Home Sale Prediction in the city of Sacramento CA

Santhosh Thirumalai

6/16/2019

Introduction

This document illustrates the usage of Linear, Regression Trees, Random Forest and K-Nearest Neighbor models on predicting the home sale price in the city of Sacramento, CA. The dataset “Sacramento” can be found in the “dslabs” package which is used in this script for processing.

The Goal of this exercise is to demonstrate the usage of Linear, Randomforest, Rpart, KNN models and to compile the RMSE values for each model.

The code is present in the following repository

- [Github] (https://github.com/sthirumalai2020/EDX-HARVARDX.git (https://github.com/sthirumalai2020/EDX-HARVARDX.git))

Dataset description

The Sacramento dataset has the following fields out of which the “price” field is the outcome

1. city - factor
2. zip - factor
3. beds - Integer
4. baths - numeric
5. sqft - integer
6. type - factor
7. latitude - numeric
8. longitude - numeric
9. price - numeric

The following code prints the observations and predictors. As you can see there are 932 rows and 9 fields as predictors.

```
dim(Sacramento)
```

```
## [1] 932   9
```

```
names(Sacramento)
```

```
## [1] "city"      "zip"       "beds"      "baths"     "sqft"      "type"
## [7] "price"     "latitude"  "longitude"
```

The summary of the Sacramento dataset can be printed as below

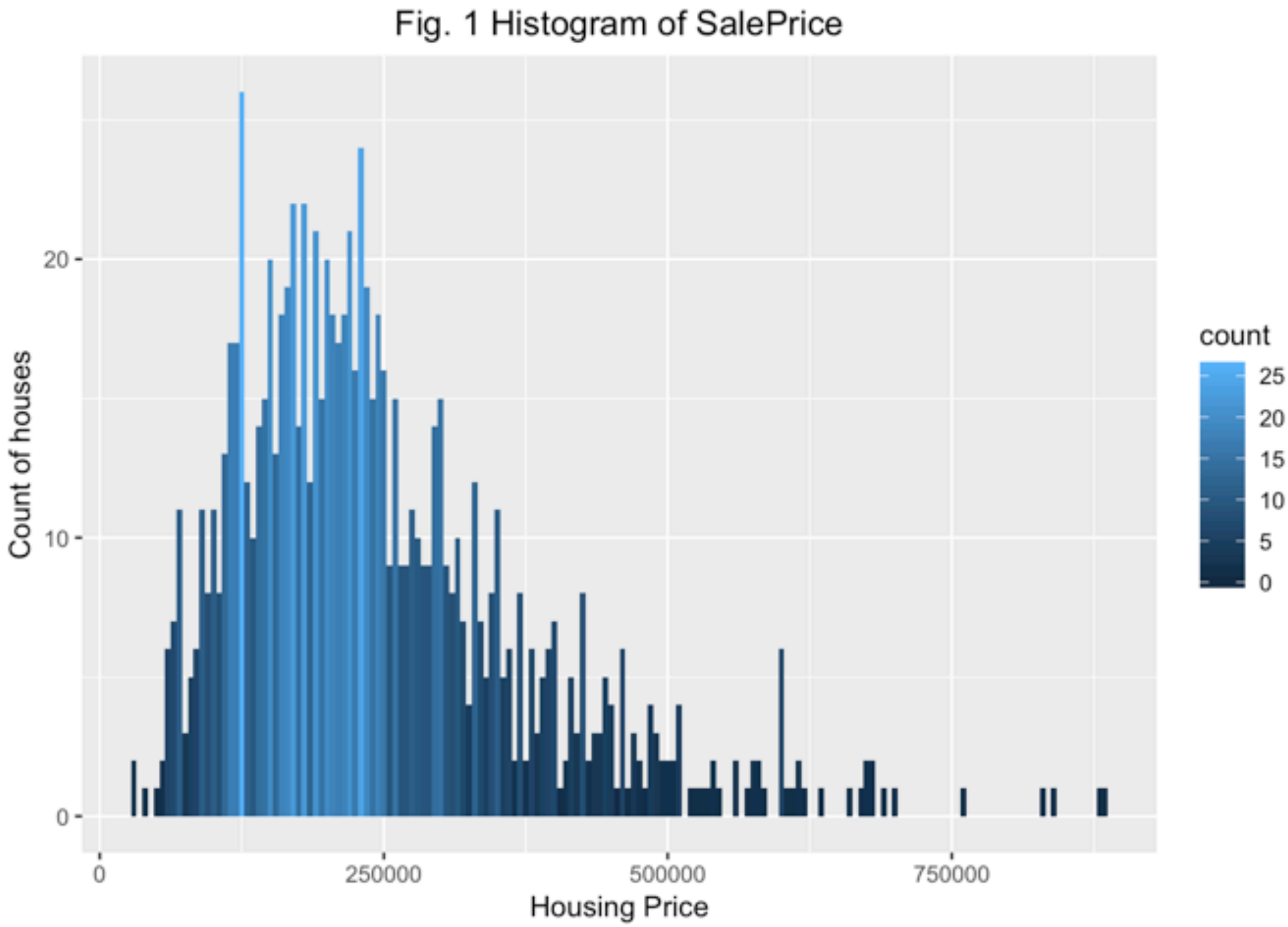
```
summary(Sacramento)
```

```
##             city          zip           beds           baths
##  SACRAMENTO    :438   z95823 : 61   Min.   :1.000   Min.   :1.000
##  ELK_GROVE     :114   z95828 : 45   1st Qu.:3.000   1st Qu.:2.000
##  ROSEVILLE     : 48   z95758 : 44   Median :3.000   Median :2.000
##  CITRUS_HEIGHTS: 35   z95835 : 37   Mean   :3.276   Mean   :2.053
##  ANTELOPE      : 33   z95838 : 37   3rd Qu.:4.000   3rd Qu.:2.000
##  RANCHO_CORDOVA: 28   z95757 : 36   Max.   :8.000   Max.   :5.000
##  (Other)       :236   (Other):672
##       sqft                 type         price           latitude
##  Min.   : 484   Condo       : 53   Min.   : 30000   Min.   :38.24
##  1st Qu.:1167   Multi_Family: 13   1st Qu.:156000   1st Qu.:38.48
##  Median :1470   Residential :866   Median :220000   Median :38.62
##  Mean   :1680                      Mean   :246662   Mean   :38.59
##  3rd Qu.:1954                      3rd Qu.:305000   3rd Qu.:38.69
##  Max.   :4878                      Max.   :884790   Max.   :39.02
##
##    longitude
##  Min.   :-121.6
##  1st Qu.:-121.4
##  Median :-121.4
##  Mean   :-121.4
##  3rd Qu.:-121.3
##  Max.   :-120.6
##
```

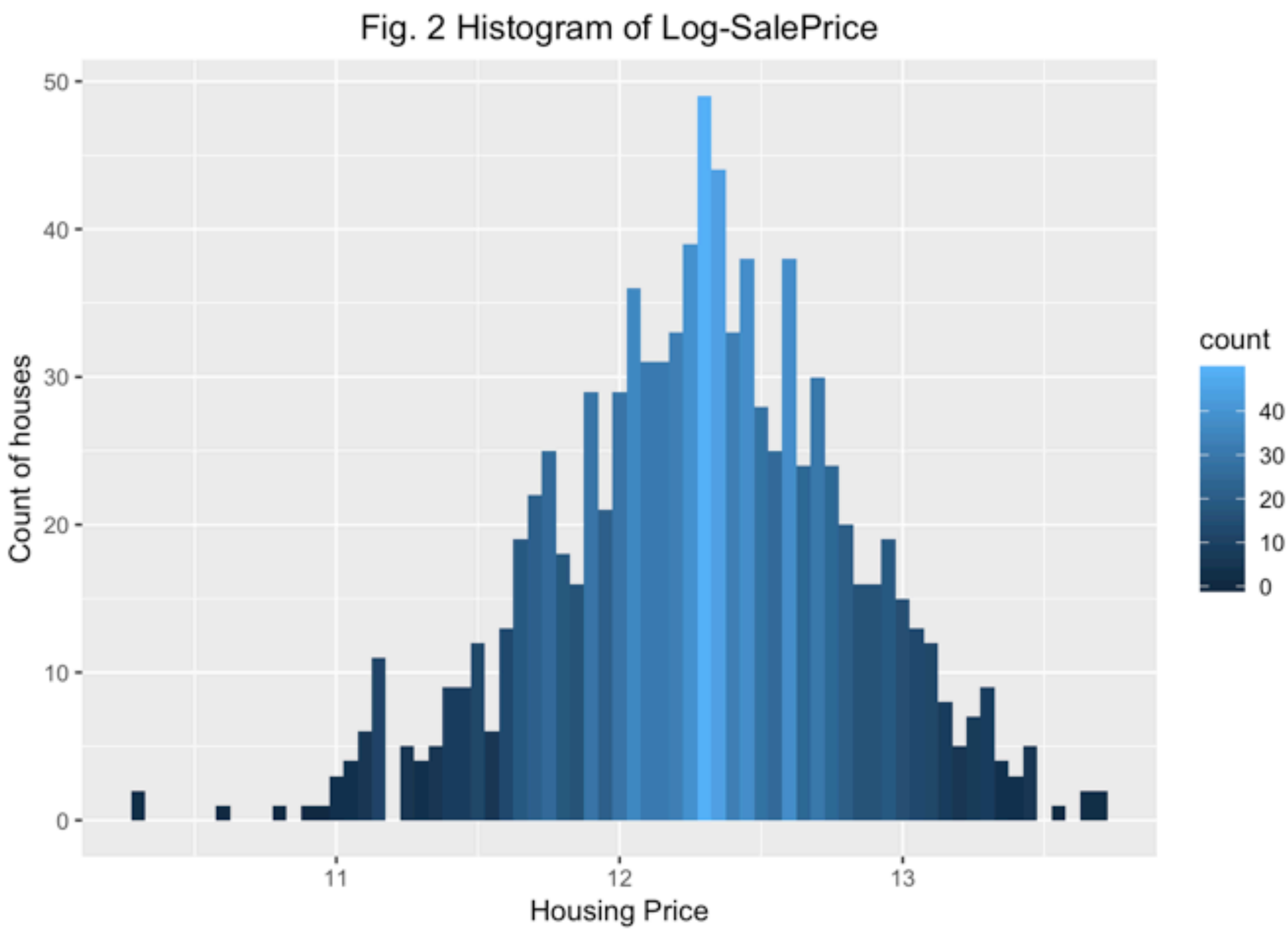
The summary shows that there are 3 types of homes with more sales in Sacramento city, the sqft ranges from 484 to 4878. The beds in the homes ranges from 1 to 8 and baths are between 1 and 5. The price ranges are from 30000$ to a max of 885000$.

Visualization

Let’s find out the predictors for the outcomes using vizualization.



The above plot shows the Home prices and number of homes are normally disctributed with some right skewness due to the price range. Hence the price field will be transformed to a log format and used throughout this exercise.

