Project Test Coding

Daniel Scott

2025-10-15

## R Markdown

Question (Q): What is the problem we’re investigating? We’re investigating how the size of properties in New York is predicted by various variables. These variables are: which neighborhood the property is in, the tax class, the building classification at the time of sale, the zip code of the property, the month the property was sold in, the number of residential and commercial units, the sales price, and the year the property was built.

Q: Are we focused on making predictions or are we trying to describe the relationship between multiple variables? We’re focused on helping people with budgets in mind understand what size of property they can buy. So I would say we’re worried about making predictions.

Q: What kind of statistical learning are we performing? We want to perform Predictive Statistical learning. We’re more worried about accuracy than interpretability. We want to lower our bias and increase our variance.

Q: What sort of analyses do we intend to perform? We’re doing a GLM to help us predict how our variables impact the Gross Square Feet of units.

# We're 1st going to import our dataset. The data set is called "nyc-rolling-sales\_initialcleaning(1)".  
DATA <- read.csv("C:\\Users\\Daniel Scott\\Desktop\\College\\Fall 2025\\Statistical Learning\\Ordered NYC Data.csv")  
  
# Next, we generate a summary. We're also checking how many observations we have and how many variables.   
summary(DATA)

## BOROUGH NEIGHBORHOOD BUILDING.CLASS.CATEGORY ZIP.CODE   
## Min. :1.000 Length:10079 Length:10079 Min. :10001   
## 1st Qu.:4.000 Class :character Class :character 1st Qu.:10305   
## Median :4.000 Mode :character Mode :character Median :10312   
## Mean :4.202 Mean :10739   
## 3rd Qu.:5.000 3rd Qu.:11419   
## Max. :5.000 Max. :11694   
## RESIDENTIAL.UNITS COMMERCIAL.UNITS GROSS.SQUARE.FEET YEAR.BUILT   
## Min. : 0.000 Min. : 0.000 Min. : 200 Min. :1800   
## 1st Qu.: 1.000 1st Qu.: 0.000 1st Qu.: 1240 1st Qu.:1920   
## Median : 1.000 Median : 0.000 Median : 1600 Median :1944   
## Mean : 3.291 Mean : 0.303 Mean : 5635 Mean :1950   
## 3rd Qu.: 2.000 3rd Qu.: 0.000 3rd Qu.: 2280 3rd Qu.:1980   
## Max. :894.000 Max. :318.000 Max. :1617206 Max. :2017   
## TAX.CLASS.AT.TIME.OF.SALE BUILDING.CLASS.AT.TIME.OF.SALE SALE.PRICE   
## Min. :1.000 Length:10079 Min. :1.000e+03   
## 1st Qu.:1.000 Class :character 1st Qu.:3.836e+05   
## Median :1.000 Mode :character Median :5.060e+05   
## Mean :1.213 Mean :2.394e+06   
## 3rd Qu.:1.000 3rd Qu.:7.100e+05   
## Max. :4.000 Max. :2.210e+09

dim(DATA)

## [1] 10079 11

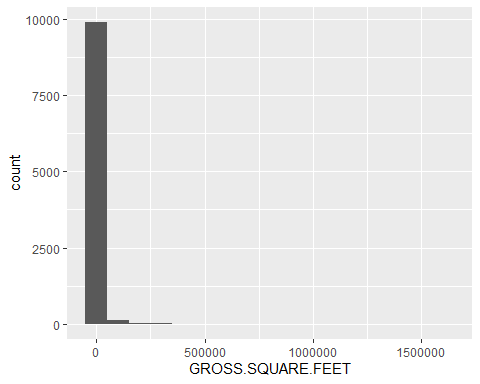
names(DATA)

## [1] "BOROUGH" "NEIGHBORHOOD"   
## [3] "BUILDING.CLASS.CATEGORY" "ZIP.CODE"   
## [5] "RESIDENTIAL.UNITS" "COMMERCIAL.UNITS"   
## [7] "GROSS.SQUARE.FEET" "YEAR.BUILT"   
## [9] "TAX.CLASS.AT.TIME.OF.SALE" "BUILDING.CLASS.AT.TIME.OF.SALE"  
## [11] "SALE.PRICE"

# So we have 10,079 observations, and 11 variables. The names() command tells you the names.  
  
# Now, we want to look at what our target variable looks like. Our target variable = Gross.Square.Feet  
summary(DATA$GROSS.SQUARE.FEET)

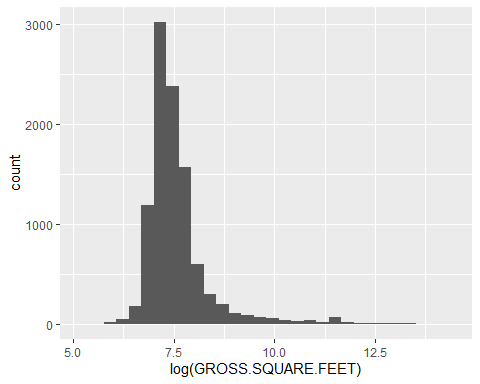
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 200 1240 1600 5635 2280 1617206

library(ggplot2)  
ggplot(DATA, aes(x = GROSS.SQUARE.FEET)) + geom\_histogram(binwidth = 100000)



# So it looks like there may be some outliers. Clearly, something is up because these bins are really really large.The range of our data is very very very large which means it may be tough to graphically look at our data.  
  
