install.packages("magrittr") install.packages("ggmap") install.packages("geosphere") install.packages("httr")

In [1]:
```
install.packages("rpart.plot")
install.packages("inTrees")
```

```
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
also installing the dependencies ‘RRF’, ‘arules’, ‘gbm’

Warning message in install.packages("inTrees"):
“installation of package ‘RRF’ had non-zero exit status”Warning message
in install.packages("inTrees"):
“installation of package ‘arules’ had non-zero exit status”Warning mess
age in install.packages("inTrees"):
“installation of package ‘gbm’ had non-zero exit status”Warning message
in install.packages("inTrees"):
“installation of package ‘inTrees’ had non-zero exit status”Updating HT
ML index of packages in '.Library'
Making 'packages.html' ... done
```

King County Housing Prices

Sruthi Jogi, Jonathan McFadden, Angela Zhao

TCSS-551: Big Data Analytics -:- ***Autumn 2017***

Introduction

Overview

For our final project, we have chosen to analyze data covering housing sales in King County. To do this, we are using the data from the **Kaggle** *King County House Sales Prediction* page at

https://www.kaggle.com/harlfoxem/housesalesprediction

From this page, we sign-up for an account (*free, but required for downloading*) and then download the *zip* file containing the *CSV* file with the data.

Our goal is to use this data to create models for home sales in King County based on the feature information provided in the obtained data file. Our eventual goal is two-fold. First, we wish to create a model or models which will enable us to quantitatively predict house sale prices, using this data set as the basis for our model or models. Our other goal is to determine, based on the obtained data, which features are most important to the sale price of a house.

Data File

Our first task is to import, examine, and then give an overall description of the data. We are especially interested in the size and descriptive contents of the data file. Specifially, we want to know the number of sales contined within the data file and, especially, what parameters the data file uses to describe each house sale. Furthermore, we want to check the import to ensure that the data was initially complete, that it was then imported correctly, and that ***R*** is interpreting the imported data properly.

Import and First-Look

We begin by importing the data file into the '***houseDFo***( )' data frame. This data frame will serve as an intial data-frame, not the working one. This is because we may need an initial frame to reload as a we clean the data, allowing us to avoid having to reimport the CSV file over ang over again. Thus, we now import the CSV file into this initial data frame.

In [2]:
```
houseDFo <- read.csv("../houseData.csv")
```

We are now interested in the number of data-points contined within the data file. Thus, we want to see how many row ***R*** has imported.

In [3]:
```
nrow(houseDFo)
```

21613

We also want to see how many descriptors the imported data uses to describe each house sale. Thus we want to see how many columns ***R*** has imported.

In [4]:
```
ncol(houseDFo)
```

21

In addition, we want to see what the labels for those columns are and what type of values the elements of each column have (*interger, numeric, string, etc.*)

In [5]:
```
sapply(houseDFo, class)
```

id	'numeric'
date	'factor'
price	'numeric'
bedrooms	'integer'
bathrooms	'numeric'
sqft_living	'integer'
sqft_lot	'integer'
floors	'numeric'
waterfront	'integer'
view	'integer'
condition	'integer'
grade	'integer'
sqft_above	'integer'
sqft_basement	'integer'
yr_built	'integer'
yr_renovated	'integer'
zipcode	'integer'
lat	'numeric'
long	'numeric'
sqft_living15	'integer'
sqft_lot15	'integer'

From above, it is clear that the **date** column did not import as a *date*, instead importing as a *factor*. Therefore, we will now examine the first few rows of the imported data to see what may have caused the issues with imporation.

In [6]:
```
head(houseDFo)
```

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floor
7129300520	20141013T000000	221900	3	1.00	1180	5650	1
6414100192	20141209T000000	538000	3	2.25	2570	7242	2
5631500400	20150225T000000	180000	2	1.00	770	10000	1
2487200875	20141209T000000	604000	4	3.00	1960	5000	1
1954400510	20150218T000000	510000	3	2.00	1680	8080	1
7237550310	20140512T000000	1225000	4	4.50	5420	101930	1

Clearly, some elements of the data file did not import correctly; therefore, we must clean the data before we can proceed to analysis.

Clean the Data

Missing Data

First, we will check to see if there are any missing data points.

In [7]:
```
houseDFo[!complete.cases(houseDFo),]
```

```
Warning message in cbind(parts$left, ellip_h, parts$right, deparse.leve
l = 0L):
“number of rows of result is not a multiple of vector length (arg 2)”Wa
rning message in cbind(parts$left, ellip_h, parts$right, deparse.level
= 0L):
“number of rows of result is not a multiple of vector length (arg 2)”Wa
rning message in cbind(parts$left, ellip_h, parts$right, deparse.level
= 0L):
“number of rows of result is not a multiple of vector length (arg 2)”Wa
rning message in cbind(parts$left, ellip_h, parts$right, deparse.level
= 0L):
“number of rows of result is not a multiple of vector length (arg 2)”
```

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	⋯	gr

Since there are no missing data points, we can move on to the dates.

Dates

From the first few rows of the data table seen above, it is clear that we must first strip the "*T000000*" string at the end of every date. To do this, we require the **stringr** library. Thus, we import **stringr**

In [8]:
```
library(stringr)
```

so we can now strip the offending substrings. Before stripping these substrings, we create a copy of our initial data frame, ***houseDFo***( ), so that our initial import data frame will remain untouched, and therefore available for reloading other data frames. Thus, we create the copy and strip the substrings, storing the result in the copied data frame ***houseDFo1***( ).