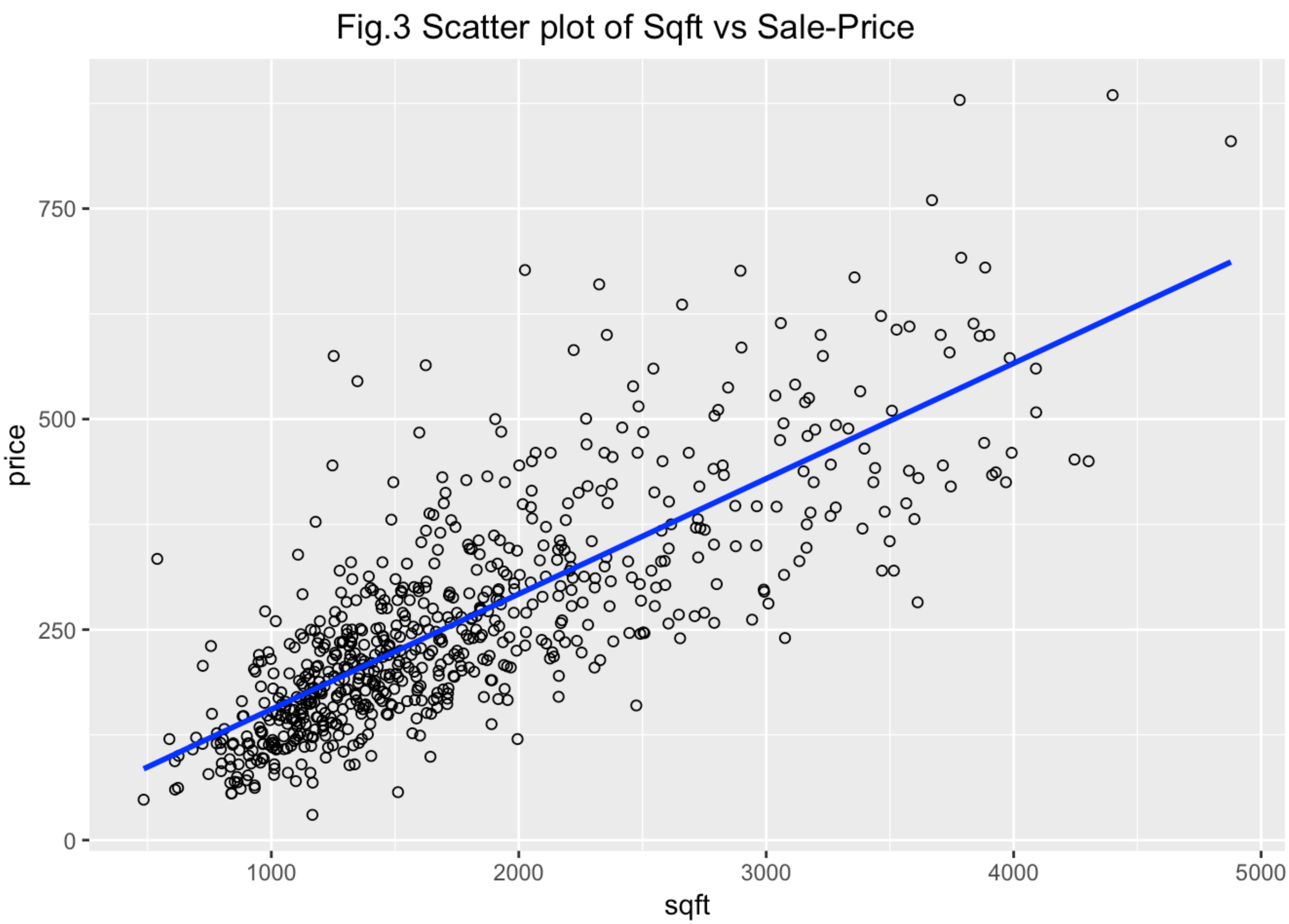


After the log transforms the data looks normally distributed.

Finding the predictors using visualization

1. sqft vs price

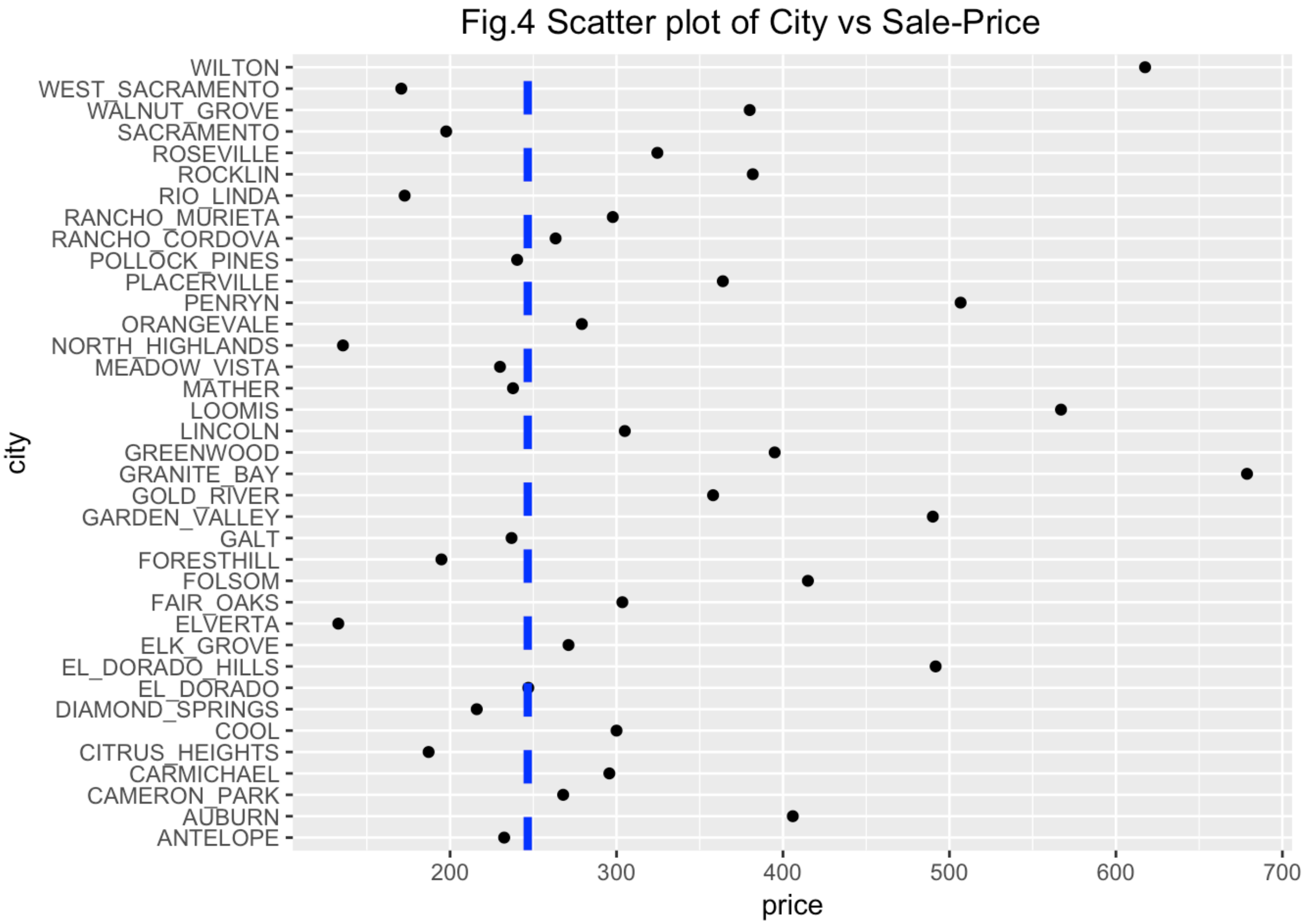
Let's plot the price vs Sqft relationship to see if the sqft can be used as a predictor. Let's use the scatter plot to determine this note that the sale price is transformed to 100s and the mean for the price for sqfts are plotted.



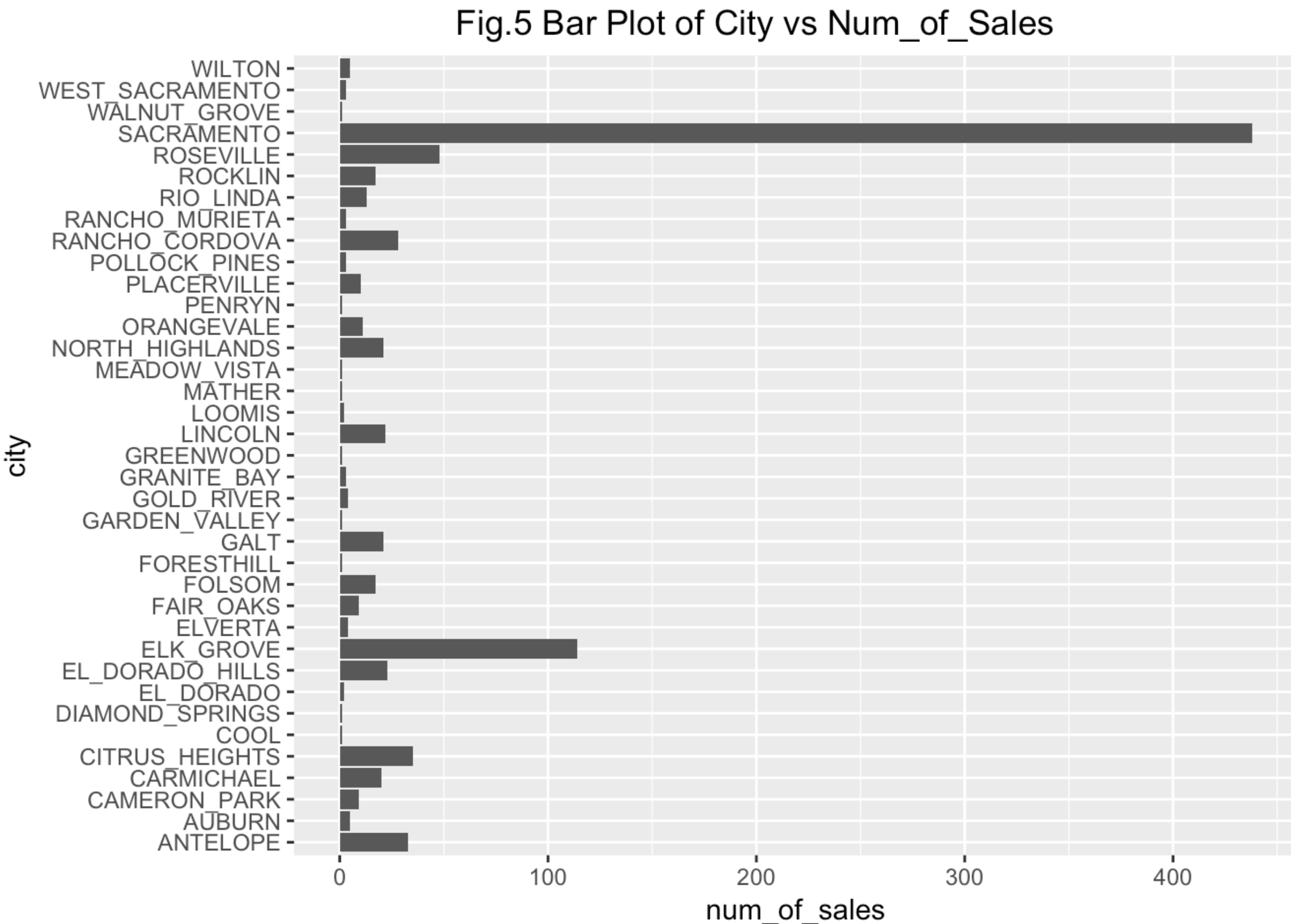
As the scatter plot (Fig.3) shows there is a linear relationship and the sqft can be used as a predictor which we can confirm using heat map for correlations.