# By looking at our mean and median, we see that the mean is MUCH larger than our median. This indicates a positively skewed data set. Since no values of our target variable are 0, we can use a logarithmic transformation to try and make our variable more normal. Let's see how that looks now.  
ggplot(DATA, aes(x = log(GROSS.SQUARE.FEET))) + geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value `binwidth`.



# Alright. This looks waaaayyy better. It still looks positively skewed but we're doing much better.  
  
# Now, let's look at our correlation matrix. Our numerical predictors are in columns: five, six, eight and 11. Our target variable is in column 7  
cor.matrix <- cor(DATA[, c(5, 6, 8, 11, 7 )])  
round(cor.matrix, digits = 3)

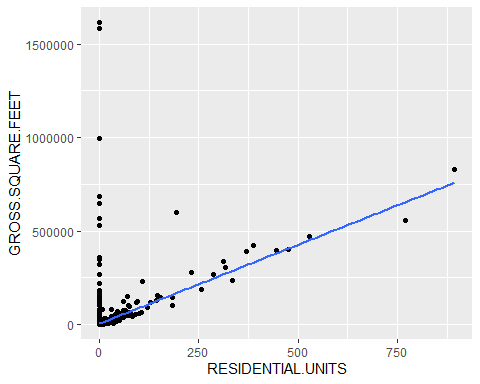
## RESIDENTIAL.UNITS COMMERCIAL.UNITS YEAR.BUILT SALE.PRICE  
## RESIDENTIAL.UNITS 1.000 0.025 -0.032 0.164  
## COMMERCIAL.UNITS 0.025 1.000 0.004 0.204  
## YEAR.BUILT -0.032 0.004 1.000 -0.015  
## SALE.PRICE 0.164 0.204 -0.015 1.000  
## GROSS.SQUARE.FEET 0.508 0.269 0.027 0.735  
## GROSS.SQUARE.FEET  
## RESIDENTIAL.UNITS 0.508  
## COMMERCIAL.UNITS 0.269  
## YEAR.BUILT 0.027  
## SALE.PRICE 0.735  
## GROSS.SQUARE.FEET 1.000

# The correlations for our numeric predictors are not very high, which is what we want. We don't want to have to deal with high collinearities, if I am remembering properly.  
# Notice that sale price is strongly correlated with gross square feet, and residential units is also strongly correlated as well.

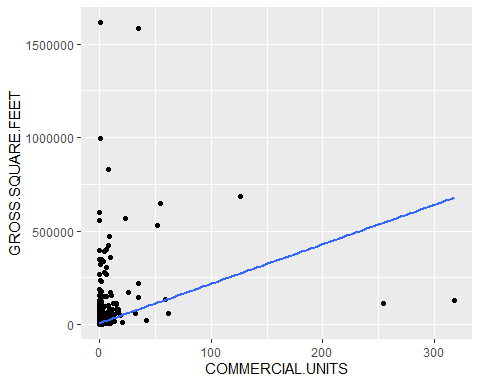
We’re now moving onto visualizing the relationships between our numeric predictors and gross square feet.

# First we're saving the names of the numeric predictors (N.P.) as a vector. Then we're plotting each N.P. against Gross.Square.Feet.  
vars.numeric <- colnames(DATA[, c(5, 6, 8, 11)])  
for (i in vars.numeric){  
plot <- ggplot(DATA, aes(x = DATA[, i], y = GROSS.SQUARE.FEET)) +  
geom\_point() +  
geom\_smooth(method = "lm", se = FALSE) +  
labs(x = i)  
print(plot)  
}

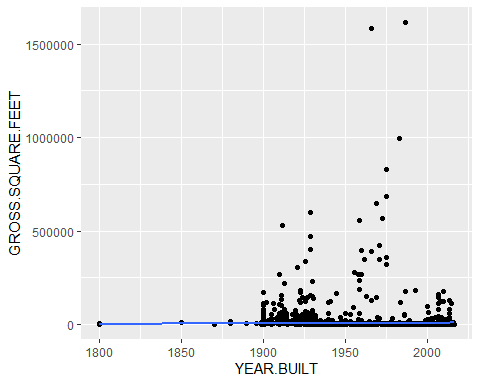
## `geom\_smooth()` using formula = 'y ~ x'



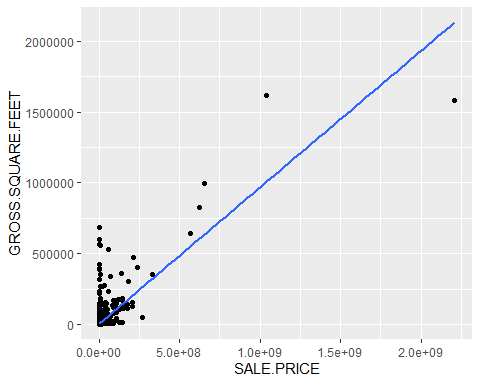
## `geom\_smooth()` using formula = 'y ~ x'



## `geom\_smooth()` using formula = 'y ~ x'



## `geom\_smooth()` using formula = 'y ~ x'



# We see that all of the N.P.'s, except year built have a positive correlation with gross square feet.  
# We see that year built has almost a 0 correlation with gross square feet. So we'll keep that in the back of our mind.

