NYC-Property-Sales.R

kpvp2

2020-05-20

```
# Name: Kishan Patel
# Project (New York Property Sales)
setwd("~/Masters/Data Science Methods/Project/nyc-property-sales")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(ggcorrplot)
```

```
## Warning: package 'ggcorrplot' was built under R version 3.6.3
```

```
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 3.6.3
```

```
## Loading required package: magrittr
```

```
library(stringr)
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.6.2
```

```
library(e1071)
library(stats)
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.6.3
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.6.2
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 3.0-1
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
# Import and Viewing Data
ny.sales <- read_csv('nyc-rolling-sales.csv', na = c("", "NA", "NaN"))
```

```
## Warning: Missing column names filled in: 'X1' [1]

## Parsed with column specification:
## cols(
##   .default = col_character(),
##   X1 = col_double(),
##   BOROUGH = col_double(),
##   BLOCK = col_double(),
##   LOT = col_double(),
##   `EASE-MENT` = col_logical(),
##   `ZIP CODE` = col_double(),
##   `RESIDENTIAL UNITS` = col_double(),
##   `COMMERCIAL UNITS` = col_double(),
##   `TOTAL UNITS` = col_double(),
##   `YEAR BUILT` = col_double(),
##   `TAX CLASS AT TIME OF SALE` = col_double(),
##   `SALE DATE` = col_datetime(format = "")
## )
```

```
## See spec(...) for full column specifications.
```

```
data.dim <- dim(ny.sales)
dim.output <- paste("The data has", data.dim[1], "rows and", data.dim[2],"columns", sep = " ")
print(dim.output)
```

```
## [1] "The data has 84548 rows and 22 columns"
```

```
head(ny.sales)
```

```
## # A tibble: 6 x 22
##      X1 BOROUGH NEIGHBORHOOD `BUILDING CLASS~ `TAX CLASS AT P~ BLOCK   LOT
##   <dbl>   <dbl> <chr>        <chr>            <chr>            <dbl> <dbl>
## 1     4       1 ALPHABET CI~ 07 RENTALS - WA~ 2A                 392     6
## 2     5       1 ALPHABET CI~ 07 RENTALS - WA~ 2                  399    26
## 3     6       1 ALPHABET CI~ 07 RENTALS - WA~ 2                  399    39
## 4     7       1 ALPHABET CI~ 07 RENTALS - WA~ 2B                 402    21
## 5     8       1 ALPHABET CI~ 07 RENTALS - WA~ 2A                 404    55
## 6     9       1 ALPHABET CI~ 07 RENTALS - WA~ 2                  405    16
## # ... with 15 more variables: `EASE-MENT` <lgl>, `BUILDING CLASS AT
## #   PRESENT` <chr>, ADDRESS <chr>, `APARTMENT NUMBER` <chr>, `ZIP
## #   CODE` <dbl>, `RESIDENTIAL UNITS` <dbl>, `COMMERCIAL UNITS` <dbl>,
## #   `TOTAL UNITS` <dbl>, `LAND SQUARE FEET` <chr>, `GROSS SQUARE
## #   FEET` <chr>, `YEAR BUILT` <dbl>, `TAX CLASS AT TIME OF SALE` <dbl>,
## #   `BUILDING CLASS AT TIME OF SALE` <chr>, `SALE PRICE` <chr>, `SALE
## #   DATE` <dttm>
```

```
# Viewing Column Names and Types
cnames <- colnames(ny.sales)
types <- sapply(ny.sales, class)
print(types)
```

```
## $X1
## [1] "numeric"
##
## $BOROUGH
## [1] "numeric"
##
## $NEIGHBORHOOD
## [1] "character"
##
## $`BUILDING CLASS CATEGORY`
## [1] "character"
##
## $`TAX CLASS AT PRESENT`
## [1] "character"
##
## $BLOCK
## [1] "numeric"
##
## $LOT
## [1] "numeric"
```

```
##
## $`EASE-MENT`
## [1] "logical"
##
## $`BUILDING CLASS AT PRESENT`
## [1] "character"
##
## $ADDRESS
## [1] "character"
##
## $`APARTMENT NUMBER`
## [1] "character"
##
## $`ZIP CODE`
## [1] "numeric"
##
## $`RESIDENTIAL UNITS`
## [1] "numeric"
##
## $`COMMERCIAL UNITS`
## [1] "numeric"
##
## $`TOTAL UNITS`
## [1] "numeric"
##
## $`LAND SQUARE FEET`
## [1] "character"
##
## $`GROSS SQUARE FEET`
## [1] "character"
##
## $`YEAR BUILT`
## [1] "numeric"
##
## $`TAX CLASS AT TIME OF SALE`
## [1] "numeric"
##
## $`BUILDING CLASS AT TIME OF SALE`
## [1] "character"
##
## $`SALE PRICE`
## [1] "character"
##
## $`SALE DATE`
## [1] "POSIXct" "POSIXt"
```

```r
# Column Names to lowercase
colnames(ny.sales) <- tolower(colnames(ny.sales))
colnames(ny.sales) <- str_replace_all(colnames(ny.sales),'\\s', '_')

cnames <- colnames(ny.sales)
# Drop Unneeded Columns
nysales <- select(ny.sales, -c(x1,'apartment_number', 'address', 'ease-ment'))
obs.pre.drop <- nrow(nysales)
```

```r
nysales <- unique(nysales)
obs.post.rop <- nrow(nysales)
drops <- paste(obs.pre.drop - obs.post.rop, "rows dropped.", sep = " ")
print(drops)
```

```
## [1] "956 rows dropped."
```

```r
# Converting DataFrame Columns
nysales[,'sale_price'] <- as.numeric(nysales$sale_price)
```

```
## Warning: NAs introduced by coercion
```

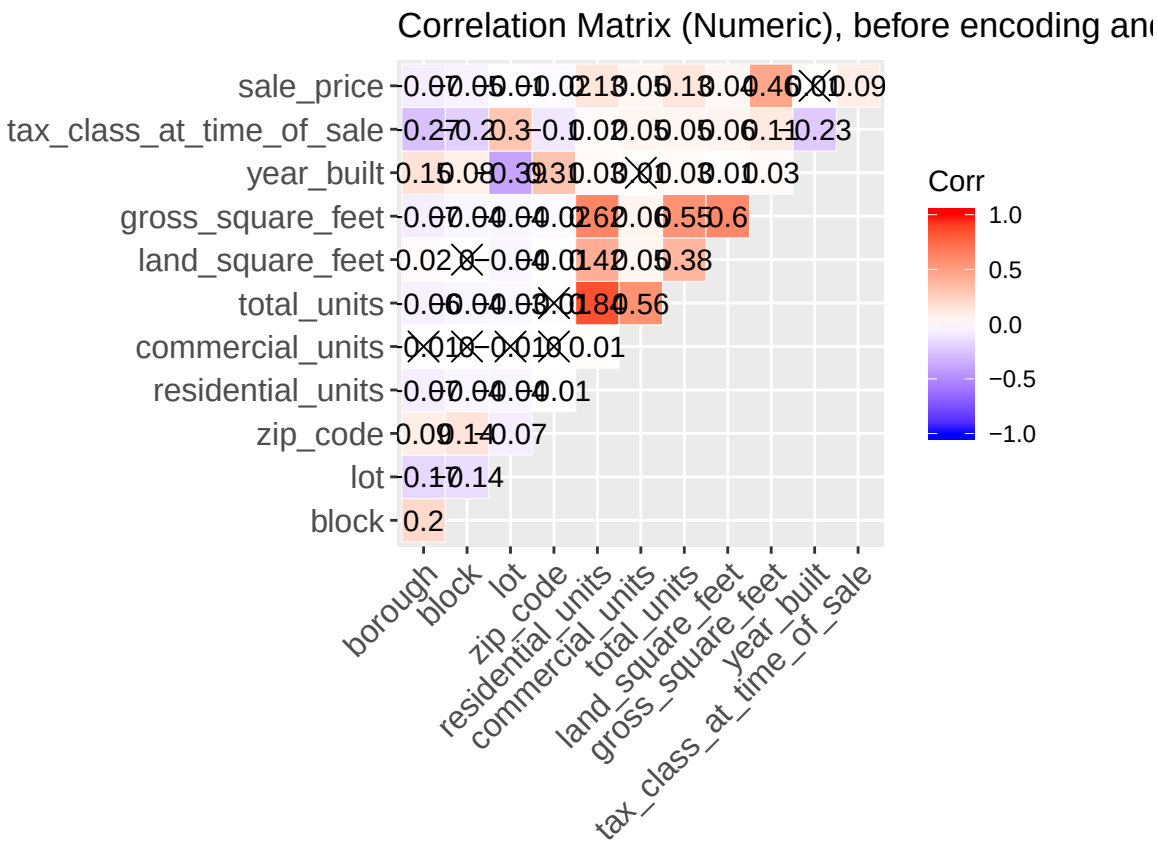
```r
nysales[,'land_square_feet'] <- as.numeric(nysales$land_square_feet)
```

```
## Warning: NAs introduced by coercion
```