2. city vs price

Let’s use the scatter plot to determine is city can be used as a predictor. Note that the sale price is transformed to 100s and the mean for the price for cities are plotted.



Note that the Fig.4 illustrates the data points are not on a linear fashion and some cities has less sales which can be proved below.




```
## # A tibble: 37 x 2
##    city            num_of_sale
##    <fct>                 <int>
##  1 ANTELOPE                 33
##  2 AUBURN                    5
##  3 CAMERON_PARK              9
##  4 CARMICHAEL               20
##  5 CITRUS_HEIGHTS           35
##  6 COOL                      1
##  7 DIAMOND_SPRINGS           1
##  8 EL_DORADO                 2
##  9 EL_DORADO_HILLS          23
## 10 ELK_GROVE               114
## # … with 27 more rows
```

The above statistics confirms that the city cannot be used as predictor.

3. Beds vs price

Let’s use the scatter plot to determine if the Beds field can be used as a predictor.

Note that the sale price is transformed to 100s and the mean for the price for beds are plotted.

Fig.6 Bar Plot of Beds vs Sale price

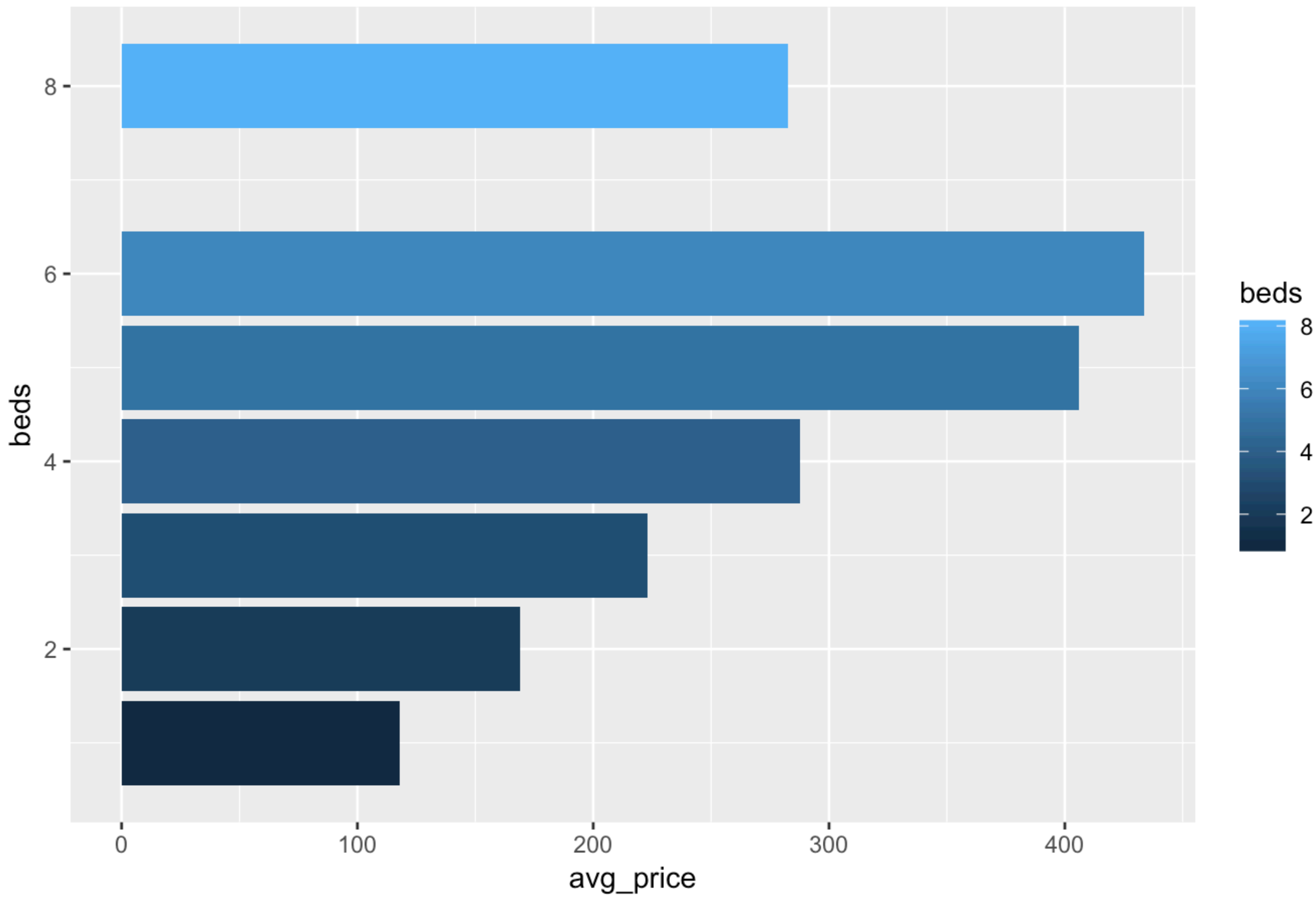
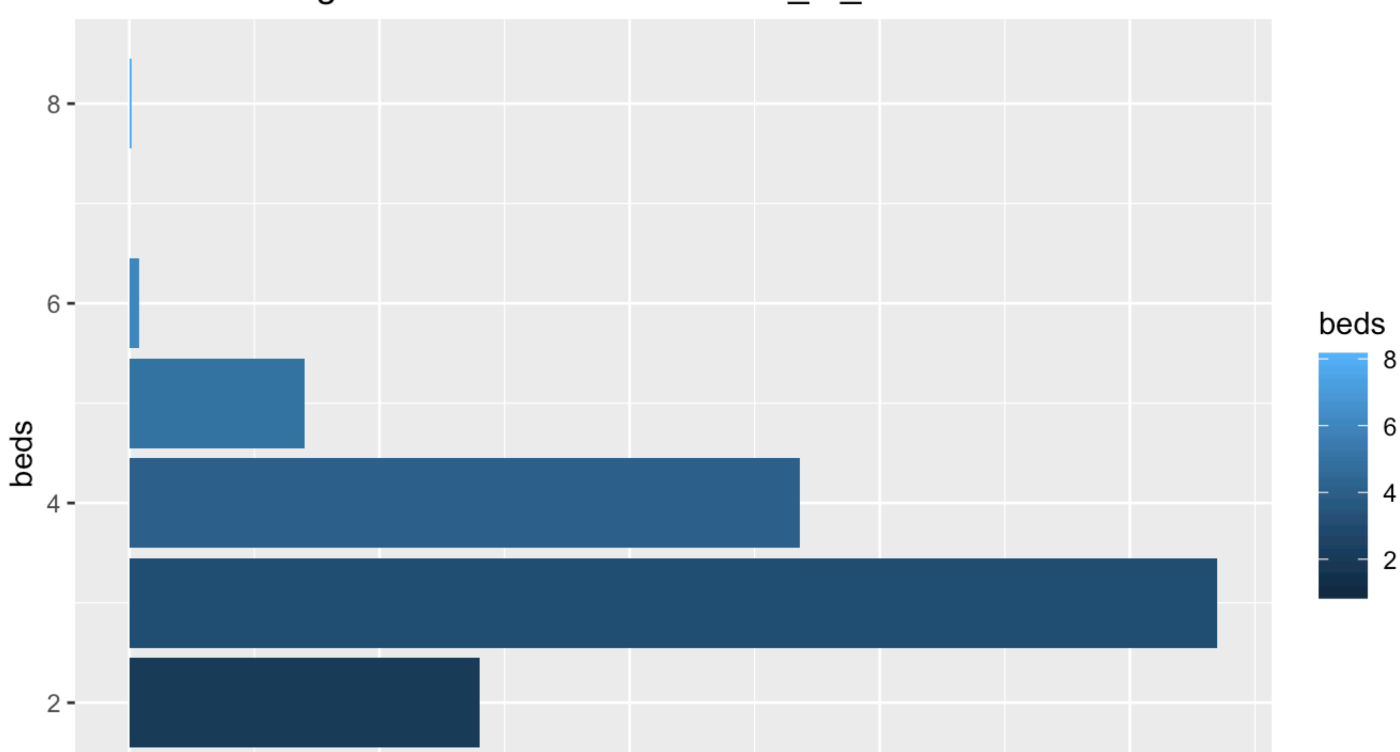
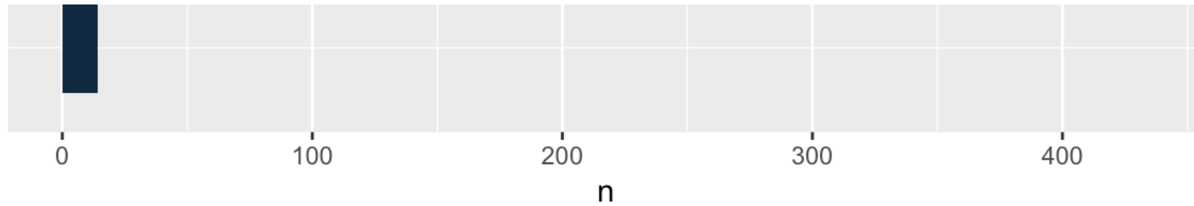


Fig.7 Bar Plot of Beds vs Num_of_Sales





The Bar plot Fig.6 clearly shows that when Beds increases the price increases and Fig.7 shows there is an outlier with bedrooms 8 beds which has only one sample which can be removed from the data.

4. Baths vs price

Let’s use the scatter plot to see if the Baths can be used as a predictor. Note that the sale price is transformed to 100s and the mean for the price for baths are plotted.

