties with a View: mean = ", df_view_yes_mean," and stadar
d deviation = ", df_view_yes_sd))
```

```
##
##  Properties with a View: mean = 939859.4 and stadard deviation = 662440
```

```
# Performing a two sample, one-sided t-test
# Hypothesis Test: houses with a view are priced lower than waterfront houses
# NULL Hypothesis: Houses without a view are priced lower than waterfront hou
ses
# Confidence Level: 95%
cat("\n")
```

```
cat("\n **** Waterfront Houses !< Houses With A View ****")
```

```
##
##  **** Waterfront Houses !< Houses With A View ****
```

```
t.test(df_waterfront_yes$price, df_view_yes$price, conf.level = 0.95, alterna
tive = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  df_waterfront_yes$price and df_view_yes$price
## t = 8.1267, df = 170.8, p-value = 1
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##    -Inf 869731
## sample estimates:
## mean of x mean of y
## 1662524.2  939859.4
```

```
# Performing a two sample, two-sided t-test
# Hypothesis Test: houses with a view and waterfront houses are priced simila
rly
```

```
# NULL Hypothesis: houses with a view and waterfront houses are NOT priced si
milarly
# Confidence Level: 95%
cat("\n**** Waterfront Houses != Houses With A View ****")

##
## **** Waterfront Houses != Houses With A View ****

t.test(df_waterfront_yes$price, df_view_yes$price, conf.level = 0.95, alterna
tive = "two.sided")

##
##  Welch Two Sample t-test
##
## data:  df_waterfront_yes$price and df_view_yes$price
## t = 8.1267, df = 170.8, p-value = 8.544e-14
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  547131.4 898198.2
## sample estimates:
## mean of x mean of y
## 1662524.2  939859.4

# Performing a two sample, one-sided t-test
# Hypothesis Test: houses with a view are priced higher than waterfront house
s
# NULL Hypothesis: Houses with a view are priced lower than waterfront houses
cat("\n**** Houses With A View !> Waterfront Houses ****")

##
## **** Houses With A View !> Waterfront Houses ****

t.test(df_waterfront_yes$price, df_view_yes$price, conf.level = 0.95, alterna
tive = "greater")

##
##  Welch Two Sample t-test
##
## data:  df_waterfront_yes$price and df_view_yes$price
## t = 8.1267, df = 170.8, p-value = 4.272e-14
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  575598.6      Inf
## sample estimates:
## mean of x mean of y
## 1662524.2  939859.4
```

Summary of Hypothesis Test Series 3:

1. Waterfront houses = 163 rows

2. Houses with a view = 2,124 rows (includes the 163 waterfront houses. Not material)

3. t-value = 8.1267, df = 170.8

4. Mean of waterfront houses = 1,662,524 and Standard Deviation = 1,120,388

5. Mean of houses with a view = 939,859 and Standard Deviation = 662,440

6. The mean variance between waterfront houses versus houses with a view is significant which supports the hypothesis that waterfront houses appear to be priced higher than houses with a view.

7. based on the p-value:

    7.a. One sided test of prices of waterfront houses < houses with a view (p-value = 1): The t.test results cannot statistically prove that waterfront houses are priced lower than houses with a view. Thus, accepting the NULL hypothesis that waterfront houses are priced higher than houses with a view.

    7.b. Two-sided test of prices of waterfront houses = houses with a view (p-value = 8.544e-14): The t.test results cannot statistically reject the hypothesis that waterfront houses are priced higher than houses with a view. Thus, rejecting the NULL hypothesis that waterfront houses and houses with a view are priced similarly.

    7.c. One sided test of prices of waterfront houses > houses with a view (p-value = 4.272e-14): The t.test results cannot statistically reject that waterfront houses are priced higher than houses with a view. Thus, accepting the NULL hypothesis that houses with a view are priced lower than waterfront houses.

The conclusions from the 3 t.test performed. The data suggests that a house with a waterfront special feature will have a greater price advantage over a house with a view. Although, having a view as special feature offers good price advantage.

Split into train / test sets

To test the model in an unbiased manner and avoid **unrealistically optimistic** results because of the machine learning the data effectively and misleading the results in a way that the model produces excellent test results when presented with similar training data. To avoid this issue, I have split our data set into two **mutually exclusive** subsets named **training data set** and the **test data set**.