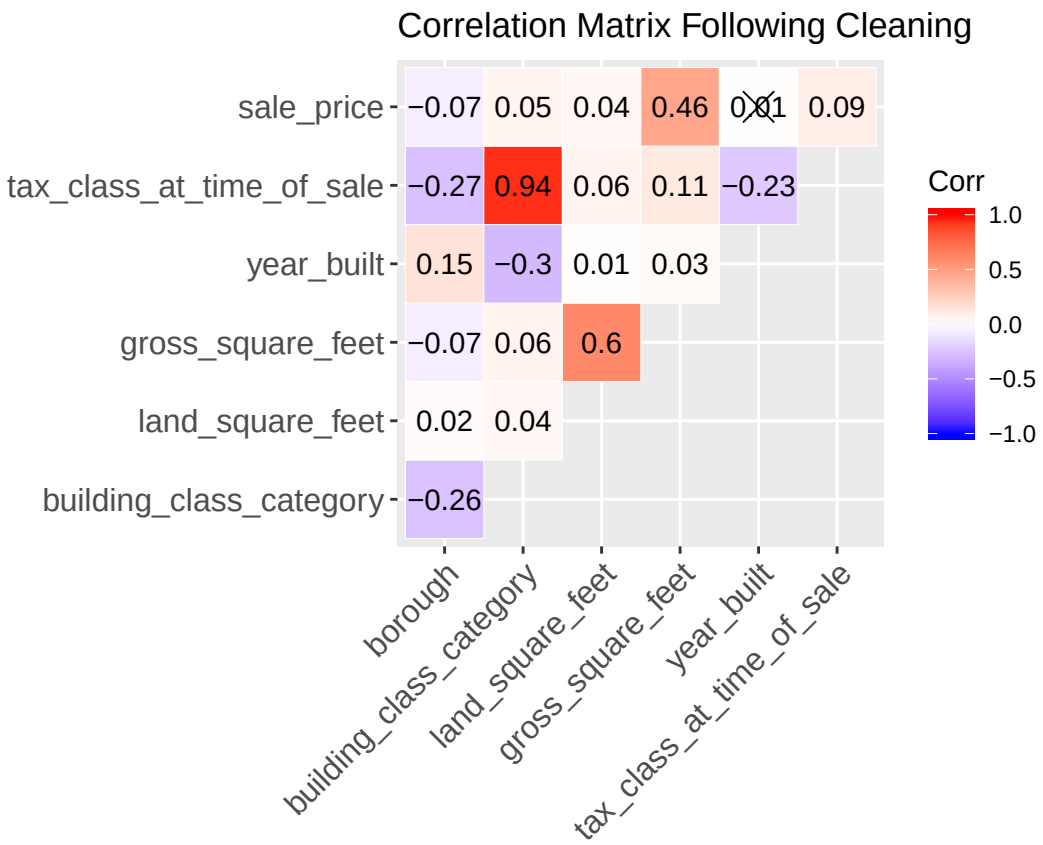
```r
nysales[,'gross_square_feet'] <- as.numeric(nysales$gross_square_feet)
```

```
## Warning: NAs introduced by coercion
```

```r
nysales[,'year_built'] <- as.numeric(nysales$year_built)

# Data Cleaning and Filtering
natest = is.na(nysales$sale_price)
nydata <- nysales[!natest,]
natest = is.na(nydata$gross_square_feet)
nydata <- nydata[!natest,]
natest = is.na(nydata$land_square_feet)
nydata <- nydata[!natest,]
natest = is.na(nydata$year_built)
nydata <- nydata[!natest,]

# Data Exploration
cnames <- colnames(nydata)
types <- sapply(nydata, class)
idx <- which(types=='numeric')
ind <- c()
for (i in 1:length(idx)){ind <- c(ind, idx[[i]])}
nydata.numeric <- nydata[,ind]

corr <- cor(nydata.numeric) # Correlation of the Numerical Data
p.mat <- cor_pmat(nydata.numeric)
ggcorrplot(corr, type = 'upper', ggtheme = ggplot2::theme_gray,
           outline.col = 'white', title = "Correlation Matrix (Numeric), before encoding and cleaning",
           p.mat = p.mat, lab = TRUE)
```




```
# Removing Variables that provide no Information
nydata <- select(nydata, -c('zip_code', 'neighborhood', 'commercial_units',
                            'residential_units','total_units',
                            'tax_class_at_present','building_class_at_present',
                            'building_class_at_time_of_sale','lot','block'))
cnames <- colnames(nydata)
types <- sapply(nydata, class)

# Encoding the Categorical Data
bct.levels <- levels(factor(nydata$building_class_category))

for (i in 1:length(bct.levels)){
  temp <- replace(nydata[,'building_class_category'], nydata[,'building_class_category']==bct.levels[i]
  nydata[,'building_class_category'] <- temp
}
nydata$building_class_category <- as.numeric(nydata$building_class_category)

corr <- cor(nydata[,-8]) # Correlation Following Removing and Encoding
p.mat <- cor_pmat(nydata[,-8])
ggcorrplot(corr, type = 'upper', ggtheme = ggplot2::theme_gray,
           outline.col = 'white', title = "Correlation Matrix Following Cleaning",
           p.mat = p.mat, lab = TRUE)
```




```
# Let's Look at the Distributions of Numerical Variables
s <- ggplot(nydata, aes(y=sale_price)) + geom_boxplot()
summary(nydata$sale_price)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.000e+00 1.000e+05 4.850e+05 1.159e+06 8.350e+05 2.210e+09
```

```
g <- ggplot(nydata, aes(y=gross_square_feet)) + geom_boxplot()
summary(nydata$gross_square_feet)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     820    1620    3502    2508 3750565
```