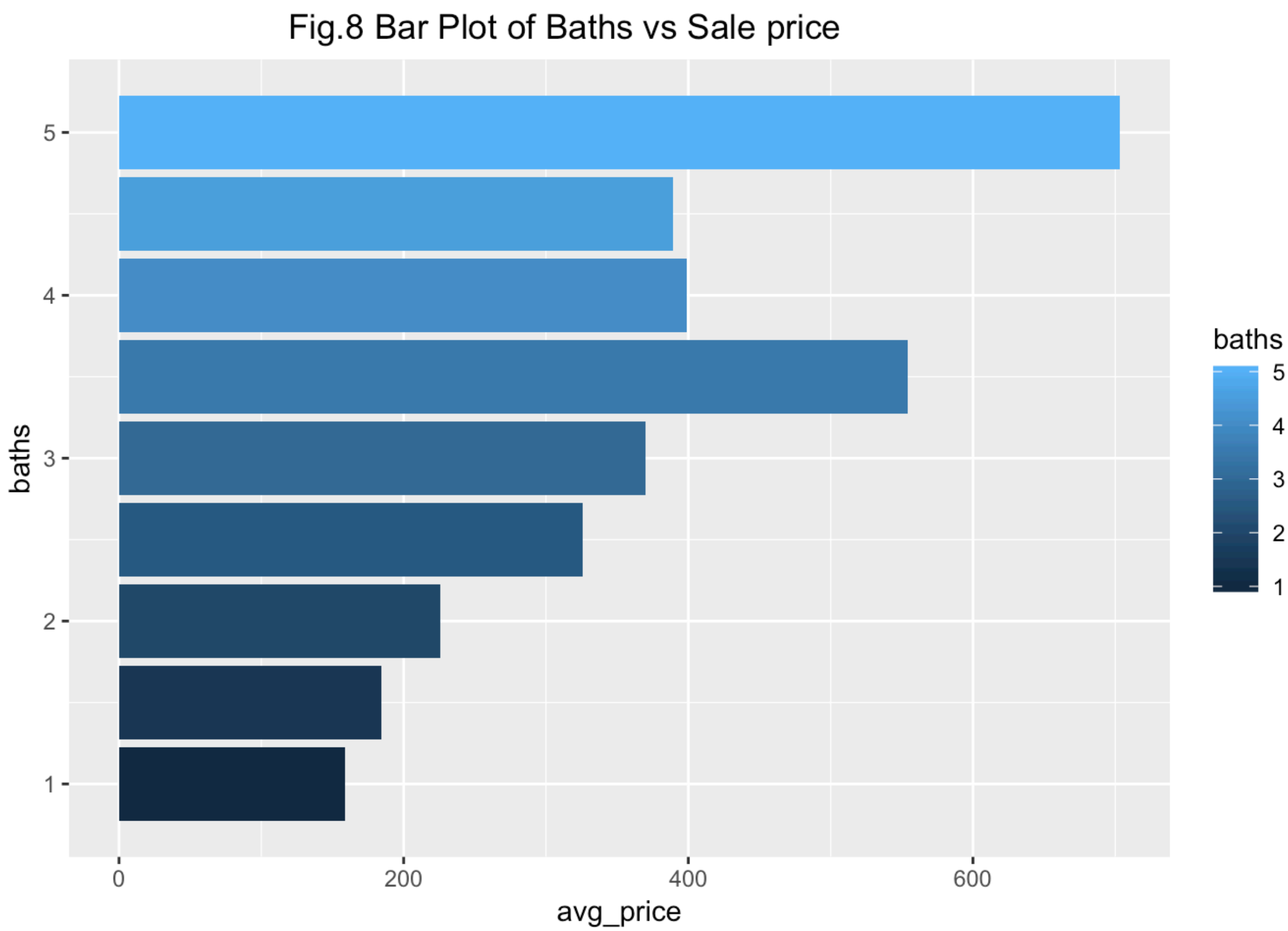


Fig.8 Bar Plot of Baths vs Sale price
baths
5
4
3
2
1
0
200
400
600
avg_price
baths
5
4
3
2
1

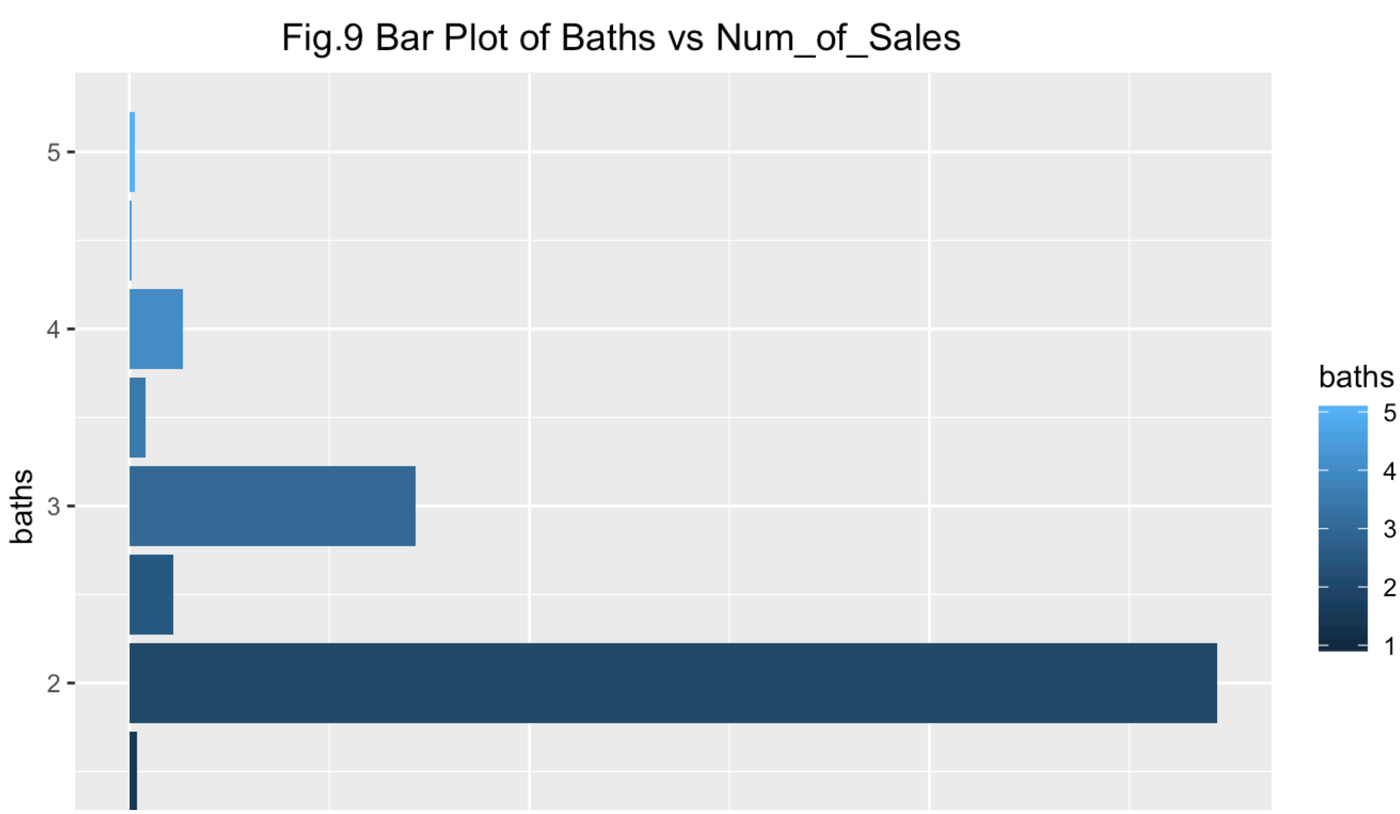
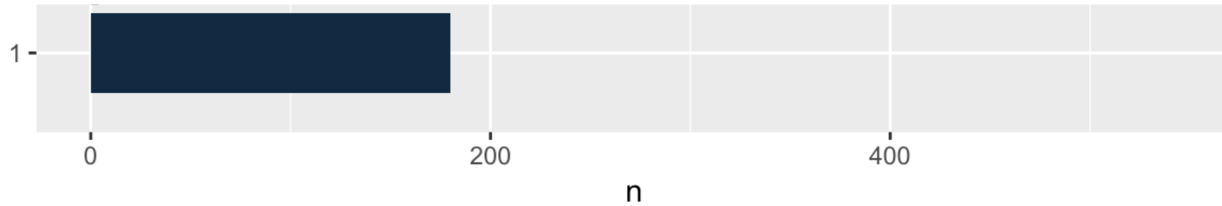


Fig.9 Bar Plot of Baths vs Num_of_Sales
baths
5
4
3
2
baths
5
4
3
2
1



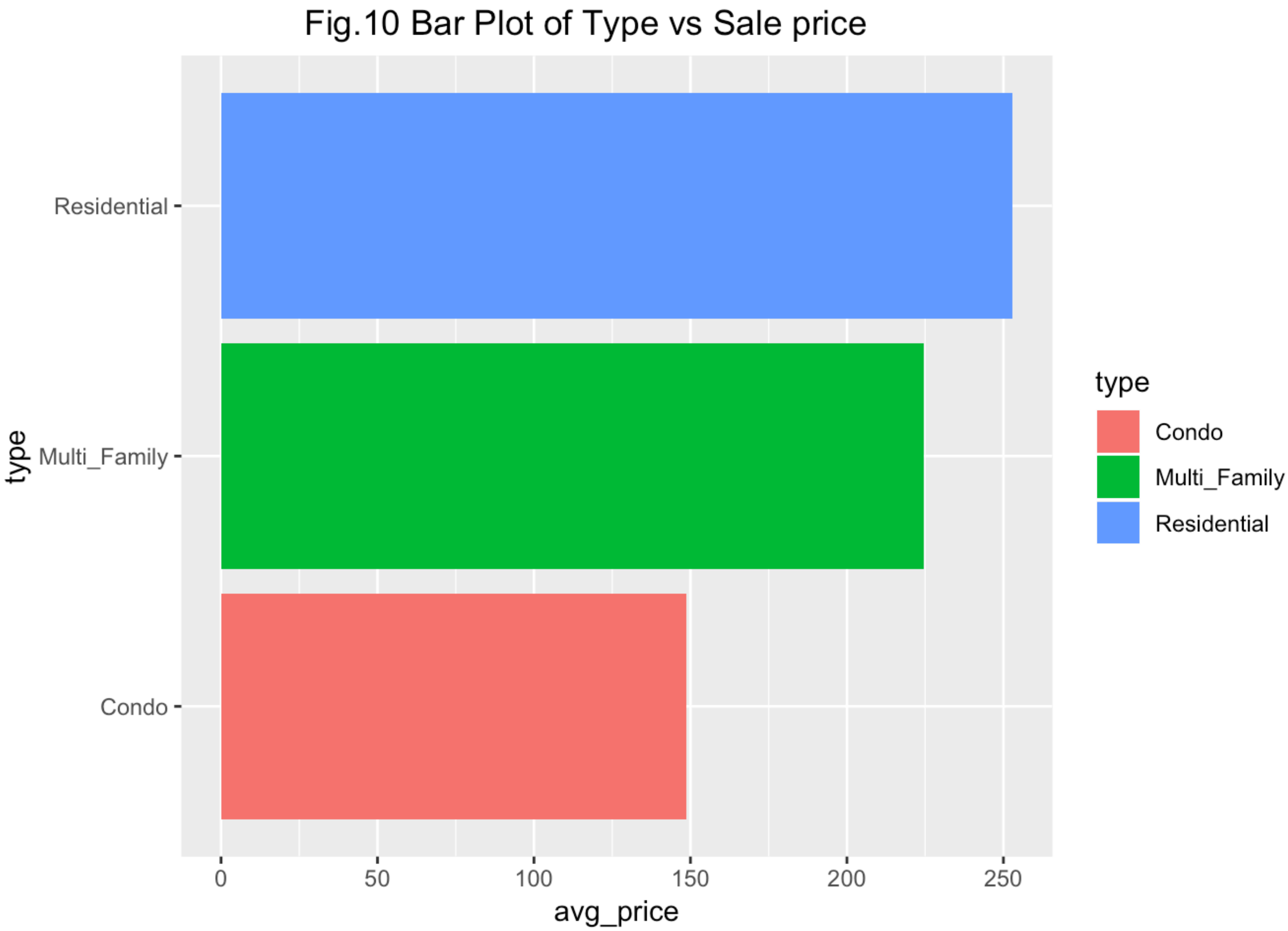

```
## # A tibble: 9 x 2
##   baths     n
##   <dbl> <int>
## 1   1     180
## 2   1.5     4
## 3   2     544
## 4   2.5    22
## 5   3     143
## 6   3.5     8
## 7   4      27
## 8   4.5     1
## 9   5       3
```

The Bar plot Fig.8 clearly shows that when baths increases the price increases and Fig.9 shows there is an outlier with 4.5 baths which has only one sample which can be removed from the data.

