Homework 1 - CMTH 642

Christopher Graham

October 22, 2015

Prediction Model for Manhattan Real Estate Prices

Summary

This is an attempt to build a price prediction model using linear regression for a data set consisting of real estate sales data in the borough of Manhattan for August 2012-2013. Data was downloaded from http://www1.nyc.gov/site/finance/taxes/property-rolling-sales-data.page, and an abbreviate code book is available on the site.

The paper goes through the process of cleaning and transforming the data, some visualization and other exploratory analysis.

Next, we build and compare 8 different regression models using a variety of methods and compare the results. (Spoiler alert - none of them are particularly good).

Finally, we give some thought to why our models were so poor and what might be done to build a better predictive model based on this data.

Required libraries

The following libraries are needed to run all code in this report:

- qdap
- lubridate
- ggplot2
- corrplot
- lattice
- leaps
- caret
- MASS

Data import and cleaning

Since read.xlsx is painfully slow on my computer, the spreadsheet was first converted to csv format in Excel so it could be imported more efficiently.
All transformations are documented in the code below. No changes were made in Excel.

Data Import

```
m_sales <- read.csv('rollingsales_manhattan.csv',
                    skip = 4, na.strings= c('', ' '), strip.white=TRUE,
                    stringsAsFactors = FALSE)
```

Basic Data Cleaning

```
# Change sales price to number
m_sales$SALE.PRICE <- gsub('$', '', m_sales$SALE.PRICE, fixed=TRUE)
m_sales$SALE.PRICE <- as.numeric(gsub('\\,', '', m_sales$SALE.PRICE))
# Change Square Foot values to Numeric
m_sales$GROSS.SQUARE.FEET <-
    as.numeric(gsub('\\,', '', m_sales$GROSS.SQUARE.FEET))
m_sales$LAND.SQUARE.FEET <-
    as.numeric(gsub('\\,', '', m_sales$LAND.SQUARE.FEET))

# Change units to numeric class
m_sales$TOTAL.UNITS <- as.numeric(m_sales$TOTAL.UNITS)
m_sales$RESIDENTIAL.UNITS <- as.numeric(m_sales$RESIDENTIAL.UNITS)
m_sales$COMMERCIAL.UNITS <- as.numeric(m_sales$COMMERCIAL.UNITS)
# remove extra spaces in string fields
require(qdap)
m_sales$ADDRESS <- Trim(clean(m_sales$ADDRESS))
m_sales$NEIGHBORHOOD <- Trim(clean(m_sales$NEIGHBORHOOD))
m_sales$BUILDING.CLASS.CATEGORY <- Trim(clean(m_sales$BUILDING.CLASS.CATEGORY))
# convert date fields to date class
require(lubridate)
m_sales$SALE.DATE <- ymd(m_sales$SALE.DATE)

# Remove any record with SALE.PRICE <= $100,000
m_sales <- m_sales[m_sales$SALE.PRICE > 1e+5,]
nrow(m_sales)
```

```
## [1] 16945
```

Note that since we are using this data to develop a model predicting house sale prices, I decided to remove any property in which sale price was listed as $0. According to nyc.gov, these are non-cash transfers, and as such have no relation to a price-predictor model[1].

Similarly, a number of the non-zero sales prices are suspiciously low: $2, $10, $283, etc. Given that the median sales price for a resale property for Q1-2003 in Manhattan was $765,000, with a mean sale price of $1.3 million[2], it seems reasonable to assume that these are data entry errors or some kind of non-open-market sales and to ignore all sales prices below $100,000.[3]

Identifying missing data

```
# identify where data is missing
na_vals <- NULL; zero_vals <- NULL
for (col in 1:ncol(m_sales)) {
    na_vals <- c(na_vals, sum(is.na(m_sales[,col])))
    zero_vals <- c(zero_vals, sum(m_sales[,col] == 0))
}
names(na_vals) <- colnames(m_sales)
```

[1] see code book referenced in summary
[2] source: http://www.millersamuel.com/files/2013/04/Manhattan_1Q_2013.pdf
[3] $100,000 is probably too low, but I have not been able to find reliable data to use as an alternative cutoff.

```r
names(zero_vals) <- colnames(m_sales)

# instances of zero values or NA values
na_vals[na_vals > 0]
```

```
##              NEIGHBORHOOD   BUILDING.CLASS.CATEGORY
##                        17                       263
##      TAX.CLASS.AT.PRESENT                 EASE.MENT
##                       115                     16945
## BUILDING.CLASS.AT.PRESENT         APART.MENT.NUMBER
##                       115                     10314
```

```r
zero_vals[!is.na(zero_vals)][zero_vals[!is.na(zero_vals)] > 0]
```

```
##          ZIP.CODE RESIDENTIAL.UNITS  COMMERCIAL.UNITS       TOTAL.UNITS
##                 1              9616             15859              8650
##  LAND.SQUARE.FEET GROSS.SQUARE.FEET        YEAR.BUILT
##             15206             15276              2169
```

```r
# Is BOROUGH all 1?
summary(m_sales$BOROUGH)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```

```r
# How often does TOTAL.UNITS != COMMERCIAL.UNITS + RESIDENTIAL.UNITS
sum(m_sales$TOTAL.UNITS != m_sales$COMMERCIAL.UNITS +
        m_sales$RESIDENTIAL.UNITS)
```

```
## [1] 346
```

```r
# How often do building classes differ?
sum(m_sales$BUILDING.CLASS.AT.PRESENT !=
        m_sales$BUILDING.CLASS.AT.TIME.OF.SALE,
    na.rm=TRUE)
```

```
## [1] 82
```

```r
nlevels(as.factor(m_sales$BUILDING.CLASS.AT.TIME.OF.SALE))
```

```
## [1] 105
```

```r
nlevels(as.factor(m_sales$BUILDING.CLASS.AT.PRESENT))
```

```
## [1] 105
```

```
# How often does TAX.CLASS.AT.PRESENT differ from TAX.CLASS.AT.TIME.OF.SALE?
sum(m_sales$TAX.CLASS.AT.PRESENT !=
        m_sales$TAX.CLASS.AT.TIME.OF.SALE,
    na.rm=TRUE)
```

```
## [1] 944
```

```
nlevels(as.factor(m_sales$TAX.CLASS.AT.TIME.OF.SALE))
```

```
## [1] 3
```

```
nlevels(as.factor(m_sales$TAX.CLASS.AT.PRESENT))
```

```
## [1] 8
```

This shows us the following:

1. **BOROUGH** is the same value for all records and can be deleted
2. **EASE.MENT** is missing for all records and can be deleted
3. **GROSS.SQUARE.FEET** is missing for 90% of records and cannot be inferred. If this variable is important, we will have to discard all records with missing data.
4. **LAND.SQUARE.FEET** is missing for 90% of records and cannot be inferred. If this variable is important, we will have to discard all records with missing data.
5. **APART.MENT.NUMBER** is missing for many entries. While this could be an interesting variable to include in analysis (floor height may impact price), the actual values are too messy to be easily turned into floor numbers. As such, I have decided to remove this variable.
6. **NEIGHBORHOOD** can easily be inferred as the modal value from ZIP.CODE.
7. **ZIP.CODE** is only missing for one record. This record can be safely excluded.
8. **YEAR.BUILT:** A manual Google search on some of the addresses with missing data does not imply that these values are related to anything (my first thought was that values might be missing for old buildings). For example, 309 West 118 Street is missing this value and was built in 2004[4].
   In some cases, the data can be filled in from other properties with the same address (code below). In cases where this value cannot be inferred, I will use the mean value for the entire data set.
9. **BUILDING.CLASS.AT.PRESENT** is the same as BUILDING.CLASS.AT.TIME.OF.SALE in all but 82 cases. NA values can safely be replaced with this latter value. Note that there are 105 values for Building class.
10. **TAX.CLASS.AT.PRESENT** differs from TAX.CLASS.AT.TIME.OF.SALE in about 5% of cases. Also, note that the number of tax classes differ in these two cases. This is a bit more difficult to deal with and my approach to cleaning these variables is detailed and explained below.
11. **BUILDING.CLASS.CATEGORY** There is no clear way to impute this variable from the information provided. Therefore, should the variable be useful for prediction, the NA value records will be excluded

[4] http://www.elliman.com/new-york-city/brownstone-lane-309-west-118-street-manhattan

Cleaning up TAX.CLASS

```
table(as.factor(m_sales$TAX.CLASS.AT.PRESENT))
```

```
##
##     1    1A    1C     2    2A    2B    2C     4
##   293     2    10 14529   139   174   615  1068
```

```
table(as.factor(m_sales$TAX.CLASS.AT.TIME.OF.SALE))
```

```
##
##     1     2     4
##   304 15570  1071
```

```
# Simplify Present tax class to uber-class
m_sales$TAX.CLASS.PRESENT.SIMPLE <-
    substr(m_sales$TAX.CLASS.AT.PRESENT, 1, 1)

# Do uber-class tax classes change?
sum(m_sales$TAX.CLASS.PRESENT.SIMPLE !=
        m_sales$TAX.CLASS.AT.TIME.OF.SALE,
    na.rm=TRUE)
```

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

TAX.CLASS.AT.PRESENT is divided into more sub-classes than TAX.CLASS.AT.TIME.OF.SALE. It is unclear from New York City's property tax documents[5] whether these sub-classes have an impact on tax rates.

What we do see is that only 5 buildings have a different top-level (1,2,3,4) tax class at present, compared with time of sale. NA Values in TAX.CLASS.AT.PRESENT will thus be replaced with the value of TAX.CLASS.AT.TIME.OF.SALE.

Implementation of Missing Value Replacement

```
# delete BOROUGH, EASE.MENT, APART.MENT.NUMBER
m_sales <- subset(m_sales, select =
                      -c(BOROUGH, EASE.MENT, APART.MENT.NUMBER))

# remove record with missing zip code
m_sales <- m_sales[m_sales$ZIP.CODE != 0, ]

# helper function to compute mode
mode <- function(vect) {
    unq_vect <- unique(vect)
    return(unq_vect[which.max(tabulate(match(vect, unq_vect)))])
}
```

[5] see: http://www1.nyc.gov/site/finance/taxes/definitions-of-property-assessment-terms.page#t and http://www1.nyc.gov/site/finance/taxes/property-tax-rates.page

```
# replace Neighbourhood NA values with mode for Zip Code
fix_nbhd <- is.na(m_sales$NEIGHBORHOOD)

for(i in 1:sum(fix_nbhd)) {
    zip <- m_sales[fix_nbhd,][i,'ZIP.CODE']
    compare_group <- m_sales[m_sales$ZIP.CODE == zip &
                                !is.na(m_sales$NEIGHBORHOOD),
                            'NEIGHBORHOOD']
    new_neighborhood = mode(compare_group)
    m_sales[fix_nbhd,][i,'NEIGHBORHOOD'] = new_neighborhood
}

# replace YEAR.BUILT with modal value for address or overall df mean
fix_yr_blt <- m_sales$YEAR.BUILT == 0
mean_year <- as.integer(mean(m_sales[!fix_yr_blt,
                                     'YEAR.BUILT'], na.rm = TRUE))
for(i in 1:sum(fix_yr_blt)) {
    address <- m_sales[fix_yr_blt,][i,'ADDRESS']
    compare_group <- m_sales[m_sales$ADDRESS == address &
                                m_sales$YEAR.BUILT != 0,
                            'YEAR.BUILT']
    if (length(compare_group) == 0) {
        new_year = mean_year
    } else {
        new_year = mode(compare_group)
    }
    m_sales[fix_yr_blt,][i,'YEAR.BUILT'] = new_year
}

# replace missing BUILDING.CLASS.AT.PRESENT
m_sales[is.na(m_sales$BUILDING.CLASS.AT.PRESENT),
        'BUILDING.CLASS.AT.PRESENT'] <-
    m_sales[is.na(m_sales$BUILDING.CLASS.AT.PRESENT),
        'BUILDING.CLASS.AT.TIME.OF.SALE']

# replace mising TAX.CLASS.AT.PRESENT
m_sales[is.na(m_sales$TAX.CLASS.AT.PRESENT),
        'TAX.CLASS.AT.PRESENT'] <-
    m_sales[is.na(m_sales$TAX.CLASS.AT.PRESENT),
        'TAX.CLASS.AT.TIME.OF.SALE']

# Update TAX.CLASS.PRESENT.SIMPLE
m_sales$TAX.CLASS.PRESENT.SIMPLE <-
    as.integer(substr(m_sales$TAX.CLASS.AT.PRESENT, 1, 1))
```

Other Variable Issues for Regression

Address and Lot are both too granular to be of use. Lot is just a sub-set of Block information and meaningless in the absence of block information.

Block information is also quite granular (1354 unique values), and not particularly well correlated with location, other than generally having lower numbers at the south end of Manhattan and larger numbers at

the north end of the island.[6]

These variables **could** be useful in creating an alternative clustering of property locations. But that is beyond the expectations of this assignment. As such, the variables will be dropped.