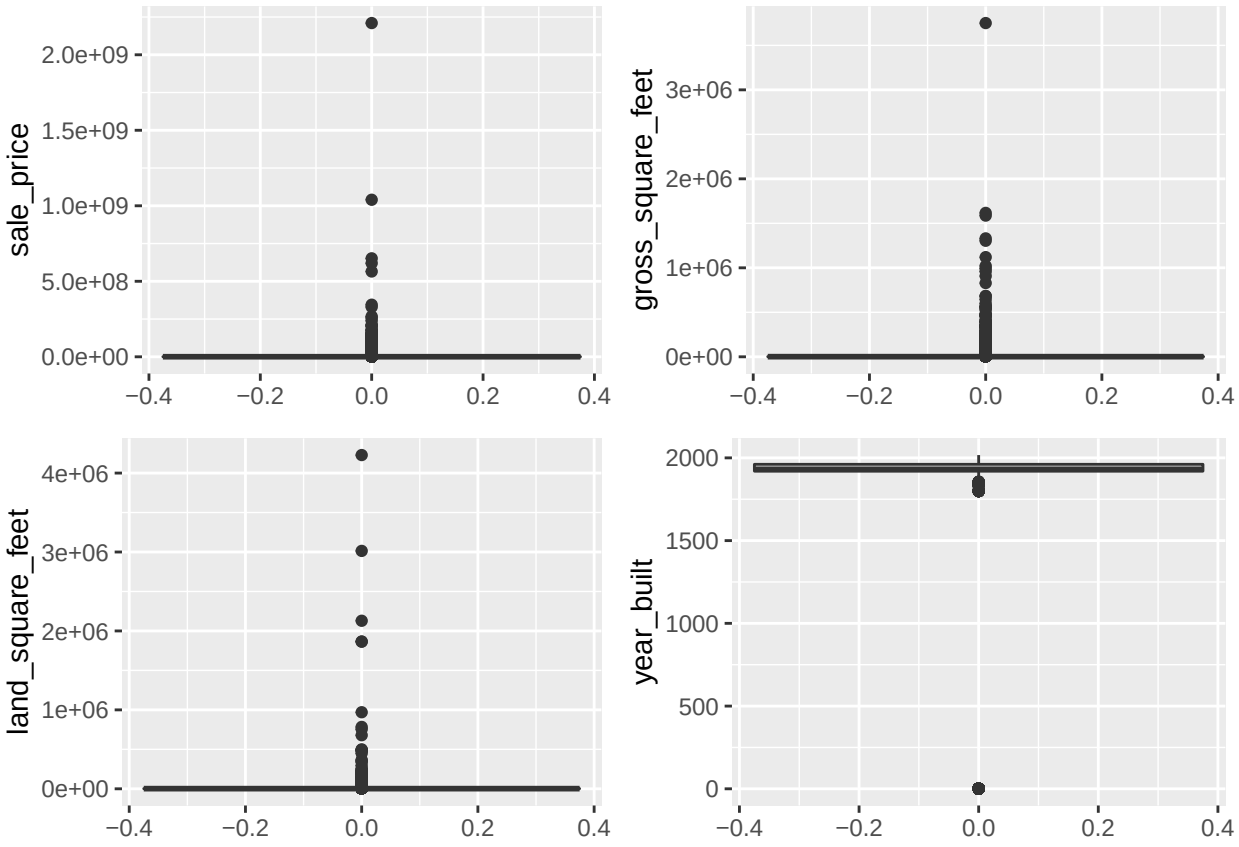
```
l <- ggplot(nydata, aes(y=land_square_feet)) + geom_boxplot()
summary(nydata$land_square_feet)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    1405    2133    3264    3050 4228300
```

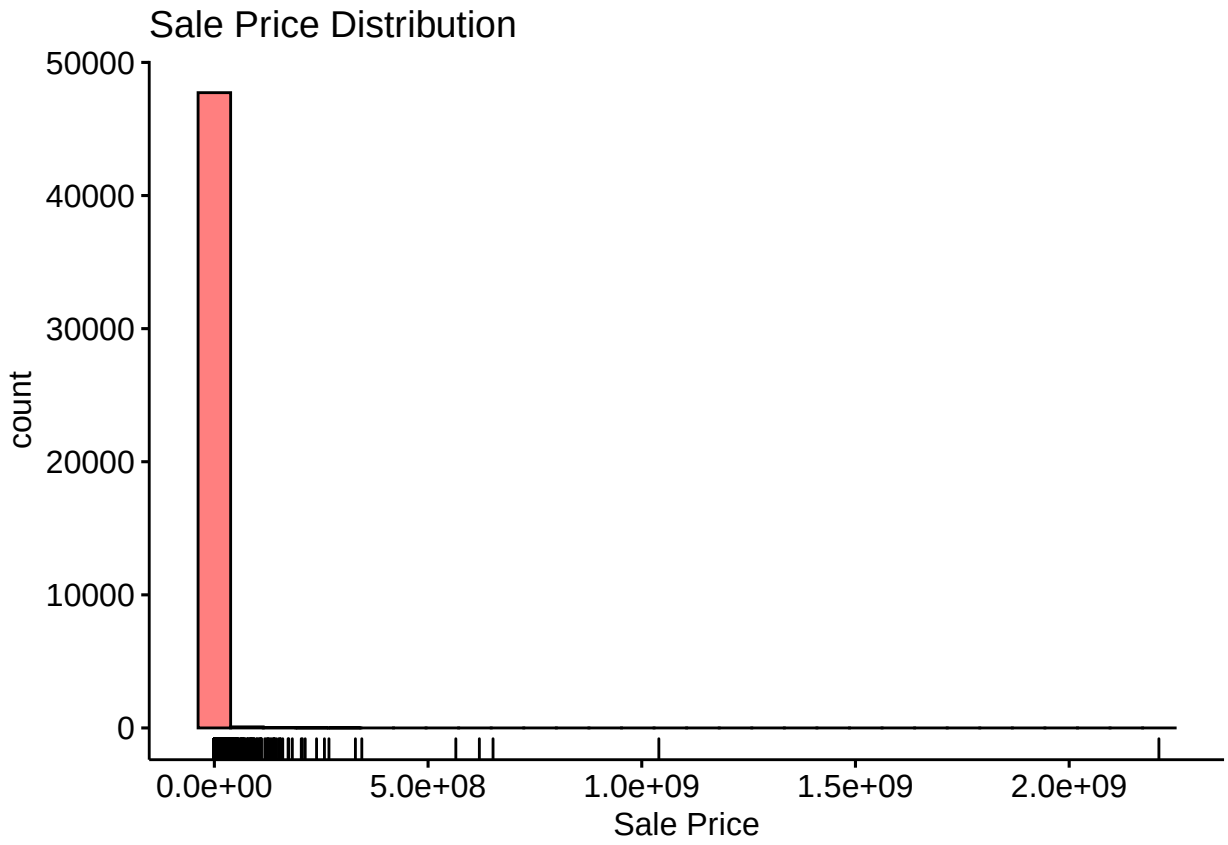
```
ye <- ggplot(nydata, aes(y=year_built)) + geom_boxplot()
summary(nydata$year_built)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    1920    1931    1828    1961    2017
```

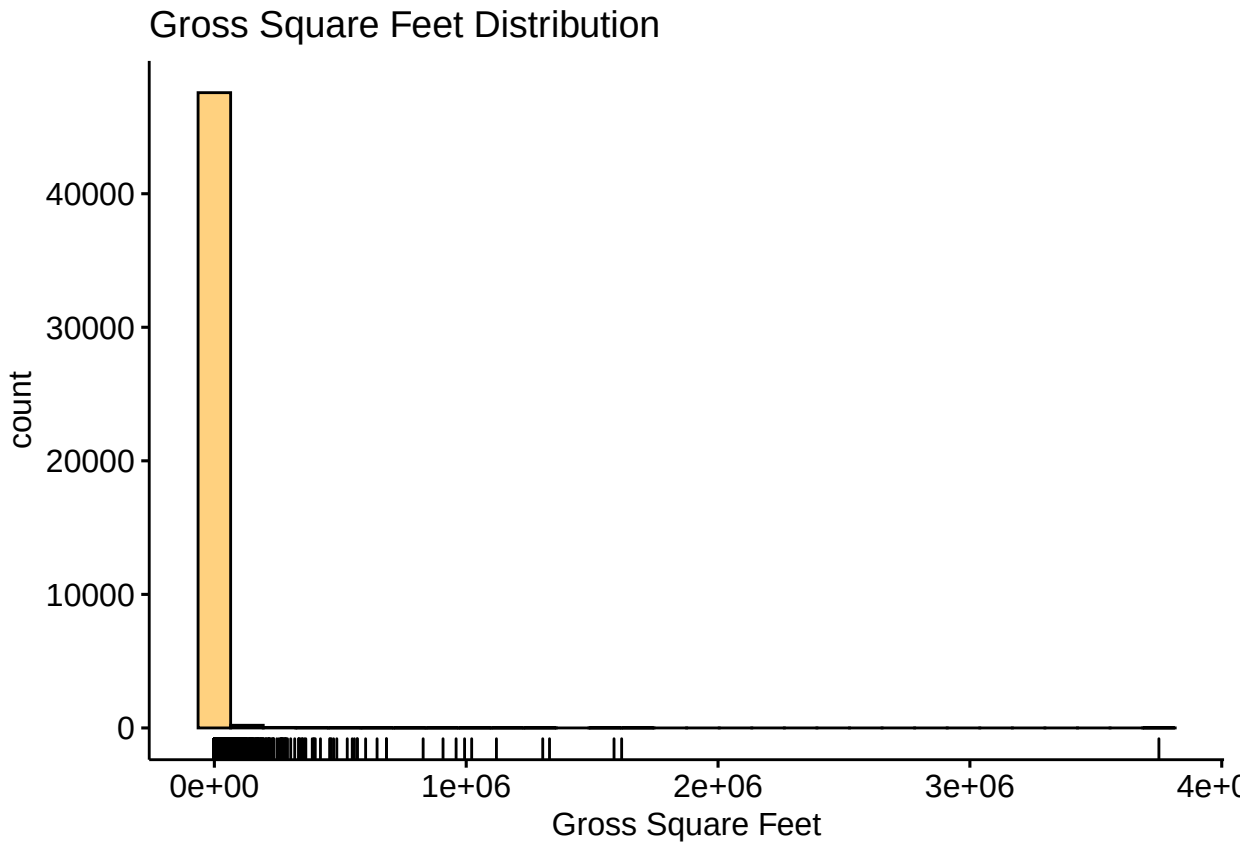
```
ggarrange(s, g, l, ye, ncol= 2, nrow = 2)
```