In [9]:
```
houseDFo1 <- houseDFo
houseDFo1$date = str_replace(houseDFo$date, "T000000", "")
```

We now examine the result of this

In [10]:
```
head(houseDFo1)
```

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	water
7129300520	20141013	221900	3	1.00	1180	5650	1	0
6414100192	20141209	538000	3	2.25	2570	7242	2	0
5631500400	20150225	180000	2	1.00	770	10000	1	0
2487200875	20141209	604000	4	3.00	1960	5000	1	0
1954400510	20150218	510000	3	2.00	1680	8080	1	0
7237550310	20140512	1225000	4	4.50	5420	101930	1	0

The dates are now just strings of numbers with the format '*yyyymmdd*'; therefore, we can use the date conversion method from ***R*** to convert these dates.

In [11]:
```
houseDFo1 <- transform(houseDFo1, date = as.Date(date, "%Y%m%d"))
```

To ensure that the conversion to dates happend properly, we will no check the column data types followed by looking at the first few rows of the data.

In [12]:
```
sapply(houseDFo1, class)
head(houseDFo1)
```

id	'numeric'
date	'Date'
price	'numeric'
bedrooms	'integer'
bathrooms	'numeric'
sqft_living	'integer'
sqft_lot	'integer'
floors	'numeric'
waterfront	'integer'
view	'integer'
condition	'integer'
grade	'integer'
sqft_above	'integer'
sqft_basement	'integer'
yr_built	'integer'
yr_renovated	'integer'
zipcode	'integer'
lat	'numeric'
long	'numeric'
sqft_living15	'integer'
sqft_lot15	'integer'

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfron
7129300520	2014-10-13	221900	3	1.00	1180	5650	1	0
6414100192	2014-12-09	538000	3	2.25	2570	7242	2	0
5631500400	2015-02-25	180000	2	1.00	770	10000	1	0
2487200875	2014-12-09	604000	4	3.00	1960	5000	1	0
1954400510	2015-02-18	510000	3	2.00	1680	8080	1	0
7237550310	2014-05-12	1225000	4	4.50	5420	101930	1	0

Since the results for the date conversions are as desired, we can now store the data in a final data frame followed by moving on to begining our analysis.

In [13]:
```
houseDF <- houseDFo1
```

We will also create a version of the data with the **ID** column stripped out.

In [14]:
```
houseDFa <- houseDF[-c(1)]
```

Initial Analysis

To begin our analysis, we will look at the basic statistics of every column (*except the date*).

In [15]:
```
sapply(houseDFa[-c(1)], function(x) list(mean=mean(x),
                                          stdev=sd(x, na.rm=TRUE)))
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vie
mean	540088.1	3.370842	2.114757	2079.9	15106.97	1.494309	0.007541757	0.2
stdev	367127.2	0.9300618	0.7701632	918.4409	41420.51	0.5399889	0.0865172	0.7

and get a summary of the entire

In [16]:
```
summary(houseDFa)
```