Now we are moving onto exploring our categorical variables. We have six categorical variables: borough, neighborhood, building class category, zip code, tax class, and building class at time of sale.

We’re going to calculate the mean and median of our target variable (TV), which is Gross Square Feet. This will be split by the different levels of each of our categorical variables. First we need to install tidyverse. #{r} #install.packages("tidyverse") # Now that it is installed, we can move onto actually using it.

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.1 ✔ stringr 1.5.2  
## ✔ lubridate 1.9.4 ✔ tibble 3.3.0  
## ✔ purrr 1.1.0 ✔ tidyr 1.3.1  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

# Now, we're going to save the names of our categorical predictors (CP) as a vector. The column numbers that correspond to our CP's are one through four, then nine and ten.  
# Then our function will go ahead and generate the mean and median of Gross Square Feet (GSF) for each level of each CP.  
vars.categorical <- colnames(DATA[, c(1:4, 9:10)])  
for (i in vars.categorical) {  
x <- DATA %>%  
group\_by\_(i) %>%  
summarise(  
mean = mean(GROSS.SQUARE.FEET),  
median = median(GROSS.SQUARE.FEET),  
n = n()  
)  
print(x)  
}

## Warning: `group\_by\_()` was deprecated in dplyr 0.7.0.  
## ℹ Please use `group\_by()` instead.  
## ℹ See vignette('programming') for more help  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## # A tibble: 4 × 4  
## BOROUGH mean median n  
## <int> <dbl> <dbl> <int>  
## 1 1 37644. 8065 965  
## 2 2 1652 1587 5  
## 3 4 2119. 1504 4168  
## 4 5 2354. 1544 4941

## Warning: `group\_by\_()` was deprecated in dplyr 0.7.0.  
## ℹ Please use `group\_by()` instead.  
## ℹ See vignette('programming') for more help  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## # A tibble: 111 × 4  
## NEIGHBORHOOD mean median n  
## <chr> <dbl> <dbl> <int>  
## 1 ALPHABET CITY 10205. 6024 12  
## 2 ANNADALE 2101. 1848 103  
## 3 ARDEN HEIGHTS 1532. 1378 223  
## 4 ARROCHAR 2113. 1766 24  
## 5 ARROCHAR-SHORE ACRES 1521. 1543 14  
## 6 BATHGATE 1652 1587 5  
## 7 BULLS HEAD 1758. 1595 213  
## 8 CASTLETON CORNERS 1620. 1367 99  
## 9 CHELSEA 21020. 6330 38  
## 10 CHINATOWN 11371. 8550 9  
## # ℹ 101 more rows

## Warning: `group\_by\_()` was deprecated in dplyr 0.7.0.  
## ℹ Please use `group\_by()` instead.  
## ℹ See vignette('programming') for more help  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## # A tibble: 27 × 4  
## BUILDING.CLASS.CATEGORY mean median n  
## <chr> <dbl> <dbl> <int>  
## 1 "01 ONE FAMILY DWELLINGS " 1498. 1326 5733  
## 2 "02 TWO FAMILY DWELLINGS " 2047. 1960 2906  
## 3 "03 THREE FAMILY DWELLINGS " 2719. 2643 260  
## 4 "05 TAX CLASS 1 VACANT LAND " 2451. 2292 7  
## 5 "06 TAX CLASS 1 - OTHER " 956. 642 4  
## 6 "07 RENTALS - WALKUP APARTMENTS " 8892. 6040 499  
## 7 "08 RENTALS - ELEVATOR APARTMENTS " 93304. 43896 98  
## 8 "09 COOPS - WALKUP APARTMENTS " 16389. 12200 5  
## 9 "10 COOPS - ELEVATOR APARTMENTS " 221933. 139270. 12  
## 10 "11A CONDO-RENTALS " 54496. 37450 6  
## # ℹ 17 more rows

## Warning: `group\_by\_()` was deprecated in dplyr 0.7.0.  
## ℹ Please use `group\_by()` instead.  
## ℹ See vignette('programming') for more help  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## # A tibble: 84 × 4  
## ZIP.CODE mean median n  
## <int> <dbl> <dbl> <int>  
## 1 10001 119004. 62457 22  
## 2 10002 13129. 10230 29  
## 3 10003 19896. 6932 33  
## 4 10004 568650. 568650. 2  
## 5 10005 601447. 400531 5  
## 6 10006 130054 130054 1  
## 7 10007 15362. 15362. 2  
## 8 10009 10937. 6440 17  
## 9 10010 44362. 14013 10  
## 10 10011 16753. 5955 45  
## # ℹ 74 more rows

## Warning: `group\_by\_()` was deprecated in dplyr 0.7.0.  
## ℹ Please use `group\_by()` instead.  
## ℹ See vignette('programming') for more help  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## # A tibble: 3 × 4  
## TAX.CLASS.AT.TIME.OF.SALE mean median n  
## <int> <dbl> <dbl> <int>  
## 1 1 1713. 1512 8910  
## 2 2 24873. 6937 682  
## 3 4 50460. 7000 487

## Warning: `group\_by\_()` was deprecated in dplyr 0.7.0.  
## ℹ Please use `group\_by()` instead.  
## ℹ See vignette('programming') for more help  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## # A tibble: 104 × 4  
## BUILDING.CLASS.AT.TIME.OF.SALE mean median n  
## <chr> <dbl> <dbl> <int>  
## 1 A0 1199. 1162. 94  
## 2 A1 1542. 1385 2276  
## 3 A2 1221. 1100 880  
## 4 A3 3702. 3556. 92  
## 5 A4 4867. 4122 47  
## 6 A5 1404. 1352 1923  
## 7 A6 699. 640 61  
## 8 A7 6955 5900 3  
## 9 A9 1404. 1292 305  
## 10 B1 2262. 2082 321  
## # ℹ 94 more rows

# A couple of notes:  
# Most of our CP's have some levels with small n's that we may want to combine to reduce dimensionality.  
# There are some levels that seem to make a large difference in the mean and median of GSF.  
  