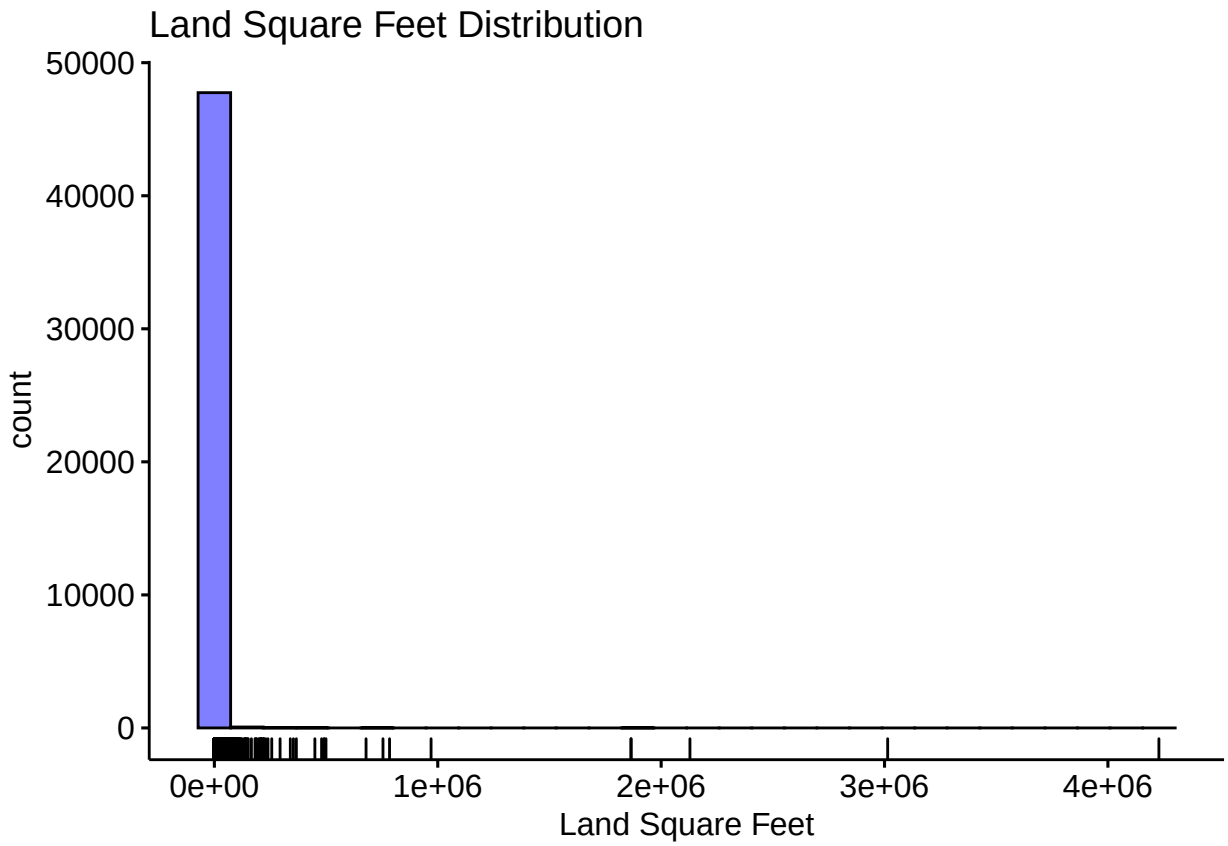

```
sp_hist1 <- gghistogram(nydata, x='sale_price', rug=TRUE,
                        color = 'black', fill = 'red',
                        xlab='Sale Price', title = 'Sale Price Distribution', bins=30)
sp_hist1
```



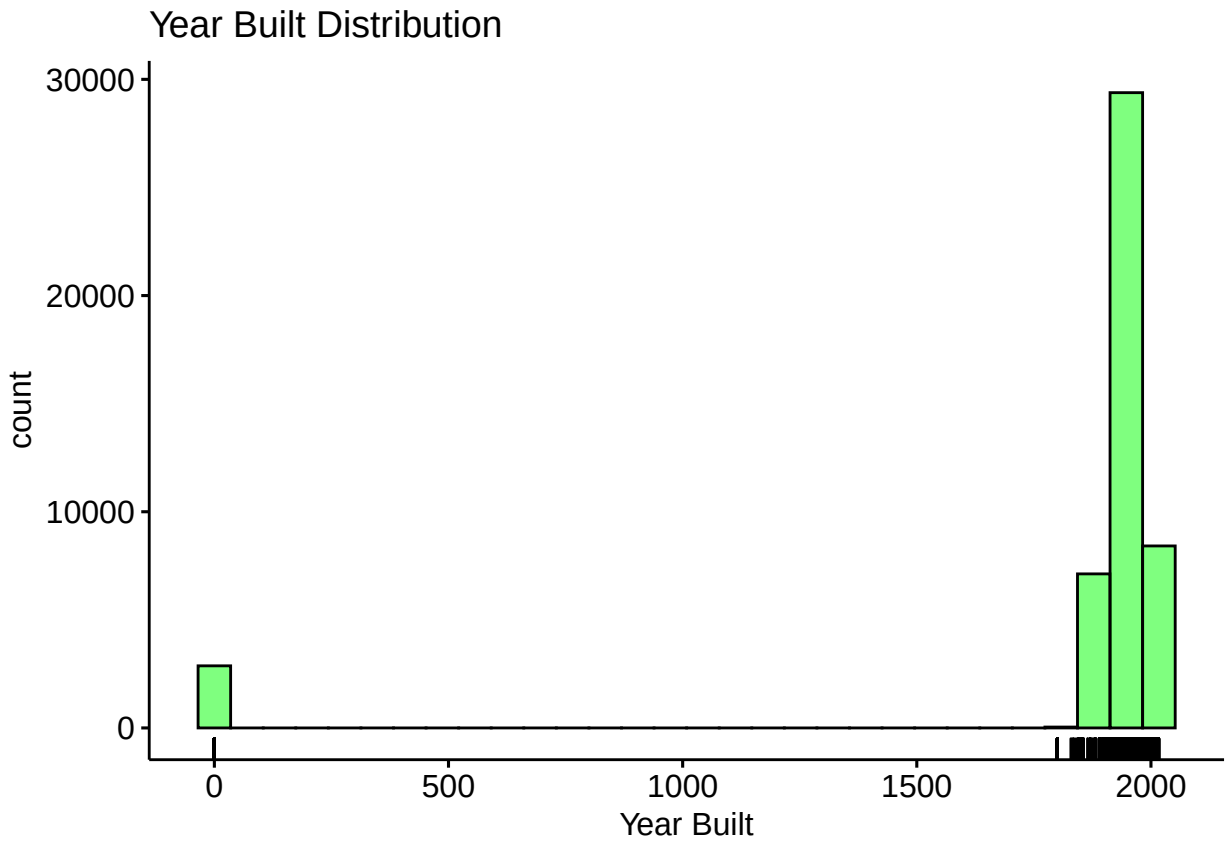

```
gsf_hist1 <- gghistogram(nydata, x='gross_square_feet', rug=TRUE,
                         color = 'black', fill = 'orange',
                         xlab='Gross Square Feet', title = 'Gross Square Feet Distribution', bins=30)
gsf_hist1
```




```
lsf_hist1 <- gghistogram(nydata, x='land_square_feet', rug=TRUE,
                         color = 'black', fill = 'blue',
                         xlab='Land Square Feet', title = 'Land Square Feet Distribution', bins=30)
lsf_hist1
```




```
yb_hist1 <- gghistogram(nydata, x='year_built', rug=TRUE,
                        color = 'black', fill = 'green',
                        xlab='Year Built', title = 'Year Built Distribution', bins=30)
yb_hist1
```




```
# Have to remove values which can throw our model off (Subsetting)
nydata <- nydata[nydata$sale_price > 10000,]
nydata <- nydata[nydata$gross_square_feet!=0,]
nydata <- nydata[nydata$land_square_feet!=0,]
nydata <- nydata[nydata$year_built > 1880,]

s <- ggplot(nydata, aes(y=sale_price)) + geom_boxplot()
summary(nydata$sale_price)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 1.000e+04 4.398e+05 6.350e+05 1.691e+06 9.700e+05 2.210e+09
```

```
g <- ggplot(nydata, aes(y=gross_square_feet)) + geom_boxplot()
summary(nydata$gross_square_feet)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     120    1360    1870    4346    2652 3750565
```