5. Type vs price

Let's use the scatter plot to see if the Baths can be used as a predictor.

Note that the sale price is transformed to 100s and the mean for the price for types are plotted.



The Bar plot Fig.10 shows that the condos and multi family units are cheaper compared to Single family units and has a correlation between sale price and hence can be picked up as a predictor.

6. Zip vs price

Let’s plot the zip field to see if it can be used as a Predictor.

Fig.11 Bar Plot of Zip vs Sale price

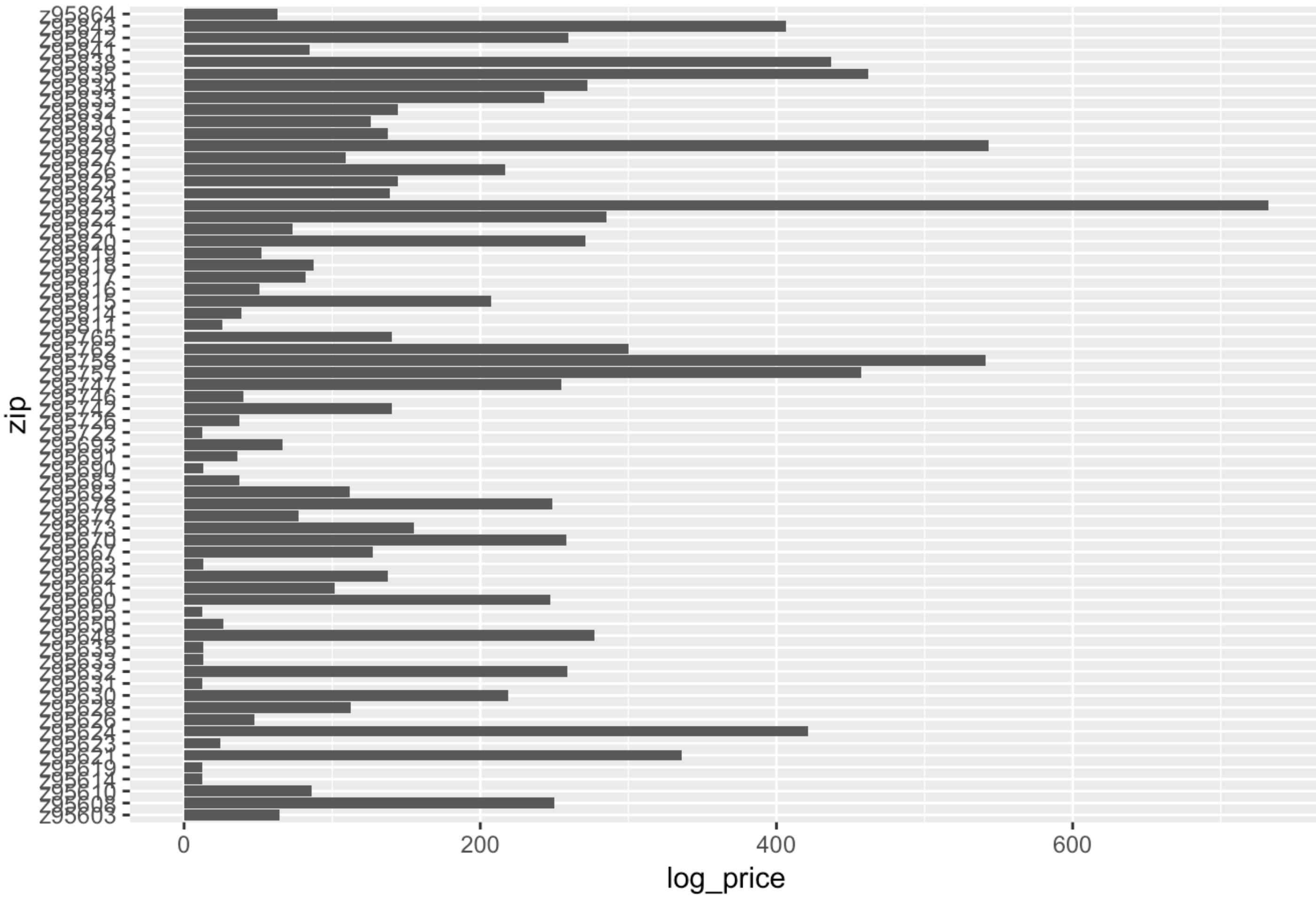
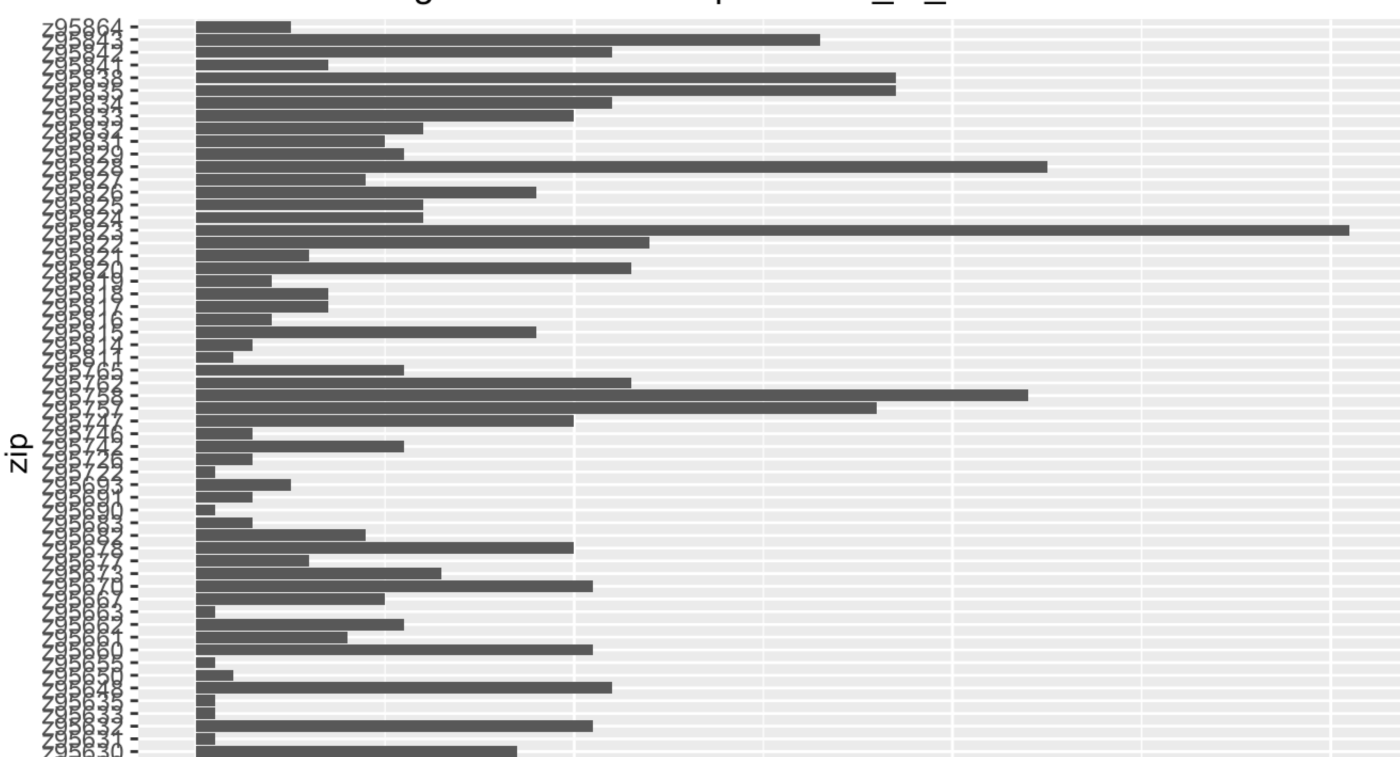
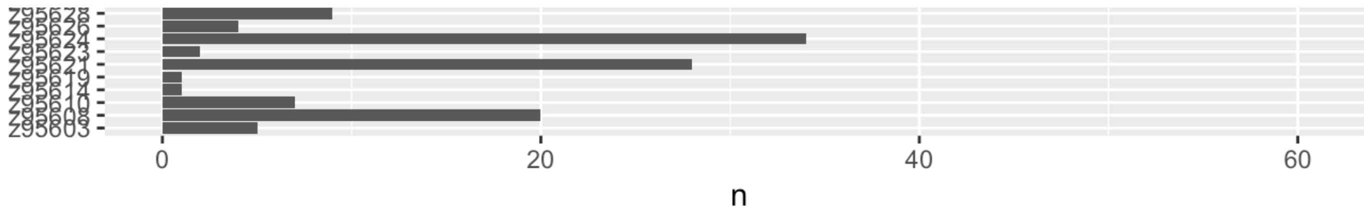


Fig.12 Bar Plot of Zip vs Num_of_Sales






```
## # A tibble: 196 x 3
## # Groups:   zip [68]
##    zip     beds n_bed
##    <fct>  <int> <int>
##  1 z95603     2     2
##  2 z95603     3     1
##  3 z95603     4     2
##  4 z95608     2     4
##  5 z95608     3    11
##  6 z95608     4     5
##  7 z95610     3     5
##  8 z95610     4     1
##  9 z95610     5     1
## 10 z95614     3     1
## # … with 186 more rows
```

Eventhough the Fig 11 shows some relationship between zip and sale price, there is no concrete evidence to support the reason for the dependency. For instance, the number of sales based on beds were summarized in an assumption that the bedroom counts may play a role in the raise in saleprice on the particular zipcode but the summary defies that, hence this can be eliminated to be a predictor.

7. latitude/longitude vs price

Let's plot the latitude and longitude field to see if it can be used as Predictors.