# So we're going to need to consider the 5% rule, which states that each factor level should contain at least 5% of the total observations. For our data set, n = 10,079. 0.05\*10,079 = 503.95, which we will round up to 504. So each factor level must contain at least 504 observations in it.  
# Let's start doing that here.  
  
# We're just going to combine Borough two, which is the Bronx with Borough four, which is Queens because they both border the East River in NYC.We're checking that everything looks good with the factor count function.  
library(forcats)  
F\_Borough <- factor(x = DATA$BOROUGH)  
fct\_count(F\_Borough)

## # A tibble: 4 × 2  
## f n  
## <fct> <int>  
## 1 1 965  
## 2 2 5  
## 3 4 4168  
## 4 5 4941

# We've created a factor for Borough. Now we want to combine levels 2 and 4. Then we'll make sure it worked with the fct\_count function again.  
F\_Borough\_Comb <- fct\_collapse(F\_Borough, "2+4"=c(2, 4))  
fct\_count(F\_Borough\_Comb, prop = TRUE)

## # A tibble: 3 × 3  
## f n p  
## <fct> <int> <dbl>  
## 1 1 965 0.0957  
## 2 2+4 4173 0.414   
## 3 5 4941 0.490

# So we've created a new factor called "Factor Borough Combined" where we combined the Bronx and Queens. Now we just need to add it to our data set.  
DATA$BOROUGH.COMBINED <- F\_Borough\_Comb  
  
  
# For our next variable "Neighborhood", we're going to want to combine many factor levels together. This will be tricky. I am thinking that we will need to use some logic to combine our factor levels together. Probably we will do it based on keep adding factor levels together until the combined factor level has at least 504 observations in it. Time to figure out how to do this.

Ok so we need to condense the neighborhoods. Let’s do it in a systematic manner. I am referring to <https://locality.nyc> for a map of New York City. I am proposing we condense these levels into clusters of neighborhoods that are geographically close to one another. The first step is to choose a starting point. I am starting with the neighborhoods in Manhattan.

# First we need to load the forcats library.  
#library(forcats)  
#F\_Neighborhood <- factor(x = DATA$NEIGHBORHOOD)  
# Next, we are going to have a count of each ordered Neighborhood.  
#Neigh\_Count <- fct\_count(DATA$NEIGHBORHOOD)  
#print(Neigh\_Count)  
  
# I am first going to sort the neighborhoods in order they appear in our data set. This is defined as "Ordered\_Neighborhood."  
#Ordered\_Neighborhood <- fct\_inorder(DATA$NEIGHBORHOOD)  
#print(Ordered\_Neighborhood)  
  
# Let's check what the factor count of ordered Neighborhood looks like.  
#fct\_count(Ordered\_Neighborhood)  
  
# Ok this is what we wanted.  
# Let's try to now combine this factor NOT based on geographic location but based on Zip Code. This will effectively allow us to include zip code's information and have some sembelence related to geographic information included. Note: Zip Codes are much more about post offices and less to do with geographic information necessarily.  
  
#fct\_reorder2(F\_Neighborhood, DATA$NEIGHBORHOOD, DATA$ZIP.CODE)  
  
#F\_ZipCode <- factor(DATA$ZIP.CODE)  
#fct\_count(F\_ZipCode)  
  
#We're going to come back to this later. We are going to skip these high dimensionality categorical variables and instead just run what we can at the moment.

Ok, so we’re skipping the neighborhood variable ATM for the sake of time. We’re also going to be skipping over: building class category, zip code, building class at time of sale. This means we have the variables: residential units, commerical units, year, sales price, combined borough, and tax class at time of sale.

This means we need to do some work on tax class at time of sale.

# First, let's look at Tax Class At Time of Sale.  
Tax\_Clss\_Sale <- factor(x = DATA$TAX.CLASS.AT.TIME.OF.SALE)  
fct\_count(Tax\_Clss\_Sale)

## # A tibble: 3 × 2  
## f n  
## <fct> <int>  
## 1 1 8910  
## 2 2 682  
## 3 4 487

# We see that this gives us one level of tax class at time of sale that is under 504. This is a little bit worrisome as the information included may potentially mess with whatever factor level it is included in. Let's combine it with factor level 2, as they have a similar median. We will do this by using the  
Tax\_Clss\_Sale\_Comb <- fct\_collapse(Tax\_Clss\_Sale, "2+4" = c(2, 4) )  
fct\_count(f = Tax\_Clss\_Sale\_Comb)