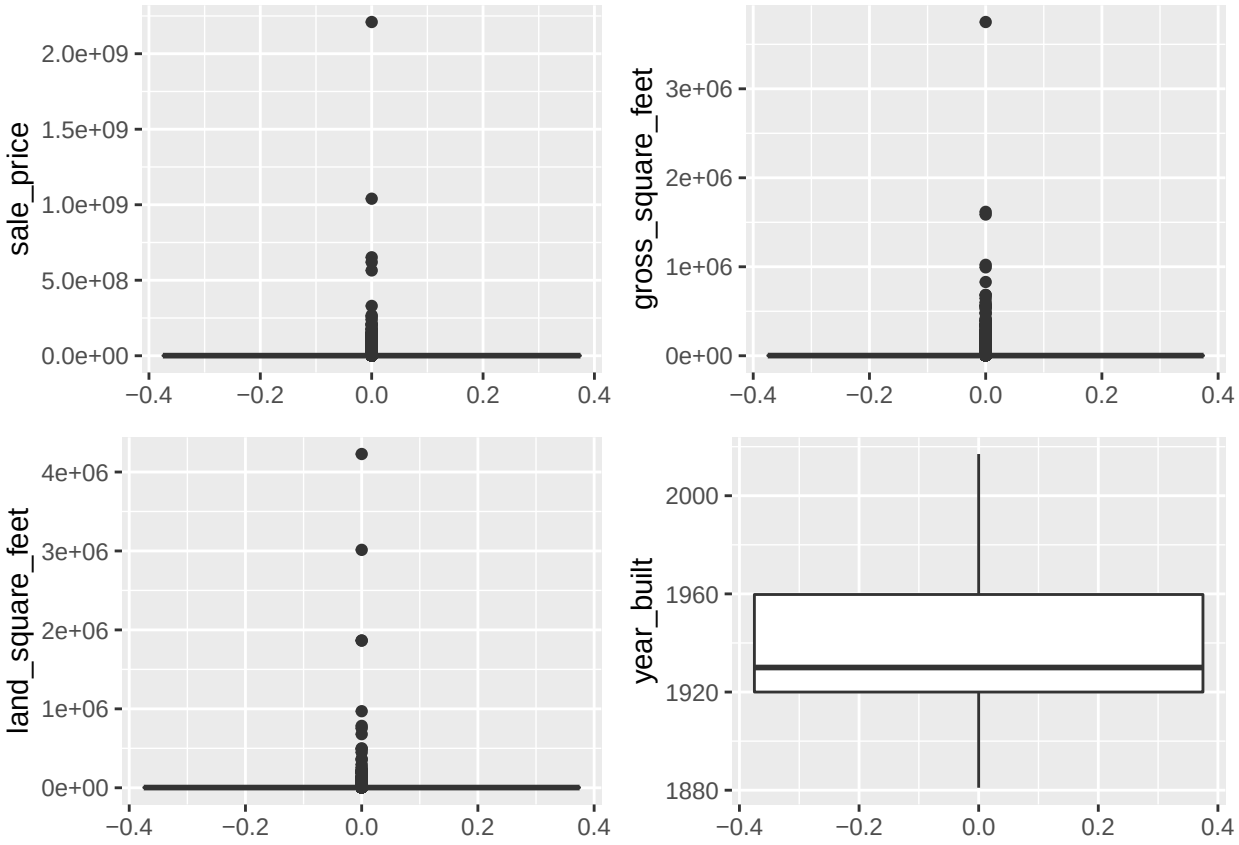
```
l <- ggplot(nydata, aes(y=land_square_feet)) + geom_boxplot()
summary(nydata$land_square_feet)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     200    2000    2500    4137    4000 4228300
```

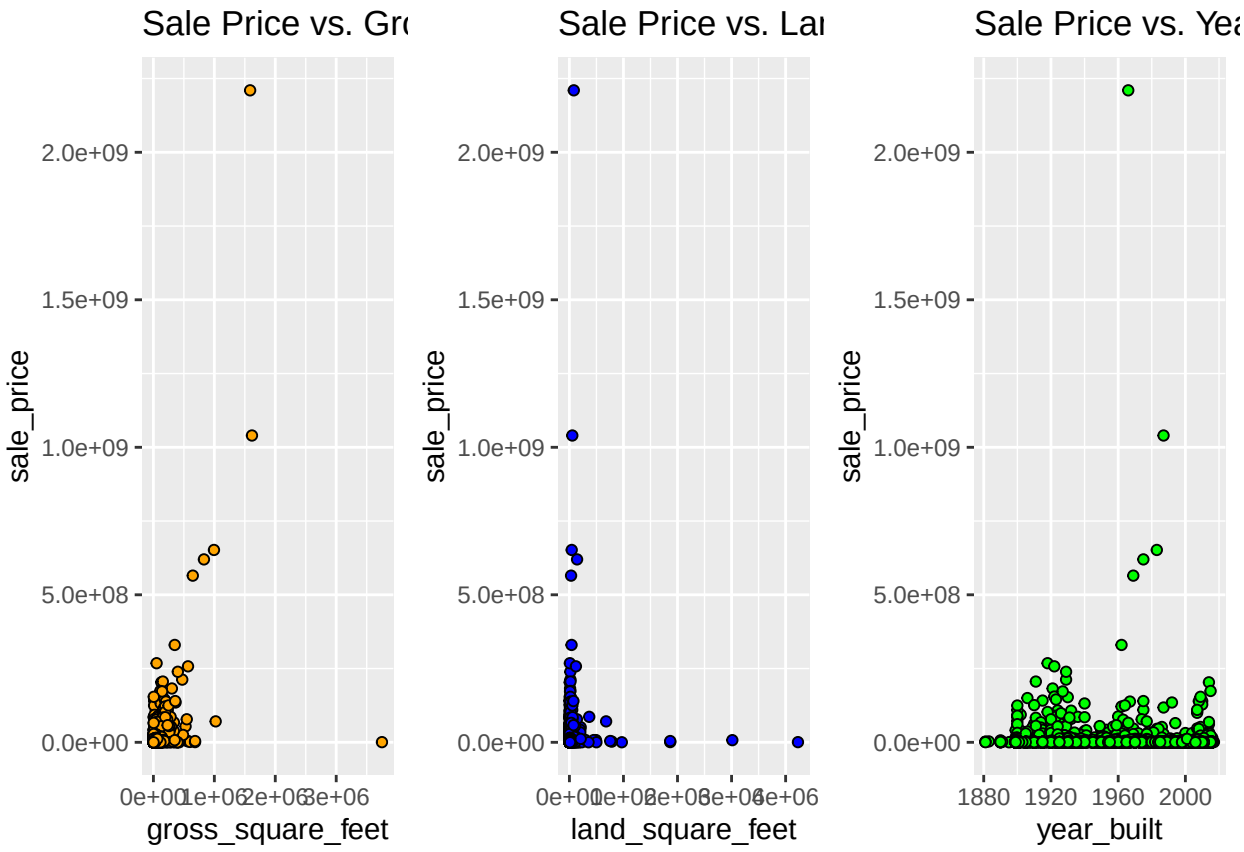
```
ye <- ggplot(nydata, aes(y=year_built)) + geom_boxplot()
summary(nydata$year_built)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1881    1920    1930    1941    1960    2017
```

```
ggarrange(s, g, l, ye, ncol= 2, nrow = 2)
```




```
sp_gsf1 <- ggplot(nydata, aes(x=gross_square_feet, y=sale_price)) + geom_point(shape = 21, fill='orange
sp_lsf1 <- ggplot(nydata, aes(x=land_square_feet, y=sale_price))  + geom_point(shape = 21, fill='blue',
sp_yb1  <- ggplot(nydata, aes(x=year_built, y=sale_price))        + geom_point(shape = 21, fill='green'
ggarrange(sp_gsf1, sp_lsf1, sp_yb1, ncol= 3, nrow = 1)
```




```
# Transforming Numerical Variables
nums <- c("sale_price", "gross_square_feet","land_square_feet","year_built")
cat <- c("borough","building_class_category","tax_class_at_time_of_sale")
sp_skew.pre <- skewness(nydata$sale_price)
gsf_skew.pre <- skewness(nydata$gross_square_feet)
lsf_skew.pre <- skewness(nydata$land_square_feet)
yb_skew.pre <- skewness(nydata$year_built)
pre.skew <- round(c(sp_skew.pre, gsf_skew.pre, lsf_skew.pre, yb_skew.pre),3)
for(i in 1:4){
  print(paste(nums[i], "has a skewness of: ", pre.skew[i], sep = " "))
}
```