Fig.13 Scatter Plot of Latitude vs Sale price

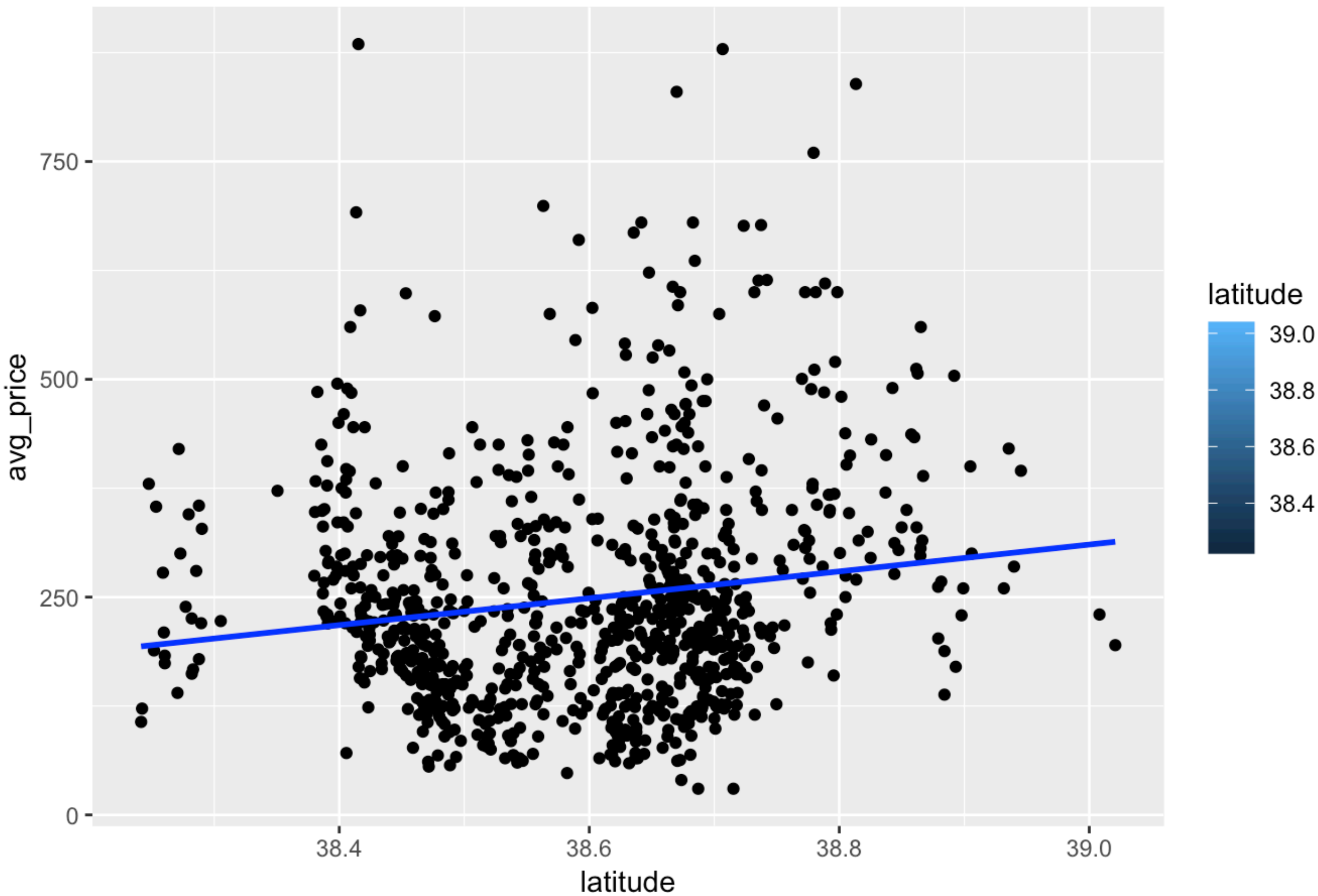
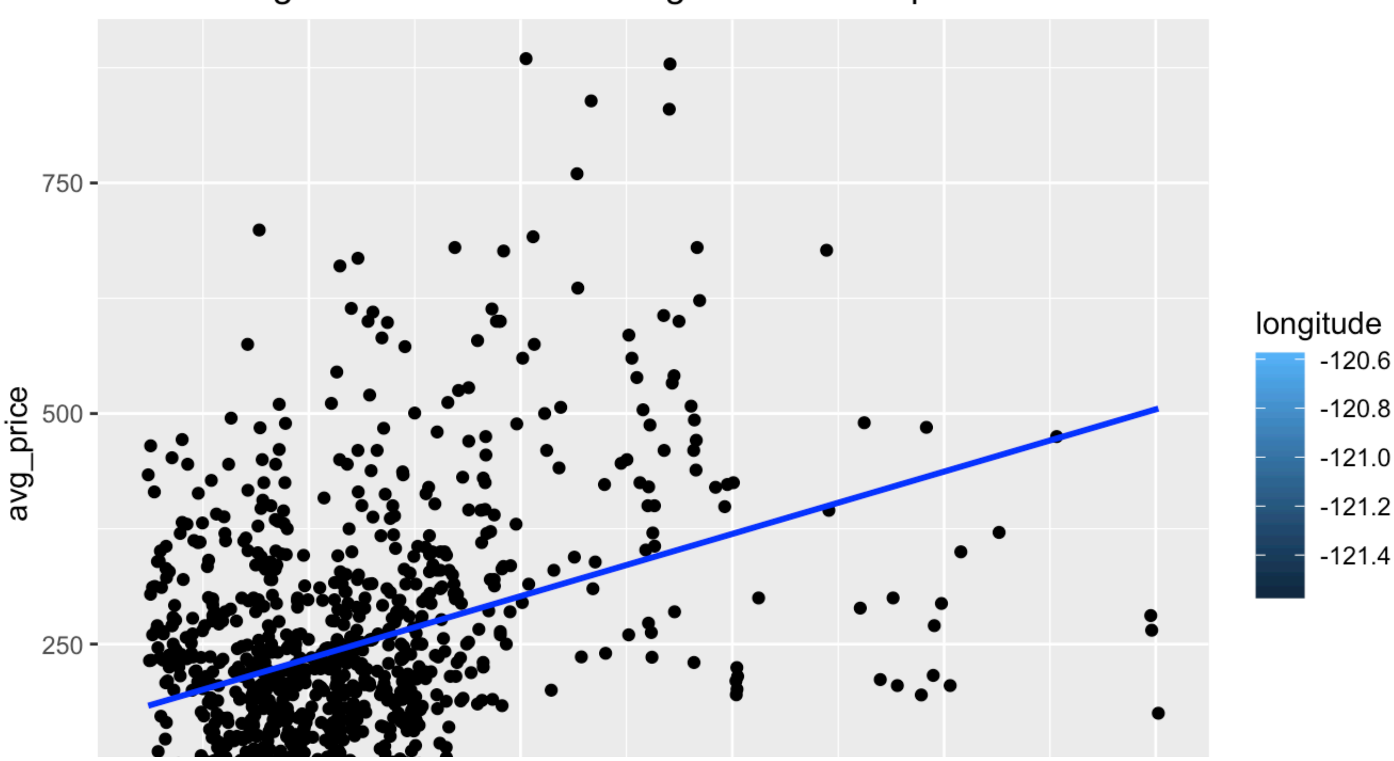
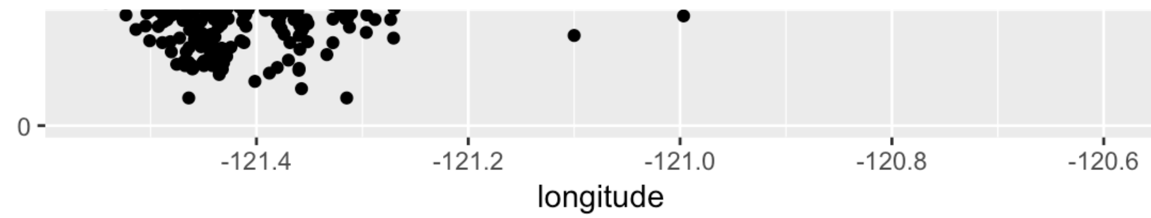


Fig.14 Scatter Plot of Longitude vs Sale price





The scatter plots of latitude and longitudes shows there is no linear relationship with prices and hence can be eliminated as predictors.

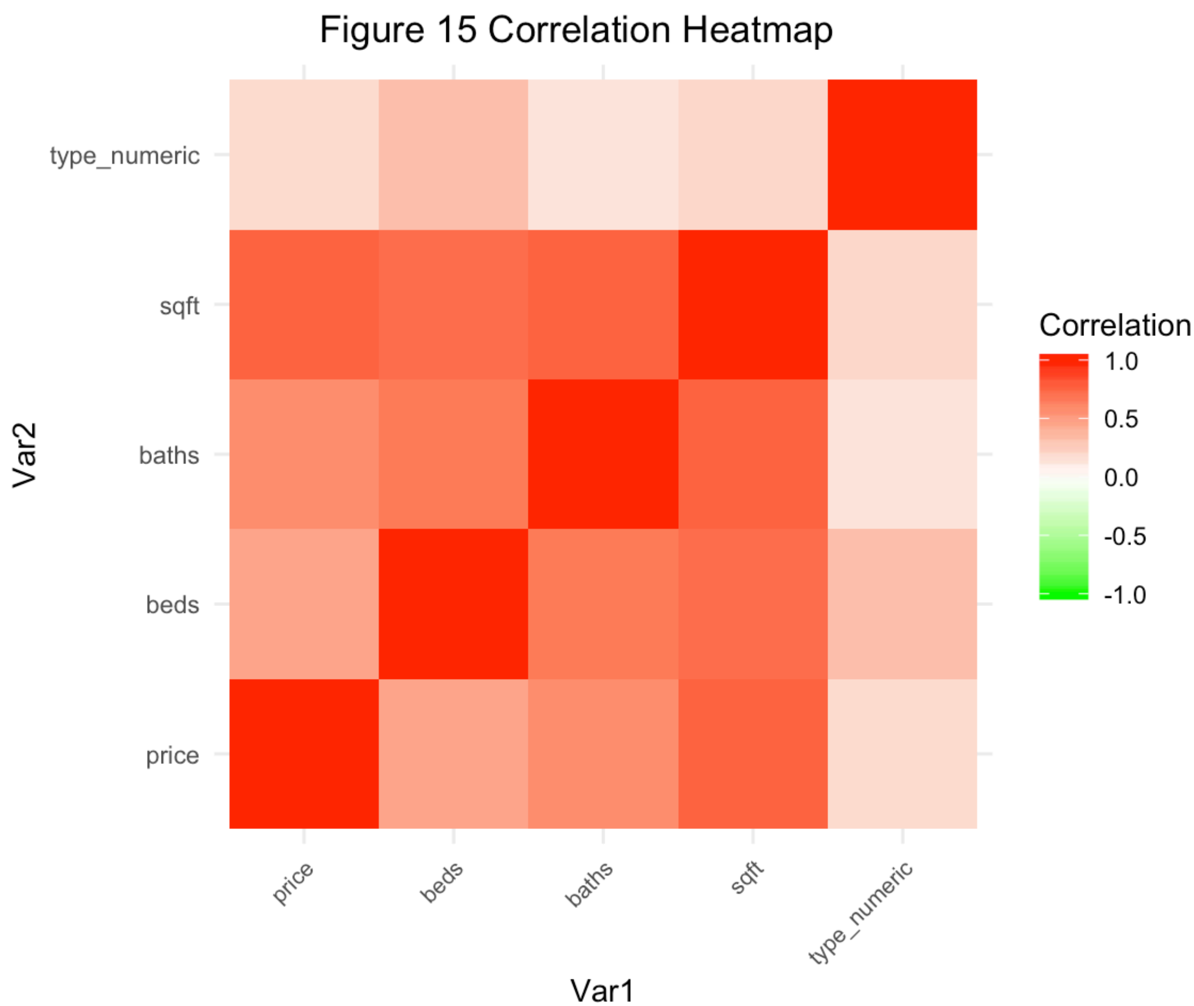
Corelations - Heat map demonstration

Let's plot the heat map to confirm the predictors we have chosen are good for our modeling.

Note that the red tiles in the diagonal shows the strong corelation between price and it's predictors.

Before we plot the Heat Map, lets convert the type to a numeric from factor.

```
Sacramento$type_numeric <-
  as.numeric(factor(Sacramento$type,levels=c("Condo","Multi_Family","Residential"),
                    labels=c(1,2,3), ordered = TRUE))
```



Data Preparation

Okay, we vizualized the data and picked the predictors.

