Week 1: Data Mining Process & Multiple Linear Regression

Tom Cook

```
packages_to_be_loaded=c("corrplot","dummies","forecast","gains","reshape","leaps")
for(i in packages_to_be_loaded){
    if(i%in%installed.packages()[,1]==F){install.packages(i)}
    require(i,character.only=T)
}
```

```
## Loading required package: corrplot
```

```
## corrplot 0.84 loaded
```

```
## Loading required package: dummies
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
## Loading required package: forecast
```

```
## Loading required package: gains
```

```
## Loading required package: reshape
```

```
## Loading required package: leaps
```

Getting Started

You will need to install the following packages for this week: *corrplot*, *reshape*, *dummies*, *leaps*, *forecast*, and *gains*.

Step 1: Develop an understanding of the data mining project

Instead of using assessed values provided by the city, can we create a model to better predict values of single family owner-occupied homes in West Roxbury, a neighborhood in southwest Boston, MA in 2014 (Shmueli et al 2018, Chapter 6)?

Step 2: Obtain the dataset to be used in the analysis

City of Boston provided total assessed value and 13 other variables for 5,000+ single family owner-occupied homes in West Roxbury.

Step 3: Explore, clean, and preprocess the data

```
#change the file path as needed
housing.df <- read.csv("WestRoxbury.csv", header=TRUE)
```

```
dim(housing.df)  # find the dimension of data frame
```

```
## [1] 5802   14
```

```r
head(housing.df)  # show the first six rows
```

```
##   TOTAL.VALUE  TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 1       344.2 4330     9965     1880       2436        1352      2     6
## 2       412.6 5190     6590     1945       3108        1976      2    10
## 3          NA 4152     7500     1890       2294        1371      2     8
## 4       498.6 6272    13773     1957       5032        2608      1     9
## 5       331.5 4170     5000     1910       2370        1438      2     7
## 6       337.4 4244     5142     1950       2124        1060      1     6
##   BEDROOMS FULL.BATH HALF.BATH KITCHEN FIREPLACE REMODEL
## 1        3         1         1       1         0    None
## 2       NA         2         1       1         0  Recent
## 3        4         1         1       1         0    None
## 4        5         1         1       1         1    None
## 5        3         2         0       1         0    None
## 6        3         1         0       1         1     Old
```

```r
names(housing.df) # get names from housing data frame
```

```
##  [1] "TOTAL.VALUE" "TAX"         "LOT.SQFT"    "YR.BUILT"    "GROSS.AREA" 
##  [6] "LIVING.AREA" "FLOORS"      "ROOMS"       "BEDROOMS"    "FULL.BATH"  
## [11] "HALF.BATH"   "KITCHEN"     "FIREPLACE"   "REMODEL"
```

```r
# Practice showing different subsets of the data
housing.df[1:10, 1]  # show the first 10 rows of the first column only
```

```
##  [1] 344.2 412.6    NA 498.6 331.5 337.4 359.4 320.4 333.5 409.4
```

```r
housing.df[1:10, ]  # show the first 10 rows of each of the columns
```

```
##    TOTAL.VALUE  TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 1        344.2 4330     9965     1880       2436        1352      2     6
## 2        412.6 5190     6590     1945       3108        1976      2    10
## 3           NA 4152     7500     1890       2294        1371      2     8
## 4        498.6 6272    13773     1957       5032        2608      1     9
## 5        331.5 4170     5000     1910       2370        1438      2     7
## 6        337.4 4244     5142     1950       2124        1060      1     6
## 7        359.4 4521     5000     1954       3220        1916      2     7
## 8        320.4 4030    10000     1950       2208        1200      1     6
## 9        333.5 4195     6835     1958       2582        1092      1     5
## 10       409.4 5150     5093     1900       4818        2992      2     8
##    BEDROOMS FULL.BATH HALF.BATH KITCHEN FIREPLACE REMODEL
## 1         3         1         1       1         0    None
## 2        NA         2         1       1         0  Recent
## 3         4         1         1       1         0    None
## 4         5         1         1       1         1    None
## 5         3         2         0       1         0    None
## 6         3         1         0       1         1     Old
## 7         3         1         1       1         0    None
## 8         3         1         0       1         0    None
## 9         3         1         0       1         1  Recent
## 10        4         2         0       1         0    None
```

```r
#housing.df$TOTAL.VALUE  # a different way to show the whole first column
housing.df$TOTAL.VALUE[1:10]  # show the first 10 rows of the first column
```

```
##  [1] 344.2 412.6    NA 498.6 331.5 337.4 359.4 320.4 333.5 409.4
```

```r
housing.df[5, 1:10]  # show the fifth row of the first 10 columns
```

```
##   TOTAL.VALUE  TAX LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 5       331.5 4170     5000     1910       2370        1438      2     7
##   BEDROOMS FULL.BATH
## 5        3         2
```

```r
housing.df[5, c(1:2, 4, 8:10)]  # show the fifth row of some columns
```

```
##   TOTAL.VALUE  TAX YR.BUILT ROOMS BEDROOMS FULL.BATH
## 5       331.5 4170     1910     7        3         2
```

```r
str(housing.df) #structure of data frame
```

```
## 'data.frame':    5802 obs. of  14 variables:
##  $ TOTAL.VALUE: num  344 413 NA 499 332 ...
##  $ TAX        : int  4330 5190 4152 6272 4170 4244 4521 4030 4195 5150 ...
##  $ LOT.SQFT   : int  9965 6590 7500 13773 5000 5142 5000 10000 6835 5093 ...
##  $ YR.BUILT   : int  1880 1945 1890 1957 1910 1950 1954 1950 1958 1900 ...
##  $ GROSS.AREA : int  2436 3108 2294 5032 2370 2124 3220 2208 2582 4818 ...
##  $ LIVING.AREA: int  1352 1976 1371 2608 1438 1060 1916 1200 1092 2992 ...
##  $ FLOORS     : num  2 2 2 1 2 1 2 1 1 2 ...
##  $ ROOMS      : int  6 10 8 9 7 6 7 6 5 8 ...
##  $ BEDROOMS   : int  3 NA 4 5 3 3 3 3 3 4 ...
##  $ FULL.BATH  : int  1 2 1 1 2 1 1 1 1 2 ...
##  $ HALF.BATH  : int  1 1 1 1 0 0 1 0 0 0 ...
##  $ KITCHEN    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ FIREPLACE  : int  0 0 0 1 0 1 0 0 1 0 ...
##  $ REMODEL    : Factor w/ 3 levels "None","Old","Recent": 1 3 1 1 1 2 1 1 3 1 ...
```

```r
summary(housing.df) #five number summary for numeric variables; count summary for factors
```

```
##   TOTAL.VALUE          TAX           LOT.SQFT        YR.BUILT   
##  Min.   : 105.0   Min.   : 1320   Min.   :  997   Min.   :   0  
##  1st Qu.: 325.1   1st Qu.: 4090   1st Qu.: 4772   1st Qu.:1920  
##  Median : 375.9   Median : 4728   Median : 5683   Median :1935  
##  Mean   : 392.7   Mean   : 4939   Mean   : 6278   Mean   :1937  
##  3rd Qu.: 438.8   3rd Qu.: 5520   3rd Qu.: 7022   3rd Qu.:1955  
##  Max.   :1217.8   Max.   :15319   Max.   :46411   Max.   :2011  
##  NA's   :1                                                      
##    GROSS.AREA    LIVING.AREA       FLOORS          ROOMS       
##  Min.   : 821   Min.   : 504   Min.   :1.000   Min.   : 3.000  
##  1st Qu.:2347   1st Qu.:1308   1st Qu.:1.000   1st Qu.: 6.000  
##  Median :2700   Median :1548   Median :2.000   Median : 7.000  
##  Mean   :2925   Mean   :1657   Mean   :1.684   Mean   : 6.995  
##  3rd Qu.:3239   3rd Qu.:1874   3rd Qu.:2.000   3rd Qu.: 8.000  
##  Max.   :8154   Max.   :5289   Max.   :3.000   Max.   :14.000  
##                                                                
##     BEDROOMS       FULL.BATH       HALF.BATH         KITCHEN     
##  Min.   :1.00   Min.   :1.000   Min.   :0.0000   Min.   :1.000  
##  1st Qu.:3.00   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:1.000  
##  Median :3.00   Median :1.000   Median :1.0000   Median :1.000  
##  Mean   :3.23   Mean   :1.297   Mean   :0.6139   Mean   :1.015  
##  3rd Qu.:4.00   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:1.000  
##  Max.   :9.00   Max.   :5.000   Max.   :3.0000   Max.   :2.000  
##  NA's   :1                                                      
##    FIREPLACE        REMODEL    
##  Min.   :0.0000   None  :4346  
##  1st Qu.:0.0000   Old   : 581  
##  Median :1.0000   Recent: 875  
##  Mean   :0.7399                
##  3rd Qu.:1.0000                
##  Max.   :4.0000                
## 
```

```
class(housing.df$REMODEL)
```

```
## [1] "factor"
```

```
levels(housing.df$REMODEL)
```

```
## [1] "None"   "Old"    "Recent"
```

```
summary(housing.df$BEDROOMS) #one missing value in row #2
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##    1.00    3.00    3.00    3.23    4.00    9.00       1 
```

```
# housing.df$BEDROOMS[is.na(housing.df$BEDROOMS)]=median(housing.df$BEDROOMS,na.rm=TRUE)
```

```
summary(housing.df$TOTAL.VALUE) #one missing value in row #3
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   105.0   325.1   375.9   392.7   438.8  1218.0       1 
```

```
# housing.df$TOTAL.VALUE[is.na(housing.df$TOTAL.VALUE)]=mean(housing.df$TOTAL.VALUE,na.rm = TRUE)
```