```
      date                price            bedromos          bathrooms
 Min.   :2014-05-02   Min.   :  75000   Min.   : 0.000   Min.   :0.000
 1st Qu.:2014-07-22   1st Qu.: 321950   1st Qu.: 3.000   1st Qu.:1.750
 Median :2014-10-16   Median : 450000   Median : 3.000   Median :2.250
 Mean   :2014-10-29   Mean   : 540088   Mean   : 3.371   Mean   :2.115
 3rd Qu.:2015-02-17   3rd Qu.: 645000   3rd Qu.: 4.000   3rd Qu.:2.500
 Max.   :2015-05-27   Max.   :7700000   Max.   :33.000   Max.   :8.000
  sqft_living       sqft_lot           floors         waterfront
 Min.   :  290   Min.   :    520   Min.   :1.000   Min.   :0.000000
 1st Qu.: 1427   1st Qu.:   5040   1st Qu.:1.000   1st Qu.:0.000000
 Median : 1910   Median :   7618   Median :1.500   Median :0.000000
 Mean   : 2080   Mean   :  15107   Mean   :1.494   Mean   :0.007542
 3rd Qu.: 2550   3rd Qu.:  10688   3rd Qu.:2.000   3rd Qu.:0.000000
 Max.   :13540   Max.   :1651359   Max.   :3.500   Max.   :1.000000
      view          condition          grade          sqft_above
 Min.   :0.0000   Min.   :1.000   Min.   : 1.000   Min.   : 290
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000   1st Qu.:1190
 Median :0.0000   Median :3.000   Median : 7.000   Median :1560
 Mean   :0.2343   Mean   :3.409   Mean   : 7.657   Mean   :1788
 3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000   3rd Qu.:2210
 Max.   :4.0000   Max.   :5.000   Max.   :13.000   Max.   :9410