Now lets remove the outliers as explained and split the data into train and test sets.

```
#====================================================================#
# Remove the house with beds = 8 and bath = 4.5
#====================================================================#

Sacramento_data <- Sacramento %>% filter((beds != 8)) %>% filter((baths != 4.5))

#====================================================================#
# Confirm the outliers were removed
#====================================================================#
unique(Sacramento_data$baths)
```

```
## [1] 1.0 2.0 3.0 4.0 5.0 1.5 2.5 3.5
```

```
unique(Sacramento_data$beds)
```

```
## [1] 2 3 1 4 5 6
```

```
#==========================================================================#
# Select the required data for modeling
#==========================================================================#

Sacramento_model_data <- Sacramento_data %>%
  select(log_price, beds, baths, sqft, type_numeric)

#==========================================================================#
# Partition the data into test and train datasets
#==========================================================================#

y_all <- Sacramento_model_data$log_price

index<-createDataPartition(y_all,times=1,p=0.8,list=FALSE)

train_data<- Sacramento_model_data[index,]
test_data<- Sacramento_model_data[-index,]

#==========================================================================#
#  Display the details of Train and Test datasets
#==========================================================================#

dim(train_data)
```

```
## [1] 745   5
```

```
dim(test_data)
```

```
## [1] 185   5
```

```
summary(train_data)
```

```
##    log_price          beds           baths            sqft     
##  Min.   :10.31   Min.   :1.000   Min.   :1.000   Min.   : 484  
##  1st Qu.:11.96   1st Qu.:3.000   1st Qu.:2.000   1st Qu.:1172  
##  Median :12.30   Median :3.000   Median :2.000   Median :1477  
##  Mean   :12.28   Mean   :3.289   Mean   :2.058   Mean   :1682  
##  3rd Qu.:12.63   3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:1940  
##  Max.   :13.69   Max.   :6.000   Max.   :5.000   Max.   :4400  
##   type_numeric  
##  Min.   :1.000  
##  1st Qu.:3.000  
##  Median :3.000  
##  Mean   :2.867  
##  3rd Qu.:3.000  
##  Max.   :3.000
```

```
summary(test_data)
```

```
##    log_price          beds           baths            sqft     
##  Min.   :10.99   Min.   :2.000   Min.   :1.000   Min.   : 539  
##  1st Qu.:11.95   1st Qu.:3.000   1st Qu.:2.000   1st Qu.:1140  
##  Median :12.30   Median :3.000   Median :2.000   Median :1440  
##  Mean   :12.30   Mean   :3.189   Mean   :2.011   Mean   :1657  
##  3rd Qu.:12.61   3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:1980  
##  Max.   :13.63   Max.   :6.000   Max.   :5.000   Max.   :4878  
##   type_numeric  
##  Min.   :1.000  
##  1st Qu.:3.000  
##  Median :3.000  
##  Mean   :2.897  
##  3rd Qu.:3.000  
##  Max.   :3.000
```

Methods - Modelling

1.Linear model

Let us use the linear model to predict the RMSE to make sure the squared errors and mimimal, so that we can use the data for other models to predict the fit

```
linreg <- lm(log_price~.,data = train_data)
summary(linreg)
```

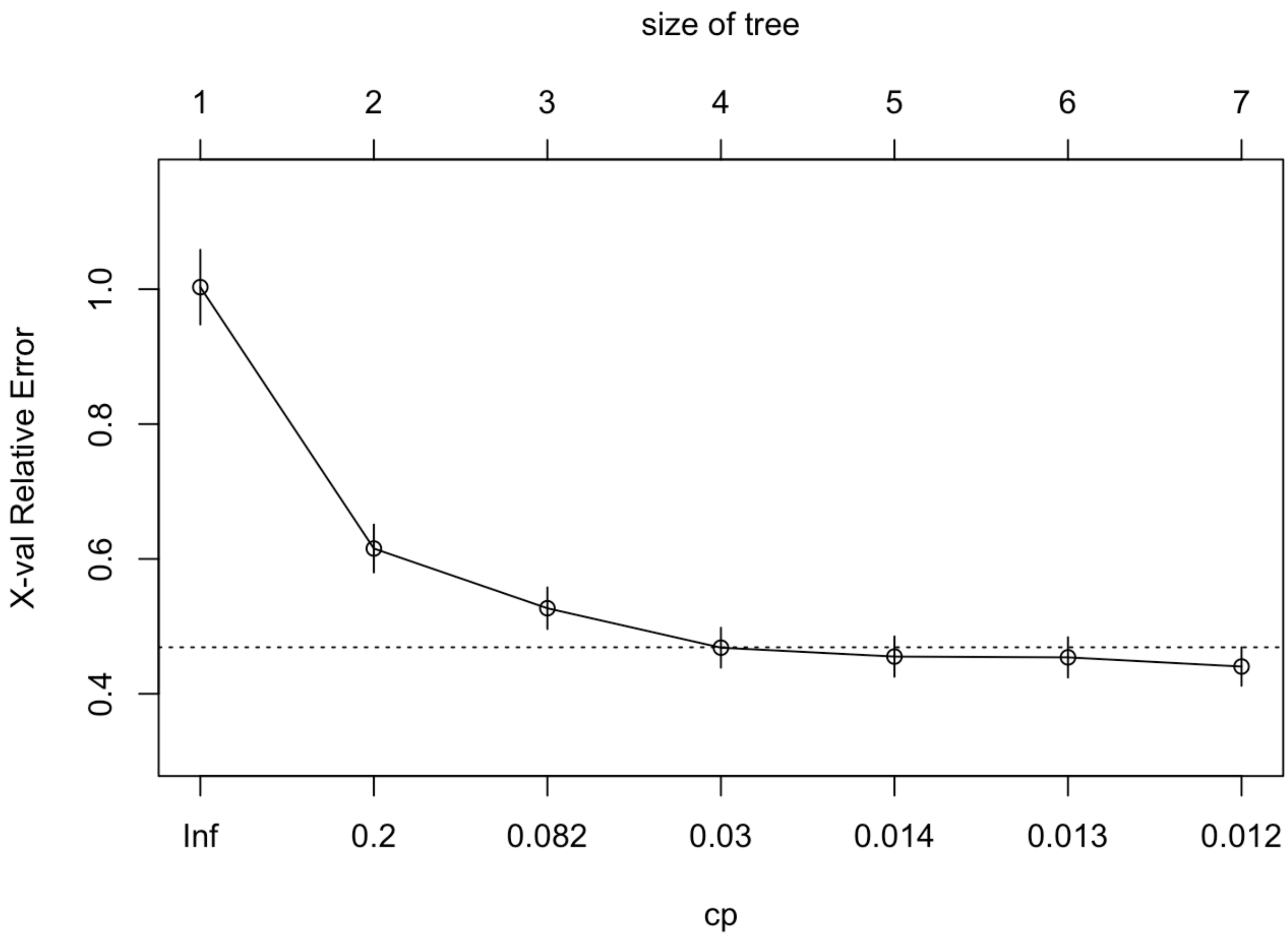
```
##
## Call:
## lm(formula = log_price ~ ., data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.75587 -0.22238  0.00903  0.23030  1.22231
##
## Coefficients:
##                Estimate  Std. Error t value             Pr(>|t|)
## (Intercept)  11.10717828  0.08399718 132.233 < 0.0000000000000002 ***
## beds         -0.05055495  0.02273543  -2.224             0.026475 *
## baths         0.09921736  0.02933540   3.382             0.000757 ***
## sqft          0.00049507  0.00003063  16.165 < 0.0000000000000002 ***
## type_numeric  0.10490029  0.02913622   3.600             0.000339 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3579 on 740 degrees of freedom
## Multiple R-squared:  0.5509, Adjusted R-squared:  0.5485
## F-statistic: 226.9 on 4 and 740 DF,  p-value: < 0.00000000000000022
```

```
pred_lm_lp <- predict(linreg,test_data,type="response")
```

2.Regression trees - Recursive partitioning

Lets pick the confusion parameter for rpart function for cross validation.

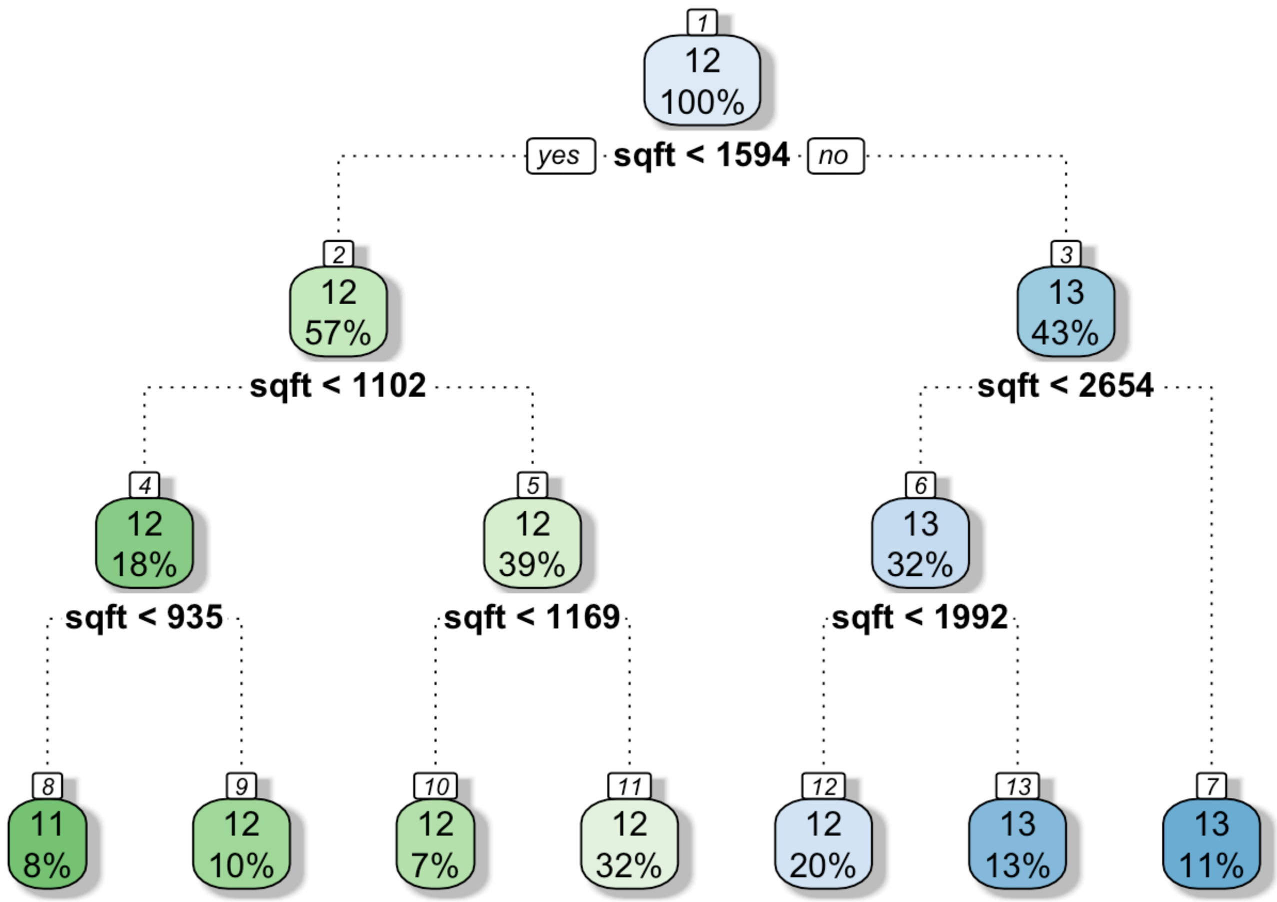
```
rpart_cp = rpart.control(cp=0.01)

rpart_tree <- rpart(log_price ~ .,data=train_data,control = rpart_cp)
```




```
##
## Regression tree:
## rpart(formula = log_price ~ ., data = train_data, control = rpart_cp)
##
## Variables actually used in tree construction:
## [1] sqft
##
## Root node error: 211.02/745 = 0.28325
##
## n= 745
##
##         CP nsplit rel error  xerror     xstd
## 1 0.394402      0   1.00000 1.00304 0.055433
## 2 0.099840      1   0.60560 0.61544 0.035568
## 3 0.066656      2   0.50576 0.52689 0.030900
## 4 0.013766      3   0.43910 0.46840 0.029624
## 5 0.013514      4   0.42534 0.45522 0.029884
## 6 0.013442      5   0.41182 0.45403 0.029849
## 7 0.010000      6   0.39838 0.44044 0.028513
```

The plot shows the relative error is getting minimized when cp=0.011.