```
# Remove ADDRESS, LOT & BLOCK
m_sales <- subset(m_sales, select = -c(ADDRESS, LOT, BLOCK))
```

Data Transformation

Having cleaned up the data, there are now a two additional variables we will introduce in order to create a potentially richer analysis.

1. Although there are small numbers of changes in property or tax class, these changes may have an impact on price. To do this, we create two new binary variables: TAX.CLASS.CHANGE and BUILDING.CLASS.CHANGE
2. Because we are looking at one year of data, the actual calendar date is probably not that useful. However, seasonality is known to impact real estate prices.[7] As such, we will introduce a 'MONTH' variable that may give insight as to whether seasonality is useful predictor of price.
3. Finally, to facilitate analysis, all character class variables will be changed into factors

```
m_sales$TAX.CLASS.CHANGE <- m_sales$TAX.CLASS.PRESENT.SIMPLE ==
    m_sales$TAX.CLASS.AT.TIME.OF.SALE

m_sales$PROPERTY.CLASS.CHANGE <- m_sales$BUILDING.CLASS.AT.PRESENT ==
    m_sales$BUILDING.CLASS.AT.TIME.OF.SALE

m_sales$MONTH <- as.integer(format(m_sales$SALE.DATE, "%m"))
```

Finally, in order to make regression work correctly in R, we need to indicate that the following are nominal variables (at this point in the analysis, R is treating them as continuous numeric variables). This will be done by coercing them to factorial variables in R.

- ZIP.CODE
- TAX.CLASS.AT.TIME.OF.SALE
- TAX.CLASS.PRESENT.SIMPLE
- MONTH

Also, all character vectors will be coerced to factor variables.

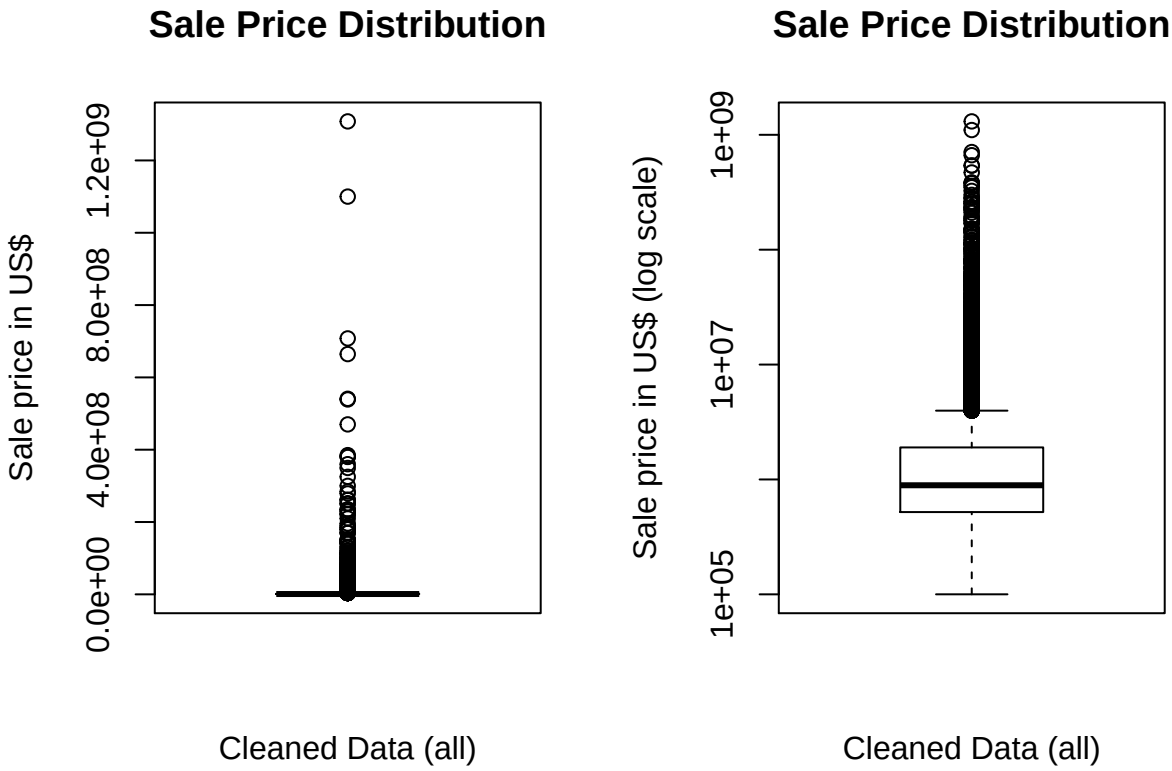
```
change_cols <- z <- c('ZIP.CODE', 'TAX.CLASS.AT.TIME.OF.SALE',
                      'TAX.CLASS.PRESENT.SIMPLE', 'MONTH')
for (i in change_cols) {
    m_sales[,i] <- as.factor(m_sales[,i])
}


# convert characters to factors
m_sales <- as.data.frame(unclass(m_sales))
```

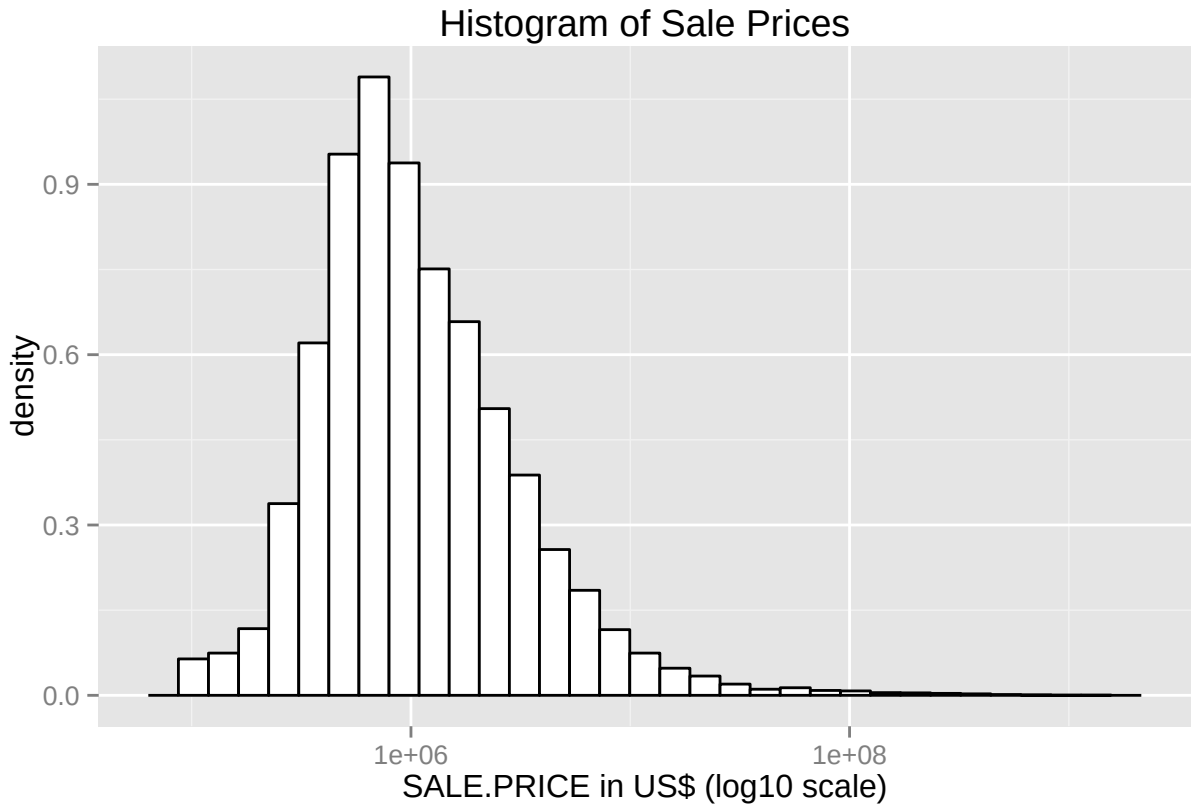
[6] http://gis.nyc.gov/taxmap/map.htm
[7] http://www.fciq.ca/pdf/Carrefour/definitions/en/saisonalite_a.pdf

Data Visualization

What does the price data look like?

```
par(mfrow = c(1,2))

boxplot(m_sales$SALE.PRICE,
        ylab="Sale price in US$",
        xlab="Cleaned Data (all)",
        main='Sale Price Distribution')

boxplot(m_sales$SALE.PRICE, log='y',
        ylab="Sale price in US$ (log scale)",
        xlab="Cleaned Data (all)",
        main='Sale Price Distribution')

par(mfrow = c(1,1))

require(ggplot2)

ggplot(m_sales, aes(x=SALE.PRICE)) +
    geom_histogram(aes(y=..density..), color='black', fill='white') +
    scale_x_log10() +
    ggtitle("Histogram of Sale Prices") +
    xlab("SALE.PRICE in US$ (log10 scale)")
```



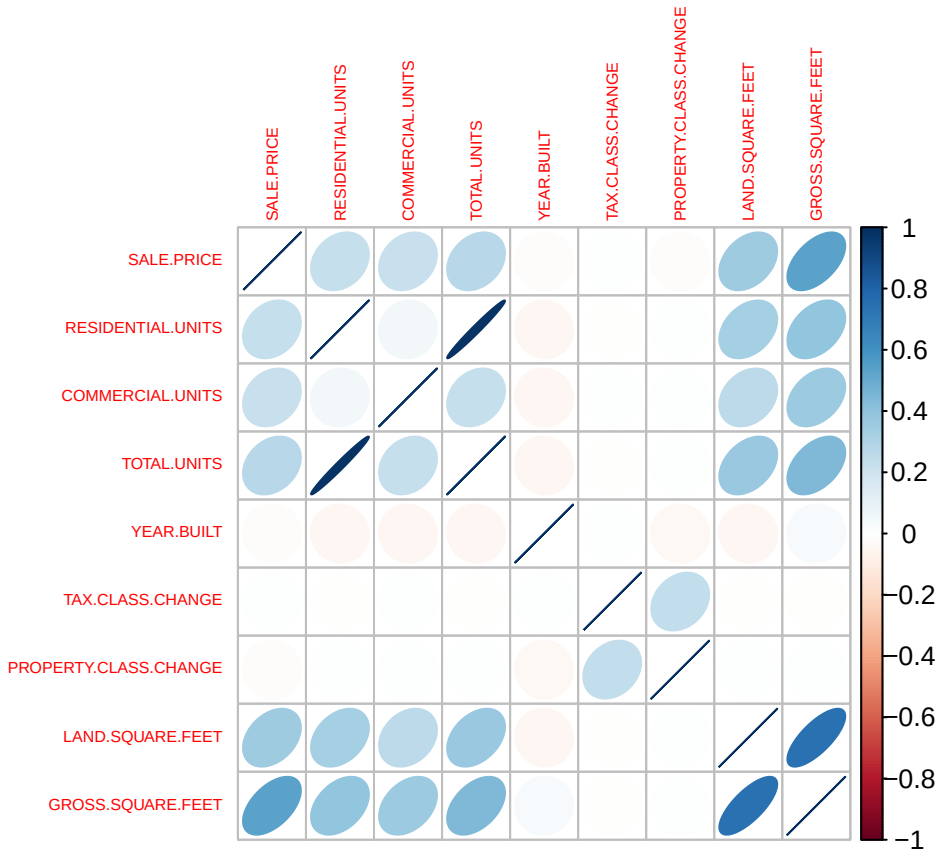


House prices have a significantly long right tail to the distribution, although showing a fairly normal Gaussian shape.

The skew is so strong that to get any kind of useful visualization, we need to transform to a Log10 scale

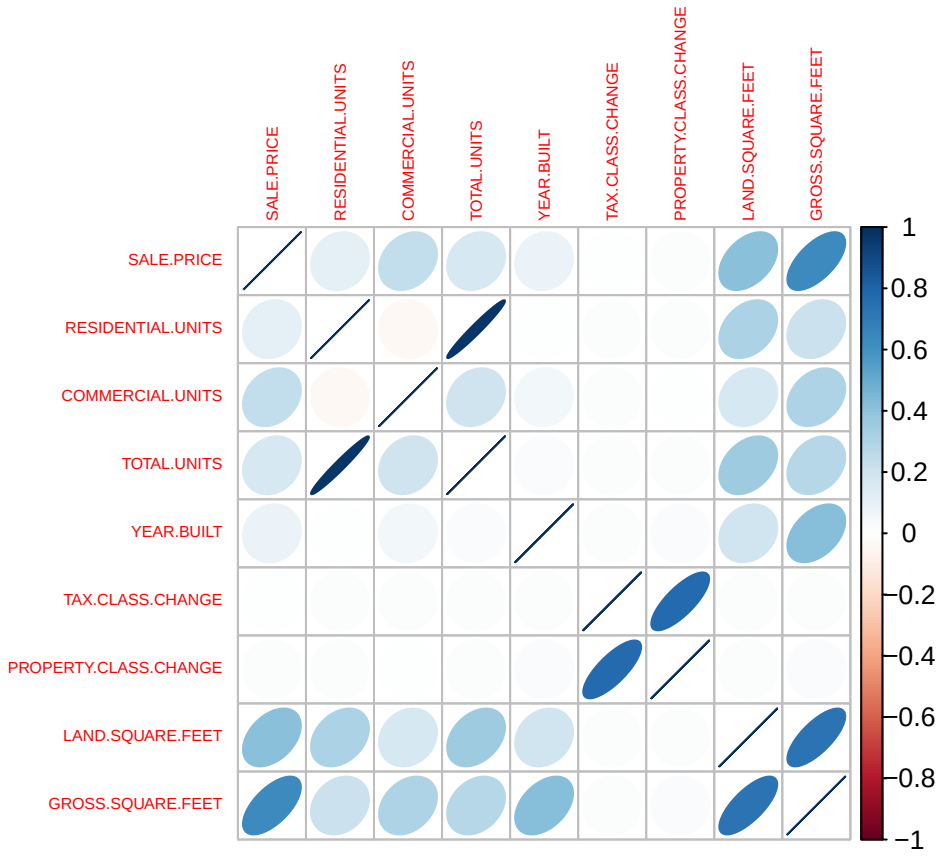
Correlation Comparison Among Numeric Variables

```
require(corrplot)

corrplot(cor(m_sales[,c('SALE.PRICE',
                        'RESIDENTIAL.UNITS',
                        'COMMERCIAL.UNITS',
                        'TOTAL.UNITS',
                        'YEAR.BUILT',
                        'TAX.CLASS.CHANGE',
                        'PROPERTY.CLASS.CHANGE',
                        'LAND.SQUARE.FEET',
                        'GROSS.SQUARE.FEET')]),
         method='ellipse', tl.cex = 0.5)
```



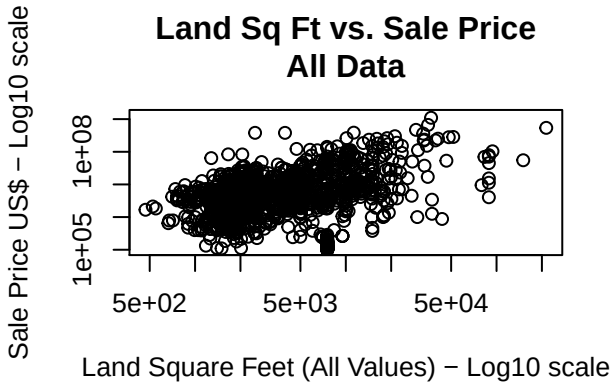