## # A tibble: 2 × 2  
## f n  
## <fct> <int>  
## 1 1 8910  
## 2 2+4 1169

DATA$COMBINED.TAX.CLASS.SALE <-Tax\_Clss\_Sale\_Comb  
# Now I want to see how this affected the mean and median of the different groups.  
library(tidyverse)  
  
# This first function will show us the mean and median of the Tax Class Variable.  
by\_Tax\_Clss\_Sale <- DATA %>%   
group\_by(TAX.CLASS.AT.TIME.OF.SALE) %>%  
summarise(  
 mean = mean(GROSS.SQUARE.FEET),  
 median = median(GROSS.SQUARE.FEET)  
)  
print(by\_Tax\_Clss\_Sale)

## # A tibble: 3 × 3  
## TAX.CLASS.AT.TIME.OF.SALE mean median  
## <int> <dbl> <dbl>  
## 1 1 1713. 1512  
## 2 2 24873. 6937  
## 3 4 50460. 7000

# This function will show us the mean and median of the Combined Tax Class Variable.  
by\_Tax\_Clss\_Sale\_Combined <- DATA %>%   
group\_by(COMBINED.TAX.CLASS.SALE) %>%  
summarise(  
 mean = mean(GROSS.SQUARE.FEET),  
 median = median(GROSS.SQUARE.FEET)  
)  
print(by\_Tax\_Clss\_Sale\_Combined)

## # A tibble: 2 × 3  
## COMBINED.TAX.CLASS.SALE mean median  
## <fct> <dbl> <dbl>  
## 1 1 1713. 1512  
## 2 2+4 35532. 7000

# We see that the mean changes for both groups significantly but the median remains the same!  
# So we've combined levels 2 and 4 of tax class sale and have put it into our data set.

Now, we have our variables ready for some analyses. We’re going to rerun one of our codes from earlier to help us assess if our, now two, categorical variables will have an impact on GSF.

library(tidyverse)  
# First, we're going to save the names of our categorical predictors (CP) as a vector. The column numbers that correspond to our CP's are column 12 and column 13.  
# Then our function will go ahead and generate the mean and median of Gross Square Feet (GSF) for each level of each CP.  
vars.categorical2 <- c("COMBINED.TAX.CLASS.SALE", "BOROUGH.COMBINED")  
for (i in vars.categorical2) {  
x2 <- DATA %>%  
group\_by\_(i) %>%  
summarise(  
mean = mean(GROSS.SQUARE.FEET),  
median = median(GROSS.SQUARE.FEET),  
n = n()  
)  
print(x2)  
}

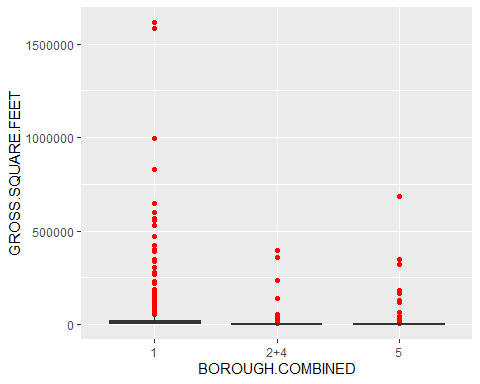
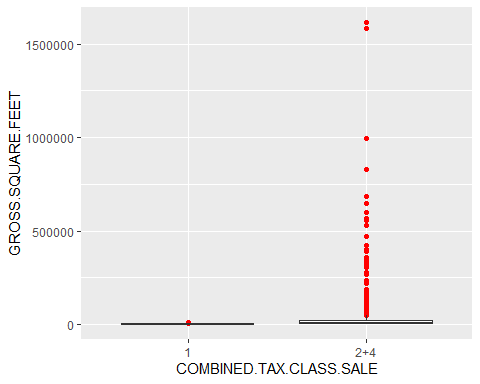
## Warning: `group\_by\_()` was deprecated in dplyr 0.7.0.  
## ℹ Please use `group\_by()` instead.  
## ℹ See vignette('programming') for more help  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## # A tibble: 2 × 4  
## COMBINED.TAX.CLASS.SALE mean median n  
## <fct> <dbl> <dbl> <int>  
## 1 1 1713. 1512 8910  
## 2 2+4 35532. 7000 1169

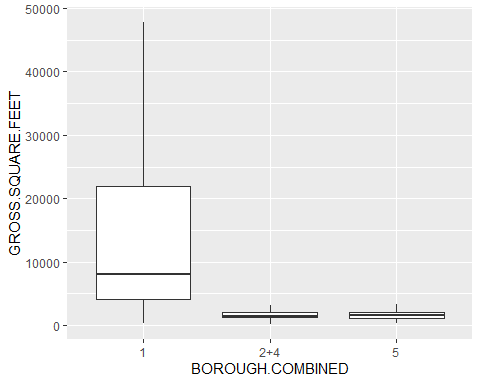
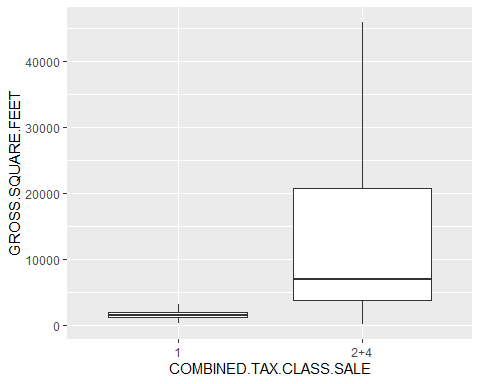
## Warning: `group\_by\_()` was deprecated in dplyr 0.7.0.  
## ℹ Please use `group\_by()` instead.  
## ℹ See vignette('programming') for more help  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## # A tibble: 3 × 4  
## BOROUGH.COMBINED mean median n  
## <fct> <dbl> <int> <int>  
## 1 1 37644. 8065 965  
## 2 2+4 2119. 1504 4173  
## 3 5 2354. 1544 4941

# We see a couple of things. 1st: Combined Tax Class Sale seems to have a pretty sizeable impact on GSF. 2nd: Borough Combined also seems to have a noticeable impact on GSF.  
  