What patterns do you see in the scatter plots below?

```
png(filename = "scatterplots1.png")
plot(housing.df[,c(1,2,3,5)])
dev.off()
```

```
## pdf 
##   2
```

```
png(filename = "scatterplots2.png")
plot(housing.df[,c(1,6,7,8)])
dev.off()
```

```
## pdf 
##   2
```

```
png(filename = "scatterplots3.png")
plot(housing.df[,c(1,9,10,11,12,13,14)])
dev.off()
```

```
## pdf 
##   2
```

```
# plot(housing.df[,c(1,2,3,5)])
# plot(housing.df[,c(1,6,7,8)])
# plot(housing.df[,c(1,9,10,11,12,13,14)])
```

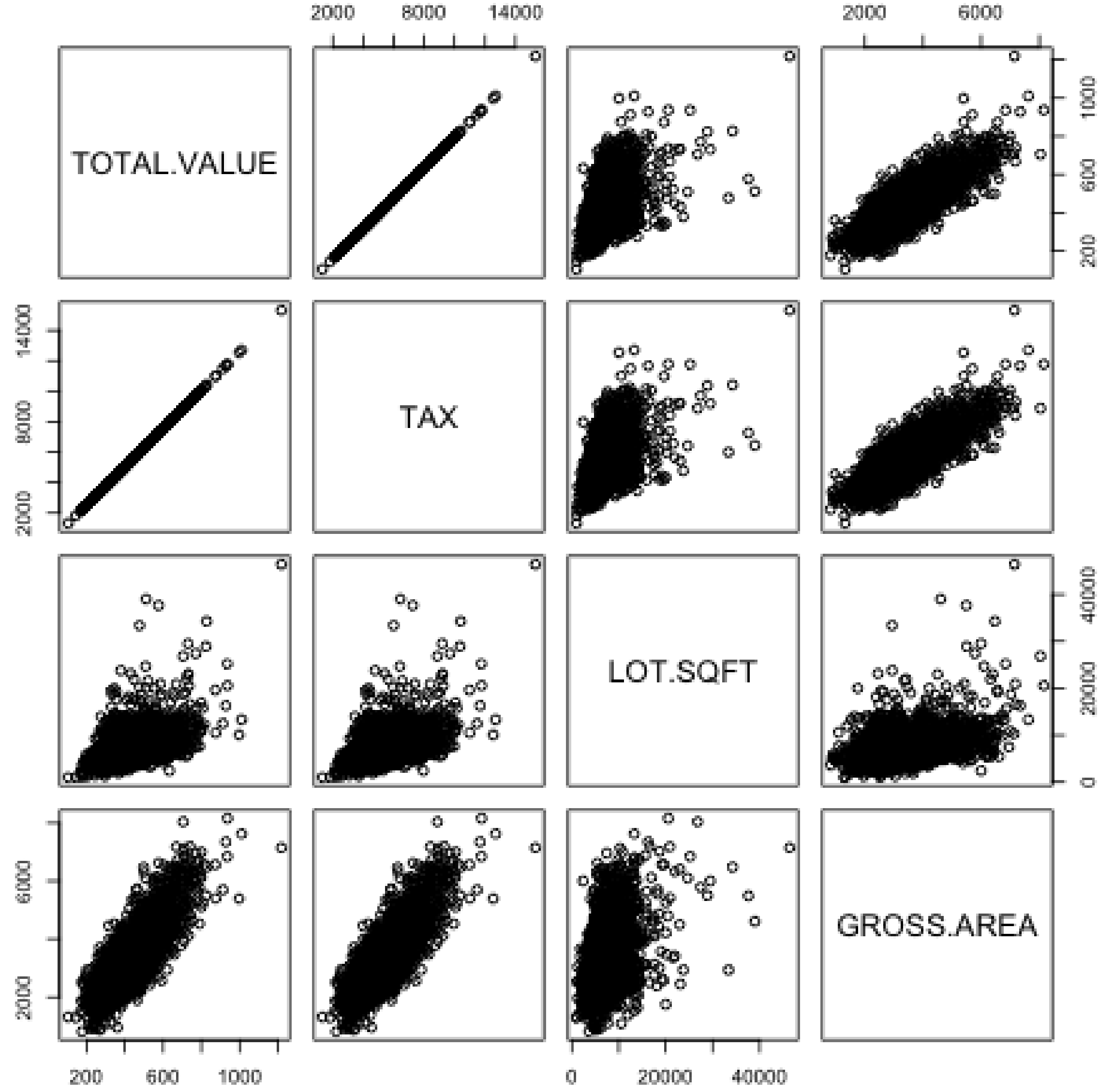