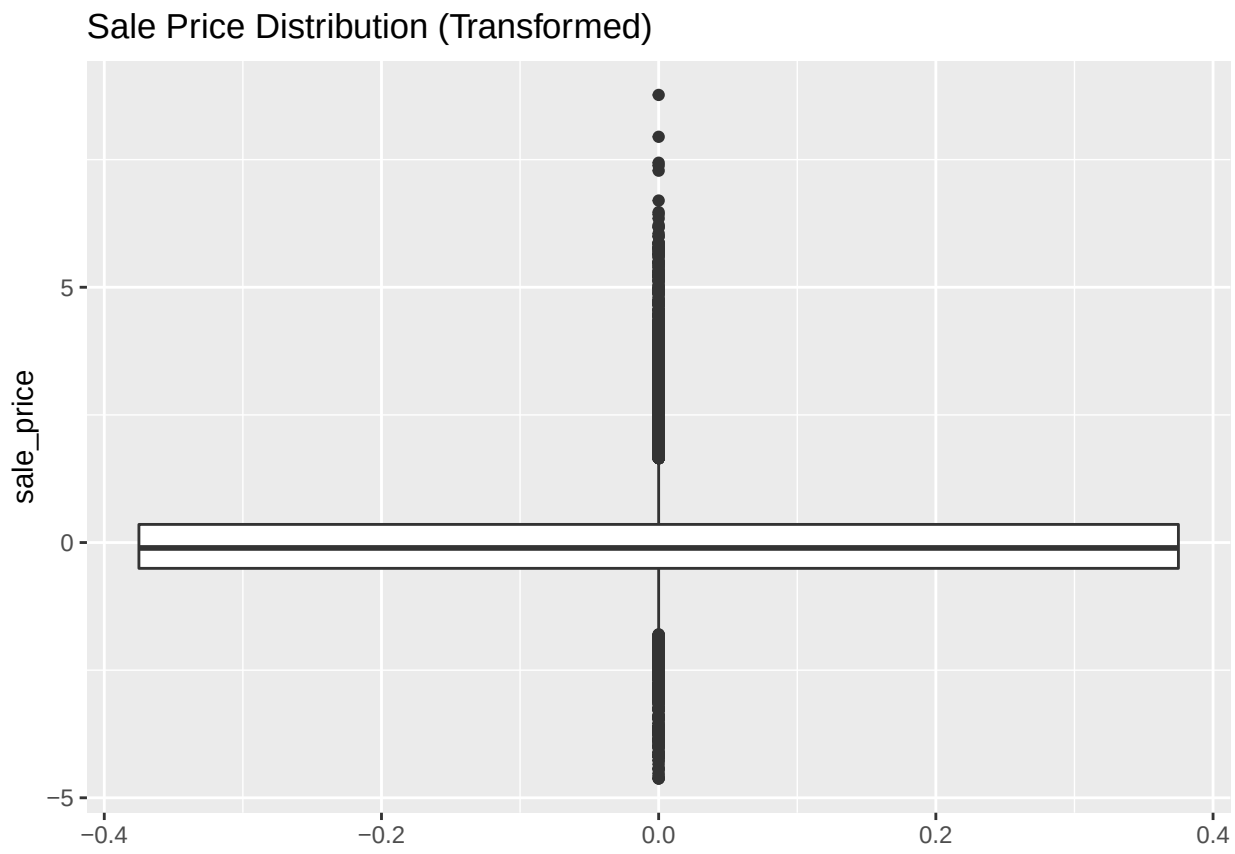
```
## [1] "sale_price has a skewness of:  87.313"
## [1] "gross_square_feet has a skewness of:  63.275"
## [1] "land_square_feet has a skewness of:  83.238"
## [1] "year_built has a skewness of:  0.84"
```

```
nydata[,'sale_price'] <- log1p(nydata$sale_price) # Transforming using log(x + 1); to revert exp(x-1)
nydata[,'gross_square_feet'] <- log1p(nydata$gross_square_feet) # Transforming using log(x + 1); to rev
nydata[,'land_square_feet'] <- log1p(nydata$land_square_feet) # Transforming using log(x + 1); to rever
nydata[,'year_built'] <- log1p(nydata$year_built) # Transforming using log(x + 1); to revert exp(x-1)

sp_skew.post <- skewness(nydata$sale_price)
gsf_skew.post <- skewness(nydata$gross_square_feet)
lsf_skew.post <- skewness(nydata$land_square_feet)
```

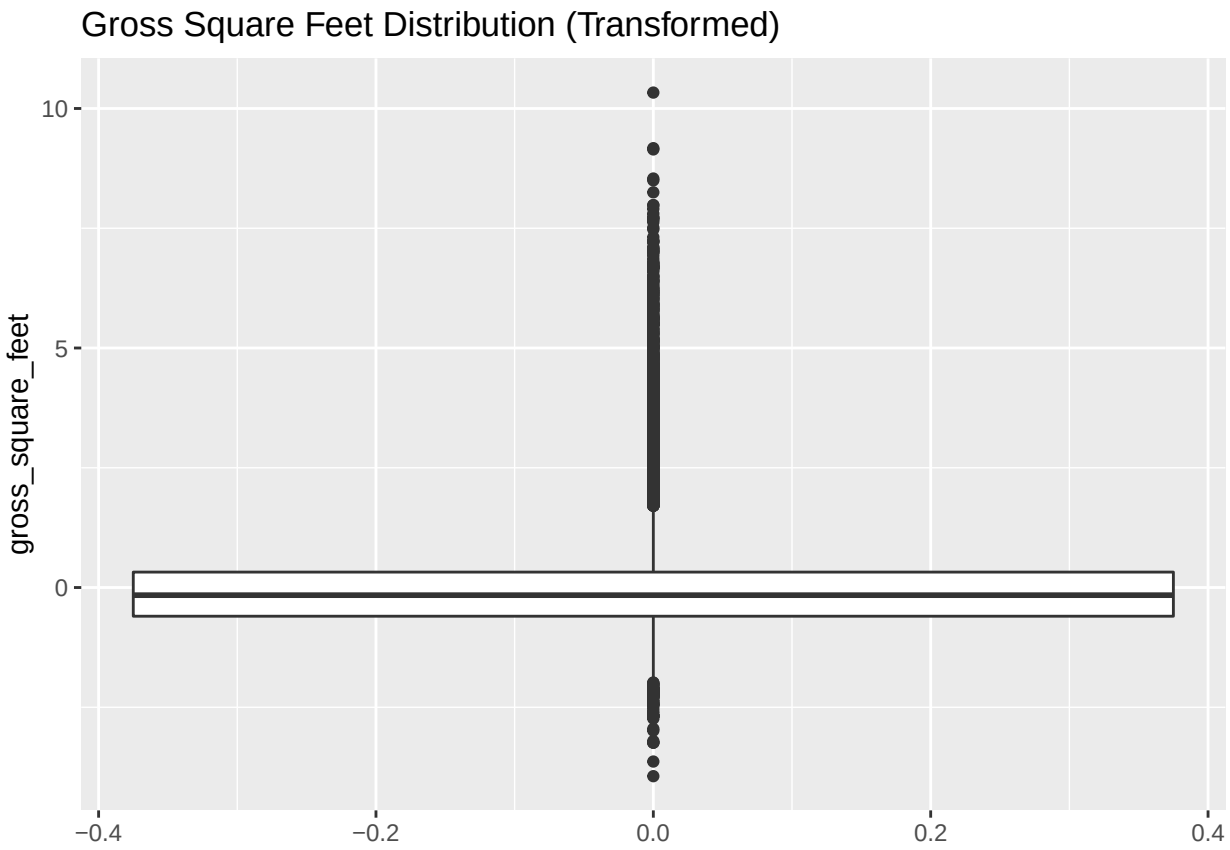
```r
yb_skew.post <- skewness(nydata$year_built)
post.skew <- round(c(sp_skew.post, gsf_skew.post, lsf_skew.post, yb_skew.post),3)
for(i in 1:4){
  print(paste(nums[i], "has a skewness of: ", post.skew[i], sep = " "))
}
```

```
## [1] "sale_price has a skewness of:  1.088"
## [1] "gross_square_feet has a skewness of:  2.658"
## [1] "land_square_feet has a skewness of:  1.87"
## [1] "year_built has a skewness of:  0.813"
```