 sqft_basement       yr_built     yr_renovated        zipcode
 Min.   :   0.0   Min.   :1900   Min.   :   0.0   Min.   :98001
 1st Qu.:   0.0   1st Qu.:1951   1st Qu.:   0.0   1st Qu.:98033
 Median :   0.0   Median :1975   Median :   0.0   Median :98065
 Mean   : 291.5   Mean   :1971   Mean   :  84.4   Mean   :98078
 3rd Qu.: 560.0   3rd Qu.:1997   3rd Qu.:   0.0   3rd Qu.:98118
 Max.   :4820.0   Max.   :2015   Max.   :2015.0   Max.   :98199
      lat             long         sqft_living15    sqft_lot15
 Min.   :47.16   Min.   :-122.5   Min.   : 399   Min.   :   651
 1st Qu.:47.47   1st Qu.:-122.3   1st Qu.:1490   1st Qu.:  5100
 Median :47.57   Median :-122.2   Median :1840   Median :  7620
 Mean   :47.56   Mean   :-122.2   Mean   :1987   Mean   : 12768
 3rd Qu.:47.68   3rd Qu.:-122.1   3rd Qu.:2360   3rd Qu.: 10083
 Max.   :47.78   Max.   :-121.3   Max.   :6210   Max.   :871200
```

Model A

We also run a simple linear model on the *entire* dataset so that we can see how significant each variable is to determining the price (*basically running a t-Test on all variables*). To do this, we need to **nnet** library, so we load it

In [17]:
```
library(nnet)
```

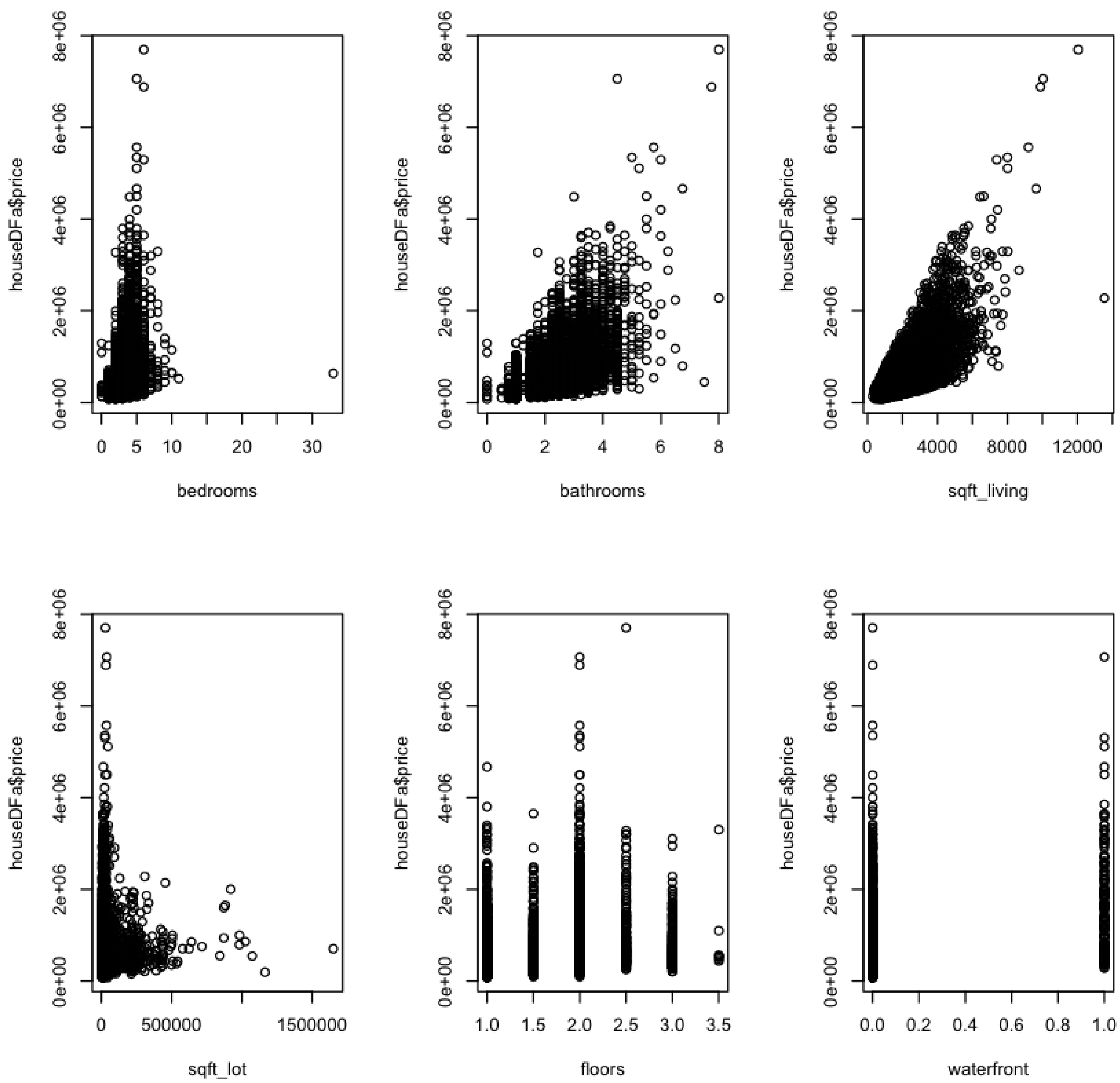
Then we run the model, our first, and display the results.

In [18]:
```
modelA <- lm(price ~., data=houseDFa)
summary(modelA)
```

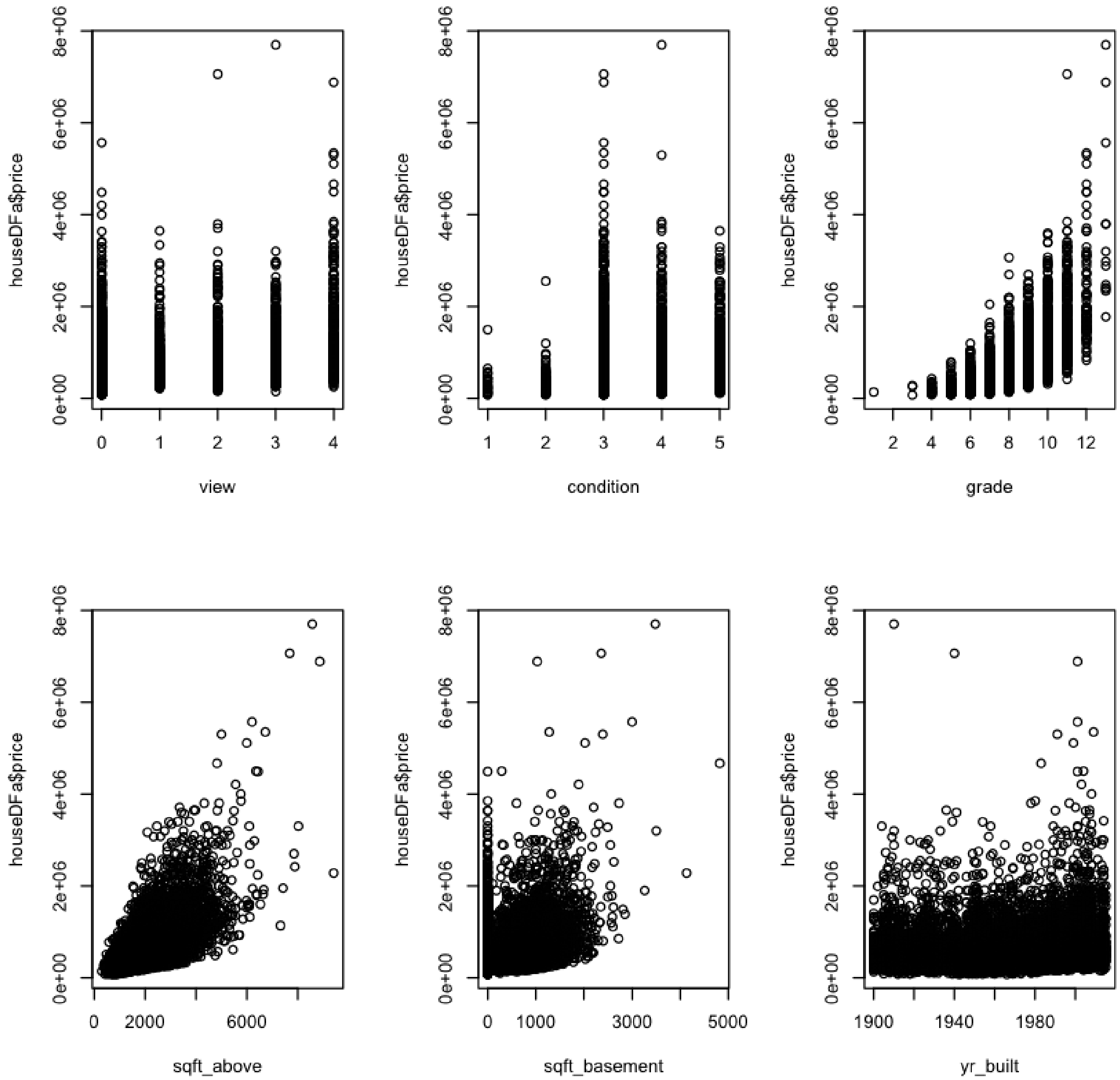
```
Call:
lm(formula = price ~ ., data = houseDFa)