```
# Count Rows in the data set
count_rows = nrow(house.price)
count_rows

## [1] 21613

# Setting seed
set.seed(1222)
```

```
# 70% of the data for training
train_id = sample(1:count_rows, 0.7*count_rows)
#train_id

# 30% remaining of the data for testing
train_data = house.price[train_id,]
test_data = house.price[-train_id,]

# Display row counts of train data and test data
print(str_c("house.price.train rows: ", nrow(train_data)))

## [1] "house.price.train rows: 15129"

print(str_c("house.price.test rows: ", nrow(test_data)))

## [1] "house.price.test rows: 6484"

# Compare train data and test data
nrow(test_data) + nrow(train_data) == nrow(house.price)

## [1] TRUE
```

Summary of the Data Split: Training Data Set contains 15,219 rows, while the Testing Data Set contains 6,848 rows out of the 21,613 and the data is ready to train the **generalized linear model (glm)**.

Train and test models

Model 1 – Regression Model

It is challenging to build a prediction model to estimate the house prices (label) based on one or few features as the price is driven by many features. As mentioned at the beginning of this analysis the house price data available presents features that I have segment into Geographical, Basic, Age/Condition, and Special Features to which are all applied in the Linear Regression Model with multiple variables shown below.

Linear Regression Model 1.1

Multiple Variables

Data set: house. price

Label: Price

Independent Variables: House Basic Features + Special Features + Geographical Feature + Age/Condition Feature Features removed due to their significant codes are sqft. basement, and year renovated

```
# Linear Regression Model 1.1
# Multiple Variables
```

```r
# Data set: house.price
# Label: Price
# Independent Variables: House Basic Features + Special Features + Geographic
al Feature + Age/Condition Feature
# Features removed due to their significant codes are sqft.basement, and year
renovated
# Understand the relation between house price and its features

# Create linear model
# Price ~ Geographical Age/Condition, Basic, and Special Features
house.price.regressionmod1.1 = lm(price ~ zipcode + bedrooms + bathrooms + sq
ft.living + sqft.lot + floors + grade + yr.built + waterfront_yes_no + view_y
es_no, data = house.price)
summary (house.price.regressionmod1.1)

##
## Call:
## lm(formula = price ~ zipcode + bedrooms + bathrooms + sqft.living +
##     sqft.lot + floors + grade + yr.built + waterfront_yes_no +
##     view_yes_no, data = house.price)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1379162  -110967    -9849    90773  4264707
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.193e+07  3.028e+06   3.941 8.14e-05 ***
## zipcode              -5.299e+01  3.042e+01  -1.742   0.0815 .
## bedrooms             -3.884e+04  2.036e+03 -19.073  < 2e-16 ***
## bathrooms             4.881e+04  3.418e+03  14.280  < 2e-16 ***
## sqft.living           1.712e+02  3.295e+00  51.961  < 2e-16 ***
## sqft.lot             -2.576e-01  3.672e-02  -7.014 2.38e-12 ***
## floors                2.154e+04  3.446e+03   6.251 4.15e-10 ***
## grade                 1.250e+05  2.146e+03  58.257  < 2e-16 ***
## yr.built             -3.816e+03  6.778e+01 -56.306  < 2e-16 ***
## waterfront_yes_noyes  6.472e+05  1.777e+04  36.412  < 2e-16 ***
## view_yes_noyes        9.961e+04  5.478e+03  18.185  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 217400 on 21602 degrees of freedom
## Multiple R-squared:   0.65,  Adjusted R-squared:  0.6498
## F-statistic:  4011 on 10 and 21602 DF,  p-value: < 2.2e-16

cat('The coefficient confidence intervals')