# Let's look at some graphical displays as well.  
for (i in vars.categorical2){  
 cat.plot <- ggplot(DATA, aes(x = DATA[, i], y = GROSS.SQUARE.FEET)) +  
 geom\_boxplot(outlier.color = "red") +  
 labs(x = i)  
 print(cat.plot)  
}



# HMMM.... This is making me think that there are definetly some outliers in our data set. Let's see what happens if we remove these outliers.  
for (i in vars.categorical2){  
 cat.plot <- ggplot(DATA, aes(x = DATA[, i], y = GROSS.SQUARE.FEET)) +  
 geom\_boxplot(outliers = FALSE) +  
 labs(x = i)  
 print(cat.plot)  
}



# Clearly there's some effect that both of these are having on GSF.

We need a way to address our outliers.

# Calculate the quantiles and the IQR based on GSF.  
DATA$logGROSS.SQUARE.FEET <- log(DATA$GROSS.SQUARE.FEET)  
Q1 <- quantile(DATA$logGROSS.SQUARE.FEET, 0.25)  
Q3 <- quantile(DATA$logGROSS.SQUARE.FEET, 0.75)  
IQR <- Q3 - Q1  
print("Q1")

## [1] "Q1"

print(Q1)

## 25%   
## 7.122867

print("Q3")

## [1] "Q3"

print(Q3)

## 75%   
## 7.73215

print("IQR")

## [1] "IQR"

print(IQR)

## 75%   
## 0.6092833

# Now we will define the bounds of our data. If anything falls outside these bounds, we will remove them for being outliers.  
lower\_bound <- Q1 - 1.5 \* IQR  
print("lowerbound")

## [1] "lowerbound"

print(lower\_bound)

## 25%   
## 6.208942

upper\_bound <- Q3 + 1.5 \* IQR  
print(upper\_bound)

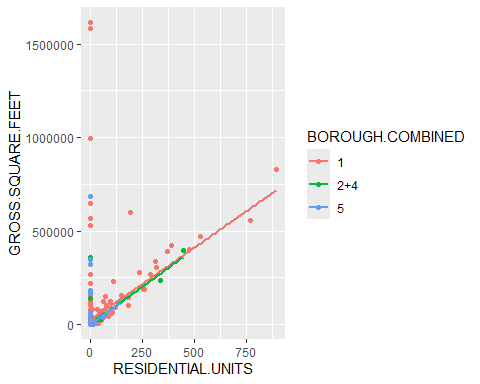
## 75%   
## 8.646075

# Now, we will identify the outliers. We're going to omit the lower bound because we can't have negative gross square feet.  
outliers <- which(DATA$logGROSS.SQUARE.FEET < lower\_bound|DATA$logGROSS.SQUARE.FEET > upper\_bound)  
  
#Using a different method to identify outliers  
#Box\_1 <- boxplot.stats(DATA$GROSS.SQUARE.FEET)  
#Box\_1  
#Testing\_Box <- boxplot.stats(DATA$GROSS.SQUARE.FEET)$out  
#print("Outliers Boxplot stats")  
#Testing\_Box  
  
#We've identified 24 outliers, now we need to remove them. Fixed\_Data will refer to our data without these outliers in them.  
Fixed\_DATA <- DATA[-outliers, ]

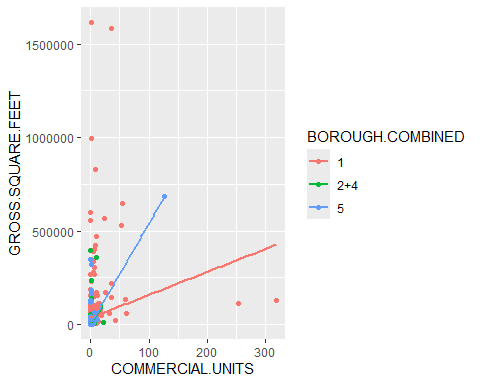
Let’s make some scatterplots for our numerical variables.