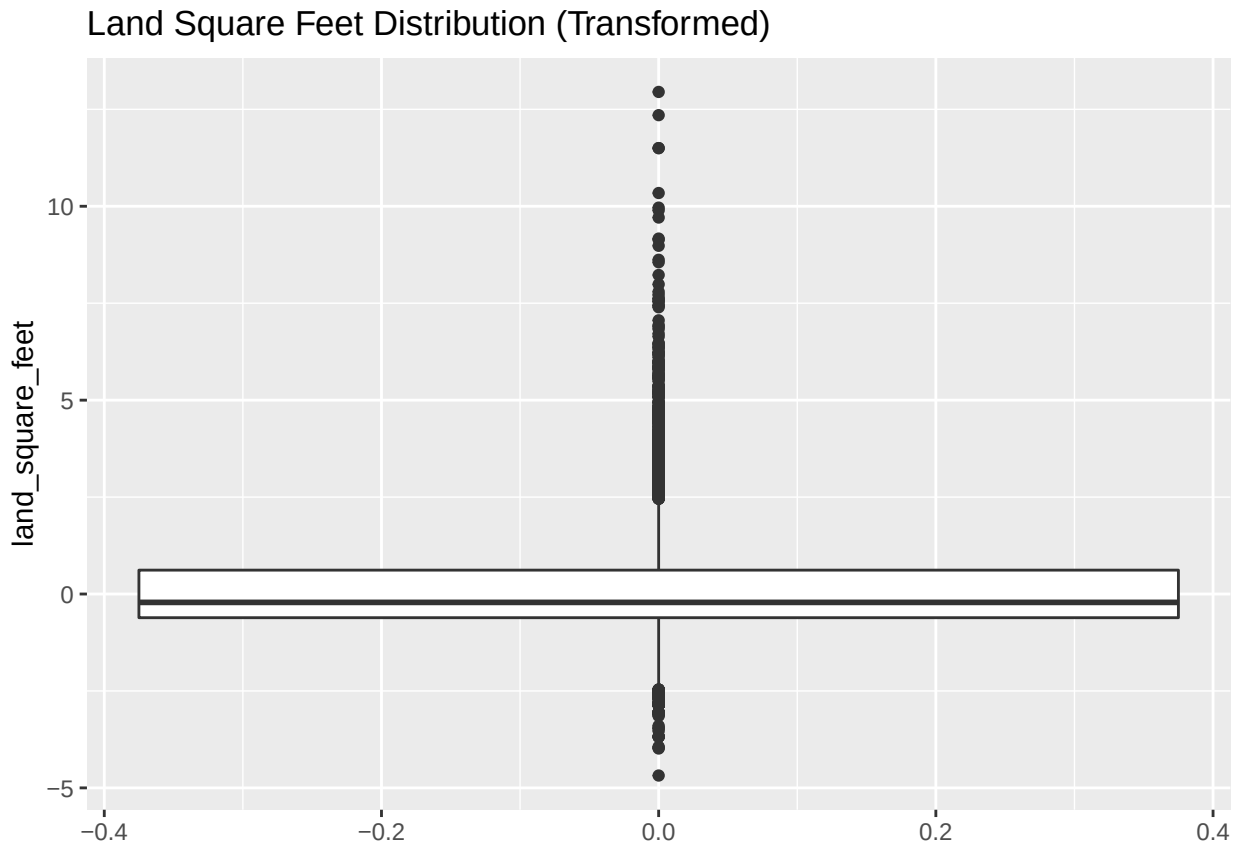
```r
sp_scaled <- scale(nydata$sale_price)
gsf_scaled <- scale(nydata$gross_square_feet)
lsf_scaled <- scale(nydata$land_square_feet)
yb_scaled <- scale(nydata$year_built)

nydata[,'sale_price'] <- sp_scaled # Scaling
nydata[,'gross_square_feet'] <- gsf_scaled # To revert: (x * attr(x,'sclaed:scale) + attr(x,'scaled:cen
nydata[,'land_square_feet'] <- lsf_scaled # Scaling
nydata[,'year_built'] <- yb_scaled # To revert: (x * attr(x,'sclaed:scale) + attr(x,'scaled:center'))


sp_bp <- ggplot(nydata, aes(y=sale_price)) + geom_boxplot() + ggtitle("Sale Price Distribution (Transfo
sp_bp
```



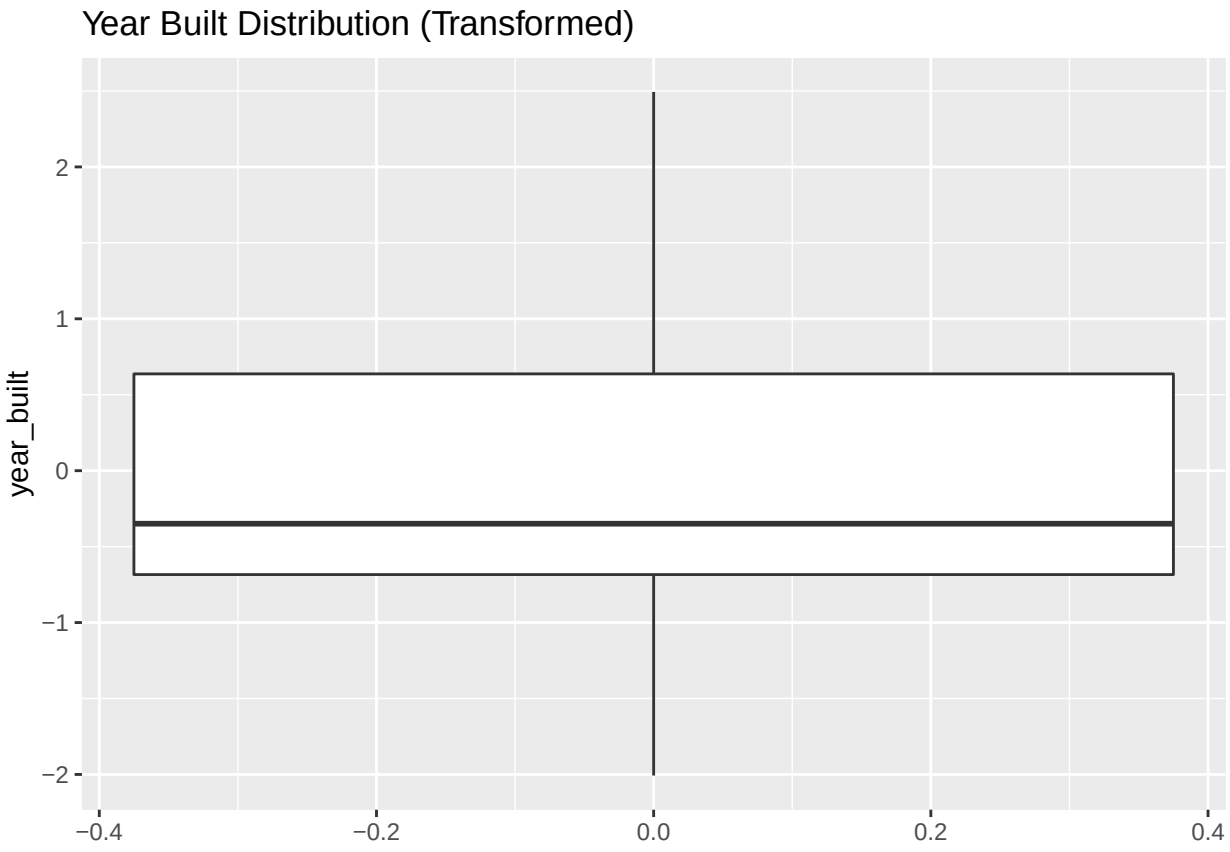
```
gsf_bp <-ggplot(nydata, aes(y=gross_square_feet)) + geom_boxplot() + ggtitle("Gross Square Feet Distribu
gsf_bp
```