Figure 1: scatterplots1.png

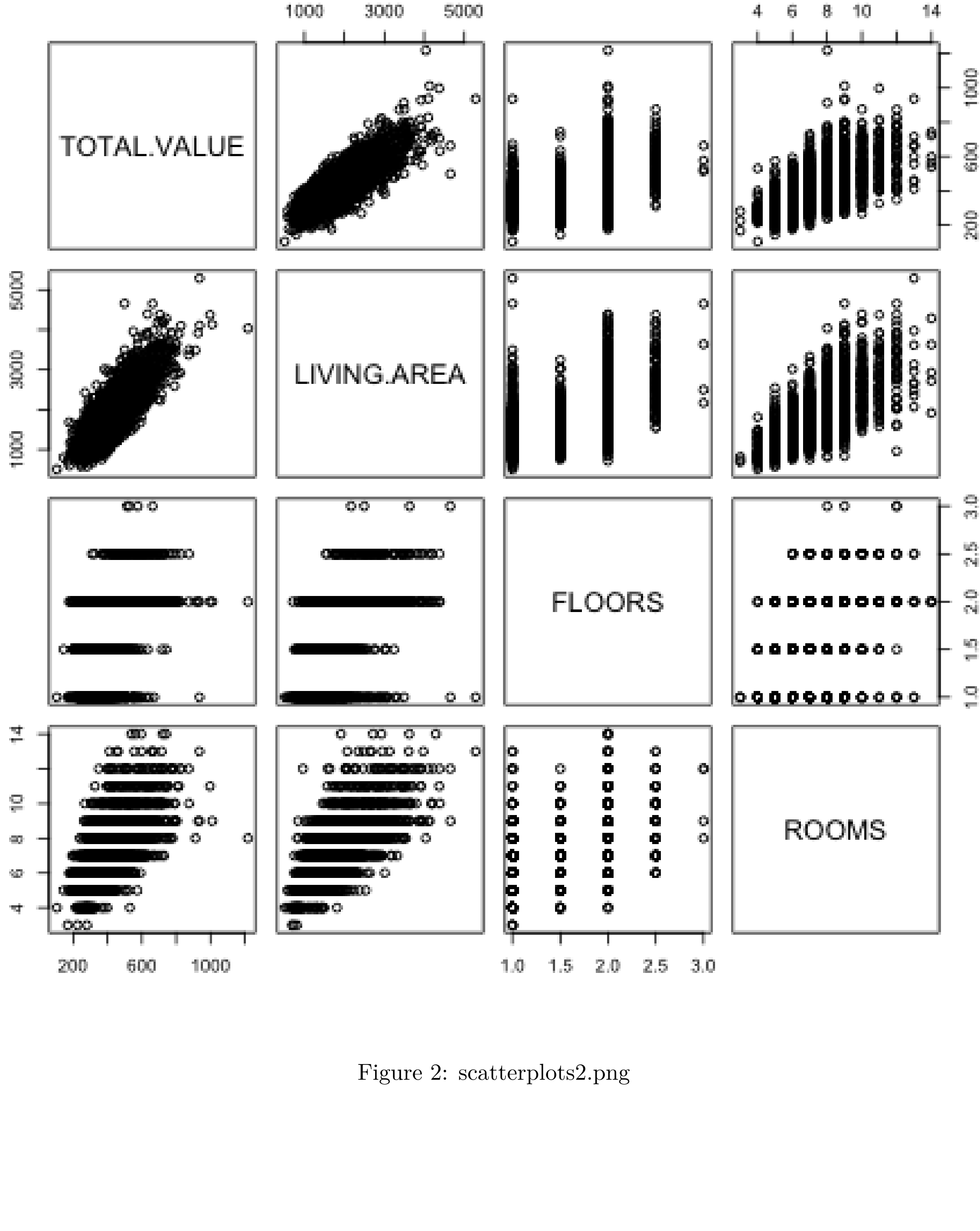


Figure 2: scatterplots2.png

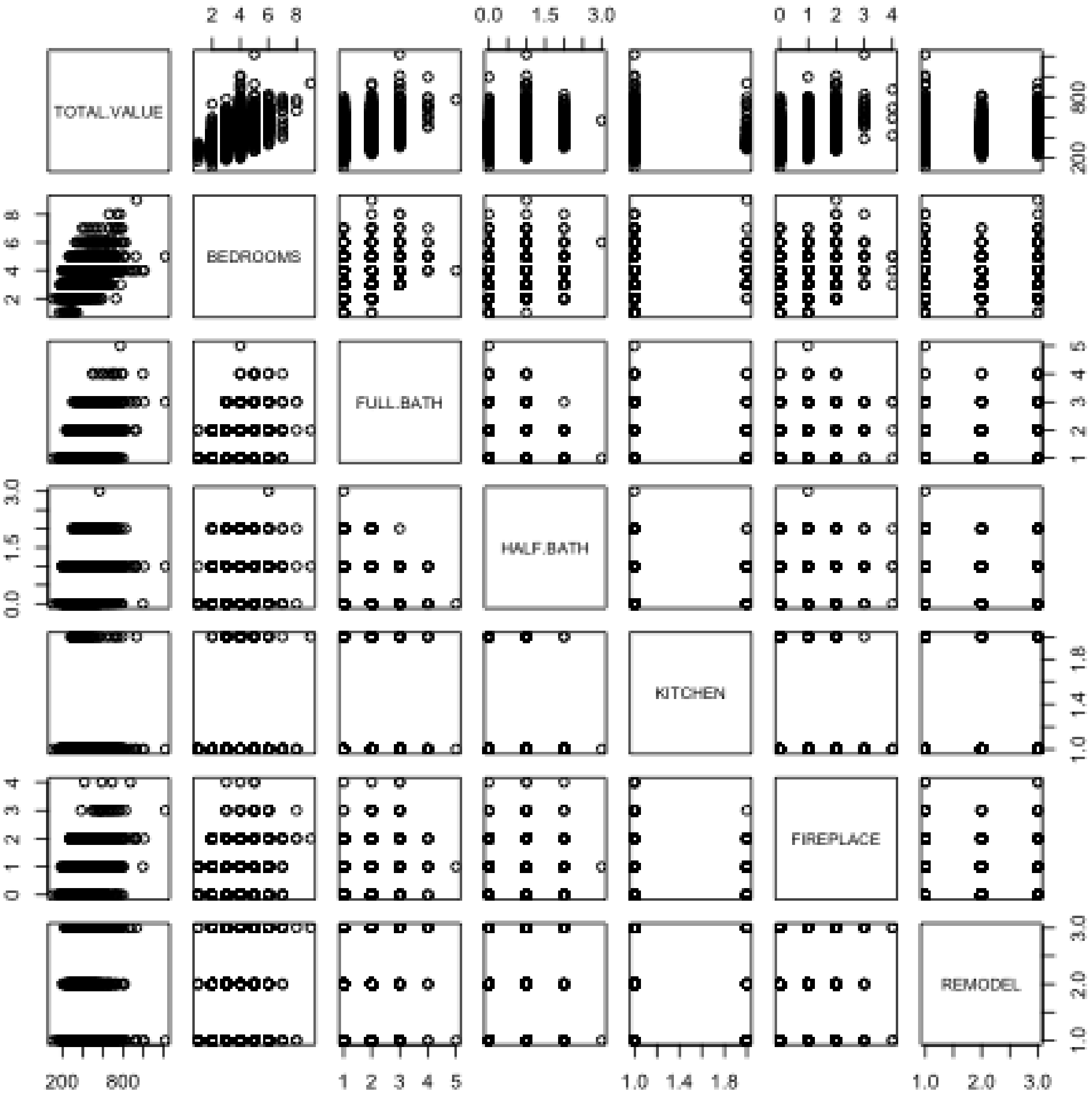
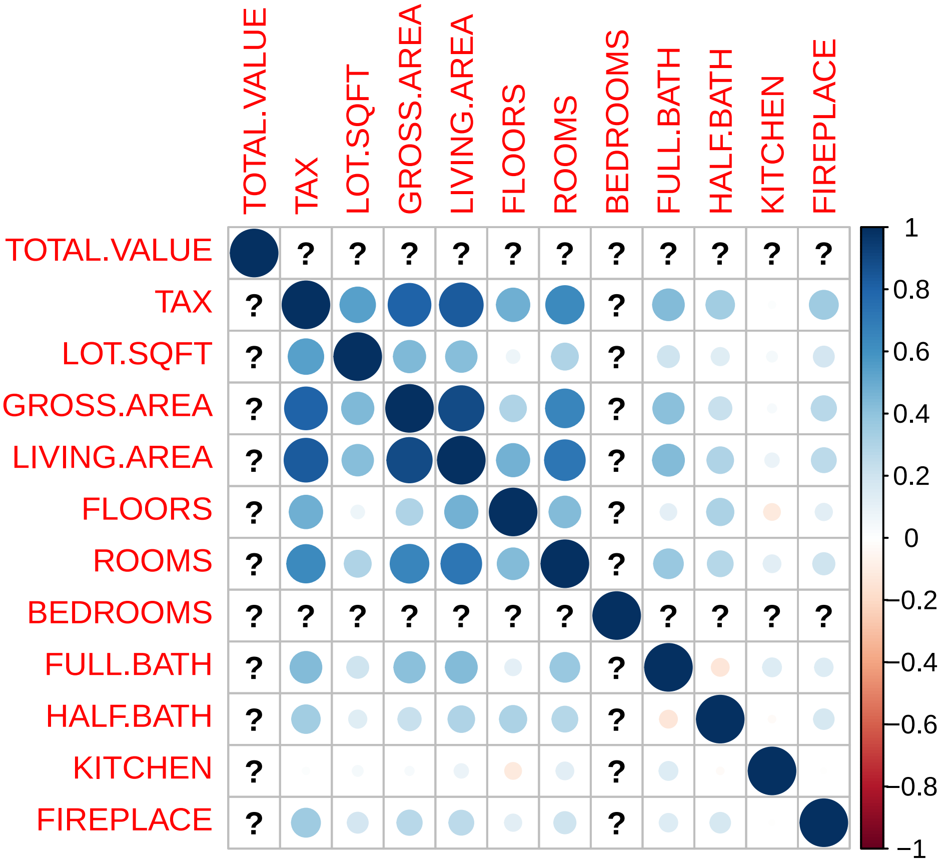


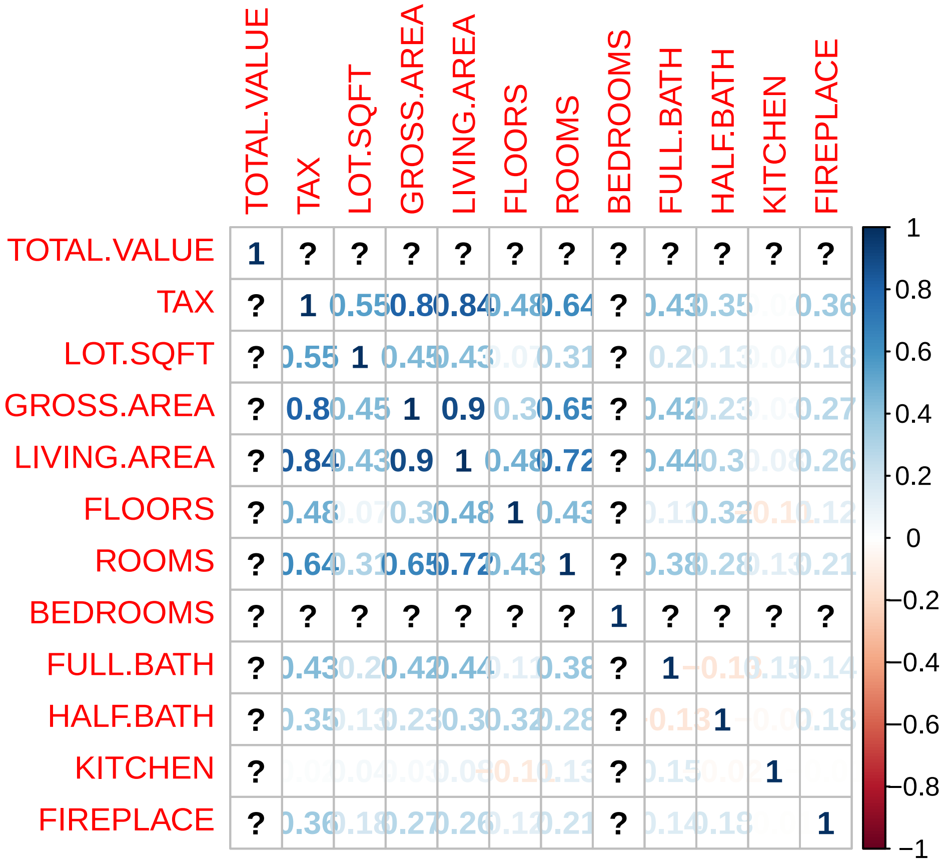
Figure 3: scatterplots3.png

Correlation plot

```
# using functions from library corrplot
corrs <- cor(housing.df[,c(1:3,5:13)]) #did not include YR.BUILT and REMODEL
corrs.matrix <- as.matrix(corrs)
corrplot(corrs.matrix)
```

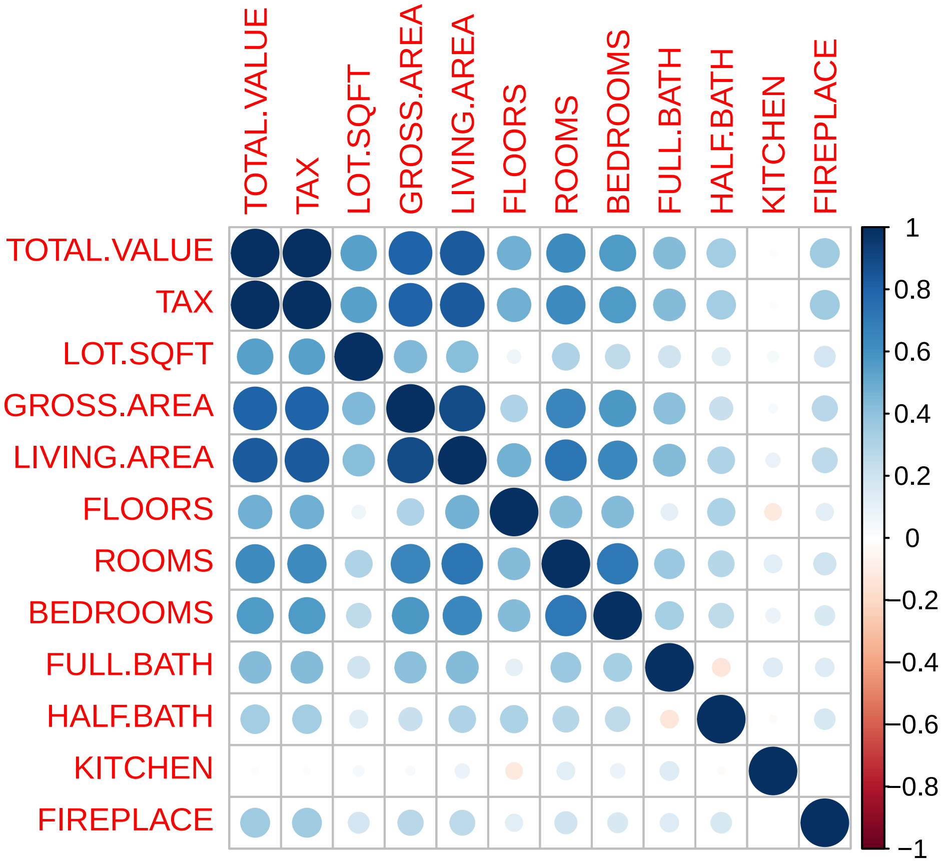


```
corrplot(corrs.matrix, method="number")
```