Print the tree split which shows the split based on sqft. Note that the rpart() function ignored other predictors since those gave the same split as like sqft.

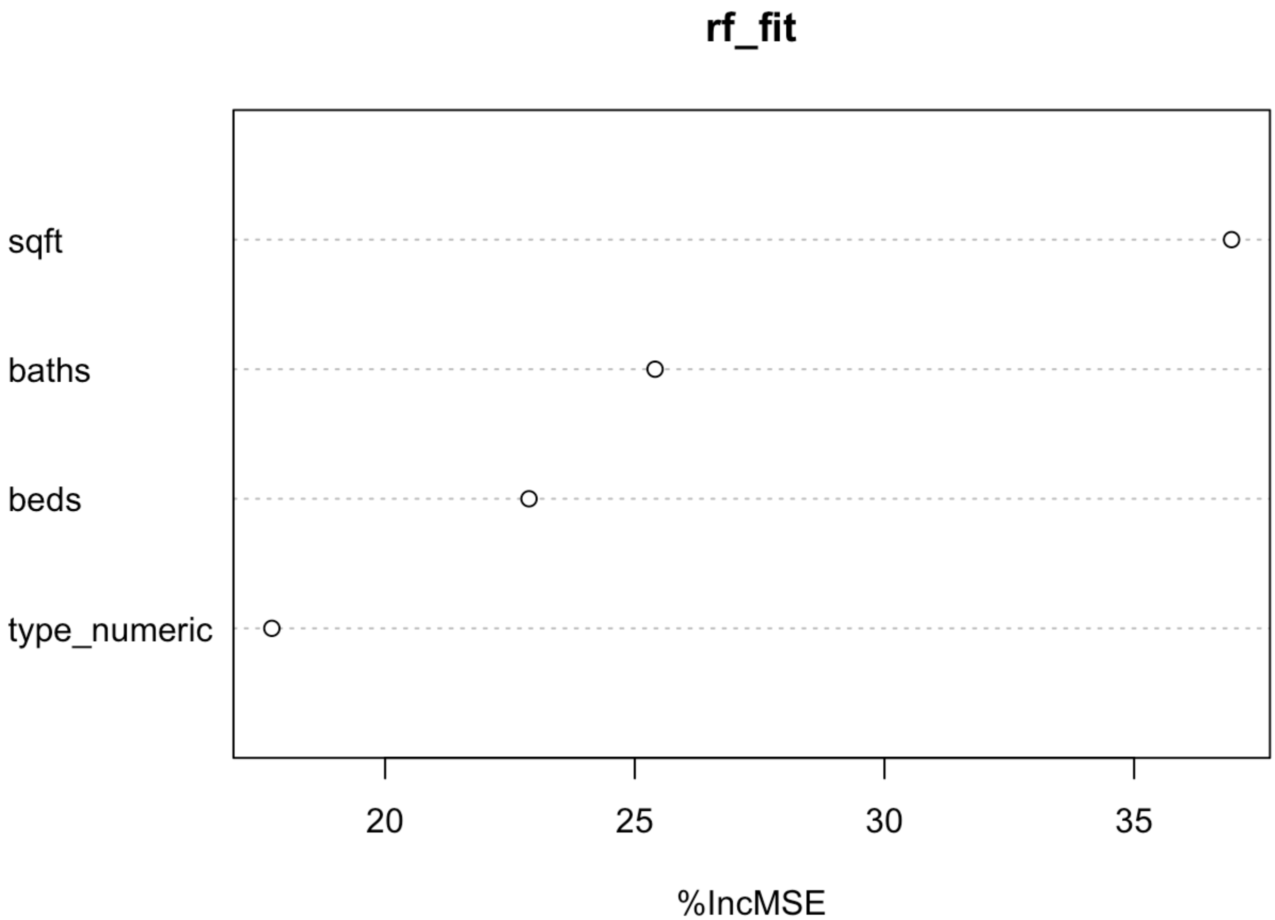


3. Random Forest - Model

Fit the training data using Random Forest algorithm.

```
rf_fit <- randomForest(log_price ~ .,data=train_data,
                       importance =TRUE,ntree=500,
                       nodesize=7, na.action=na.roughfix)
```

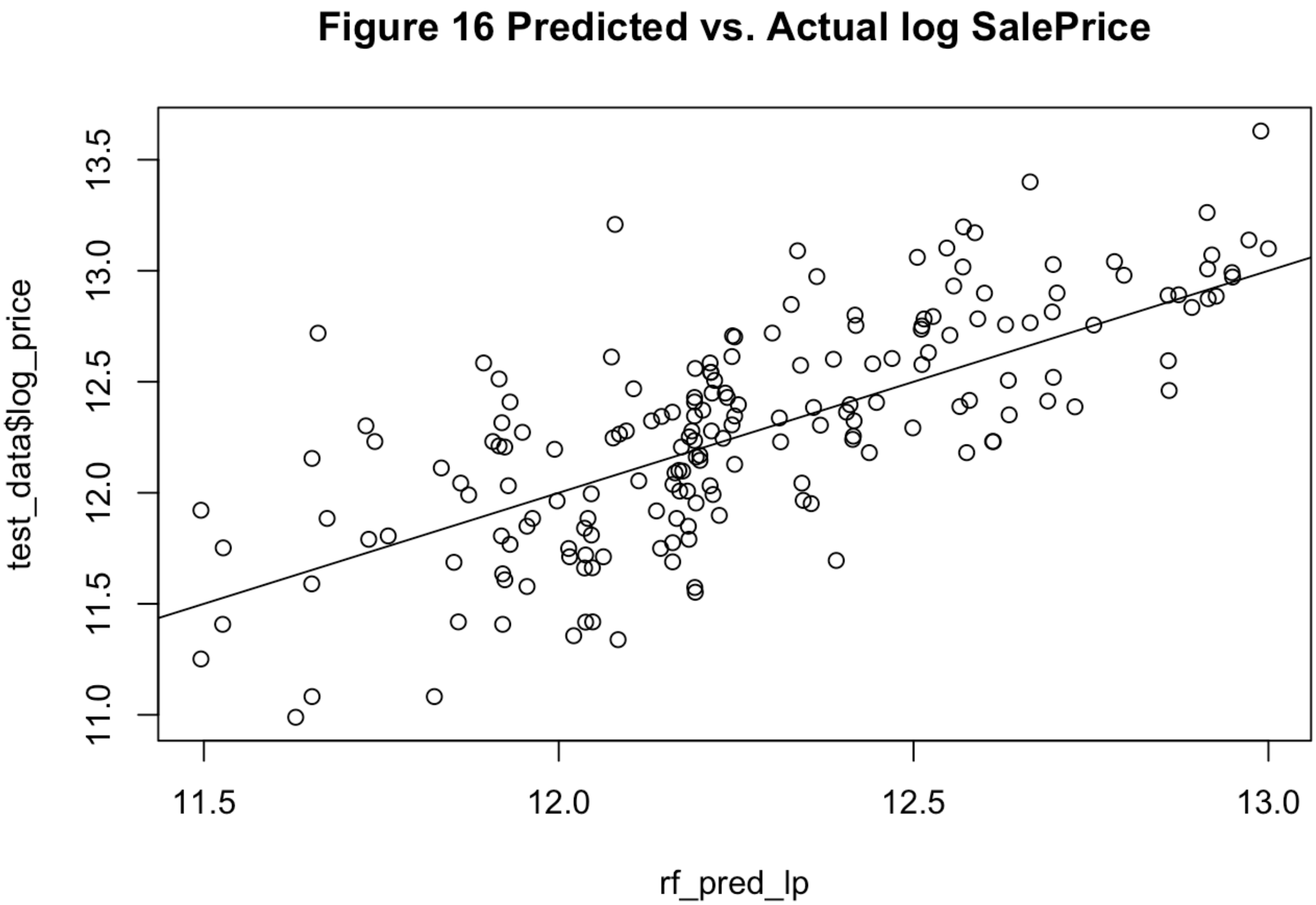
Plot the variable importance graph.



Now, let's predict for the test data

```
rf_pred_lp <- predict(rf_fit, newdata=test_data )
```

Plot the predicted and actual values for Random Forest prediction.



4.KNN model

Create the control for Cross Validation Kfold = 10 with 90% of data for training.

```
control_knn<-trainControl(method = "cv", number = 10, p=0.9)
```

The following code will determine the “K-Value” from the cross validation

```
train_knn <- train(log_price~.,data=train_data,method="knn",
                   tuneGrid = data.frame(k=seq(1,10,by=0.25)),
                   trControl = control_knn)
k_value <- as.numeric(train_knn$bestTune)
print(k_value)
```

```
## [1] 9.75
```

Now, let’s fit the KNN using the correct K-Value

```
fit_knn<-train(log_price~.,data=train_data,method="knn",
               tuneGrid=data.frame(k=k_value))
```

Predict the KNN fit using the test data.

```
knn_pred_lp<-predict(fit_knn,newdata = test_data)
```

Great!. We are done with fitting various models on the test and train datasets.

Now let's see the results.

Results

Display the RMSE values for each model used in this script.

```
accuracy_details %>% knitr::kable()
```

Method	RMSE
LM	0.3418035
RPART	0.3390634
RF	0.3425370
KNN	0.3483518

Please note that the predictions for each model is combined into the dataframe "combined_prediction".

Conclusion

As explained in the Introduction section, we reached our goal by selecting the dataset, used various visualization techniques on the data to pick the correct predictors, removed the outliers and got the heat map to confirm that we picked the correct predictors.

The data is then cleaned by removing the outliers, split that into test and training sets for predictions.

The following models are used to predict the outcomes and the results were printed to reach the goal.

1. Linear Model.
2. Regression Trees.
3. Random Forest.
4. K- Nearest Neighbors.