# Similar Process to what we were doing above except with a scatterplot function.  
# We're now going to be trying to see if we can graphically detect any interaction effects.  
# Recall:vars.numeric <- colnames(DATA[, c(5, 6, 8, 11)])   
  
for (i in vars.numeric){  
plot <- ggplot(DATA, aes(x = DATA[, i], y = GROSS.SQUARE.FEET, color = BOROUGH.COMBINED)) +  
geom\_point() +  
geom\_smooth(method = "lm", se = FALSE) +  
labs(x = i)  
print(plot)  
}

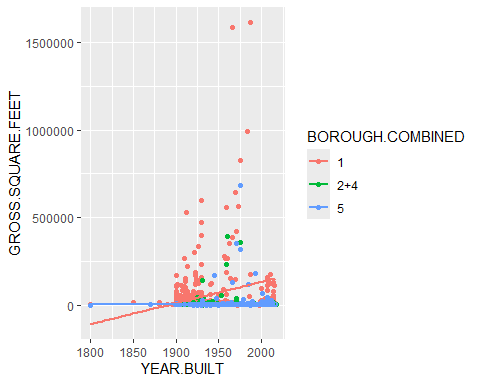
## `geom\_smooth()` using formula = 'y ~ x'



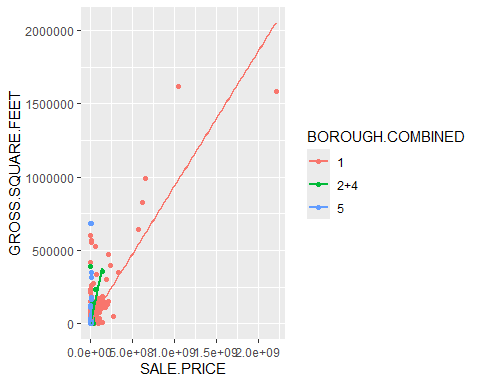
## `geom\_smooth()` using formula = 'y ~ x'



## `geom\_smooth()` using formula = 'y ~ x'



## `geom\_smooth()` using formula = 'y ~ x'



# Residential Units doesn't appear to interact with Borough  
# Commercial units may interact with Borough  
# Something odd is happening with year built... Particularly with Borough 1. Those outliers appear to be skewing things drastically.  
# Sale Price may interact with borough.  
# I think these provide more evidence of outliers causing problems in our data set.  
#We will have to address them.

Here we address the outliers. We are going to remove them for the time being.

Now we can finally, initially split our data into the test and training data sets.

#This is creating a test and training data split based on a 70% 30% data split. It is also reporting the mean and median of our training and testing data.  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

partition <- createDataPartition(DATA$GROSS.SQUARE.FEET, p = 0.7, list = FALSE)  
data.train = DATA[partition, ]  
data.test = DATA[-partition, ]  
print("data.train")

## [1] "data.train"

mean(data.train$GROSS.SQUARE.FEET)

## [1] 6101.014

median(data.train$GROSS.SQUARE.FEET)

## [1] 1600

print("data.test")

## [1] "data.test"

mean(data.test$GROSS.SQUARE.FEET)

## [1] 4547.814

median(data.test$GROSS.SQUARE.FEET)

## [1] 1600

# We're doing this to create a partition just to see what it would look like using the fixed data.  
library(caret)  
partition2 <- createDataPartition(Fixed\_DATA$GROSS.SQUARE.FEET, p = 0.7, list = FALSE)  
data.train\_F = Fixed\_DATA[partition2, ]  
data.test\_F = Fixed\_DATA[-partition2, ]  
print("data.train\_F")

## [1] "data.train\_F"

mean(data.train\_F$GROSS.SQUARE.FEET)

## [1] 1788.621

median(data.train\_F$GROSS.SQUARE.FEET)

## [1] 1550

print("data.test\_F")

## [1] "data.test\_F"

mean(data.test\_F$GROSS.SQUARE.FEET)

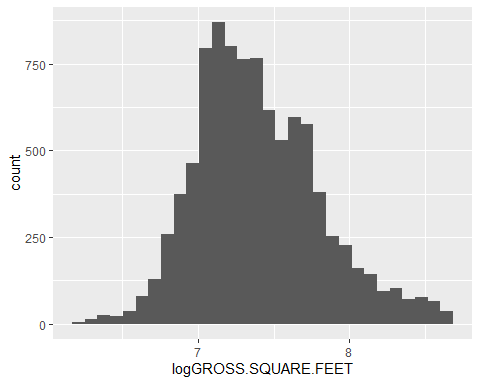
## [1] 1780.589

median(data.test\_F$GROSS.SQUARE.FEET)

## [1] 1550.5

#The means are much closer, so I would recommend using the second data set.  
  
#Just making a histogram to see what our logarithmic model look like with our fixed data.  
ggplot(Fixed\_DATA, aes(x = logGROSS.SQUARE.FEET)) + geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value `binwidth`.



#This looks much better.

Now we move onto to doing a Multiple Linear Regression.

# This is the linear regression for the full model.  
model.full <- lm(  
 logGROSS.SQUARE.FEET ~ BOROUGH.COMBINED + RESIDENTIAL.UNITS + COMMERCIAL.UNITS + YEAR.BUILT + SALE.PRICE + COMBINED.TAX.CLASS.SALE, data = Fixed\_DATA  
 )  
summary(model.full)