Residuals:
     Min       1Q   Median       3Q      Max 
-1306672   -98900    -8963    77327  4330103 

Coefficients: (1 not defined because of singularities)
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)    4.618e+06  2.933e+06   1.574  0.11539    
date           1.165e+02  1.213e+01   9.608  < 2e-16 ***
bedrooms      -3.588e+04  1.888e+03 -19.005  < 2e-16 ***
bathrooms      4.137e+04  3.247e+03  12.741  < 2e-16 ***
sqft_living    1.502e+02  4.376e+00  34.327  < 2e-16 ***
sqft_lot       1.257e-01  4.782e-02   2.629  0.00858 ** 
floors         7.158e+03  3.589e+03   1.995  0.04610 *  
waterfront     5.826e+05  1.732e+04  33.628  < 2e-16 ***
view           5.260e+04  2.136e+03  24.629  < 2e-16 ***
condition      2.774e+04  2.351e+03  11.799  < 2e-16 ***
grade          9.624e+04  2.149e+03  44.791  < 2e-16 ***
sqft_above     3.084e+01  4.351e+00   7.088 1.40e-12 ***
sqft_basement         NA         NA      NA       NA    
yr_built      -2.618e+03  7.251e+01 -36.113  < 2e-16 ***
yr_renovated   2.079e+01  3.649e+00   5.698 1.23e-08 ***
zipcode       -5.807e+02  3.292e+01 -17.643  < 2e-16 ***
lat            6.053e+05  1.072e+04  56.487  < 2e-16 ***
long          -2.136e+05  1.311e+04 -16.300  < 2e-16 ***
sqft_living15  2.195e+01  3.441e+00   6.381 1.79e-10 ***