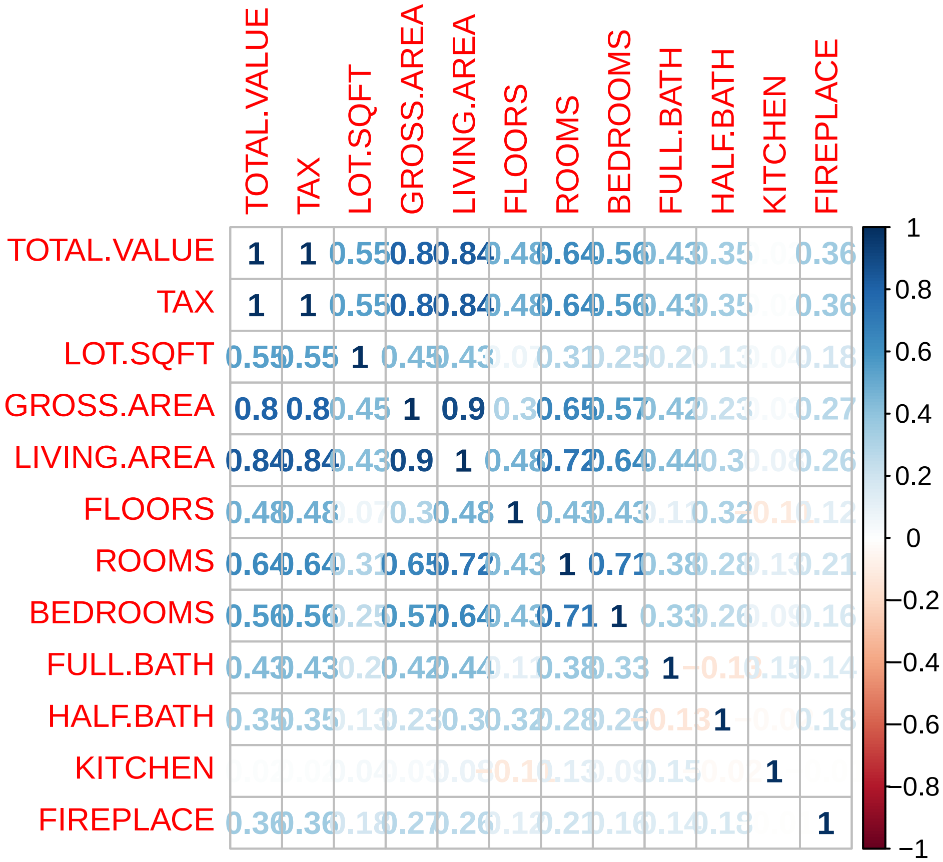


Notice that the correlation plots show "?" for the variables with NA's. We have to exclude the NAs if we want to calculate the correlation coefficients for all variables and not deal with the "?" output.

```
housing.df.no.mv <- housing.df[c(-2,-3), ]#remove rows 2 & 3
corrs2 <- cor(housing.df.no.mv[,c(1:3,5:13)]) #did not include YR.BUILT and REMODEL
corrs2.matrix <- as.matrix(corrs2)
corrplot(corrs2.matrix)
```



```
corrplot(corrs2.matrix, method="number")
```



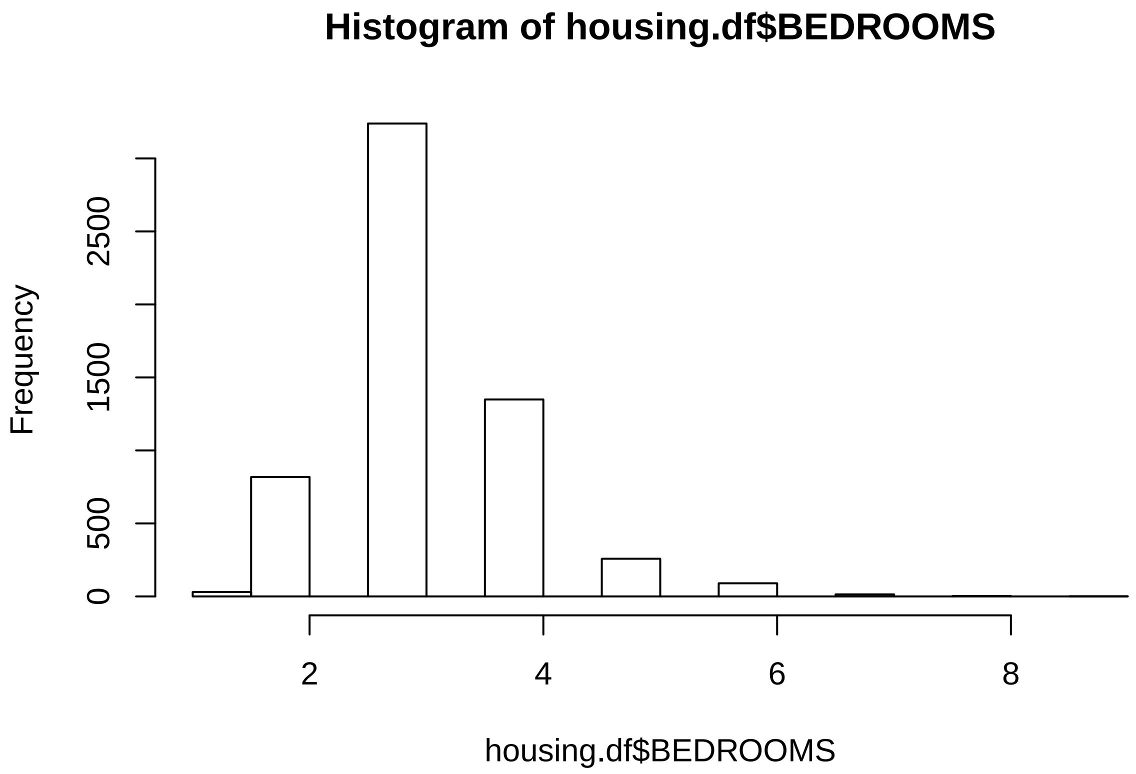
	TOTAL.VALUE	TAX	LOT.SQFT	GROSS.AREA	LIVING.AREA	FLOORS	ROOMS	BEDROOMS	FULL.BATH	HALF.BATH	KITCHEN	FIREPLACE
TOTAL.VALUE	1	1	0.55	0.8	0.84	0.48	0.64	0.56	0.43	0.35	0.02	0.36
TAX	1	1	0.55	0.8	0.84	0.48	0.64	0.56	0.43	0.35	0.02	0.36
LOT.SQFT	0.55	0.55	1	0.45	0.43	0.07	0.31	0.25	0.2	0.13	0.04	0.18
GROSS.AREA	0.8	0.8	0.45	1	0.9	0.3	0.65	0.57	0.42	0.23	0.03	0.27
LIVING.AREA	0.84	0.84	0.43	0.9	1	0.48	0.72	0.64	0.44	0.3	0.08	0.26
FLOORS	0.48	0.48	0.07	0.3	0.48	1	0.43	0.43	0.11	0.32	−0.1	0.12
ROOMS	0.64	0.64	0.31	0.65	0.72	0.43	1	0.71	0.38	0.28	0.13	0.21
BEDROOMS	0.56	0.56	0.25	0.57	0.64	0.43	0.71	1	0.33	0.26	0.09	0.16
FULL.BATH	0.43	0.43	0.2	0.42	0.44	0.11	0.38	0.33	1	−0.13	0.15	0.14
HALF.BATH	0.35	0.35	0.13	0.23	0.3	0.32	0.28	0.26	−0.13	1	−0.02	0.18
KITCHEN	0.02	0.02	0.04	0.03	0.08	−0.1	0.13	0.09	0.15	−0.02	1	[illegible]
FIREPLACE	0.36	0.36	0.18	0.27	0.26	0.12	0.21	0.16	0.14	0.18	[illegible]	1

Missing Values

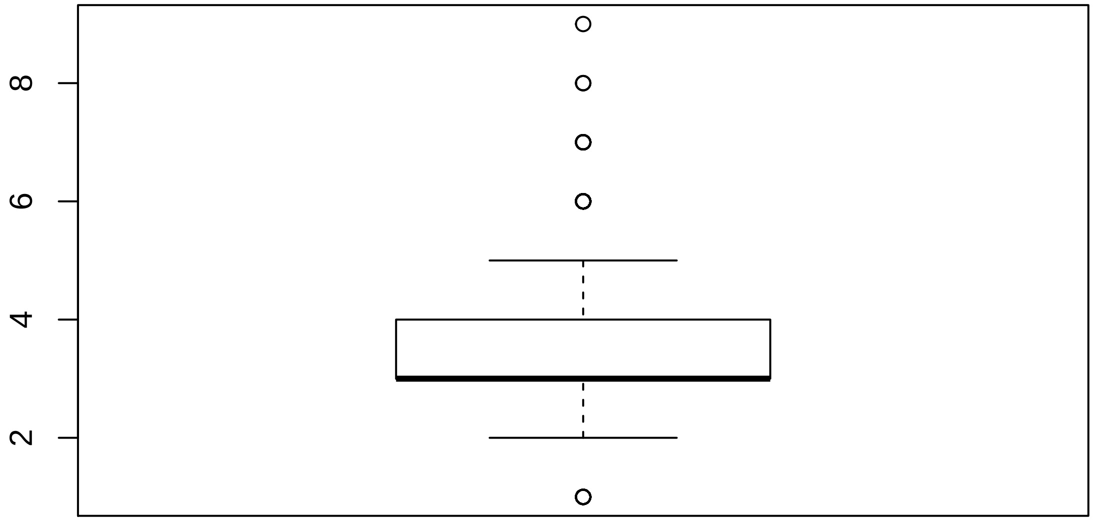
Two approaches: 1) delete the missing values but will lose these observations or 2) impute missing values with mean or median value. If the variable is normally distributed, use mean. If the distribution is skewed, use the median.

```
#housing.df <- housing.df[-2,] #remove row
housing.df$BEDROOMS[is.na(housing.df$BEDROOMS)]=median(housing.df$BEDROOMS,na.rm=TRUE)
housing.df$TOTAL.VALUE[is.na(housing.df$TOTAL.VALUE)]=mean(housing.df$TOTAL.VALUE,na.rm = TRUE)
```