## The coefficient confidence intervals

confint (house.price.regressionmod1.1)
```

```
##                               2.5 %        97.5 %
## (Intercept)           5.996905e+06  1.786525e+07
## zipcode              -1.126210e+02  6.634595e+00
## bedrooms             -4.282640e+04 -3.484440e+04
## bathrooms             4.211176e+04  5.551203e+04
## sqft.living           1.647586e+02  1.776760e+02
## sqft.lot             -3.295658e-01 -1.856056e-01
## floors                1.478751e+04  2.829669e+04
## grade                 1.208068e+05  1.292190e+05
## yr.built             -3.949200e+03 -3.683497e+03
## waterfront_yes_noyes  6.123632e+05  6.820422e+05
## view_yes_noyes        8.887631e+04  1.103500e+05
```

```
# house.price.regressionmod1.1 coeficient
house.price.regressionmod1.1$coefficients
```

```
##          (Intercept)               zipcode              bedrooms
##         1.193108e+07        -5.299318e+01        -3.883540e+04
##            bathrooms          sqft.living              sqft.lot
##         4.881190e+04         1.712173e+02        -2.575857e-01
##               floors                grade              yr.built
##         2.154210e+04         1.250129e+05        -3.816348e+03
## waterfront_yes_noyes       view_yes_noyes
##         6.472027e+05         9.961318e+04
```

```
# Predict
house.price$prediction = predict(house.price.regressionmod1.1, newdata = hous
e.price)

# Computed error
error = sqrt(sum((house.price$price - house.price$prediction)^2)/nrow(house.p
rice))
cat('Computed Error: \n')
```

```
## Computed Error:
```

```
error
```

```
## [1] 217335
```

Summary of results Linear Regression Model 1.1:

1. The model appears to be **over-fit**, as the zipcode coefficient does not appear to be significant. However, it is important to retain the geographical feature for this test as the location of the house plays a role in the house price.
2. The residual standard error appears reasonably modest, given the range of the label
3. The adjusted R-squared is reasonable. Although, the coefficients predict 65% of the behavior of the house price. However, I'd like to see it closer to 1.0. Even by removing the Zip Code the number would increase materially.
4. Error is 217,335

Linear Regression Model 1.2

Note: The only difference with the previous model is that 1.2 uses training data = 70% of data set.

Multiple Variables,

Data set: Training Data

Label: Price

Independent Variables: House Basic Features + Special Features + Geographical Feature + Age/Condition Feature Features removed due to their significant codes are sqft. basement, and year renovated

```
# Linear Regression Model 1.2
# Multiple Variables
# Data set: train data which is 70% of the house.price data set chosen random
ly based on seed
# Label: Price
# Independent Variables: House Basic Features + Special Features + Geographic
al Feature + Age/Condition Feature
# Features removed due to their significant codes are sqft.basement, and year
renovated
# Understand the relation between house price and its features

# Create linear model
# Price ~ Geographical Age/Condition, Basic, and Special Features
house.price.regressionmod1.2 = lm(price ~ zipcode + bedrooms + bathrooms + sq
ft.living + sqft.lot + floors + grade + yr.built + waterfront_yes_no + view_y
es_no, data = train_data)
summary (house.price.regressionmod1.2)

##
## Call:
## lm(formula = price ~ zipcode + bedrooms + bathrooms + sqft.living +
##     sqft.lot + floors + grade + yr.built + waterfront_yes_no +
##     view_yes_no, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1332314  -110114   -10303    90448  4294048
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.146e+07  3.622e+06   3.164  0.00156 **
## zipcode              -4.795e+01  3.640e+01  -1.317  0.18778
## bedrooms             -3.651e+04  2.386e+03 -15.301  < 2e-16 ***
## bathrooms             4.842e+04  4.073e+03  11.887  < 2e-16 ***
## sqft.living           1.677e+02  3.910e+00  42.883  < 2e-16 ***
## sqft.lot             -2.748e-01  4.288e-02  -6.409 1.50e-10 ***
```