sqft_lot15    -3.825e-01  7.311e-02  -5.232 1.69e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 200800 on 21594 degrees of freedom
Multiple R-squared:  0.701,	Adjusted R-squared:  0.7008 
F-statistic:  2813 on 18 and 21594 DF,  p-value: < 2.2e-16
```

We also plot the data to look for potential relationships and outliers.

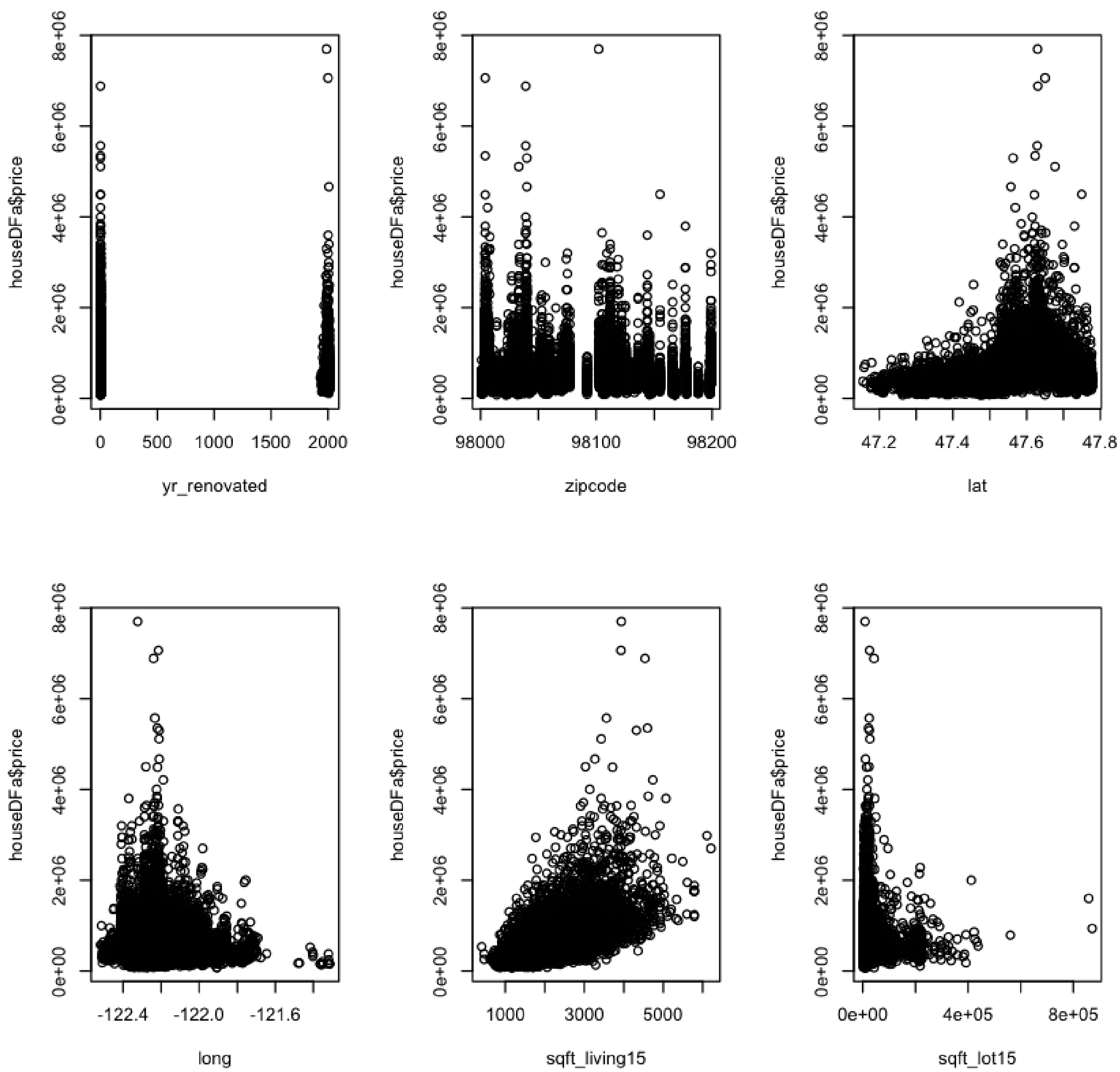
In [19]:
```
par(mfrow=c(2,3))
for(i in 3:8) {plot(houseDFa[,i], houseDFa$price, xlab=names(houseDFa[i
]), ylab=names(houseDFa$price))}
```



In [20]:
```
par(mfrow=c(2,3))
for(i in 9:14) {plot(houseDFa[,i], houseDFa$price, xlab=names(houseDFa[i
]), ylab=names(houseDFa$price))}
```



In [21]:
```
par(mfrow=c(2,3))
for(i in 15:20) {plot(houseDFa[,i], houseDFa$price, xlab=names(houseDFa[
i]), ylab=names(houseDFa$price))}
```



Model B

For our second model, we chose a decision tree, to determine which home characteristics were most important, followed by a random forest to provide a more accurate regression that what a lone decision tree would provide. First, we need the decision treet library

In [22]:
```
library(rpart)
```

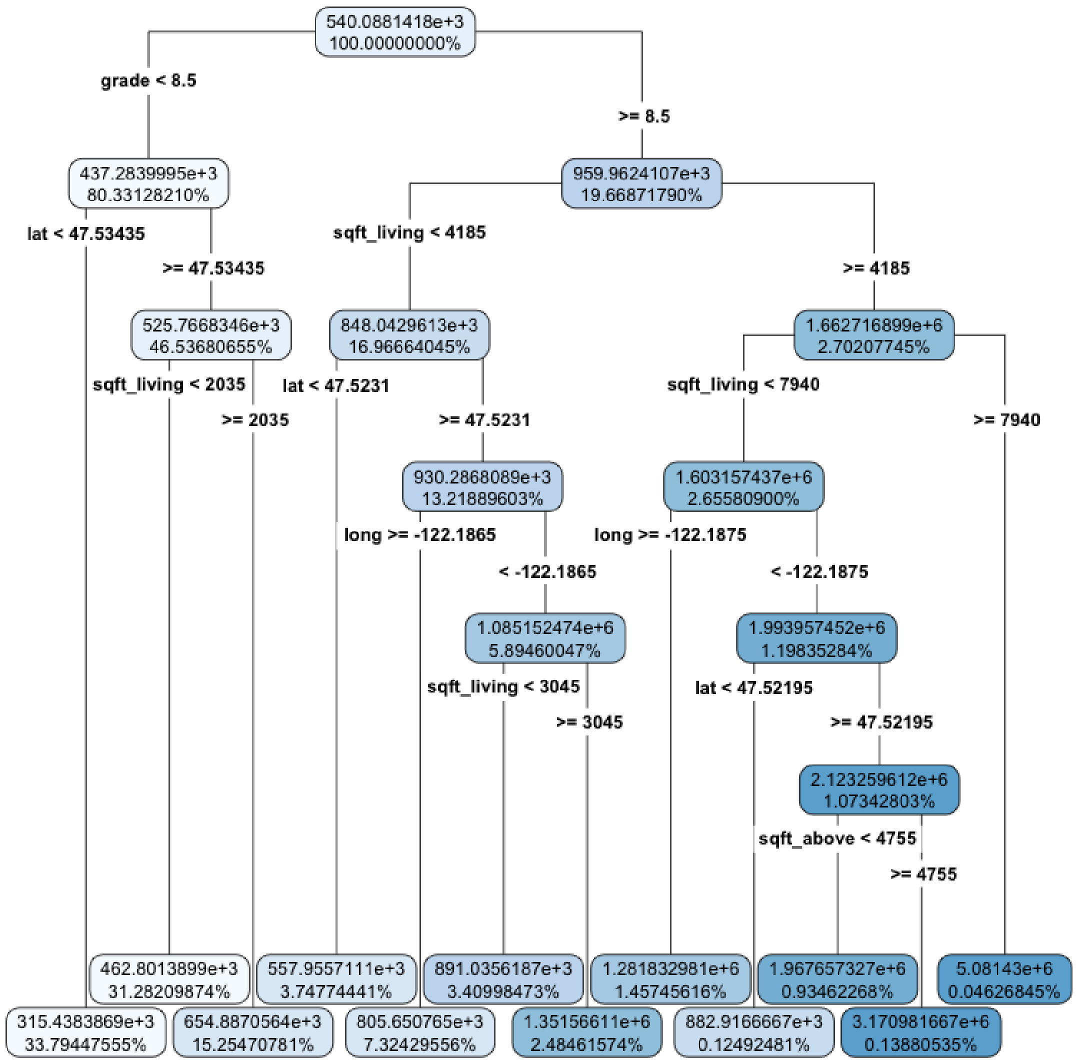
and then we create the decision tree

In [23]:
```
modelB1 <- rpart(houseDFa$price ~., houseDFa[,-c(1,2)], na.action = na.r
part)
modelB1
```

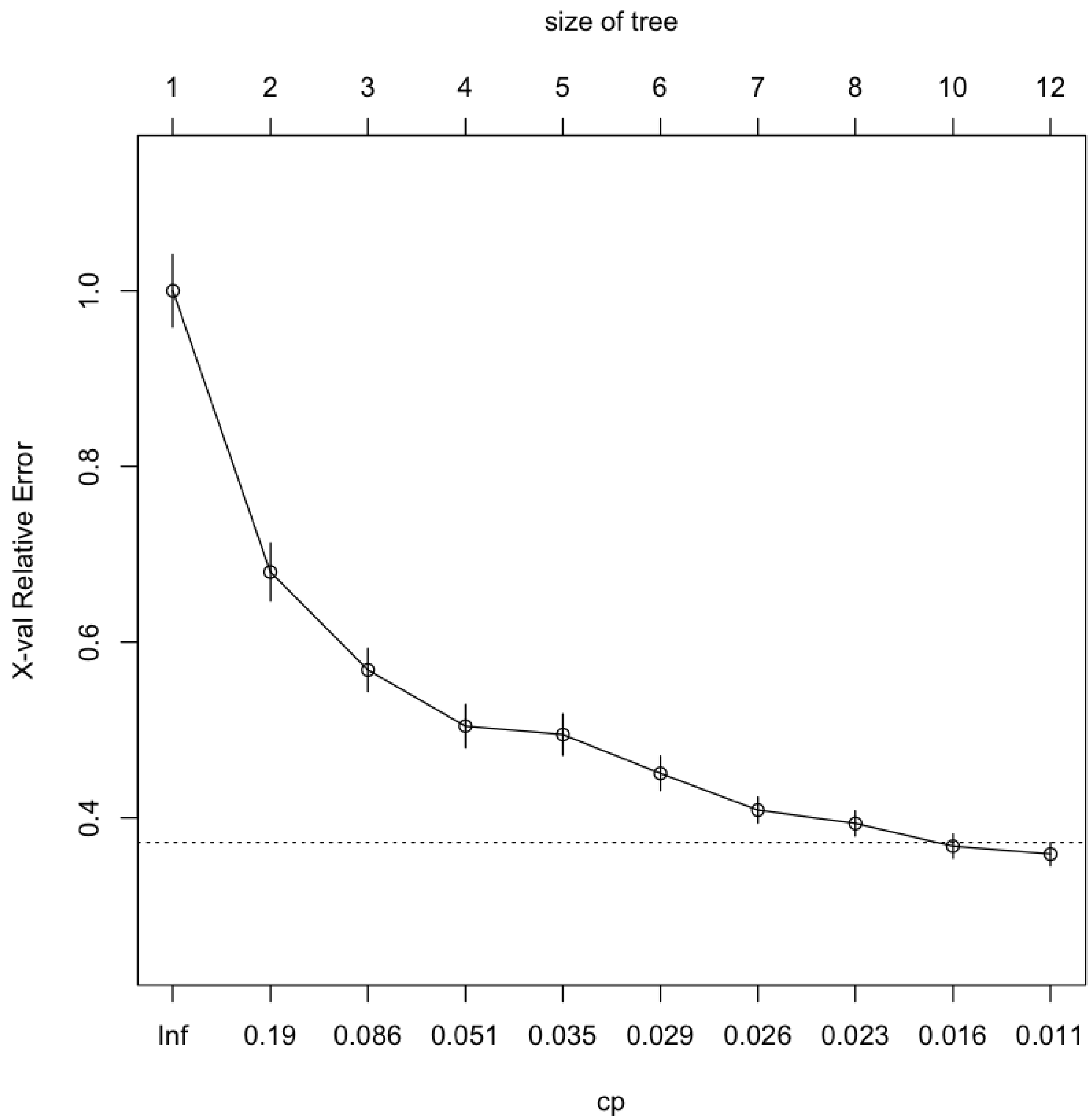
```
n= 21613