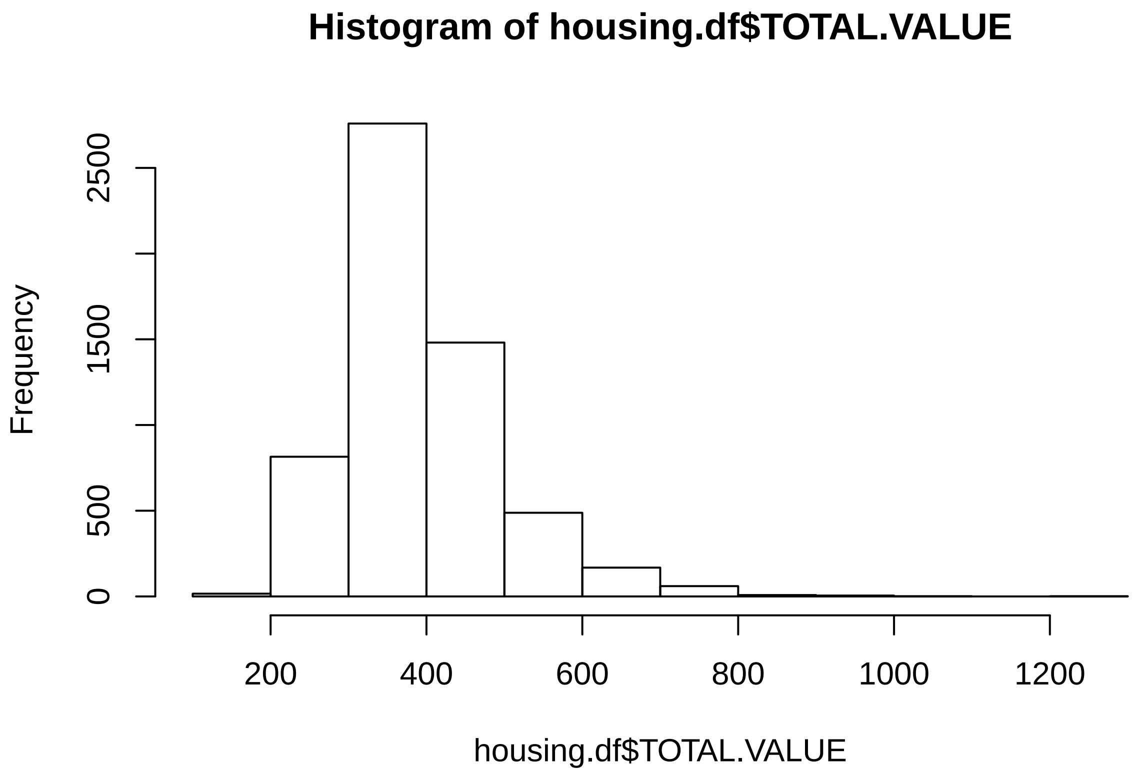
```
hist(housing.df$BEDROOMS)
```



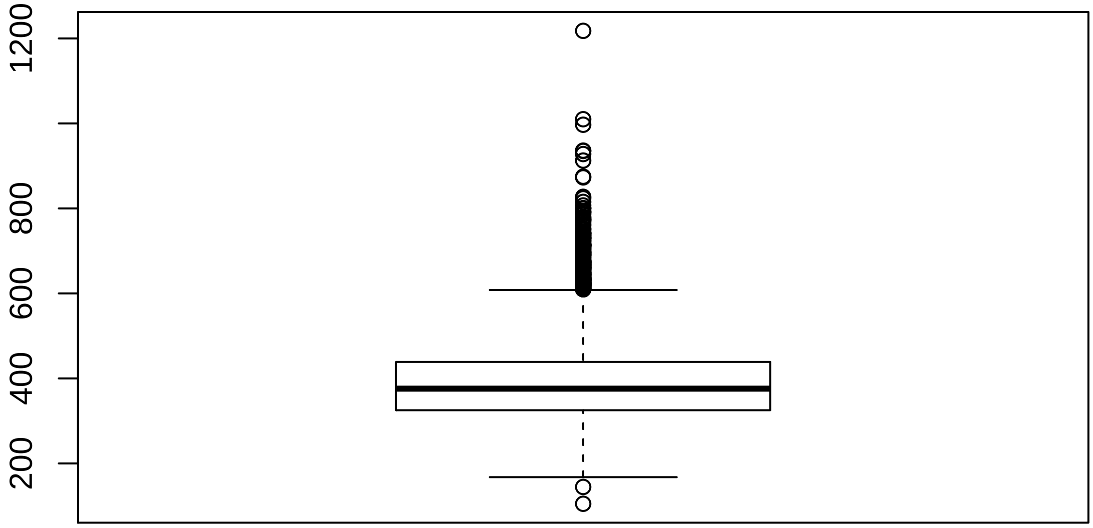
```
boxplot(housing.df$BEDROOMS)
```



```
hist(housing.df$TOTAL.VALUE)
```



```
boxplot(housing.df$TOTAL.VALUE)
```



```
#Using the aggregate function
aggregate(housing.df$BEDROOMS, by=list(housing.df$FULL.BATH), FUN=mean, na.rm=TRUE)
```

```
##   Group.1        x
## 1       1 3.075312
## 2       2 3.578270
## 3       3 4.264286
## 4       4 5.076923
## 5       5 4.000000
```

```
#Using melt and cast to create a Pivot Table
# using functions from library reshape
```

```r
summary(housing.df$ROOMS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   6.000   7.000   6.995   8.000  14.000
```

```r
housing.df$ROOMS.bin <- .bincode(housing.df$ROOMS, c(2:14)) #create bins
mlt <- melt(housing.df, id = c("ROOMS.bin", "REMODEL"), measure=c("TOTAL.VALUE","FLOORS"))
cast(mlt, ROOMS.bin~REMODEL, subset= variable=="TOTAL.VALUE",
     median, na.rm=TRUE)
```

```
##    ROOMS.bin   None    Old Recent
## 1          1 223.85     NA 225.80
## 2          2 277.40 264.10 358.95
## 3          3 295.30 295.80 343.95
## 4          4 341.20 339.20 372.05
## 5          5 378.40 386.20 414.10
## 6          6 407.25 426.00 459.95
## 7          7 464.95 481.85 501.60
## 8          8 511.80 511.25 588.35
## 9          9 540.00 530.15 666.40
## 10        10 599.70 626.30 666.30
## 11        11 462.20 692.45 660.00
## 12        12 639.30 597.30 642.20
```

Step 4: Reduce the data dimension

We have 14 variables in total with only one categorical variable: REMODEL. The REMODEL variable has three categories: NONE, OLD, and RECENT. This is not too bad! If we were dealing with a large number of variables (think +40 variables) or a data set with many categorical variables (and many categories), we will need to determine whether some variables can be grouped together or remove to reduce the dimension of the data set. We will return to this topic in Week #5.

Step 5: Determine the data mining task

We want to predict TOTAL.VALUE using the given predictor variables. Since we have a target variable, this is a supervised learning task. We will not use the TAX variable. TAX is determined by using the home value, which is what we are trying to predict. We will also remove ROOMS.bin, which is a variable we created earlier to make a Pivot Table.

```r
housing.df.model <- housing.df[,c(-2,-15)]
```

Step 6: Partition the data (for supervised tasks)

We are using an 80-20 split, which is 80% of the data will be used to train our prediction model. The other 20% of the data will be used to "score" our prediction model. Note that we do not have a test set since we are not comparing multiple supervised models at the moment.

```r
set.seed(123) #ensure we always get the same output
housing.index <- housing.df.model[order(runif(5802)), ]#randomized the observations
train <- housing.index[1:4641, ] #create training set
valid <- housing.index[4642:5802, ] #create validation set

dim(train)
```

```
## [1] 4641   13
```

```r
dim(valid)
```

```
## [1] 1161   13
```

Step 7: Choose the data mining techniques to be used

We will be using multiple linear regression to build a prediction model.