```
## floors                2.312e+04  4.085e+03   5.659 1.55e-08 ***
## grade                 1.258e+05  2.561e+03  49.110  < 2e-16 ***
## yr.built             -3.832e+03  8.103e+01 -47.287  < 2e-16 ***
## waterfront_yes_noyes  6.118e+05  2.076e+04  29.473  < 2e-16 ***
## view_yes_noyes        9.604e+04  6.547e+03  14.669  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 217500 on 15118 degrees of freedom
## Multiple R-squared:  0.6435, Adjusted R-squared:  0.6433
## F-statistic:  2729 on 10 and 15118 DF,  p-value: < 2.2e-16

cat('The coefficient confidence intervals')

## The coefficient confidence intervals

confint (house.price.regressionmod1.2)

##                              2.5 %        97.5 %
## (Intercept)           4.359433e+06  1.855921e+07
## zipcode              -1.193097e+02  2.340351e+01
## bedrooms             -4.118322e+04 -3.183000e+04
## bathrooms             4.043527e+04  5.640355e+04
## sqft.living           1.599973e+02  1.753244e+02
## sqft.lot             -3.589049e-01 -1.907974e-01
## floors                1.510887e+04  3.112214e+04
## grade                 1.207375e+05  1.307761e+05
## yr.built             -3.990660e+03 -3.672991e+03
## waterfront_yes_noyes  5.711089e+05  6.524837e+05
## view_yes_noyes        8.320785e+04  1.088754e+05

test_data$prediction = predict(house.price.regressionmod1.2, newdata = test_d
ata)
test_data

## # A tibble: 6,484 x 22
##            id date                price bedrooms bathrooms sqft.living s
qft.lot
##         <dbl> <dttm>              <dbl>    <dbl>     <dbl>       <dbl>
<dbl>
##  1 6414100192 2014-12-09 00:00:00  538000        3      2.25        2570
7242
##  2 7237550310 2014-05-12 00:00:00 1230000        4      4.5         5420
101930
##  3 1321400060 2014-06-27 00:00:00  257500        3      2.25        1715
6819
##  4 9212900260 2014-05-27 00:00:00  468000        2      1           1160
6000
##  5 6054650070 2014-10-07 00:00:00  400000        3      1.75        1370
9680
##  6 9297300055 2015-01-24 00:00:00  650000        4      3           2950
```

```
5000
##  7 6865200140 2014-05-29 00:00:00  485000        4      1           1600
4300
##  8 7137970340 2014-07-03 00:00:00  285000        5      2.5         2270
6300
##  9 1794500383 2014-06-26 00:00:00  937000        3      1.75        2450
2691
## 10 5101402488 2014-06-24 00:00:00  438000        3      1.75        1520
6380
## # ... with 6,474 more rows, and 15 more variables: floors <dbl>,
## #   waterfront <dbl>, view <dbl>, condition <dbl>, grade <dbl>,
## #   sqft.above <dbl>, sqft.basement <dbl>, yr.built <dbl>, yr.renovated <d
bl>,
## #   zipcode <dbl>, waterfront_yes_no <fct>, view_yes_no <fct>,
## #   view_TRUE_FALSE <fct>, price.log <dbl>, prediction <dbl>
```

```
# Computed error
error = sqrt(sum((test_data$price - test_data$prediction)^2)/nrow(test_data))
cat('Computed Error: \n')
```