node), split, n, deviance, yval
      * denotes terminal node

  1) root 21613 2.912917e+15  540088.1
    2) grade< 8.5 17362 6.656597e+14  437284.0
      4) lat< 47.53435 7304 1.003407e+14  315438.4 *
      5) lat>=47.53435 10058 3.781350e+14  525766.8
       10) sqft_living< 2035 6761 1.361306e+14  462801.4 *
       11) sqft_living>=2035 3297 1.602317e+14  654887.1 *
    3) grade>=8.5 4251 1.314336e+15  959962.4
      6) sqft_living< 4185 3667 4.934437e+14  848043.0
       12) lat< 47.5231 810 3.289091e+13  557955.7 *
       13) lat>=47.5231 2857 3.730659e+14  930286.8
         26) long>=-122.1865 1583 6.476840e+13  805650.8 *
         27) long< -122.1865 1274 2.531521e+14 1085152.0
           54) sqft_living< 3045 737 7.221140e+13  891035.6 *
           55) sqft_living>=3045 537 1.150553e+14 1351566.0 *
      7) sqft_living>=4185 584 4.865430e+14 1662717.0
       14) sqft_living< 7940 574 3.372812e+14 1603157.0
         28) long>=-122.1875 315 7.499599e+13 1281833.0 *
         29) long< -122.1875 259 1.902060e+14 1993957.0
           58) lat< 47.52195 27 6.625440e+12  882916.7 *
           59) lat>=47.52195 232 1.463726e+14 2123260.0
            118) sqft_above< 4755 202 8.983921e+13 1967657.0 *
            119) sqft_above>=4755 30 1.871090e+13 3170982.0 *
       15) sqft_living>=7940 10 3.034967e+13 5081430.0 *
```

and plot it

In [24]:
```
library(rpart.plot)
rpart.plot(modelB1,digits=10,fallen.leaves=TRUE,type=4)
```



In [25]:
```
plotcp(modelB1)
```



We now load the random forest library

In [26]:
```
library(randomForest)
```

```
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.
```

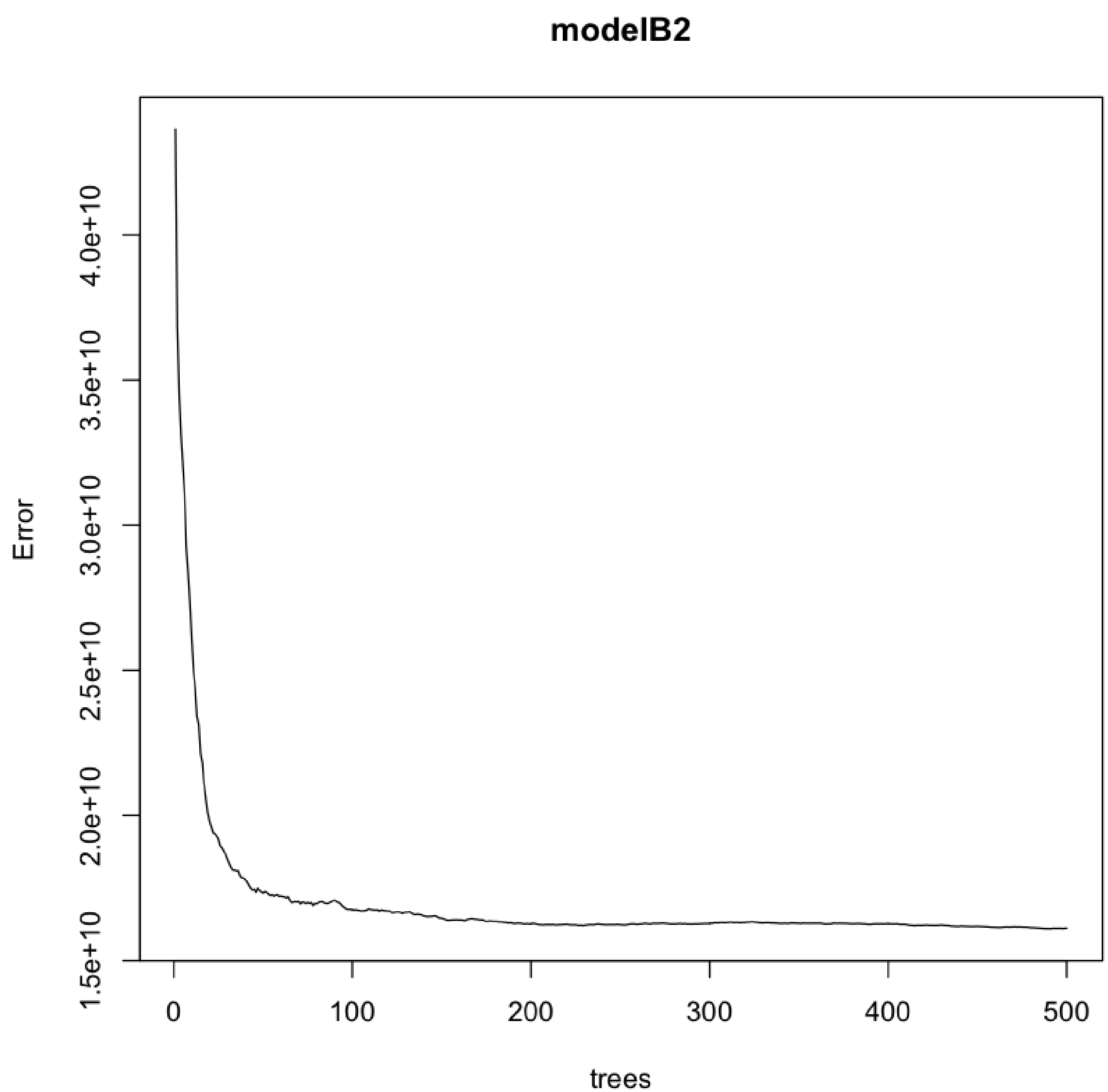
and then create the random forest model for our data

In [27]:
```
modelB2 <- randomForest(houseDFa$price ~., houseDFa[,-c(1,2)])
modelB2
```