```
# Only including records where we have Sq Foot values
corrplot(cor(m_sales[m_sales$GROSS.SQUARE.FEET != 0 &
                       m_sales$LAND.SQUARE.FEET != 0,
                     c('SALE.PRICE',
                       'RESIDENTIAL.UNITS',
                       'COMMERCIAL.UNITS',
                       'TOTAL.UNITS',
                       'YEAR.BUILT',
                       'TAX.CLASS.CHANGE',
                       'PROPERTY.CLASS.CHANGE',
```

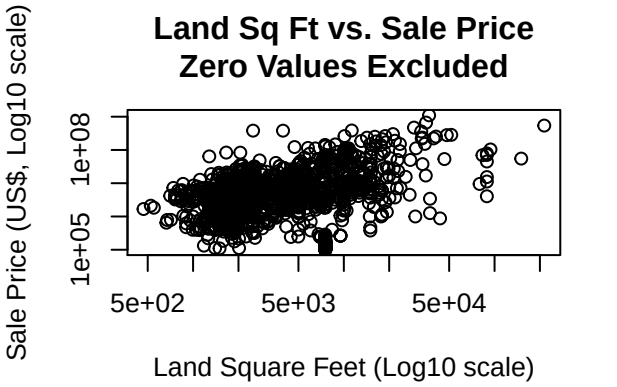
```
                        'LAND.SQUARE.FEET',
                        'GROSS.SQUARE.FEET')]),
         method='ellipse', tl.cex = 0.5)
```

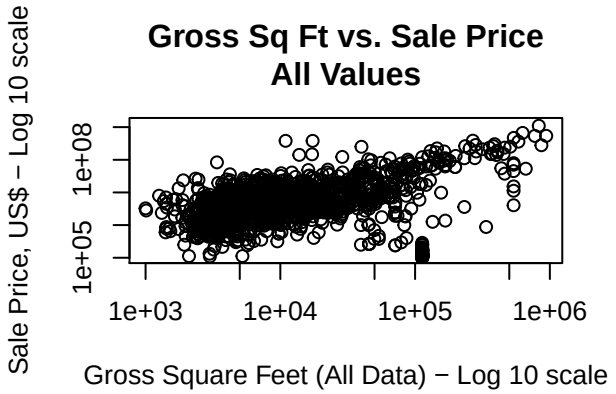


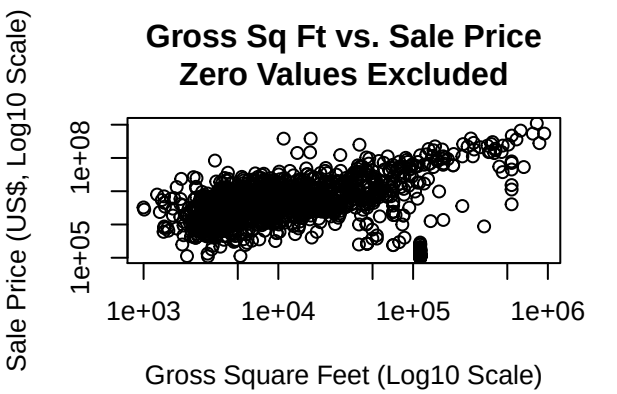

```
# Closer look at Square Foot Values vs. Price
par(mfrow = c(2,2))
par(mar=c(5,4,4,2))

plot(x=m_sales$LAND.SQUARE.FEET,
     y=m_sales$SALE.PRICE,
     log = 'xy',
     xlab = 'Land Square Feet (All Values) - Log10 scale',
     ylab = "Sale Price US$ - Log10 scale",
     main='Land Sq Ft vs. Sale Price\nAll Data')

plot(x=m_sales[m_sales$LAND.SQUARE.FEET>0,]$LAND.SQUARE.FEET,
     y=m_sales[m_sales$LAND.SQUARE.FEET>0,]$SALE.PRICE,
     log = 'xy',
     xlab = 'Land Square Feet (Log10 scale)',
     ylab = "Sale Price (US$, Log10 scale)",
     main='Land Sq Ft vs. Sale Price\nZero Values Excluded')

plot(x=m_sales$GROSS.SQUARE.FEET,
     y=m_sales$SALE.PRICE,
     xlab = 'Gross Square Feet (All Data) - Log 10 scale',
     ylab = "Sale Price, US$ - Log 10 scale",
```

```
     log = 'xy',
     main='Gross Sq Ft vs. Sale Price\nAll Values')

plot(x=m_sales[m_sales$GROSS.SQUARE.FEET>0,]$GROSS.SQUARE.FEET,
     y=m_sales[m_sales$GROSS.SQUARE.FEET>0,]$SALE.PRICE,
     log = 'xy',
     xlab = 'Gross Square Feet (Log10 Scale)',
     ylab = "Sale Price (US$, Log10 Scale)",
     main='Gross Sq Ft vs. Sale Price\nZero Values Excluded')
```

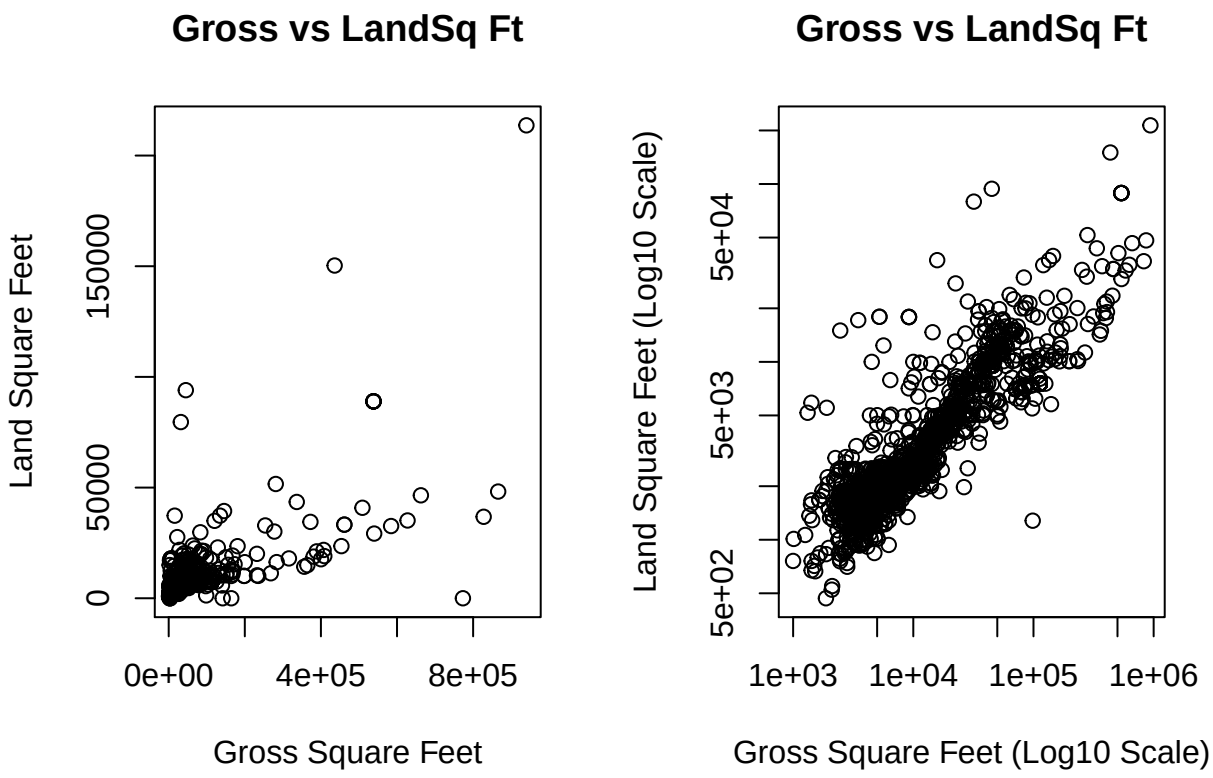




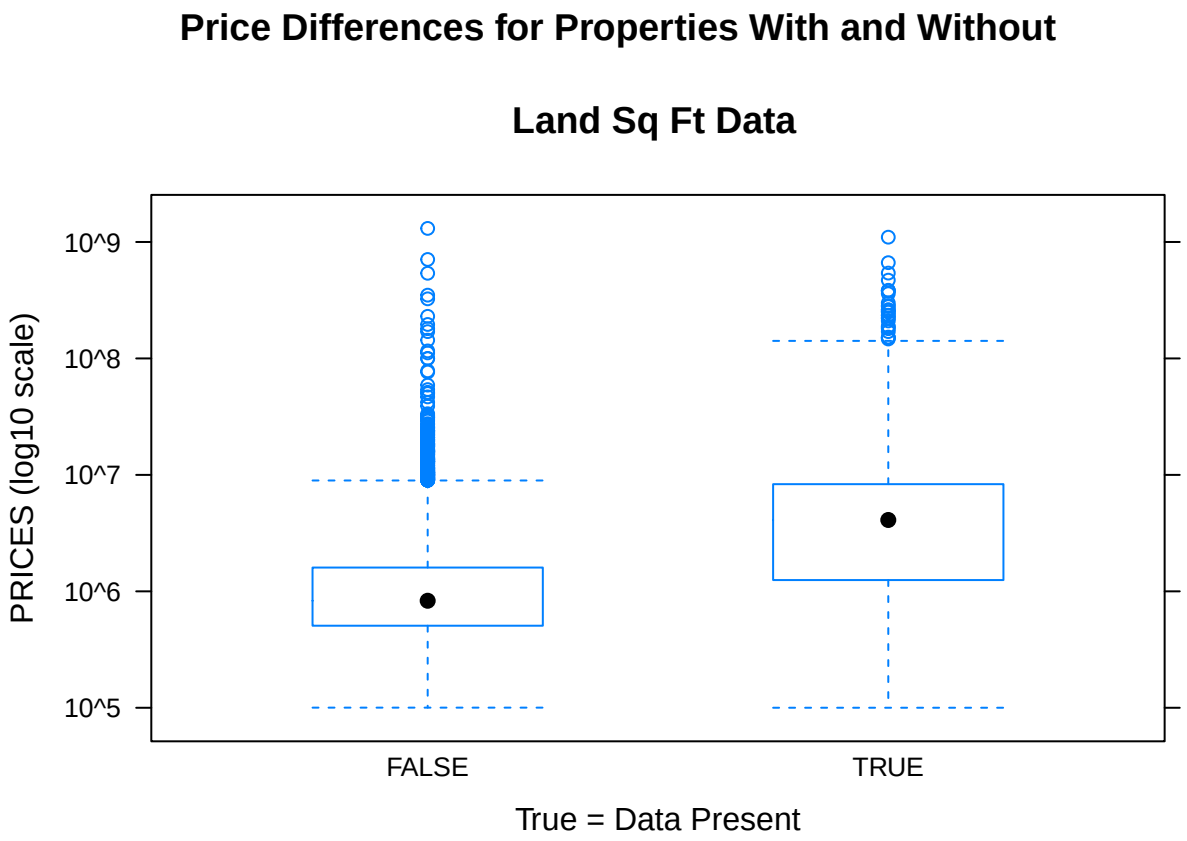


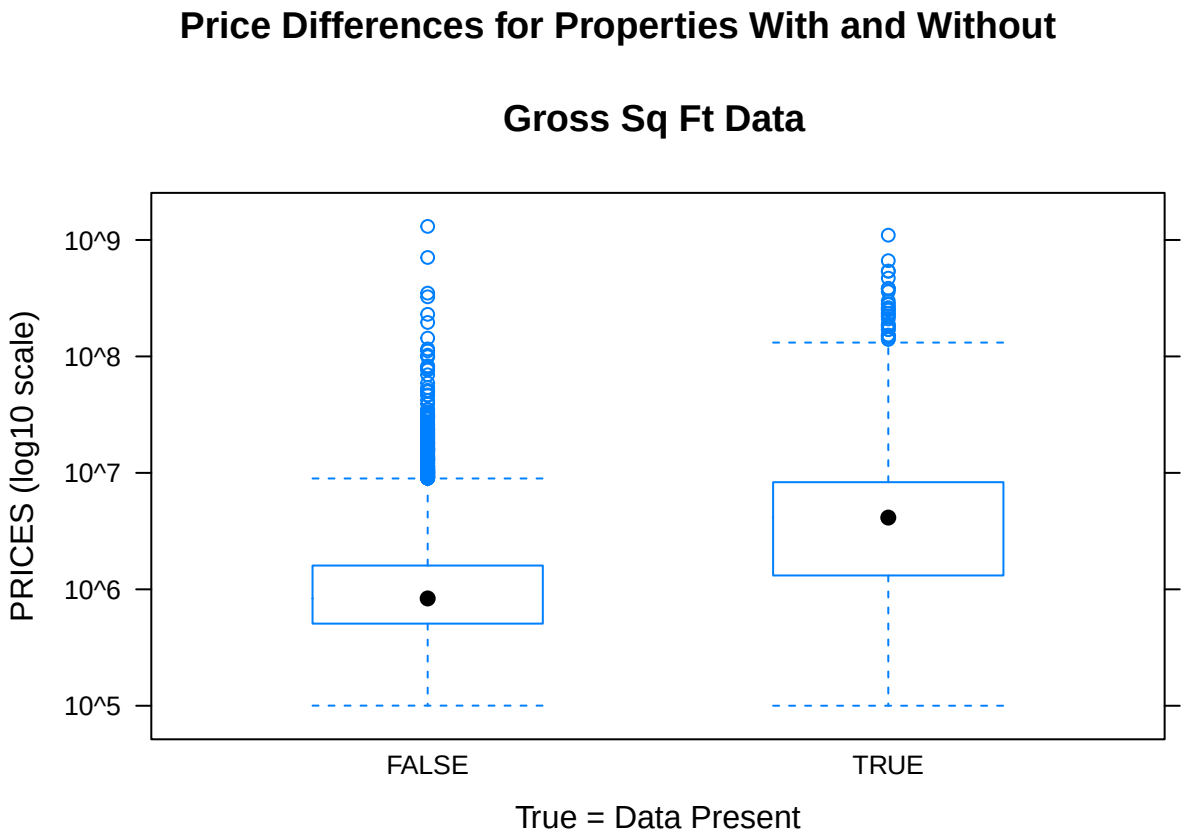