Step 8: Use algorithms to perform the task

```r
options(scipen = 999) #ensure no scientific notation is displayed
housing.lm <- lm(TOTAL.VALUE~., data=train)
summary(housing.lm)
```

```
## 
## Call:
## lm(formula = TOTAL.VALUE ~ ., data = train)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -262.465  -25.998    0.172   25.499  286.379 
## 
## Coefficients:
##                   Estimate  Std. Error t value             Pr(>|t|)    
## (Intercept)   -256.444736   58.129072  -4.412          0.000010492 ***
## LOT.SQFT         0.008312    0.000276  30.118 < 0.0000000000000002 ***
## YR.BUILT         0.154863    0.029188   5.306          0.000000117 ***
## GROSS.AREA       0.032850    0.001840  17.851 < 0.0000000000000002 ***
## LIVING.AREA      0.050366    0.003373  14.934 < 0.0000000000000002 ***
## FLOORS          41.874505    1.922567  21.781 < 0.0000000000000002 ***
## ROOMS            1.857450    0.736357   2.522               0.0117 *  
## BEDROOMS        -1.536559    1.113320  -1.380               0.1676    
## FULL.BATH       17.813581    1.533957  11.613 < 0.0000000000000002 ***
## HALF.BATH       17.873728    1.400222  12.765 < 0.0000000000000002 ***
## KITCHEN        -13.378231    5.639191  -2.372               0.0177 *  
## FIREPLACE       19.390206    1.200613  16.150 < 0.0000000000000002 ***
## REMODELOld       4.001778    2.148203   1.863               0.0625 .  
## REMODELRecent   25.074921    1.867098  13.430 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 43.11 on 4627 degrees of freedom
## Multiple R-squared:  0.8143, Adjusted R-squared:  0.8137 
## F-statistic:  1560 on 13 and 4627 DF,  p-value: < 0.00000000000000022
```

Questions to answer:

1. What is the goodness-of-fit?

2. Which predictors are important?

3. Is there a simpler prediction model (i.e. less predictors) that perform “just as well”?

Why should we reduce the number of predictors when computing cost is cheap?

1. It may be expensive or unfeasible to collect all predictors for future prediction exercises.

2. We may be able to measure fewer predictors more accurately.

3. More predictors = more chances of missing values = more imputations or record deletions.

4. Occam’s Razor

5. Estimates of regression coefficients are unstable due to multicollinearity. Multicollinearity is the presence of two or more predictors sharing the same linear relationship with the outcome variable.

6. Bias-variance tradeoff: Predictors uncorrelated with the target variable increases the *variance* of predictions. At the same time, dropping variables that are correlated with the target variable can increase the average error (*bias*) of predictions.

Exhaustive Search for Reducing Predictors

Before we can proceed, we have to create dummy variables for the REMODEL variable. (Remember that R’s lm function does it automatically for us earlier.) In addition, we also have to remove one of the three newly created dummy variables: REMODELNone, REMODELOld, and REMODELRecent in order to run the exhaustive search algorithm. The reason is because linear regression models compare desired classes against a *reference* or *base* class. If we want to know the effect of recent remodelling on the total value of a home, we have to compare it against a reference/base class. The base class could be old remodeled home or never remodeled home. It does not matter which class, but we have to pick one. For this exercise, we will compare the *old* and *recent* remodeled homes against the base class of *never remodeled*.

```
# using functions from library dummies
train.dummies<-dummy("REMODEL", train)
valid.dummies<-dummy("REMODEL", valid)

train <- cbind(train, train.dummies) #prepped data + dummies
valid <- cbind(valid, valid.dummies) #prepped data + dummies

train.search <-train[,c(-13,-14)] #remove REMODEL & REMODELNone
valid.search <-valid[,c(-13,-14)] #remove REMODEL & REMODELNone

names(train.search) #checking column names
```

```
##  [1] "TOTAL.VALUE"   "LOT.SQFT"      "YR.BUILT"      "GROSS.AREA"
##  [5] "LIVING.AREA"   "FLOORS"        "ROOMS"         "BEDROOMS"
##  [9] "FULL.BATH"     "HALF.BATH"     "KITCHEN"       "FIREPLACE"
## [13] "REMODELOld"    "REMODELRecent"
```

```
names(valid.search) #checking column names
```

```
##  [1] "TOTAL.VALUE"   "LOT.SQFT"      "YR.BUILT"      "GROSS.AREA"
##  [5] "LIVING.AREA"   "FLOORS"        "ROOMS"         "BEDROOMS"
##  [9] "FULL.BATH"     "HALF.BATH"     "KITCHEN"       "FIREPLACE"
## [13] "REMODELOld"    "REMODELRecent"
```

The exhaustive search algorithm runs linear regression models on all possible subsets of predictors. We have to pick among all the subset models to find a desired model. Adjusted R-square is a popular method used to find the desired model.

The drawback of exhaustive search is that it is time consuming to search through all the possible subsets.

```
# Functions from library leaps
search <- regsubsets(TOTAL.VALUE~.,data=train.search, nbest=1, nvmax=dim(train.search)[2], method="exhau
sum <- summary(search)
sum$which
```

```
##    (Intercept) LOT.SQFT YR.BUILT GROSS.AREA LIVING.AREA FLOORS ROOMS
## 1         TRUE    FALSE    FALSE      FALSE        TRUE  FALSE FALSE
## 2         TRUE     TRUE    FALSE      FALSE        TRUE  FALSE FALSE
## 3         TRUE     TRUE    FALSE      FALSE        TRUE  FALSE FALSE
## 4         TRUE     TRUE    FALSE      FALSE        TRUE   TRUE FALSE
## 5         TRUE     TRUE    FALSE       TRUE        TRUE   TRUE FALSE
## 6         TRUE     TRUE    FALSE       TRUE        TRUE   TRUE FALSE
## 7         TRUE     TRUE    FALSE       TRUE        TRUE   TRUE FALSE
## 8         TRUE     TRUE    FALSE       TRUE        TRUE   TRUE FALSE
## 9         TRUE     TRUE     TRUE       TRUE        TRUE   TRUE FALSE
## 10        TRUE     TRUE     TRUE       TRUE        TRUE   TRUE FALSE
## 11        TRUE     TRUE     TRUE       TRUE        TRUE   TRUE  TRUE
## 12        TRUE     TRUE     TRUE       TRUE        TRUE   TRUE  TRUE
## 13        TRUE     TRUE     TRUE       TRUE        TRUE   TRUE  TRUE
##    BEDROOMS FULL.BATH HALF.BATH KITCHEN FIREPLACE REMODELOld REMODELRecent
## 1     FALSE     FALSE     FALSE   FALSE     FALSE      FALSE         FALSE
## 2     FALSE     FALSE     FALSE   FALSE     FALSE      FALSE         FALSE
## 3     FALSE     FALSE     FALSE   FALSE      TRUE      FALSE         FALSE
## 4     FALSE     FALSE     FALSE   FALSE      TRUE      FALSE         FALSE
## 5     FALSE     FALSE     FALSE   FALSE      TRUE      FALSE         FALSE
## 6     FALSE     FALSE     FALSE   FALSE      TRUE      FALSE          TRUE
## 7     FALSE      TRUE      TRUE   FALSE      TRUE      FALSE         FALSE
## 8     FALSE      TRUE      TRUE   FALSE      TRUE      FALSE          TRUE
## 9     FALSE      TRUE      TRUE   FALSE      TRUE      FALSE          TRUE
## 10    FALSE      TRUE      TRUE    TRUE      TRUE      FALSE          TRUE
## 11    FALSE      TRUE      TRUE    TRUE      TRUE      FALSE          TRUE
## 12    FALSE      TRUE      TRUE    TRUE      TRUE       TRUE          TRUE
## 13     TRUE      TRUE      TRUE    TRUE      TRUE       TRUE          TRUE
```

```
sum$adjr2
```

```
##  [1] 0.7043680 0.7435176 0.7612452 0.7779056 0.7903347 0.7992009 0.8057831
##  [8] 0.8123564 0.8133216 0.8134521 0.8136000 0.8136983 0.8137347
```

Model #8 seems to be the model where the adjusted Rsquare does not improve much further. Here are the important predictors in Model #8: LOT.SQFT,GROSS.AREA, LIVING.AREA, FLOORS, FULL.BATH, HALF.BATH, FIREPLACE, and REMODELRecent. It is important to note that we are only including one dummy variable: REMODELRecent. Since there is no REMODELOld in this model, we are–in effect–comparing recently remodeled homes to a reference group of *never* and *old* remodeled homes. There is a difference in interpretation between Model #8 with the subsequent models shown for backward elimination, forward selection, and stepwise regression models below.

```
housing.lm.exhaust <- lm(TOTAL.VALUE~LOT.SQFT + GROSS.AREA + LIVING.AREA + FLOORS + FULL.BATH + HALF.BAT
summary(housing.lm.exhaust)
```

```
##
## Call:
## lm(formula = TOTAL.VALUE ~ LOT.SQFT + GROSS.AREA + LIVING.AREA +
##     FLOORS + FULL.BATH + HALF.BATH + FIREPLACE + REMODELRecent,
##     data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -260.757  -26.487   -0.095   25.752  296.220
##
## Coefficients:
##                 Estimate Std. Error t value            Pr(>|t|)
## (Intercept)   38.2029870  3.2970199   11.59 <0.0000000000000002 ***
## LOT.SQFT       0.0082320  0.0002763   29.79 <0.0000000000000002 ***
## GROSS.AREA     0.0312547  0.0017585   17.77 <0.0000000000000002 ***
## LIVING.AREA    0.0526537  0.0031795   16.56 <0.0000000000000002 ***
## FLOORS        39.8621284  1.7620663   22.62 <0.0000000000000002 ***
## FULL.BATH     20.1551955  1.4424798   13.97 <0.0000000000000002 ***
## HALF.BATH     20.0536171  1.3378261   14.99 <0.0000000000000002 ***
## FIREPLACE     20.5149029  1.1856781   17.30 <0.0000000000000002 ***
## REMODELRecent 23.4432010  1.8345329   12.78 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.27 on 4632 degrees of freedom
## Multiple R-squared:  0.8127, Adjusted R-squared:  0.8124
## F-statistic:  2512 on 8 and 4632 DF,  p-value: < 0.00000000000000022
```

Backward Elimination Algorithm

The algorithm starts with a set of given predictors and then eliminate the least useful predictors at each step. "Least useful" is defined as variables not statistically significant.

Backward elimination is time consuming and unstable when you start with a large number of predictors.