##   
## Call:  
## lm(formula = logGROSS.SQUARE.FEET ~ BOROUGH.COMBINED + RESIDENTIAL.UNITS +   
## COMMERCIAL.UNITS + YEAR.BUILT + SALE.PRICE + COMBINED.TAX.CLASS.SALE,   
## data = Fixed\_DATA)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.2740 -0.2145 -0.0204 0.2119 1.4788   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3.685e+00 2.413e-01 15.27 <2e-16 \*\*\*  
## BOROUGH.COMBINED2+4 -2.970e-01 2.383e-02 -12.46 <2e-16 \*\*\*  
## BOROUGH.COMBINED5 -2.945e-01 2.503e-02 -11.77 <2e-16 \*\*\*  
## RESIDENTIAL.UNITS 1.575e-01 4.226e-03 37.28 <2e-16 \*\*\*  
## COMMERCIAL.UNITS 2.851e-01 1.385e-02 20.59 <2e-16 \*\*\*  
## YEAR.BUILT 1.906e-03 1.257e-04 15.16 <2e-16 \*\*\*  
## SALE.PRICE 3.369e-08 2.660e-09 12.66 <2e-16 \*\*\*  
## COMBINED.TAX.CLASS.SALE2+4 3.465e-02 2.222e-02 1.56 0.119   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3356 on 9329 degrees of freedom  
## Multiple R-squared: 0.3328, Adjusted R-squared: 0.3323   
## F-statistic: 664.7 on 7 and 9329 DF, p-value: < 2.2e-16

Here we see that the only non-significant predictor is the tax class at time of sale. Year built doesn’t seem to be adding much useful information to our data though.

round(confint(model.full, level = 0.95), 2)

## 2.5 % 97.5 %  
## (Intercept) 3.21 4.16  
## BOROUGH.COMBINED2+4 -0.34 -0.25  
## BOROUGH.COMBINED5 -0.34 -0.25  
## RESIDENTIAL.UNITS 0.15 0.17  
## COMMERCIAL.UNITS 0.26 0.31  
## YEAR.BUILT 0.00 0.00  
## SALE.PRICE 0.00 0.00  
## COMBINED.TAX.CLASS.SALE2+4 -0.01 0.08

Now we need to relevel our factors and binarize.

for (i in vars.categorical2){  
# Use the table() function to calculate the frequencies for each factor  
table <- as.data.frame(table(Fixed\_DATA[, i]))  
# Determine the level with the highest frequency  
max <- which.max(table[, 2])  
# Save the name of the level with the highest frequency  
level.name <- as.character(table[max, 1])  
# Set the baseline level to the most populous level  
Fixed\_DATA[, i] <- relevel(Fixed\_DATA[, i], ref = level.name)  
}  
print("Dimension of the Fixed Dataset")

## [1] "Dimension of the Fixed Dataset"

dim(Fixed\_DATA)

## [1] 9337 14

print("Categorical Summary")

## [1] "Categorical Summary"

summary(Fixed\_DATA[, vars.categorical2])

## COMBINED.TAX.CLASS.SALE BOROUGH.COMBINED  
## 1 :8839 5 :4865   
## 2+4: 498 1 : 362   
## 2+4:4110

#Notice that Manhattan's Borough is now too low. We will have to address this later as it under 5% of the observations.

# To make sure factors in the training set are releveled  
data.train\_F <- Fixed\_DATA[partition, ]  
model.full <- lm(  
 logGROSS.SQUARE.FEET ~ BOROUGH.COMBINED + RESIDENTIAL.UNITS + COMMERCIAL.UNITS + YEAR.BUILT + SALE.PRICE + COMBINED.TAX.CLASS.SALE, data = Fixed\_DATA  
 )  
summary(model.full)

##   
## Call:  
## lm(formula = logGROSS.SQUARE.FEET ~ BOROUGH.COMBINED + RESIDENTIAL.UNITS +   
## COMMERCIAL.UNITS + YEAR.BUILT + SALE.PRICE + COMBINED.TAX.CLASS.SALE,   
## data = Fixed\_DATA)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.2740 -0.2145 -0.0204 0.2119 1.4788   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3.391e+00 2.476e-01 13.697 <2e-16 \*\*\*  
## BOROUGH.COMBINED1 2.945e-01 2.503e-02 11.765 <2e-16 \*\*\*  
## BOROUGH.COMBINED2+4 -2.444e-03 8.278e-03 -0.295 0.768   
## RESIDENTIAL.UNITS 1.575e-01 4.226e-03 37.282 <2e-16 \*\*\*  
## COMMERCIAL.UNITS 2.851e-01 1.385e-02 20.591 <2e-16 \*\*\*  
## YEAR.BUILT 1.906e-03 1.257e-04 15.158 <2e-16 \*\*\*  
## SALE.PRICE 3.369e-08 2.660e-09 12.663 <2e-16 \*\*\*  
## COMBINED.TAX.CLASS.SALE2+4 3.465e-02 2.222e-02 1.560 0.119   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3356 on 9329 degrees of freedom  
## Multiple R-squared: 0.3328, Adjusted R-squared: 0.3323   
## F-statistic: 664.7 on 7 and 9329 DF, p-value: < 2.2e-16

#Now we're binarizing these variables.  
library(caret)  
binarizer <- dummyVars((~ BOROUGH.COMBINED + COMBINED.TAX.CLASS.SALE),  
data = Fixed\_DATA, fullRank = TRUE)  
binarized\_vars <- data.frame(predict(binarizer, newdata = Fixed\_DATA))  
head(binarized\_vars)

## BOROUGH.COMBINED.1 BOROUGH.COMBINED.2.4 COMBINED.TAX.CLASS.SALE.2.4  
## 3 1 0 1  
## 4 1 0 1  
## 9 1 0 1  
## 10 1 0 1  
## 11 1 0 1  
## 12 1 0 1

Data.bin <- cbind()