```
par(mfrow = c(1,2))

# Comparison of the two area variables
plot(x=m_sales[m_sales$GROSS.SQUARE.FEET>0,]$GROSS.SQUARE.FEET,
     y=m_sales[m_sales$GROSS.SQUARE.FEET>0,]$LAND.SQUARE.FEET,
     xlab = 'Gross Square Feet',
     ylab = "Land Square Feet",
     main='Gross vs LandSq Ft')

plot(x=m_sales[m_sales$GROSS.SQUARE.FEET>0,]$GROSS.SQUARE.FEET,
     y=m_sales[m_sales$GROSS.SQUARE.FEET>0,]$LAND.SQUARE.FEET,
     log = 'xy',
     xlab = 'Gross Square Feet (Log10 Scale)',
     ylab = "Land Square Feet (Log10 Scale)",
     main='Gross vs LandSq Ft')
```




```
par(mfrow = c(1,1))

# Are Prices similar for properties with and without size info?
require(lattice)
bwplot(SALE.PRICE~LAND.SQUARE.FEET!=0, data=m_sales,
       main = "Price Differences for Properties With and Without\n
       Land Sq Ft Data",
       ylab = "PRICES (log10 scale)",
       xlab = "True = Data Present",
       scales = list(y = list(log=10)),
       horizontal=F)

bwplot(SALE.PRICE~GROSS.SQUARE.FEET!=0, data=m_sales,
       main = "Price Differences for Properties With and Without\n
       Gross Sq Ft Data",
       ylab = "PRICES (log10 scale)",
       xlab = "True = Data Present",
       scales = list(y = list(log=10)),
       horizontal=F)
```





Note that the above corrplotgraphs only account for numeric variables, but we can see that sales price seems

to correlate most strongly with GROSS.SQUARE.FEET, once we remove the 0-value records.

However,, it appears doubtful that the information from properties with area data are in fact representative of all property sales, as there seems like there is a large difference in terms of price (keep in mind that the plots use a log scale for price data). This will need to be addressed further.

There appears to be a fairly significant correlation between both area variables as well, meaning there is likely a high degree of collinearity, and we should be careful about including both values in an analysis.

A quick t-test shows us that it is practically certain that the differences in values seen is NOT random ($p < 2.2e\text{-}16$ in both cases), so we will have to proceed very carefully in generalizing any insights that may be provided by either of these two variables.

```
t.test(SALE.PRICE ~ GROSS.SQUARE.FEET == 0, data = m_sales)
```

```
##
##  Welch Two Sample t-test
##
## data:  SALE.PRICE by GROSS.SQUARE.FEET == 0
## t = 9.9005, df = 1693.4, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   9877294 14757661
## sample estimates:
## mean in group FALSE  mean in group TRUE
##            14076401             1758923
```

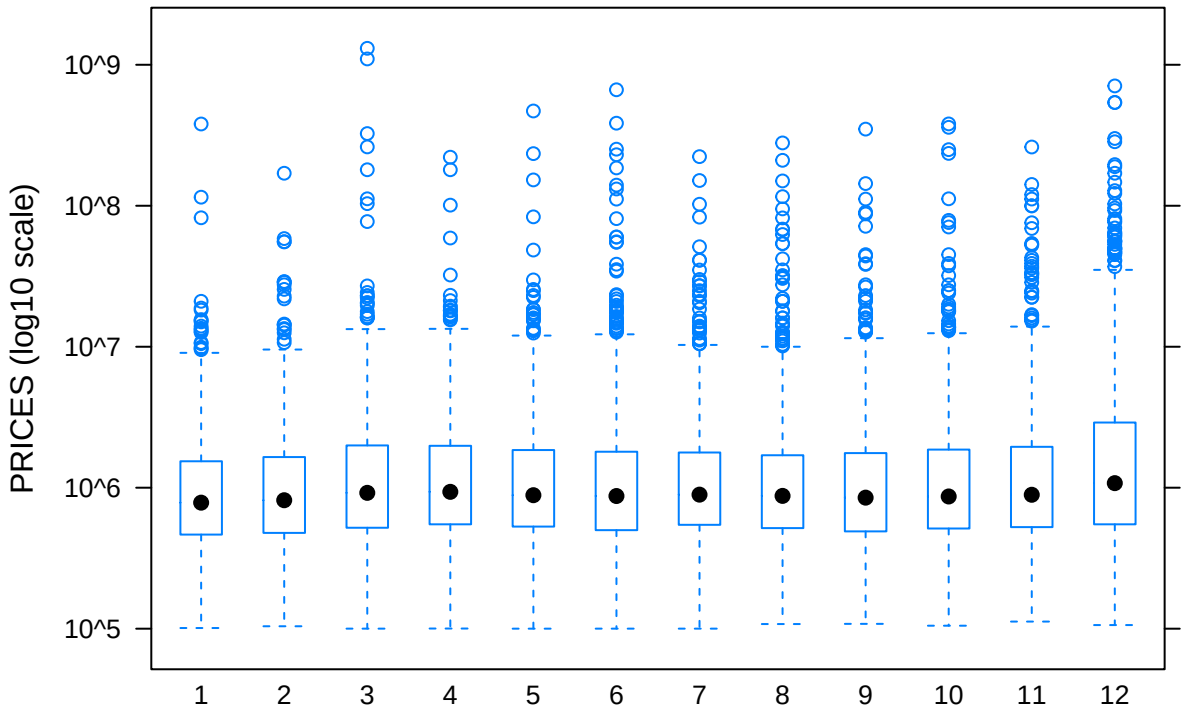
```
t.test(SALE.PRICE ~ LAND.SQUARE.FEET == 0, data = m_sales)
```

```
##
##  Welch Two Sample t-test
##
## data:  SALE.PRICE by LAND.SQUARE.FEET == 0
## t = 10.172, df = 1771.6, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   9475734 14002515
## sample estimates:
## mean in group FALSE  mean in group TRUE
##            13507212             1768087
```

Relations of price to some non-numeric variables

First we'll look at date information

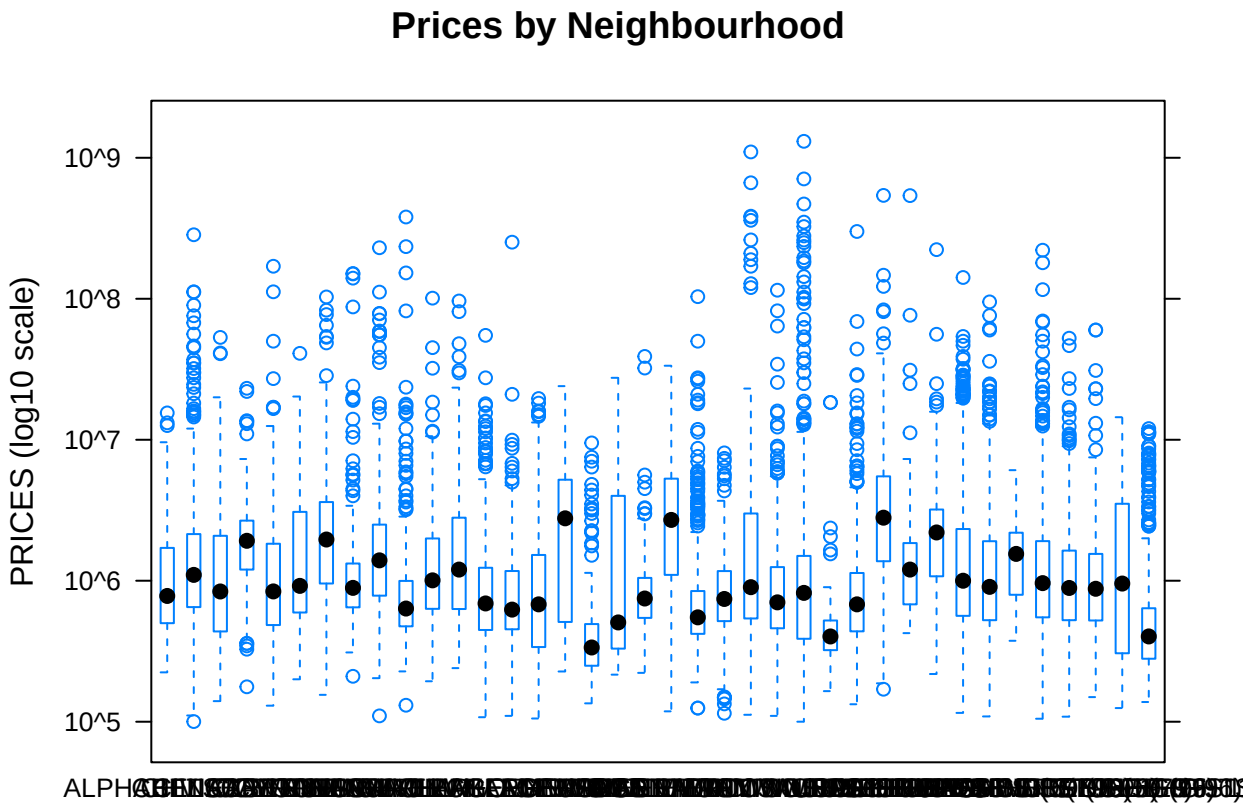
```
# Sale date vs. price
ggplot(m_sales, aes(x=SALE.DATE, y=SALE.PRICE)) +
    geom_point(alpha=0.1) +
    scale_y_log10() +
    ggtitle('Sales Price over Time')

bwplot(SALE.PRICE~MONTH, data=m_sales,
       main = "Prices by month",
       ylab = "PRICES (log10 scale)",
       scales = list(y = list(log=10)),
       horizontal=F)
```

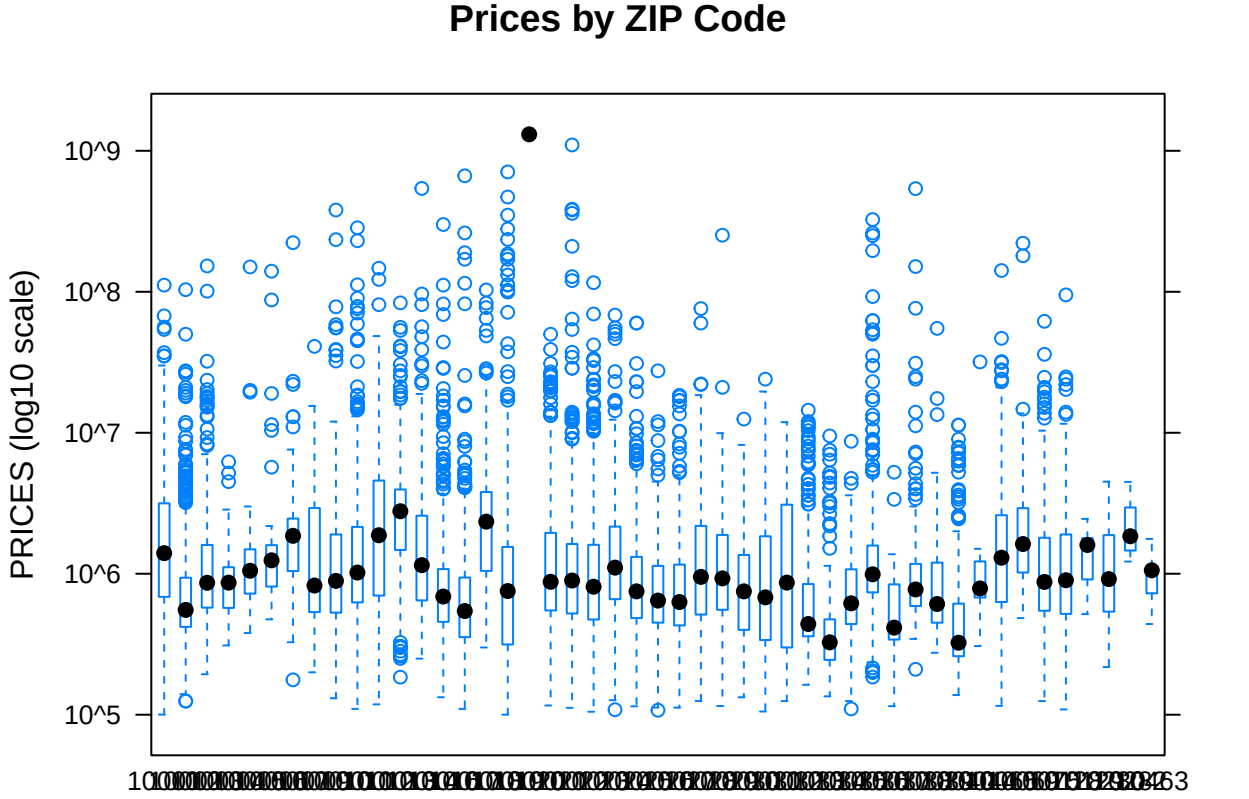


From the looks of it, there do not seem to be any large differences either over time or when sub-setting by month.

Now some of the location information