Is there any difference than our earlier regression model?

```r
bwdreg <- step(housing.lm, direction="backward")
```

```
## Start:  AIC=34949.97
## TOTAL.VALUE ~ LOT.SQFT + YR.BUILT + GROSS.AREA + LIVING.AREA +
##     FLOORS + ROOMS + BEDROOMS + FULL.BATH + HALF.BATH + KITCHEN +
##     FIREPLACE + REMODEL
##
##               Df Sum of Sq     RSS   AIC
## - BEDROOMS     1      3541 8604209 34950
## <none>                     8600668 34950
## - KITCHEN      1     10462 8611130 34954
## - ROOMS        1     11827 8612495 34954
## - YR.BUILT     1     52327 8652995 34976
## - FULL.BATH    1    250674 8851342 35081
## - HALF.BATH    1    302879 8903547 35109
## - REMODEL      2    335464 8936132 35124
## - LIVING.AREA  1    414570 9015238 35166
## - FIREPLACE    1    484832 9085500 35202
## - GROSS.AREA   1    592312  9192980 35257
## - FLOORS       1    881798  9482466 35401
## - LOT.SQFT     1   1686130 10286798 35779
##
## Step:  AIC=34949.88
## TOTAL.VALUE ~ LOT.SQFT + YR.BUILT + GROSS.AREA + LIVING.AREA +
##     FLOORS + ROOMS + FULL.BATH + HALF.BATH + KITCHEN + FIREPLACE +
##     REMODEL
##
##               Df Sum of Sq      RSS   AIC
## <none>                      8604209 34950
## - ROOMS        1      8511  8612719 34952
## - KITCHEN      1     10728  8614937 34954
## - YR.BUILT     1     53424  8657633 34977
## - FULL.BATH    1    247852  8852061 35080
## - HALF.BATH    1    300842  8905050 35107
## - REMODEL      2    335500  8939709 35123
## - LIVING.AREA  1    411041  9015250 35164
## - FIREPLACE    1    485968  9090177 35203
## - GROSS.AREA   1    590170  9194379 35256
## - FLOORS       1    881655  9485864 35401
## - LOT.SQFT     1   1688554 10292763 35780
```

```
summary(bwdreg)
```

```
##
## Call:
## lm(formula = TOTAL.VALUE ~ LOT.SQFT + YR.BUILT + GROSS.AREA +
##     LIVING.AREA + FLOORS + ROOMS + FULL.BATH + HALF.BATH + KITCHEN +
##     FIREPLACE + REMODEL, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -261.888  -26.049    0.142   25.440  288.491
##
## Coefficients:
##                  Estimate  Std. Error t value             Pr(>|t|)
## (Intercept)   -259.525517   58.091875  -4.468          0.000008103 ***
## LOT.SQFT         0.008317    0.000276  30.137 < 0.0000000000000002 ***
## YR.BUILT         0.156369    0.029170   5.361          0.000000087 ***
## GROSS.AREA       0.032777    0.001840  17.817 < 0.0000000000000002 ***
## LIVING.AREA      0.049961    0.003360  14.869 < 0.0000000000000002 ***
## FLOORS          41.542938    1.907685  21.777 < 0.0000000000000002 ***
## ROOMS            1.428150    0.667500   2.140               0.0324 *
## FULL.BATH       17.675099    1.530821  11.546 < 0.0000000000000002 ***
## HALF.BATH       17.800829    1.399362  12.721 < 0.0000000000000002 ***
## KITCHEN        -13.544593    5.638453  -2.402               0.0163 *
## FIREPLACE       19.411331    1.200633  16.168 < 0.0000000000000002 ***
## REMODELOld       3.984854    2.148378   1.855               0.0637 .
## REMODELRecent   25.075908    1.867281  13.429 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.12 on 4628 degrees of freedom
## Multiple R-squared:  0.8142, Adjusted R-squared:  0.8137
## F-statistic:  1690 on 12 and 4628 DF,  p-value: < 0.00000000000000022
```

Forward Selection Algorithm

The algorithm starts with no predictors and then add predictors one at a time. The predictor added at each step has the largest contribution to R-square on top of the predictors that are already in it. The algorithm stops when the contribution of additional predictors are not statistically significant.

The weakeness of this algorithm is that it will miss pairs of predictors that perform well together but poorly as single predictors.

```
fwdreg <- step(housing.lm, direction="forward")
```

```
## Start:  AIC=34949.97
## TOTAL.VALUE ~ LOT.SQFT + YR.BUILT + GROSS.AREA + LIVING.AREA +
##     FLOORS + ROOMS + BEDROOMS + FULL.BATH + HALF.BATH + KITCHEN +
##     FIREPLACE + REMODEL
```

```
summary(fwdreg)
```

```
##
## Call:
## lm(formula = TOTAL.VALUE ~ LOT.SQFT + YR.BUILT + GROSS.AREA +
##     LIVING.AREA + FLOORS + ROOMS + BEDROOMS + FULL.BATH + HALF.BATH +
##     KITCHEN + FIREPLACE + REMODEL, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -262.465  -25.998    0.172   25.499  286.379
##
## Coefficients:
##                  Estimate  Std. Error t value             Pr(>|t|)
## (Intercept)   -256.444736   58.129072  -4.412          0.000010492 ***
## LOT.SQFT         0.008312    0.000276  30.118 < 0.0000000000000002 ***
## YR.BUILT         0.154863    0.029188   5.306          0.000000117 ***
## GROSS.AREA       0.032850    0.001840  17.851 < 0.0000000000000002 ***
## LIVING.AREA      0.050366    0.003373  14.934 < 0.0000000000000002 ***
## FLOORS          41.874505    1.922567  21.781 < 0.0000000000000002 ***
## ROOMS            1.857450    0.736357   2.522               0.0117 *
## BEDROOMS        -1.536559    1.113320  -1.380               0.1676
## FULL.BATH       17.813581    1.533957  11.613 < 0.0000000000000002 ***
## HALF.BATH       17.873728    1.400222  12.765 < 0.0000000000000002 ***
## KITCHEN        -13.378231    5.639191  -2.372               0.0177 *
## FIREPLACE       19.390206    1.200613  16.150 < 0.0000000000000002 ***
## REMODELOld       4.001778    2.148203   1.863               0.0625 .
## REMODELRecent   25.074921    1.867098  13.430 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.11 on 4627 degrees of freedom
## Multiple R-squared:  0.8143, Adjusted R-squared:  0.8137
## F-statistic:  1560 on 13 and 4627 DF,  p-value: < 0.00000000000000022
```

Stepwise Selection Algorithm

The algorithm is a combination of forward selection and backward elimination algorithms. At each step, the algorithm adds additional predictors while removing ones that are not statistically significant.

Stepwise algorithm has the same weakenesses as forward selection and backward elimination algorithms.

```
stepreg <- step(housing.lm, direction="both")
```

```
## Start:  AIC=34949.97
## TOTAL.VALUE ~ LOT.SQFT + YR.BUILT + GROSS.AREA + LIVING.AREA +
##     FLOORS + ROOMS + BEDROOMS + FULL.BATH + HALF.BATH + KITCHEN +
##     FIREPLACE + REMODEL
##
##               Df Sum of Sq      RSS   AIC
## - BEDROOMS     1      3541  8604209 34950
## <none>                      8600668 34950
## - KITCHEN      1     10462  8611130 34954
## - ROOMS        1     11827  8612495 34954
## - YR.BUILT     1     52327  8652995 34976
## - FULL.BATH    1    250674  8851342 35081
## - HALF.BATH    1    302879  8903547 35109
## - REMODEL      2    335464  8936132 35124
## - LIVING.AREA  1    414570  9015238 35166
## - FIREPLACE    1    484832  9085500 35202
## - GROSS.AREA   1    592312  9192980 35257
## - FLOORS       1    881798  9482466 35401
## - LOT.SQFT     1   1686130 10286798 35779
##
## Step:  AIC=34949.88
## TOTAL.VALUE ~ LOT.SQFT + YR.BUILT + GROSS.AREA + LIVING.AREA +
##     FLOORS + ROOMS + FULL.BATH + HALF.BATH + KITCHEN + FIREPLACE +
##     REMODEL
##
##               Df Sum of Sq      RSS   AIC
## <none>                      8604209 34950
## + BEDROOMS     1      3541  8600668 34950
## - ROOMS        1      8511  8612719 34952
## - KITCHEN      1     10728  8614937 34954
## - YR.BUILT     1     53424  8657633 34977
## - FULL.BATH    1    247852  8852061 35080
## - HALF.BATH    1    300842  8905050 35107
## - REMODEL      2    335500  8939709 35123
## - LIVING.AREA  1    411041  9015250 35164
## - FIREPLACE    1    485968  9090177 35203
## - GROSS.AREA   1    590170  9194379 35256
## - FLOORS       1    881655  9485864 35401
## - LOT.SQFT     1   1688554 10292763 35780
```

```
summary(stepreg)
```

```
##
## Call:
## lm(formula = TOTAL.VALUE ~ LOT.SQFT + YR.BUILT + GROSS.AREA +
##     LIVING.AREA + FLOORS + ROOMS + FULL.BATH + HALF.BATH + KITCHEN +
##     FIREPLACE + REMODEL, data = train)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -261.888  -26.049    0.142   25.440  288.491
##
## Coefficients:
##                 Estimate  Std. Error t value            Pr(>|t|)
## (Intercept)   -259.525517   58.091875  -4.468         0.000008103 ***
## LOT.SQFT         0.008317    0.000276  30.137 < 0.0000000000000002 ***
## YR.BUILT         0.156369    0.029170   5.361         0.000000087 ***
## GROSS.AREA       0.032777    0.001840  17.817 < 0.0000000000000002 ***
## LIVING.AREA      0.049961    0.003360  14.869 < 0.0000000000000002 ***
## FLOORS          41.542938    1.907685  21.777 < 0.0000000000000002 ***
## ROOMS            1.428150    0.667500   2.140              0.0324 *
## FULL.BATH       17.675099    1.530821  11.546 < 0.0000000000000002 ***
## HALF.BATH       17.800829    1.399362  12.721 < 0.0000000000000002 ***
## KITCHEN        -13.544593    5.638453  -2.402              0.0163 *
## FIREPLACE       19.411331    1.200633  16.168 < 0.0000000000000002 ***
## REMODELOld       3.984854    2.148378   1.855              0.0637 .
## REMODELRecent   25.075908    1.867281  13.429 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.12 on 4628 degrees of freedom
## Multiple R-squared:  0.8142, Adjusted R-squared:  0.8137
## F-statistic:  1690 on 12 and 4628 DF,  p-value: < 0.00000000000000022
```

Step 9: Interpret the results

The backward, forward, and stepwise selection algorithms did not include BEDROOMS (not significant). exhaustive search model #8 also excludes ROOMS as a predictor. Here is a list of predictors that all selection algorithms show to be important: LOT.SQFT, GROSS.AREA, LIVING.AREA, FLOORS, FULL.BATH, HALF.BATH, KITCHEN, FIREPLACE, REMODELOld, and REMODELRecent. Here is a summary of what we found.