```
## Computed Error:
```

```
error
```

```
## [1] 217396.3
```

Summary of results Linear Regression Model 1.2:

1. Overall this model is very similar to 1.1 which uses the complete data set.

2. The model appears to be **over-fit**, as the zipcode coefficient does not appear to be significant. However, it is important to retain the geographical feature for this test as the location of the house plays a role in the house price.

3. The residual standard error appears reasonably modest, given the range of the label

4. The adjusted R-squared is reasonable but did not change much from model 1.1 as the coefficients predict 64.35% of the behavior of the house price. I would have liked to see a higher number. Even by removing the Zip Code the number would increase materially.

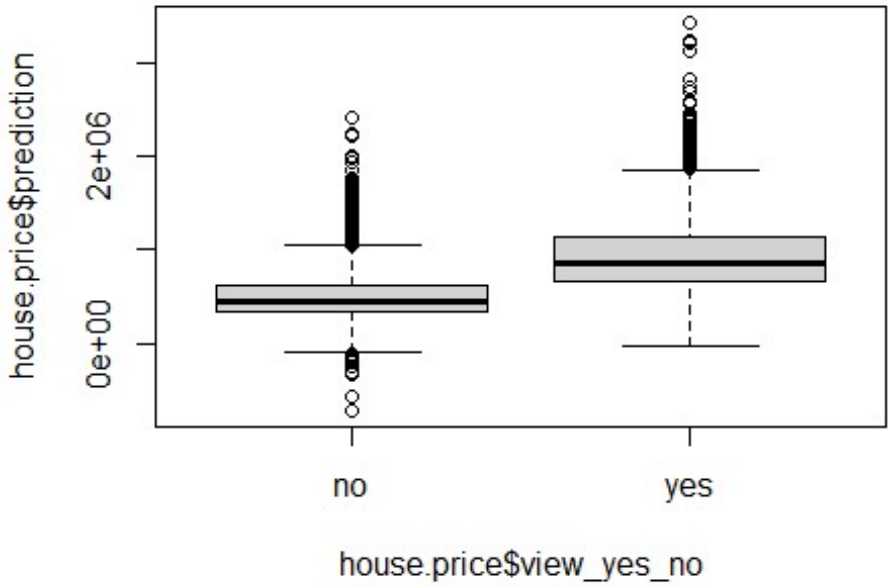
5. Error is 217,396 which unchanged (although slightly higher) from model 1.1 which shows consistency.

6. The model appears modest.

Model 2 - Classification Model

The business question is can we predict if a house has a view? Yes/No, using the existing data and Generalized Linear Model.

```r
# Classification Model 2.1
# Data set: House.price
# Label: view_yes_no
# Independent Variables: House Basic Features + Special Features + Geographic
al Feature + Age/Condition Feature
# Features removed due to their significant codes are Waterfront_yes_no, sqft
.basement, and year renovated

# Create the Model
housewithview.classificationmod2.1 = glm(view_yes_no ~ zipcode + bedrooms + b
athrooms + sqft.living + floors + grade + yr.built -1, data = train_data, fam
ily = binomial)
summary (housewithview.classificationmod2.1)

##
## Call:
## glm(formula = view_yes_no ~ zipcode + bedrooms + bathrooms +
##     sqft.living + floors + grade + yr.built - 1, family = binomial,
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9396  -0.4420  -0.3290  -0.2454   3.0303
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## zipcode      0.0003865  0.0000213  18.148  < 2e-16 ***
## bedrooms    -0.3364925  0.0402858  -8.353  < 2e-16 ***
## bathrooms    0.3403742  0.0616043   5.525 3.29e-08 ***
## sqft.living  0.0007340  0.0000572  12.831  < 2e-16 ***
## floors      -0.5235243  0.0706568  -7.409 1.27e-13 ***
## grade        0.3587864  0.0391182   9.172  < 2e-16 ***
## yr.built    -0.0221040  0.0011054 -19.996  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 20973.2  on 15129  degrees of freedom
## Residual deviance:  8243.6  on 15122  degrees of freedom
## AIC: 8257.6
##
## Number of Fisher Scoring iterations: 6