```
bwplot(SALE.PRICE~NEIGHBORHOOD, data=m_sales,
       main = 'Prices by Neighbourhood',
       ylab = 'PRICES (log10 scale)',
       scales = list(y = list(log=10)),
       horizontal = FALSE)

bwplot(SALE.PRICE~ZIP.CODE, data=m_sales,
       main = 'Prices by ZIP Code',
       ylab = 'PRICES (log10 scale)',
       scales = list(y = list(log=10)),
       horizontal = FALSE)
```

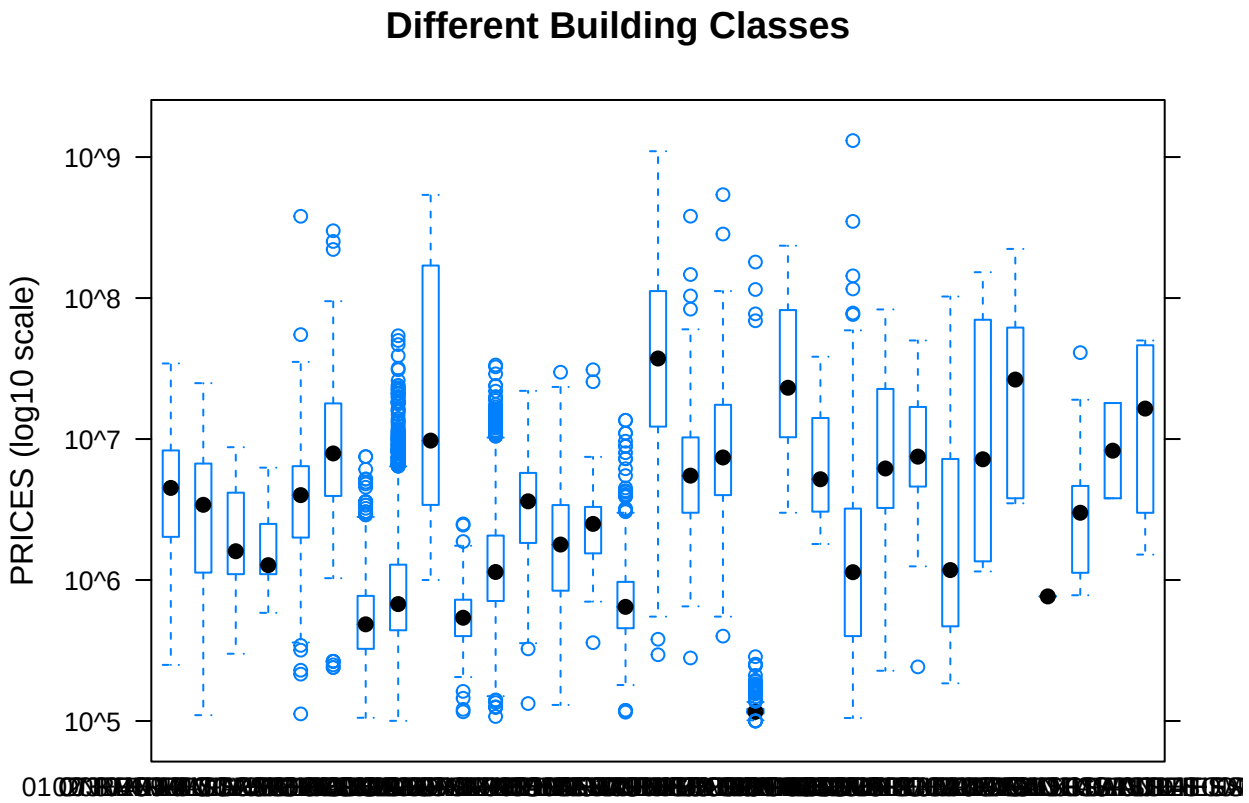


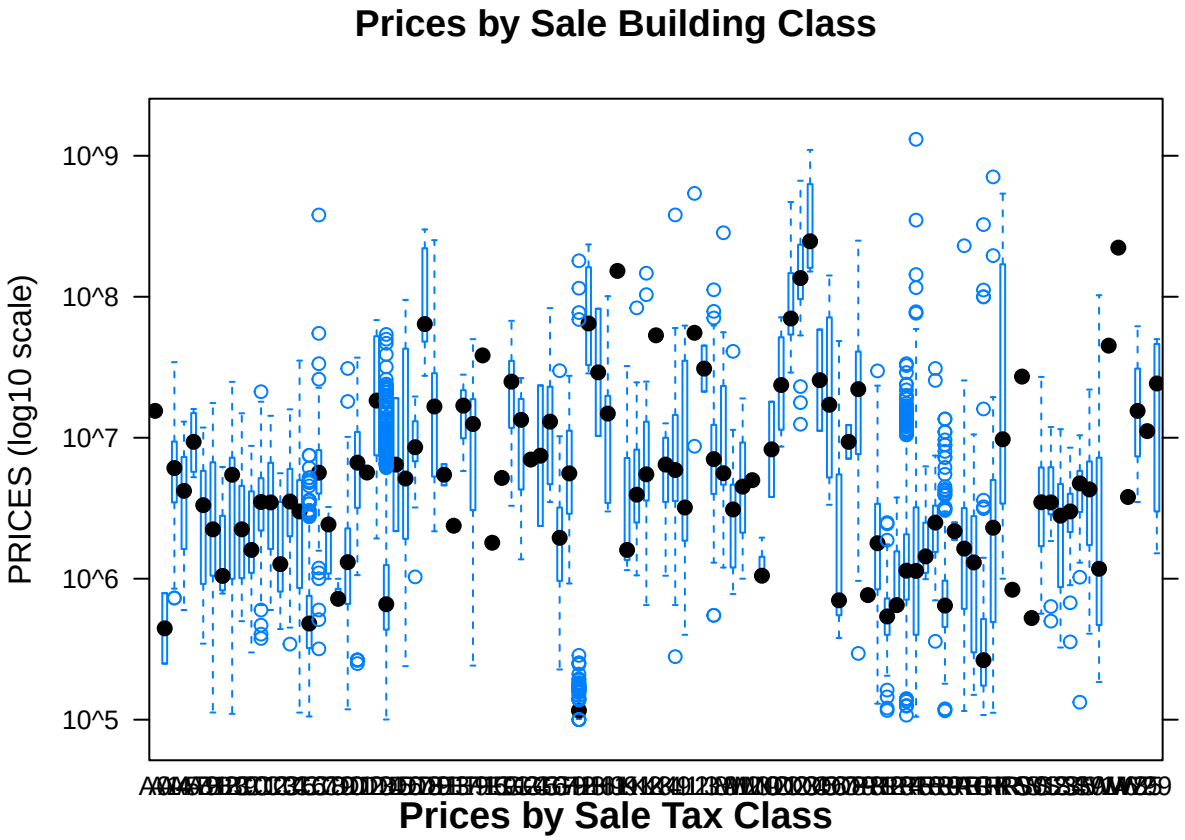


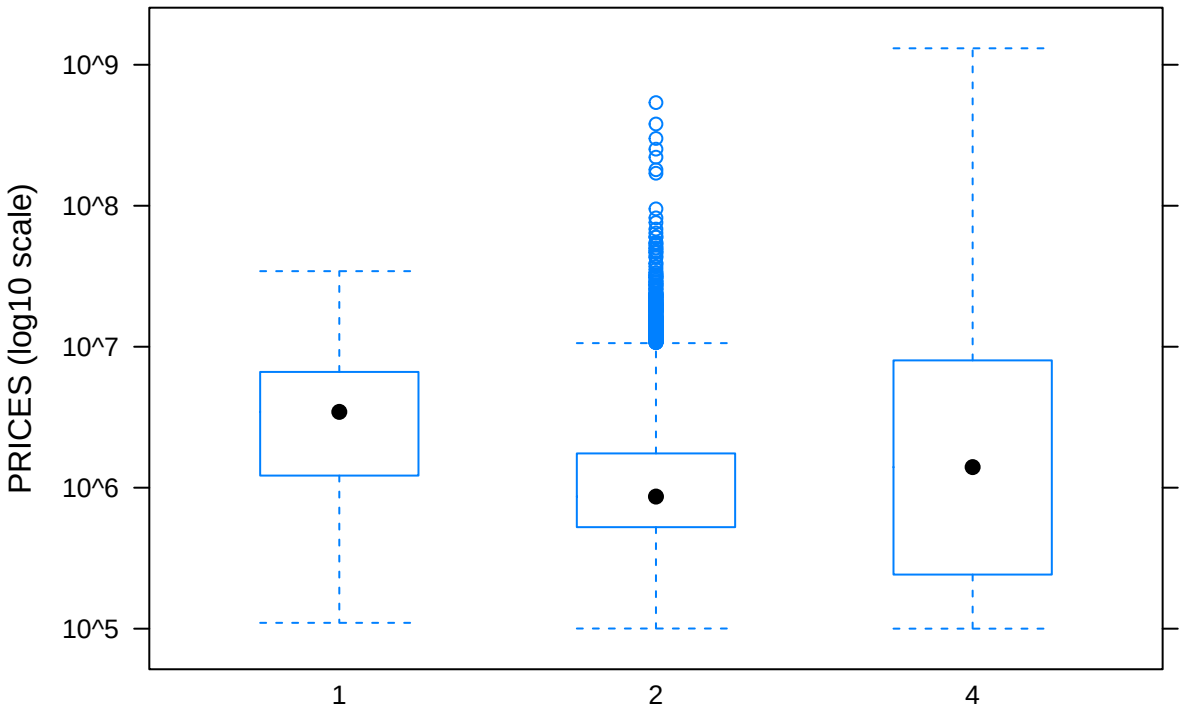
This seems more promising - there appear to be some potentially significant differences between property values based on location, even using the rather rough categories of neighborhood or zip code.

And finally, property classification variables

```
require(lattice)
bwplot(SALE.PRICE~BUILDING.CLASS.CATEGORY, data=m_sales,
       main = "Different Building Classes",
       ylab = "PRICES (log10 scale)",
       scales = list(y = list(log=10)),
       horizontal=F)

bwplot(SALE.PRICE~BUILDING.CLASS.AT.TIME.OF.SALE, data=m_sales,
       main = "Prices by Sale Building Class",
       ylab = "PRICES (log10 scale)",
       scales = list(y = list(log=10)),
       horizontal=F)

bwplot(SALE.PRICE~TAX.CLASS.AT.TIME.OF.SALE, data=m_sales,
       main = "Prices by Sale Tax Class",
       ylab = "PRICES (log10 scale)",
       scales = list(y = list(log=10)),
       horizontal=F)
```







Again, the different property and tax classes have what appears to be a significant level of difference, and hence might be good candidates for a regression variable.

Summary of insights from Visualization

Essentially, we've seen that the best correlation with Sale Price for a numeric variable is with GROSS.SQUARE.FEET, and secondarily with LAND.SQUARE.FEET. These two variables also appear to be quite strongly correlated with each other.

We have also seen that there 90% of the records supplied do not contain values for these two fields, and that, in all likelihood, the SALES.PRICE values for properties with this data are not reflective of the overall data set.

Our corrplot also shows that most of the other numeric variables seem to have a week to non-existent correlation with SALES.PRICE.

On the other hand, a number of the nominal variables seem to have some variation that might be helpful in building a regression model.

Building classes and Tax class seem likely candidates in explaining price variance, while month/time data do not seem all that useful.

These variables will be explored in the multi-variate regression part of the analysis.

Univariate Regression

As per the assignment instructions, we will first look at a regression model using only LAND.SQUARE.FEET as the input variable.

Due to the fact that 90% of the records are missing this value, we will limit our model to just the records containing a value for this field.

We will construct a 80/20 Training/Test split out of this subset of data, then build a simple regression based on the test set. We will examine the details of the model and look at how well it works on the test data.

```
# split data
uvr_data <- m_sales[m_sales$LAND.SQUARE.FEET>0,
                    c('SALE.PRICE', 'LAND.SQUARE.FEET')]
set.seed(6615)
train_crit <- sample(nrow(uvr_data),
                     floor(nrow(uvr_data) * 0.8))
train_set <- uvr_data[train_crit,]
test_set <- uvr_data[-train_crit,]

# build lm model and fit to test panel
land_model <- lm(SALE.PRICE ~ LAND.SQUARE.FEET,
                 data = train_set)
summary(land_model)
```

```
##
## Call:
## lm(formula = SALE.PRICE ~ LAND.SQUARE.FEET, data = train_set)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -204275172   -8152515   -3365310     414820 1013020153
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1323365.9  1410023.4   0.939    0.348
## LAND.SQUARE.FEET    2327.4      127.1  18.314   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 45820000 on 1388 degrees of freedom
## Multiple R-squared:  0.1946, Adjusted R-squared:  0.194
## F-statistic: 335.4 on 1 and 1388 DF,  p-value: < 2.2e-16
```

```
# Get confidence interval for Beta1
sum_coef <- summary(land_model)$coefficients
uvr_conf_int <- sum_coef[2,1] + c(-1,1) *
    qt(0.975, df = land_model$df) * sum_coef[2,2]
uvr_conf_int
```