Algorithm	Excluded Predictor(s)	R-square
Exhaustive Search	ROOMS;YR_BUILT	81.2%
Backward Elimination	BEDROOMS	81.4%
Forward Selection	BEDROOMS	81.4%
Stepwise	BEDROOMS	81.4%

It is up to you to determine which model should be chosen to deploy. Is it worthwhile to include four additional predictors to improve the linear regression model's goodness-of-fit by 0.2%?

Step 10: Deploy the model

What do the error metrics look like for the stepwise regression model?

```
pred_v <- predict(stepreg, valid)
# using functions from forecast
accuracy(pred_v, valid$TOTAL.VALUE)
```

```
##                   ME     RMSE      MAE       MPE     MAPE
## Test set -0.3543465 42.59524 31.83794 -1.079767 8.333216
```

What about the exhaustive regression model?

```
pred_v <- predict(housing.lm.exhaust, valid)
# using functions from forecast
accuracy(pred_v, valid$TOTAL.VALUE)
```

```
##                   ME     RMSE      MAE       MPE     MAPE
## Test set -0.6704036 41.73594 31.72216 -1.145796 8.304264
```

Wait… Aren't You Cheating?

Is it "cheating" if I use the RMSE (or any other error metric) to *go back* and choose an alternative model in Step #9? Yes! It's cheating. If you do choose to use the error metrics from the validation set to revise your model selection, you will need to use a third partition (i.e. the test set) to get true estimates of the error metrics.

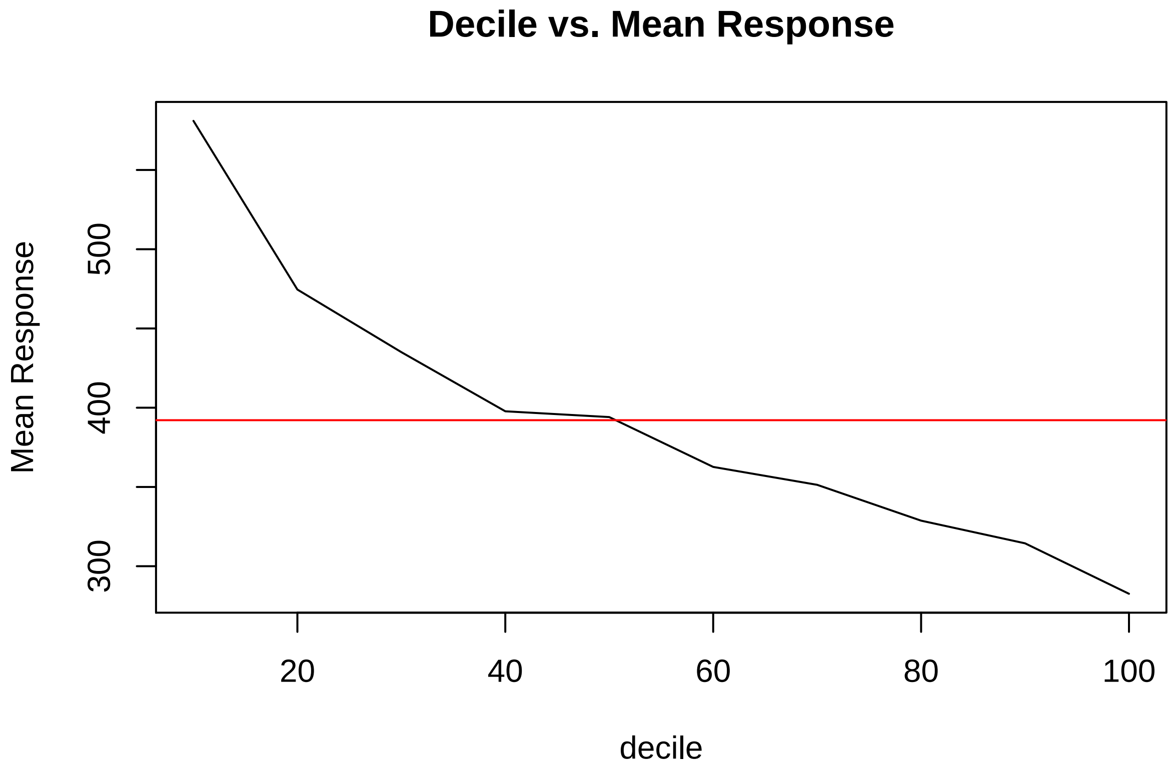
Important: As we discuss the logic of cross validation and the train/test/validate partitioning scheme, you should consider using `mlr` or a similar package that will select subsets based on out of sample performance.

Lift Chart

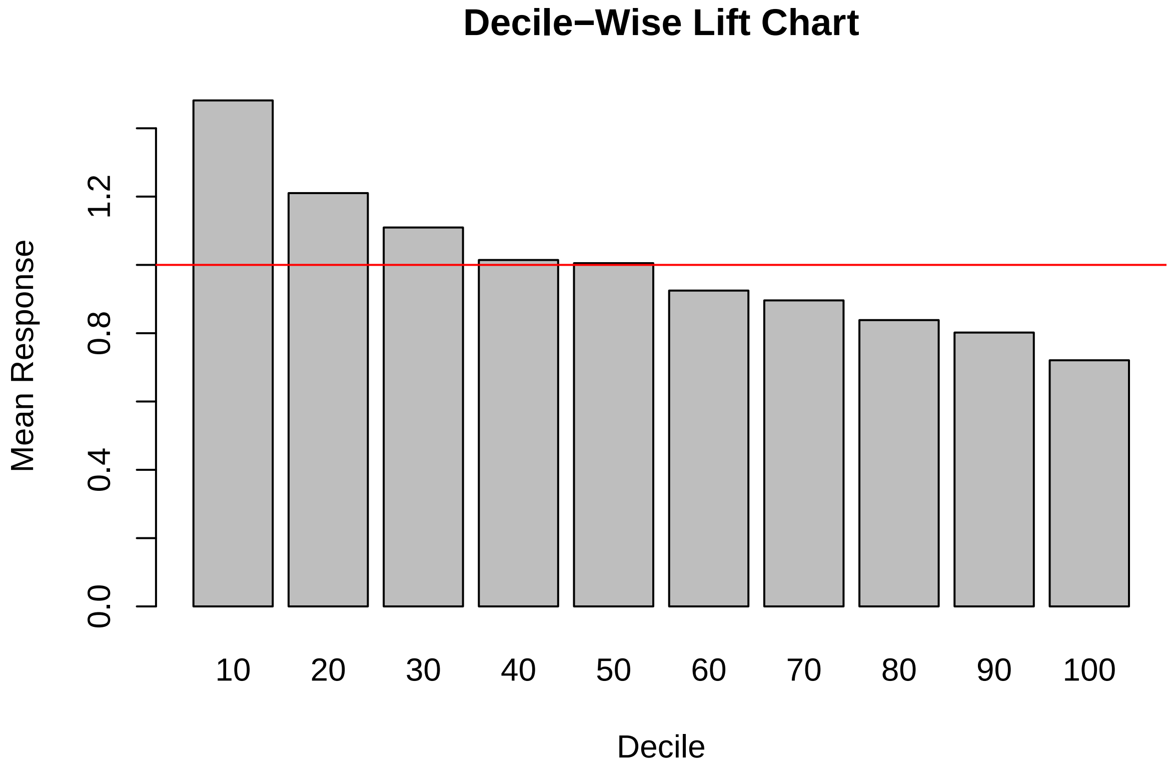
The *naive* estimate of a home's value is the mean. The mean value for a home in the validation set is $392. We can show the model's mean estimate for each decile (10%) group against the naive estimate. The decile groups are ranked from highest estimated values to lowest estimated values.

```
#using functions from gains
gain.num <-gains(valid$TOTAL.VALUE, pred_v, groups=10)
```

```
plot(gain.num$depth,gain.num$mean.resp, xlab = "decile", ylab="Mean Response", main = "Decile vs. Mean
abline(h=mean(valid$TOTAL.VALUE), col="red") #draw a horizontal line showing the naive estimate
```



```
barplot(gain.num$mean.resp/mean(valid$TOTAL.VALUE), names.arg=gain.num$depth, xlab="Decile", ylab="Mean
main="Decile-Wise Lift Chart")
abline(h=1,col='red')
```



```
#print the lift values for decile groups
gain.num$mean.resp/mean(valid$TOTAL.VALUE)
```

```
##  [1] 1.4816089 1.2101900 1.1094903 1.0143124 1.0050042 0.9248303 0.8960245
##  [8] 0.8383601 0.8018731 0.7207139
```

How much better does our model do compare to the naive estimate? For the highest 10% of the estimates, our model's estimates would *potentially* allow us to gain 1.5 times the amount of revenue, compared to choosing 10% of the homes at random.

References

Shmueli, Galit and et al (2018). Data Mining for Business Analytics: Concepts, Techniques, and Applications in R. Hoboken: Wiley. Chapter 6.