# Predict
train_data$prediction = predict(housewithview.classificationmod2.1)

# Visualize the data
boxplot(house.price$prediction ~ house.price$view_yes_no )
```




```
# Confusion matrix
test_data$prediction = predict(housewithview.classificationmod2.1, newdata =
test_data)
test_data$ampred = ifelse(test_data$prediction > 0.7, 'yes', 'no')
caret::confusionMatrix(as.factor(test_data$ampred), as.factor(test_data$view_
yes_no))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  5823  607
##        yes   23   31
##
##                Accuracy : 0.9028
##                  95% CI : (0.8954, 0.9099)
##     No Information Rate : 0.9016
##     P-Value [Acc > NIR] : 0.3792
##
##                   Kappa : 0.0754
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99607
##             Specificity : 0.04859
##          Pos Pred Value : 0.90560
##          Neg Pred Value : 0.57407
```

```
##              Prevalence : 0.90160
##          Detection Rate : 0.89806
##    Detection Prevalence : 0.99167
##       Balanced Accuracy : 0.52233
##
##        'Positive' Class : no
##
```

Summary of the results of Linear Regression Model 2.0:

The confusion matrix using prediction > 70% produced:

1. correctly predicted 5,823 True Negatives

2. Incorrectly predicted 607 False Negatives

3. Incorrectly predicted 23 False Positives

4. Correctly predicted 31 True Positives 5. Accuracy of 90.3%

Model Performance Comparison

For most cases the price values are reasonably comparable to the predicted price. Additionally, there appear to be minimal variances in predicted prices based on regression models 1.1 and 1.2. The significance of the variances may be negligible.

In conclusion, both Linear Regression Models 1.1 and 1.2 based on Geographical, Basic, Special Features as well as Age/Condition are remarkably similar. To test the model prediction, I used a random set of features such as: 4 bedrooms, 3 bathrooms, 2100 sqft. living space, floor grade of 2 and built grade 6, built in 2000 with a view but no waterfront located in zip code are 98103 which is in the Freemont neighborhood of Seattle. Linear Regression Models 1.1 predicted a house price of $341,727 whereas model 1.2 predicted a house price of $338,087.

```
# Adding predicted scores and residual values from house.price.regressionmod1
.1 to the dataframe
house.price.comp1.1 = house.price %>% mutate(score = predict (house.price.reg
ressionmod1.1, data = house.price)) %>% mutate(resids = price - score,
                                        predicted.price.regressionmod1.1 = exp(
score))
house.price.comp1.1[1:10, 3:22]
```

```
## # A tibble: 10 x 20
##      price bedrooms bathrooms sqft.living sqft.lot floors waterfront  view
##      <dbl>    <dbl>     <dbl>       <dbl>    <dbl>  <dbl>      <dbl> <dbl>
##  1  221900        3      1           1180     5650      1          0     0
##  2  538000        3      2.25        2570     7242      2          0     0
##  3  180000        2      1            770    10000      1          0     0
##  4  604000        4      3           1960     5000      1          0     0
##  5  510000        3      2           1680     8080      1          0     0
```

```
##  6 1230000        4      4.5         5420   101930      1          0     0
##  7  257500        3      2.25        1715     6819      2          0     0
##  8  291850        3      1.5         1060     9711      1          0     0
##  9  229500        3      1           1780     7470      1          0     0
## 10  323000        3      2.5         1890     6560      2          0     0
## # ... with 12 more variables: condition <dbl>, grade <dbl>, sqft.above <db
l>,
## #   sqft.basement <dbl>, yr.built <dbl>, yr.renovated <dbl>, zipcode <dbl>
,
## #   waterfront_yes_no <fct>, view_yes_no <fct>, view_TRUE_FALSE <fct>,
## #   price.log <dbl>, prediction <dbl>

glimpse(house.price.comp1.1)