```
## [1] 2078.137 2576.727
```

Analysis of the linear model results

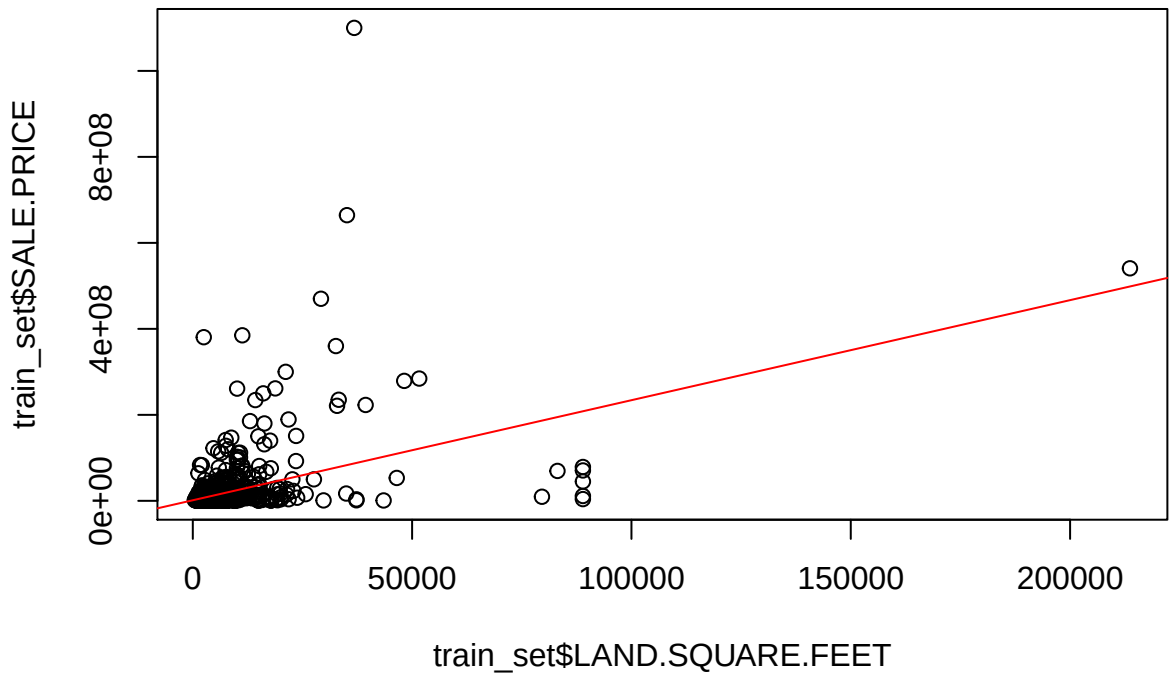
The model gives us an r^2 of 0.194, meaning a little less than 20% of the variation in our training data can be explained by the regression. The p-value of the Beta value (<2e-16) tells us that this is certainly an actual relationship between the variables and not due to chance.

Further, we can see that the model predicts that for every square foot increase in land size, we should see a US$2327 increase in sale price. Furthermore, we can say with 95% confidence, that for the training set, this increment will be in the range 2078.1367519 to 2576.7273313. This is a fairly large range, and the land variable only seems to explain a small portion of the variability, so we can likely do better.

Test set predictions

Before going on though, we need to look at how well this model predicts values in our test set.

```
land_predict <- predict(land_model, interval='prediction', newdata = test_set)
errors <- land_predict[,'fit'] - test_set$SALE.PRICE

require(ggplot2)

plot(train_set$LAND.SQUARE.FEET,
     train_set$SALE.PRICE)
abline(land_model, col='red')
```




```
rmse_land <- sqrt(sum(errors^2/nrow(test_set)))
relative_change <- 1 - ((test_set$SALE.PRICE - abs(errors)) /
                            test_set$SALE.PRICE)
pred25_land <- sum(relative_change < 0.25)/nrow(test_set)
pred10_land <- sum(relative_change < 0.1)/nrow(test_set)
```

What we can see from this plot is that, even for the training data, there is a lot of variability that is not captured by a simple linear model.

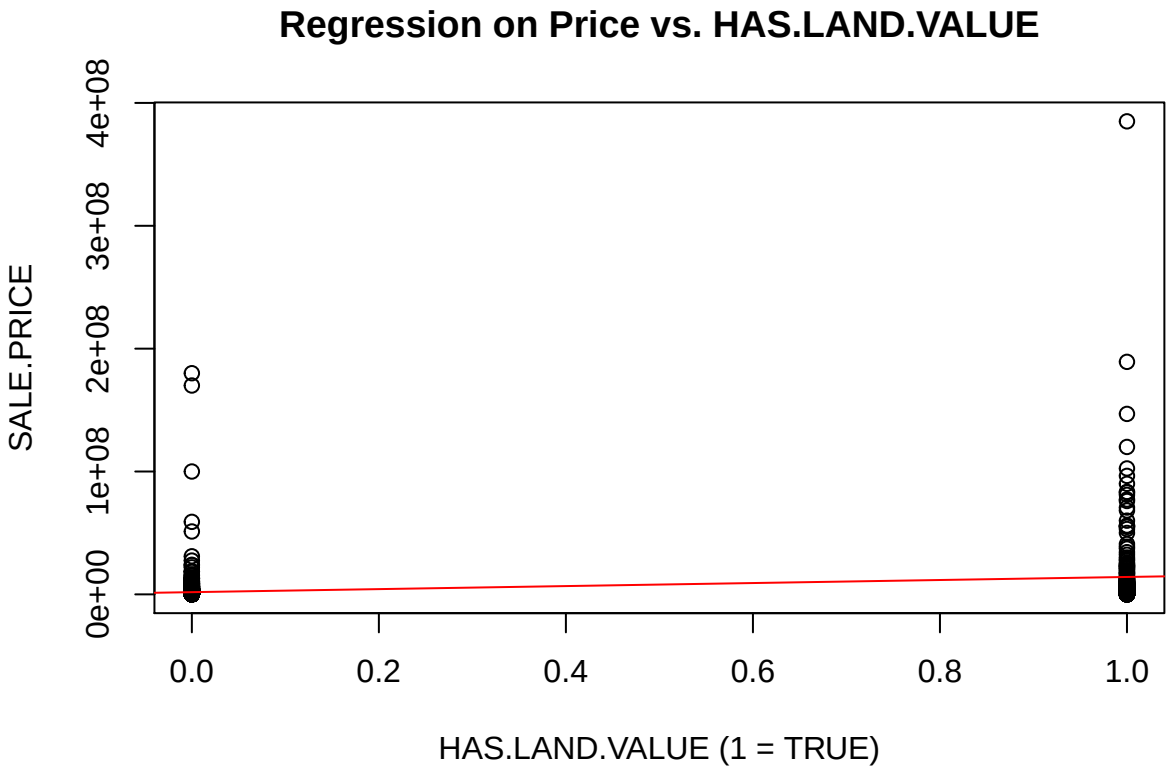
We get the following scores on the model's predictive ability:

RMSE: 34,735,740

pred(25): 0.1522989

As a comparison, let's see how this compares to using only the **presence** of a value in LAND.SQUARE.FEET as a predictor. In that case, we would get:

```
land_value_data <- data.frame(cbind(SALE.PRICE = m_sales$SALE.PRICE,
                        HAS.LAND.VALUE = m_sales$LAND.SQUARE.FEET!=0))
set.seed(6615)
train_crit <- sample(nrow(land_value_data),
                     floor(nrow(land_value_data) * 0.8))
train_set2 <- land_value_data[train_crit,]
test_set2 <- land_value_data[-train_crit,]
has_land_model <- lm(SALE.PRICE ~ HAS.LAND.VALUE, data = train_set2)
has_land_predict <- predict(has_land_model, interval='prediction',
                            newdata = test_set2)
errors2 <- has_land_predict[,'fit'] - test_set2$SALE.PRICE

rmse_has_land <- sqrt(sum(errors2^2/nrow(test_set2)))
relative_change <- 1 - ((test_set2$SALE.PRICE - abs(errors2)) /
                              test_set2$SALE.PRICE)
pred25_has_land <- sum(relative_change < 0.25)/nrow(test_set2)
pred10_has_land <- sum(relative_change < 0.1)/nrow(test_set2)

plot(test_set2$HAS.LAND.VALUE, test_set2$SALE.PRICE,
     main = 'Regression on Price vs. HAS.LAND.VALUE',
     xlab = "HAS.LAND.VALUE (1 = TRUE)",
     ylab = 'SALE.PRICE')
abline(has_land_model, col='red')
```



Finally, let's see what happens if we include the zero-values for land square feet in building a univariate predictive model:

```
train_set3 <- m_sales[train_crit,]
test_set3 <- m_sales[-train_crit,]
land_w0_model <- lm(SALE.PRICE ~ LAND.SQUARE.FEET, data = train_set3)
land_w0_predict <- predict(land_w0_model, interval='prediction',
                           newdata = test_set3)
errors3 <- land_w0_predict[,'fit'] - test_set3$SALE.PRICE

rmse_w0_land <- sqrt(sum(errors3^2/nrow(test_set3)))
relative_change <- 1 - ((test_set3$SALE.PRICE - abs(errors3)) /
                          test_set3$SALE.PRICE)
pred25_w0_land <- sum(relative_change < 0.25)/nrow(test_set3)
pred10_w0_land <- sum(relative_change < 0.1)/nrow(test_set3)

compare_frame <- data.frame(model_name = c('Land Area Model',
                                           'Has Land Area Model',
                                           'Land Model with Zeros'),
                            RMSE = c(rmse_land,
                                     rmse_has_land,
                                     rmse_w0_land),
                            Pred25 = c(pred25_land,
                                       pred25_has_land,
                                       pred25_w0_land),
                            Pred10 = c(pred10_land,
                                       pred10_has_land,
                                       pred10_w0_land),
                            rSquare = c(summary(land_model)$r.squared,
```

```
                                        summary(has_land_model)$r.squared,
                                        summary(land_w0_model)$r.squared),
                                stringsAsFactors = FALSE)

compare_frame
```

```
##              model_name     RMSE    Pred25     Pred10    rSquare
## 1       Land Area Model 34735740 0.1522989 0.05172414 0.19462134
## 2   Has Land Area Model 10661945 0.1363234 0.05429330 0.02790157
## 3 Land Model with Zeros 12005358 0.1498967 0.05724402 0.13800154
```

These are really interesting results in light of the fact that we know that the properties with a value for LAND.SQUARE.FEET are not representative of the full set of properties in the data set.

What we can see is that **even for the properties where we have land area information**, that area information is not particularly valuable as a predictor. In fact, if we just use the presence or absence of land area as a predictor over the entire data set, we can do a much better job of predicting prices - we get a much smaller RMSE, and a similar number of predictions within 10% or 25% of the actual value.

What is interesting in these predictions, however, is highlighted especially in the plot of the HAS.LAND.VALUE regression - the model is doing a poor job with high-priced properties. And given the significant tail of our price distribution to high property values, this is a big problem.

To try and deal with that, we're going to turn to multivariate regression.

Multi-Variate Regression

Given the insights from our univariate regression, we are going to focus on the entire data set for this part of the analysis. We've seen that, even on a limited data set, Land area is an inferior predictor to the presence of data on land area as a predictor.

It seems logical to think that we might see a similar result if we were to look at the GROSS.SQUARE.FEET variable.

So first, let's introduce two new variables into the data frame, that will allow us to potentially use the presence of area data as a predictor:

```
m_sales$HAS.LAND.AREA <- m_sales$LAND.SQUARE.FEET != 0
m_sales$HAS.GROSS.AREA <- m_sales$GROSS.SQUARE.FEET != 0
```

With a number of factor variables with a high number of factor levels, it's a little difficult to know where exactly to start in terms of choosing variables for a multilinear regression.

Based on the earlier visualization and our unilinear regression, some of the following seem promising:

- BUILDING.CLASS.CATEGORY
- ZIP.CODE
- TAX.CLASS.AT.TIME.OF.SALE

But first, let's see what are predicted to be the best variables to use in regression. Note that while regsubsets is helpful in this regard, it would be way too computationally expensive to use it to analyze all the variables available. The factor variables mean that it would be trying to optimize across 350ish variables, something that would quite some time. Furthermore, because it would be treating the categorical variables as a series of 0/1 variables, the results would not be that meaningful.

So, we will start by looking at what is predicted for just the non-factor variables plus the 3-level TAX.CLASS.AT.TIME.OF.SALE:

```
require(leaps)
subsets <- regsubsets(SALE.PRICE~RESIDENTIAL.UNITS +
                          COMMERCIAL.UNITS +
                          TOTAL.UNITS +
                          LAND.SQUARE.FEET +
                          GROSS.SQUARE.FEET +
                          YEAR.BUILT +
                          TAX.CLASS.AT.TIME.OF.SALE +
                          TAX.CLASS.CHANGE +
                          PROPERTY.CLASS.CHANGE +
                          HAS.LAND.AREA +
                          HAS.GROSS.AREA,
                      data=m_sales, nbest=1, nvmax=5)
sub.sum <- summary(subsets)
as.data.frame(sub.sum$outmat)
```

```
##          RESIDENTIAL.UNITS COMMERCIAL.UNITS TOTAL.UNITS LAND.SQUARE.FEET
## 1  ( 1 )
## 2  ( 1 )                                                               *
## 3  ( 1 )                                                               *
## 4  ( 1 )                                                               *
```

```
## 5  ( 1 )                                              *                *
##          GROSS.SQUARE.FEET YEAR.BUILT TAX.CLASS.AT.TIME.OF.SALE2
## 1  ( 1 )                 *
## 2  ( 1 )                 *
## 3  ( 1 )                 *          *
## 4  ( 1 )                 *          *
## 5  ( 1 )                 *                                     *
##          TAX.CLASS.AT.TIME.OF.SALE4 TAX.CLASS.CHANGETRUE
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
##          PROPERTY.CLASS.CHANGETRUE HAS.LAND.AREATRUE HAS.GROSS.AREATRUE
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )                                                              *
## 5  ( 1 )                                                              *
```

With these results, and those of our visualization in mind, we're going to try a preliminary multivariate regression model that includes the following:

- GROSS.SQUARE.FEET
- BUILDING.CLASS.CATEGORY
- TOTAL.UNITS
- HAS.LAND.AREA
- NEIGHBORHOOD

Note that because BUILDING.CLASS.CATEGORY is a 31-level factor, and NEIGHBORHOOD is a 38-level factor, this is technically a 70-variable regression model!

Note that because we did not impute values for NA values in BUILDING.CLASS.CATEGORY, we first have to remove those records from the data frame.