```
Call:
 randomForest(formula = houseDFa$price ~ ., data = houseDFa[,      -c
(1, 2)])
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 6

          Mean of squared residuals: 16114715551
                    % Var explained: 88.04
```

which we then plot

In [28]:
```
plot(modelB2)
```



In [29]:
```
#partialPlot(modelB2, houseDFa[,-c(1,2)], houseDFa$bathrooms)
```

We also want to use our random forest to see which variables are more important when using an ensemble approach. To do this, we create a table of the imporantance of each variable in the random forest

In [31]:
```
importance(modelB2)
```

	IncNodePurity
bedrooms	1.557039e+13
bathrooms	1.176603e+14
sqft_living	6.459734e+14
sqft_lot	4.835966e+13
floors	1.028707e+13
waterfront	7.023169e+13
view	7.662750e+13
condition	1.323093e+13
grade	5.707538e+14
sqft_above	2.201358e+14
sqft_basement	5.312903e+13
yr_built	9.719926e+13
yr_renovated	9.267914e+12
zipcode	7.040288e+13
lat	4.021602e+14
long	1.697445e+14
sqft_living15	2.490493e+14
sqft_lot15	5.004354e+13

Model C

Our final model was a logistic regression. We chose this model because we were hoping to find any non-linear relationship that might be in play between various properties of a house and its price. In constructing this model, we assumed that, overall, the parameters of the houses were guassian.

In [38]:
```
modelC <- glm(houseDFa$price ~., houseDFa[,-c(1,2)], family = gaussian(l
ink = "identity"))
modelC
summary(modelC)
```

```
Call:  glm(formula = houseDFa$price ~ ., family = gaussian(link = "iden
tity"),
    data = houseDFa[, -c(1, 2)])

Coefficients:
  (Intercept)       bedrooms      bathrooms    sqft_living       sqft_l
ot
    6.690e+06     -3.577e+04      4.114e+04      1.501e+02      1.286e-
01
       floors     waterfront           view      condition          gra
de
    6.690e+03      5.830e+05      5.287e+04      2.639e+04      9.589e+
04
   sqft_above  sqft_basement       yr_built   yr_renovated        zipco
de
    3.113e+01             NA     -2.620e+03      1.981e+01     -5.824e+
02
          lat           long  sqft_living15     sqft_lot15
    6.027e+05     -2.147e+05      2.168e+01     -3.826e-01

Degrees of Freedom: 21612 Total (i.e. Null);  21595 Residual
Null Deviance:      2.913e+15
Residual Deviance: 8.746e+14    AIC: 589200
```

```
Call:
glm(formula = houseDFa$price ~ ., family = gaussian(link = "identity"),

    data = houseDFa[, -c(1, 2)])

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1291725    -99229     -9739     77583   4333222

Coefficients: (1 not defined because of singularities)
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    6.690e+06  2.931e+06   2.282  0.02249 *
bedrooms      -3.577e+04  1.892e+03 -18.906  < 2e-16 ***
bathrooms      4.114e+04  3.254e+03  12.645  < 2e-16 ***
sqft_living    1.501e+02  4.385e+00  34.227  < 2e-16 ***
sqft_lot       1.286e-01  4.792e-02   2.683  0.00729 **
floors         6.690e+03  3.596e+03   1.860  0.06285 .
waterfront     5.830e+05  1.736e+04  33.580  < 2e-16 ***
view           5.287e+04  2.140e+03  24.705  < 2e-16 ***
condition      2.639e+04  2.351e+03  11.221  < 2e-16 ***
grade          9.589e+04  2.153e+03  44.542  < 2e-16 ***
sqft_above     3.113e+01  4.360e+00   7.139 9.71e-13 ***
sqft_basement         NA         NA      NA       NA
yr_built      -2.620e+03  7.266e+01 -36.062  < 2e-16 ***
yr_renovated   1.981e+01  3.656e+00   5.420 6.03e-08 ***
zipcode       -5.824e+02  3.299e+01 -17.657  < 2e-16 ***
lat            6.027e+05  1.073e+04  56.149  < 2e-16 ***
long          -2.147e+05  1.313e+04 -16.349  < 2e-16 ***
sqft_living15  2.168e+01  3.448e+00   6.289 3.26e-10 ***
sqft_lot15    -3.826e-01  7.327e-02  -5.222 1.78e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 40500645795)

    Null deviance: 2.9129e+15  on 21612  degrees of freedom
Residual deviance: 8.7461e+14  on 21595  degrees of freedom
AIC: 589244

Number of Fisher Scoring iterations: 2
```

Summary

We discovered that the most important properties of a house, with respect to influence on its price, were

- The number of bedrooms
- The number of bathrooms
- The condition of the house
- Wether the house had a view or was on waterfront property
- The amount of living space in the house, as measured in sq ft

We also found that the random forest was the most accurate method, with each method having accuracies of

- Linear regression: ~71%
- Random forest: ~88%
- Non-Linear Logistic Regression: *no accuaracy* but residuals were higher than the **Random Forest** model.

Finally, becuase there are clearly some non-linear dynamics at play in the data, we believe it would be useful to have information on what ***type*** each house is. Is it a condominium? A "*regular*" house? A townhome? We would also like data on the quality of things near the house like schools, parks, transit, and roads. These may be factors which could be either playing an unseen role in determining the price of a house ***or*** influencing other variables in such a manner as to make them appear non-linear. With this information, we might be able to linearize those variables.