## Rows: 21,613
## Columns: 25
## $ id                              <dbl> 7129300520, 6414100192, 563150040
0, 2~
## $ date                            <dttm> 2014-10-13, 2014-12-09, 2015-02-
25, ~
## $ price                           <dbl> 221900, 538000, 180000, 604000, 5
1000~
## $ bedrooms                        <dbl> 3, 3, 2, 4, 3, 4, 3, 3, 3, 3, 3,
2, 3~
## $ bathrooms                       <dbl> 1.00, 2.25, 1.00, 3.00, 2.00, 4.5
0, 2~
## $ sqft.living                     <dbl> 1180, 2570, 770, 1960, 1680, 5420
, 17~
## $ sqft.lot                        <dbl> 5650, 7242, 10000, 5000, 8080, 10
1930~
## $ floors                          <dbl> 1.0, 2.0, 1.0, 1.0, 1.0, 1.0, 2.0
, 1.~
## $ waterfront                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0~
## $ view                            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0~
## $ condition                       <dbl> 3, 3, 3, 5, 3, 3, 3, 3, 3, 3, 3,
4, 4~
## $ grade                           <dbl> 7, 7, 6, 7, 8, 11, 7, 7, 7, 7, 8,
7, ~
## $ sqft.above                      <dbl> 1180, 2170, 770, 1050, 1680, 3890
, 17~
## $ sqft.basement                   <dbl> 0, 400, 0, 910, 0, 1530, 0, 0, 73
0, 0~
## $ yr.built                        <dbl> 1955, 1951, 1933, 1965, 1987, 200
1, 1~
## $ yr.renovated                    <dbl> 0, 1991, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0~
## $ zipcode                         <dbl> 98178, 98125, 98028, 98136, 98074
, 98~
```

```
## $ waterfront_yes_no                <fct> no, no, no, no, no, no, no, no, n
o, n~
## $ view_yes_no                      <fct> no, no, no, no, no, no, no, no, n
o, n~
## $ view_TRUE_FALSE                  <fct> FALSE, FALSE, FALSE, FALSE, FALSE
, FA~
## $ price.log                        <dbl> 12.30998, 13.19561, 12.10071, 13.
3113~
## $ prediction                       <dbl> 296870.2, 635083.2, 231281.7, 453
437.~
## $ score                            <dbl> 296870.2, 635083.2, 231281.7, 453
437.~
## $ resids                           <dbl> -74970.197, -97083.198, -51281.70
2, 1~
## $ predicted.price.regressionmod1.1 <dbl> Inf, Inf, Inf, Inf, Inf, Inf, Inf
, In~
```

```
# Predict based on the linear model created, house.price.regressionmod1.1
print("Predicting house prices based on features such as: zip code 98103, 4 b
edrooms, 3 bathrooms, 2100 sqft.living space, 2 floors grade, 6 built grade,
built in 2000, no waterfront, with a view")
```

```
## [1] "Predicting house prices based on features such as: zip code 98103, 4
bedrooms, 3 bathrooms, 2100 sqft.living space, 2 floors grade, 6 built grade,
built in 2000, no waterfront, with a view"
```

```
data_new = data.frame(zipcode = 98103, bedrooms = 4, bathrooms = 3, sqft.livi
ng = 2100, sqft.lot = 5000, floors =2, grade =6, yr.built =2000, waterfront_y
es_no = 'no', view_yes_no = 'yes')

house.price.regressionmod1.1 %>% predict(newdata = data_new)
```

```
##        1
## 341727.1
```

```
# Predict based on the linear model created, house.price.regressionmod1.2
print("Predicting house prices based on features such as: zip code 98103, 4 b
edrooms, 3 bathrooms, 2100 sqft.living space, 2 floors grade, 6 built grade,
built in 2000, no waterfront, with a view")
```

```
## [1] "Predicting house prices based on features such as: zip code 98103, 4
bedrooms, 3 bathrooms, 2100 sqft.living space, 2 floors grade, 6 built grade,
built in 2000, no waterfront, with a view"
```

```
data_new = data.frame(zipcode = 98103, bedrooms = 4, bathrooms = 3, sqft.livi
ng = 2100, sqft.lot = 5000, floors =2, grade =6, yr.built =2000, waterfront_y
es_no = 'no', view_yes_no = 'yes')
house.price.regressionmod1.2 %>% predict(newdata = data_new)
```

```
##        1
## 338087.4
```