```
multi1_set <- m_sales[complete.cases(m_sales),]

# NOTE - The building class "35 INDOOR PUBLIC AND CULTURAL FACILITIES"
# appears 1x in the data set, causing errors in some test/train splits.
# as a result, we are eliminating it from the data set

multi1_set <- multi1_set[multi1_set$BUILDING.CLASS.CATEGORY!=
                           '35 INDOOR PUBLIC AND CULTURAL FACILITIES',]

set.seed(6615)
train_crit <- sample(nrow(multi1_set), floor(nrow(multi1_set) * 0.8))
train_set4 <- multi1_set[train_crit,]
test_set4 <- multi1_set[-train_crit,]

multi_model1 <- lm(SALE.PRICE ~ GROSS.SQUARE.FEET +
                     BUILDING.CLASS.CATEGORY +
                     TOTAL.UNITS +
                     HAS.LAND.AREA +
```

```
                        NEIGHBORHOOD,
                    data = train_set4)
multi1_predict <- predict(multi_model1,
                          interval = 'prediction',
                          newdata = test_set4)
errors4 <- multi1_predict[,'fit'] - test_set4$SALE.PRICE
rmse_multi1 <- sqrt(sum(errors4^2/nrow(test_set4)))
relative_change <- 1 - ((test_set4$SALE.PRICE - abs(errors4)) /
                            test_set4$SALE.PRICE)
pred25_multi1 <- sum(relative_change < 0.25) / nrow(test_set4)
pred10_multi1 <- sum(relative_change < 0.1) / nrow(test_set4)

compare_frame[4,1] <- 'Multivariate Model 1'
compare_frame[4,2] <- rmse_multi1
compare_frame[4,3] <- pred25_multi1
compare_frame[4,4] <- pred10_multi1
compare_frame[4,5] <- summary(multi_model1)$r.squared

compare_frame
```

```
##              model_name     RMSE    Pred25     Pred10    rSquare
## 1       Land Area Model 34735740 0.1522989 0.05172414 0.19462134
## 2   Has Land Area Model 10661945 0.1363234 0.05429330 0.02790157
## 3 Land Model with Zeros 12005358 0.1498967 0.05724402 0.13800154
## 4  Multivariate Model 1 24087338 0.1927458 0.08183453 0.55915637
```

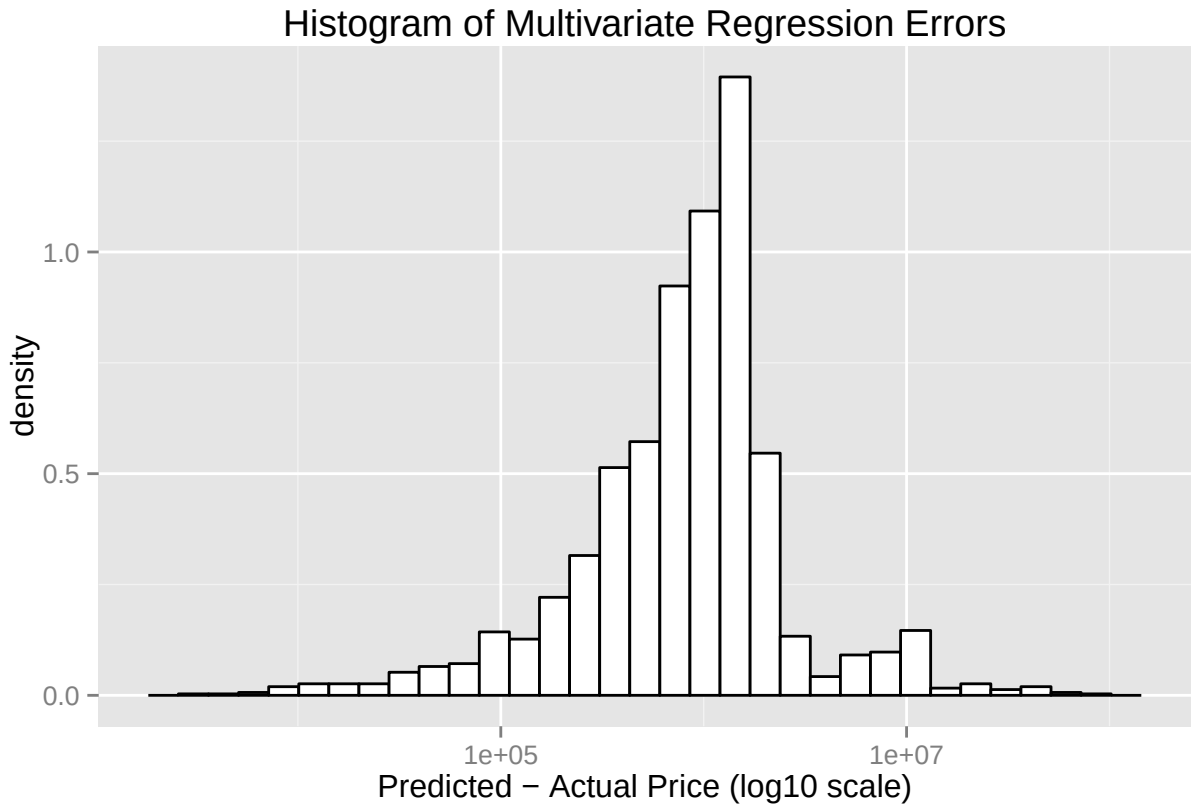
At a first level analysis, this seems is a mixed bag. Our RMSE is up to nearly $25 million. But our pred(25) and pred(10 values are up, although with a pred(25) less than 20%, this is nothing to be too excited about.

So, in terms of utility, this is a pretty weak model. We are not predicting accurately, or even, on average, remotely close to the actual sales value.

So, let's take a somewhat closer look at this model to see if, or how we might be able to improve it.

First, let's visualize the errors:

```
ggplot(as.data.frame(errors4), aes(x=errors4)) +
    geom_histogram(aes(y=..density..), color='black', fill='white') +
    scale_x_log10() +
    ggtitle("Histogram of Multivariate Regression Errors") +
    xlab("Predicted - Actual Price (log10 scale)")
```




```
summary(errors4)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -1.298e+09 -5.654e+05  3.629e+05 -4.958e+05  1.142e+06  7.285e+07
```

Again, what we're seeing is that extreme values are punishing our predictions - we have errors very far off on both extremes, but the errors are heavily weighted to under predicting prices, as the summary data shows.

Next, let's have a look at what variable are contributing to the model with any level of significance. We do this by looking at the p-values for each of the coefficients in our regression equation:

```
sort(summary(multi_model1)$coefficients[,4])
```

```
##                                                 GROSS.SQUARE.FEET
##                                                      0.000000e+00
##                           BUILDING.CLASS.CATEGORY25 LUXURY HOTELS
##                                                     1.126677e-218
##                         BUILDING.CLASS.CATEGORY21 OFFICE BUILDINGS
##                                                     1.562304e-71
##                                  BUILDING.CLASS.CATEGORY17 CONDOPS
##                                                     5.156783e-61
##                 BUILDING.CLASS.CATEGORY09 COOPS - WALKUP APARTMENTS
##                                                     1.193372e-60
##               BUILDING.CLASS.CATEGORY10 COOPS - ELEVATOR APARTMENTS
##                                                     1.659979e-60
##              BUILDING.CLASS.CATEGORY13 CONDOS - ELEVATOR APARTMENTS
##                                                     1.050279e-59
```

```
##                                                HAS.LAND.AREATRUE
##                                                     2.844932e-58
##                                                      (Intercept)
##                                                     1.555725e-56
##          BUILDING.CLASS.CATEGORY15 CONDOS - 2-10 UNIT RESIDENTIAL
##                                                     8.429692e-56
##              BUILDING.CLASS.CATEGORY12 CONDOS - WALKUP APARTMENTS
##                                                     5.059148e-54
##                       BUILDING.CLASS.CATEGORY28 COMMERCIAL CONDOS
##                                                     1.418539e-45
## BUILDING.CLASS.CATEGORY16 CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT
##                                                     1.274202e-43
##                       BUILDING.CLASS.CATEGORY04 TAX CLASS 1 CONDOS
##                                                     2.762073e-34
##                           BUILDING.CLASS.CATEGORY23 LOFT BUILDINGS
##                                                     5.280240e-22
##                                          NEIGHBORHOODMIDTOWN CBD
##                                                     1.148399e-14
##                  BUILDING.CLASS.CATEGORY33 EDUCATIONAL FACILITIES
##                                                     4.772898e-10
##           BUILDING.CLASS.CATEGORY08 RENTALS - ELEVATOR APARTMENTS
##                                                     1.408175e-09
##                       BUILDING.CLASS.CATEGORY29 COMMERCIAL GARAGES
##                                                     8.310364e-04
##                  BUILDING.CLASS.CATEGORY31 COMMERCIAL VACANT LAND
##                                                     1.112598e-03
##              BUILDING.CLASS.CATEGORY07 RENTALS - WALKUP APARTMENTS
##                                                     6.250655e-03
##                                        NEIGHBORHOODSOUTHBRIDGE
##                                                     7.762537e-03
##                                          NEIGHBORHOODCHINATOWN
##                                                     1.208236e-02
##                                              NEIGHBORHOODSOHO
##                                                     2.787801e-02
##                                           NEIGHBORHOODFLATIRON
##                                                     2.810872e-02
##                                        NEIGHBORHOODHARLEM-WEST
##                                                     3.821350e-02
##                          BUILDING.CLASS.CATEGORY22 STORE BUILDINGS
##                                                     5.462168e-02
##                           BUILDING.CLASS.CATEGORY11A CONDO-RENTALS
##                                                     5.466031e-02
##                                         NEIGHBORHOODTRIBECA
##                                                     1.183853e-01
##                                      NEIGHBORHOODHARLEM-CENTRAL
##                                                     1.192205e-01
##                                       NEIGHBORHOODHARLEM-UPPER
##                                                     1.289051e-01
##                                NEIGHBORHOODWASHINGTON HEIGHTS LOWER
##                                                     1.320672e-01
##                               BUILDING.CLASS.CATEGORY26 OTHER HOTELS
##                                                     1.478956e-01
##                         BUILDING.CLASS.CATEGORY14 RENTALS - 4-10 UNIT
##                                                     1.479610e-01
```

```
##                                       NEIGHBORHOODMIDTOWN WEST
##                                                   1.494843e-01
##                     BUILDING.CLASS.CATEGORY03 THREE FAMILY HOMES
##                                                   1.678198e-01
##          BUILDING.CLASS.CATEGORY32 HOSPITAL AND HEALTH FACILITIES
##                                                   1.958724e-01
##                                NEIGHBORHOODGREENWICH VILLAGE-WEST
##                                                   2.062016e-01
##                                      NEIGHBORHOODHARLEM-EAST
##                                                   2.076652e-01
##                               NEIGHBORHOODUPPER WEST SIDE (59-79)
##                                                   2.176334e-01
##                                          NEIGHBORHOODGRAMERCY
##                                                   2.299102e-01
##                                            NEIGHBORHOODINWOOD
##                                                   2.363924e-01
##                               NEIGHBORHOODUPPER EAST SIDE (59-79)
##                                                   2.485800e-01
##                                         NEIGHBORHOODCHELSEA
##                                                   2.715306e-01
##                              NEIGHBORHOODWASHINGTON HEIGHTS UPPER
##                                                   2.846847e-01
##                                       NEIGHBORHOODCIVIC CENTER
##                                                   3.696431e-01
##                            NEIGHBORHOODGREENWICH VILLAGE-CENTRAL
##                                                   3.752515e-01
##                                       NEIGHBORHOODLITTLE ITALY
##                                                   4.012718e-01
##                               NEIGHBORHOODUPPER EAST SIDE (79-96)
##                                                   4.162617e-01
##                         BUILDING.CLASS.CATEGORY02 TWO FAMILY HOMES
##                                                   4.775146e-01
##                              NEIGHBORHOODUPPER EAST SIDE (96-110)
##                                                   5.504670e-01
##                                       NEIGHBORHOODEAST VILLAGE
##                                                   5.764635e-01
##                               NEIGHBORHOODUPPER WEST SIDE (79-96)
##                                                   5.987147e-01
##                                       NEIGHBORHOODMIDTOWN EAST
##                                                   6.527399e-01
##                                     NEIGHBORHOODJAVITS CENTER
##                                                   6.644625e-01
##                                                    TOTAL.UNITS
##                                                   7.246655e-01
##                              NEIGHBORHOODUPPER WEST SIDE (96-116)
##                                                   7.433095e-01
##                         BUILDING.CLASS.CATEGORY38 ASYLUMS AND HOMES
##                                                   7.547036e-01
##                                         NEIGHBORHOODFASHION
##                                                   7.957453e-01
##                                  NEIGHBORHOODMORNINGSIDE HEIGHTS
##                                                   8.095245e-01
##                                       NEIGHBORHOODCLINTON
##                                                   8.099877e-01
```

```
##                                         NEIGHBORHOODKIPS BAY
##                                                8.131941e-01
##                     BUILDING.CLASS.CATEGORY37 RELIGIOUS FACILITIES
##                                                8.353621e-01
##                                       NEIGHBORHOODFINANCIAL
##                                                8.770523e-01
##                                    NEIGHBORHOODLOWER EAST SIDE
##                                                8.799618e-01
##                             BUILDING.CLASS.CATEGORY30 WAREHOUSES
##                                                9.012772e-01
##                                      NEIGHBORHOODMURRAY HILL
##                                                9.102998e-01
##                      BUILDING.CLASS.CATEGORY41 TAX CLASS 4 - OTHER
##                                                9.362884e-01
##                                  NEIGHBORHOODMANHATTAN VALLEY
##                                                9.395055e-01
##                               BUILDING.CLASS.CATEGORY27 FACTORIES
##                                                9.509769e-01
```

What we can see here is that, in general, the neighborhood values are not having a significant impact on our model, while the building class information is having a significant impact.

Perhaps, though we can find a better fit by looking at the data a bit more closely. To that end, we will approach model building with a k-folds validation approach.

Experiment with Multi-Variate Regression and k-folds Cross-Validation

k-Fold Cross-Validation gives you a better idea of the predictive value you can expect from a data set than merely using a train/test split.

So, let's start by seeing how our multi-variate model stacks up when evaluated through a k-fold validation.

```
require(caret)

set.seed(6615)
fold_set <- createFolds(multi1_set$SALE.PRICE)

rmse_k <- NULL
pred25_k <- NULL
pred10_k <- NULL
r2_k <- NULL

for (fold in fold_set) {
    # set train and test sets for iteration
    train_k <- multi1_set[-fold,]
    test_k <- multi1_set[fold,]

    # create and test model
    model_k <- lm(SALE.PRICE ~ GROSS.SQUARE.FEET +
                      BUILDING.CLASS.CATEGORY +
                      TOTAL.UNITS +
                      HAS.LAND.AREA +
                      NEIGHBORHOOD,
                  data = train_k)

    predict_k <- predict(model_k, interval='prediction',
                         newdata = test_k)

    # Calculate relevants performance stats
    errors_k <- predict_k[,'fit'] - test_k$SALE.PRICE
    rmse_k <- c(rmse_k, sqrt(sum(errors_k^2/nrow(test_k))))
    relative_change <- 1 - ((test_k$SALE.PRICE - abs(errors_k))/
                                test_k$SALE.PRICE)
    pred25_k <- c(pred25_k, sum(relative_change < 0.25) /
                      nrow(test_k))
    pred10_k <- c(pred10_k, sum(relative_change < 0.1) /
                      nrow(test_k))
    r2_k <- c(r2_k, summary(model_k)$r.squared)
}

compare_frame[5,1] <- 'k-Fold on Multi-Model'
compare_frame[5,2] <- mean(rmse_k)
compare_frame[5,3] <- mean(pred25_k)
compare_frame[5,4] <- mean(pred10_k)
compare_frame[5,5] <- mean(r2_k)

compare_frame
```

```
##              model_name     RMSE    Pred25     Pred10    rSquare
## 1       Land Area Model 34735740 0.1522989 0.05172414 0.19462134
## 2   Has Land Area Model 10661945 0.1363234 0.05429330 0.02790157
## 3 Land Model with Zeros 12005358 0.1498967 0.05724402 0.13800154
## 4  Multivariate Model 1 24087338 0.1927458 0.08183453 0.55915637
## 5 k-Fold on Multi-Model 13548759 0.1778183 0.07314099 0.43085184
```

With a bit more validation, we minimize some of the randomness that might have applied to our original evaluation of this model in the earlier section, seeing a much better RMSE score, and a somewhat worse score for both pred(25) and pred(10). But still in the same ballpark.

The question remains, however - can we do better?

2nd Model with k-fold validation

It still seems unlikely to me that location does not play a role in real estate values. After all, "Location, Location, Location" is supposedly one of the axioms of Real Estate pricing.

So for this model, we're going to see if the interaction between building class and neighborhood will help us build a better prediction model.

Because of the large number of potential interactions, k-means is going to be especially useful to help us understand the predictive nature of this model, as we wind up with a model having 2040 coefficients!.

```
require(caret)

set.seed(6615)
fold_set <- createFolds(multi1_set$SALE.PRICE)

rmse_k <- NULL
pred25_k <- NULL
pred10_k <- NULL
r2_k <- NULL

for (fold in fold_set) {
    # set train and test sets for iteration
    train_k <- multi1_set[-fold,]
    test_k <- multi1_set[fold,]

    # create and test model
    model_k <- lm(SALE.PRICE ~ GROSS.SQUARE.FEET +
                      BUILDING.CLASS.CATEGORY +
                      TOTAL.UNITS +
                      HAS.LAND.AREA +
                      YEAR.BUILT +
                      NEIGHBORHOOD +
                      NEIGHBORHOOD * BUILDING.CLASS.CATEGORY,
                  data = train_k)

    predict_k <- predict(model_k, interval='prediction',
                         newdata = test_k)

    # Calculate relevants performance stats
    errors_k <- predict_k[,'fit'] - test_k$SALE.PRICE
```

```
    rmse_k <- c(rmse_k, sqrt(sum(errors_k^2/nrow(test_k))))
    relative_change <- 1 - ((test_k$SALE.PRICE - abs(errors_k))/
                                test_k$SALE.PRICE)
    pred25_k <- c(pred25_k, sum(relative_change < 0.25) /
                        nrow(test_k))
    pred10_k <- c(pred10_k, sum(relative_change < 0.1) /
                        nrow(test_k))
    r2_k <- c(r2_k, summary(model_k)$r.squared)
}

compare_frame[6,1] <- 'k-Fold Model'
compare_frame[6,2] <- mean(rmse_k)
compare_frame[6,3] <- mean(pred25_k)
compare_frame[6,4] <- mean(pred10_k)
compare_frame[6,5] <- mean(r2_k)

compare_frame
```

```
##              model_name     RMSE    Pred25     Pred10    rSquare
## 1       Land Area Model 34735740 0.1522989 0.05172414 0.19462134
## 2   Has Land Area Model 10661945 0.1363234 0.05429330 0.02790157
## 3 Land Model with Zeros 12005358 0.1498967 0.05724402 0.13800154
## 4  Multivariate Model 1 24087338 0.1927458 0.08183453 0.55915637
## 5 k-Fold on Multi-Model 13548759 0.1778183 0.07314099 0.43085184
## 6          k-Fold Model 13867375 0.2473619 0.09850164 0.53822628
```

This really is our best model yet - with a RMSE second only to our prediction on HAS.LAND.AREA, but a significantly better value for PRED(25) - nearly 25%!, and somewhat of an improvement on Pred(10).

Still though, with our RMSE still at the $13 million dollar level, and with 75% of our predictions off by more than 25%, this model - as complicated as it is - is still not very good.

Stepwise Regression Approach

Finally, then, we'll see if our intuition has been off. Step-wise regression will allow us to look at our variables in a more structured way to see if we can find those that best predict sale price, using predicted p-values to add or remove variables.

It's important to note, however, that in much of the regression work done in this paper, we've been looking at factor variables. And when factors are converted into a linear model, they are transformed into ((number of factor levels) - 1) binary variables. In other words, a 38-level factor, like NEIGHBORHOOD generates 37 binary variables. So the calculations that follow are somewhat computationally expensive.

They also don't wind up with models that precisely fit the criteria of the question, as they are not limited to 5 variables, but rather come in with 9 variables in both the forward and backwards approaches.

Stepwise regression with ALL Variables

```
## Note that building classes F5 and W4 each have only
# one value and their inclusion induces errors in the regression
# below.  They will be removed to prevent this

multi1_set <- multi1_set[multi1_set$BUILDING.CLASS.AT.PRESENT!=
                           'F5',]
multi1_set <- multi1_set[multi1_set$BUILDING.CLASS.AT.PRESENT!=
                           'W4',]
multi1_set <- multi1_set[multi1_set$BUILDING.CLASS.AT.PRESENT!=
                           'S0',]
multi1_set <- multi1_set[multi1_set$BUILDING.CLASS.AT.TIME.OF.SALE!=
                           'F5',]
multi1_set <- multi1_set[multi1_set$BUILDING.CLASS.AT.TIME.OF.SALE!=
                           'W4',]
multi1_set <- multi1_set[multi1_set$BUILDING.CLASS.AT.TIME.OF.SALE!=
                           'S0',]
set.seed(6615)
train_crit <- sample(nrow(multi1_set), floor(nrow(multi1_set) * 0.8))
train_set5 <- multi1_set[train_crit,]
test_set5 <- multi1_set[-train_crit,]
require(MASS)
null <- lm(SALE.PRICE ~ 1, data = train_set5)
full <- lm(SALE.PRICE ~ ., data = train_set5)
step_f_model <- stepAIC(null, scope=list(lower=null, upper=full),
                        direction = 'forward', trace=FALSE)
step_b_model <- stepAIC(full, direction = 'backward', trace=FALSE)
summary(step_f_model)$call
```

```
## lm(formula = SALE.PRICE ~ BUILDING.CLASS.AT.PRESENT + ZIP.CODE +
##     GROSS.SQUARE.FEET + LAND.SQUARE.FEET + COMMERCIAL.UNITS +
##     HAS.LAND.AREA + NEIGHBORHOOD + RESIDENTIAL.UNITS + HAS.GROSS.AREA +
##     SALE.DATE, data = train_set5)
```

```
summary(step_b_model)$call
```

```
## lm(formula = SALE.PRICE ~ NEIGHBORHOOD + BUILDING.CLASS.AT.PRESENT +
```

```
##     ZIP.CODE + RESIDENTIAL.UNITS + TOTAL.UNITS + LAND.SQUARE.FEET +
##     GROSS.SQUARE.FEET + SALE.DATE + HAS.LAND.AREA + HAS.GROSS.AREA,
##     data = train_set5)
```

```
step_f_predict <- predict(step_f_model,
                          interval = 'prediction',
                          newdata = test_set5)
step_b_predict <- predict(step_b_model,
                          interval = 'prediction',
                          newdata = test_set5)
errors_f <- step_f_predict[,'fit'] - test_set5$SALE.PRICE
errors_b <- step_b_predict[,'fit'] - test_set5$SALE.PRICE
rmse_step_f <- sqrt(sum(errors_f^2/nrow(test_set5)))
rmse_step_b <- sqrt(sum(errors_b^2/nrow(test_set5)))
relative_change_f <- 1 - ((test_set5$SALE.PRICE - abs(errors_f)) /
                            test_set5$SALE.PRICE)
relative_change_b <- 1 - ((test_set5$SALE.PRICE - abs(errors_b)) /
                            test_set5$SALE.PRICE)
pred25_step_f <- sum(relative_change_f < 0.25) / nrow(test_set5)
pred25_step_b <- sum(relative_change_b < 0.25) / nrow(test_set5)
pred10_step_f <- sum(relative_change_f < 0.1) / nrow(test_set5)
pred10_step_b <- sum(relative_change_b < 0.1) / nrow(test_set5)

compare_frame[7,1] <- 'Forward Stepwise Regression'
compare_frame[8,1] <- 'Backward Stepwise Regression'
compare_frame[7,2] <- rmse_step_f
compare_frame[8,2] <- rmse_step_b
compare_frame[7,3] <- pred25_step_f
compare_frame[8,3] <- pred25_step_b
compare_frame[7,4] <- pred10_step_f
compare_frame[8,4] <- pred10_step_b
compare_frame[7,5] <- summary(step_f_model)$r.squared
compare_frame[8,5] <- summary(step_b_model)$r.squared

compare_frame
```

```
##                     model_name     RMSE    Pred25     Pred10    rSquare
## 1              Land Area Model 34735740 0.1522989 0.05172414 0.19462134
## 2          Has Land Area Model 10661945 0.1363234 0.05429330 0.02790157
## 3        Land Model with Zeros 12005358 0.1498967 0.05724402 0.13800154
## 4         Multivariate Model 1 24087338 0.1927458 0.08183453 0.55915637
## 5        k-Fold on Multi-Model 13548759 0.1778183 0.07314099 0.43085184
## 6                k-Fold Model 13867375 0.2473619 0.09850164 0.53822628
## 7  Forward Stepwise Regression  8535350 0.1732614 0.08123501 0.74655022
## 8 Backward Stepwise Regression  8535091 0.1726619 0.08093525 0.74655634
```

What we see is that, with step-wise regression, we get our best RMSE values to date - at appx $8.5 million in each case, but with a lower pred(25) and pred(10) value than we saw in our most complicated model (the second one tested under k-means validation).

In terms of the original question - identifying the 5 top attributes - that's basically impossible given the fact that this regression uses multi-level factor variables.

In regressing on multi-level factors, R automatically creates a number of temporary binary variables (1 less than the number of factor levels) and regresses on each of these. Were we to look at the p-values from the

summary of the models developed, we have 189 coefficients in the case of the forward step-wise regression, and 189 in the case of the backward step-wise regression. I'll save you having to look at this.

However, in looking at the regression equations generated (see above), the variables that appear in both the forward and backward step-wise regressions are:

- BUILDING.CLASS.AT.PRESENT
- ZIP.CODE
- GROSS.SQUARE.FEET
- LAND.SQUARE.FEET
- HAS.LAND.AREA
- HAS.GROSS.AREA
- SALE.DATE

Given that HAS.LAND.AREA is a transformation of LAND.SQUARE.FEET, and that HAS.GROSS.AREA is a transformation of GROSS.SQUARE.FEET, we should likely eliminate those two variables, leaving us with these as our likely top 5:

- BUILDING.CLASS.AT.PRESENT
- ZIP.CODE
- GROSS.SQUARE.FEET
- LAND.SQUARE.FEET
- SALE.DATE

The most surprising here is SALE.DATA, as our visualization did not seem to imply much of a relationship between this variable and sale price.

In an ideal world, we'd go back and look at our models to see if this variable could help improve performance. But this paper is probably already way longer than you wanted it to be, so I'll spare you that step.

Conclusions

The most surprising thing about all this data is just how poor a predictive model we were able to build. The best performing model was in fact the one that looked at interactions between building class and neighborhood, lending credence to the "location, location, location" axiom about real estate values. But even here, we only got about 25% of our predictions within 25% of the actual value. Hardly anything you'd want to hand to a real estate agent in trying to determine price for a property.

Location Location Location

One of the problems with our analysis is that Neighborhood and Zip Code are both rather crude measures of location. The best value would likely be block, but this gets to a very granular level, and might leave us with many orphaned values. There are nearly 3000 blocks in the borough of Manhattan. Given the limited number of sales in a given year, we could not hope to have enough data to use this information to build a predictive model with that level of granularity.

A better solution might be to pre-process the data using a geographical clustering algorithm to create a new variable that better indicates the location of properties. My guess is that this, in combination with some of the other variables we've looked at would provide significantly better performance. But that is a bit outside the scope of this project.

Outlier values & data transformation

The other problem with this analysis was just the huge range of sale prices for Manhattan properties. With a maximum sales price of $1,300,000,000 and a mean of *only* $2,900,000, we are looking at a vast range. The fact that we had to use a log10 scale on almost all the plots shows this.

A possible solution would be to do a log transformation on the data before running the regressions. However, I do not have a strong understanding of the implications, nor of best practices in carrying out such a transformation of the data, and have therefore opted not to use this approach in this assignment.