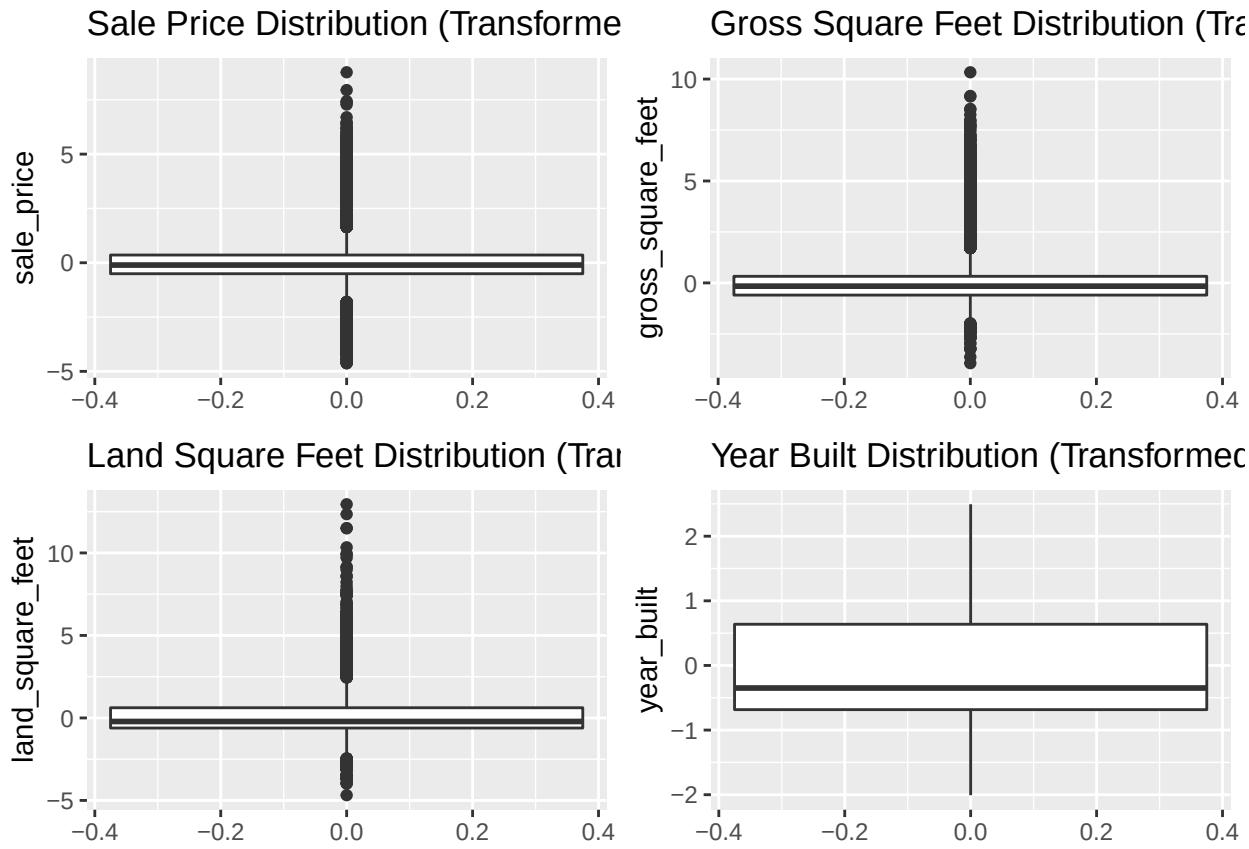

```
lsf_bp<- ggplot(nydata, aes(y=land_square_feet)) + geom_boxplot() + ggtitle("Land Square Feet Distributi
lsf_bp
```



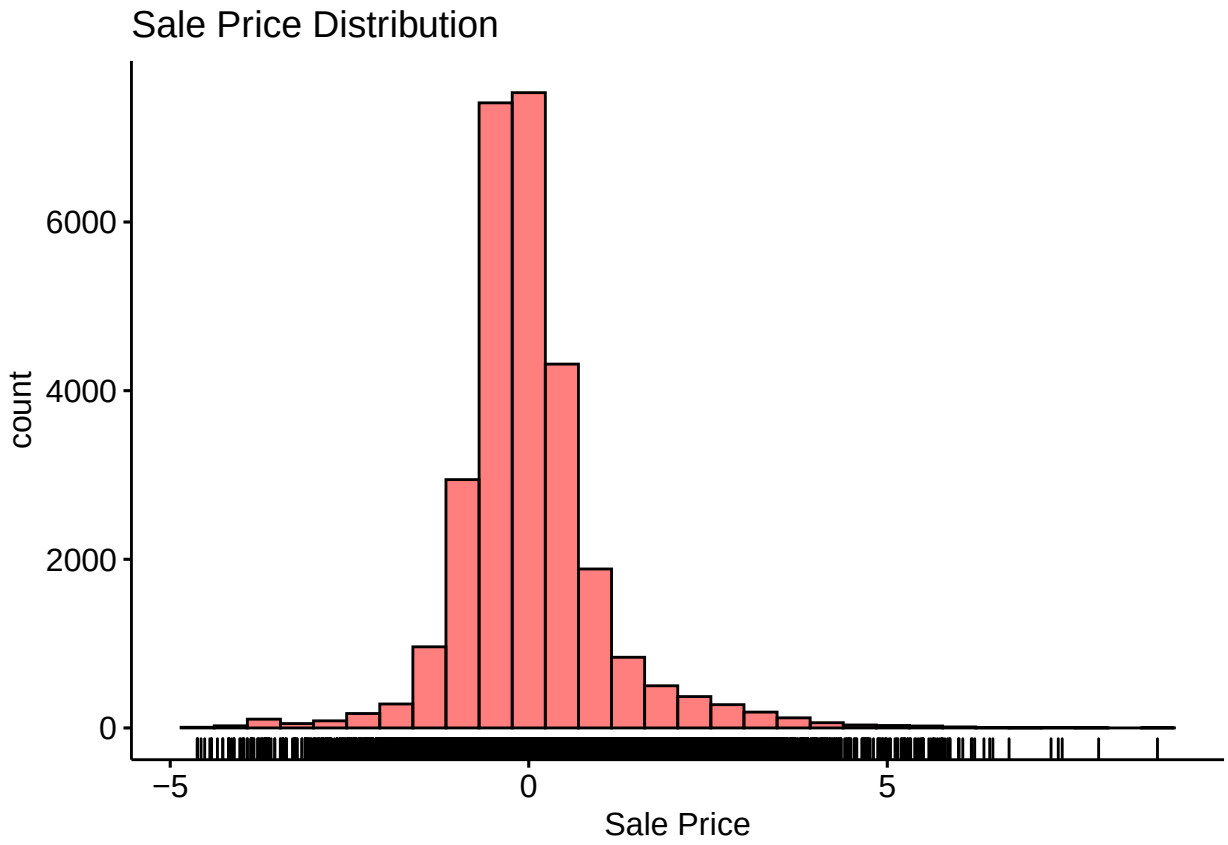

```
yb_bp <- ggplot(nydata, aes(y=year_built)) + geom_boxplot() + ggtitle("Year Built Distribution (Transfo
yb_bp
```



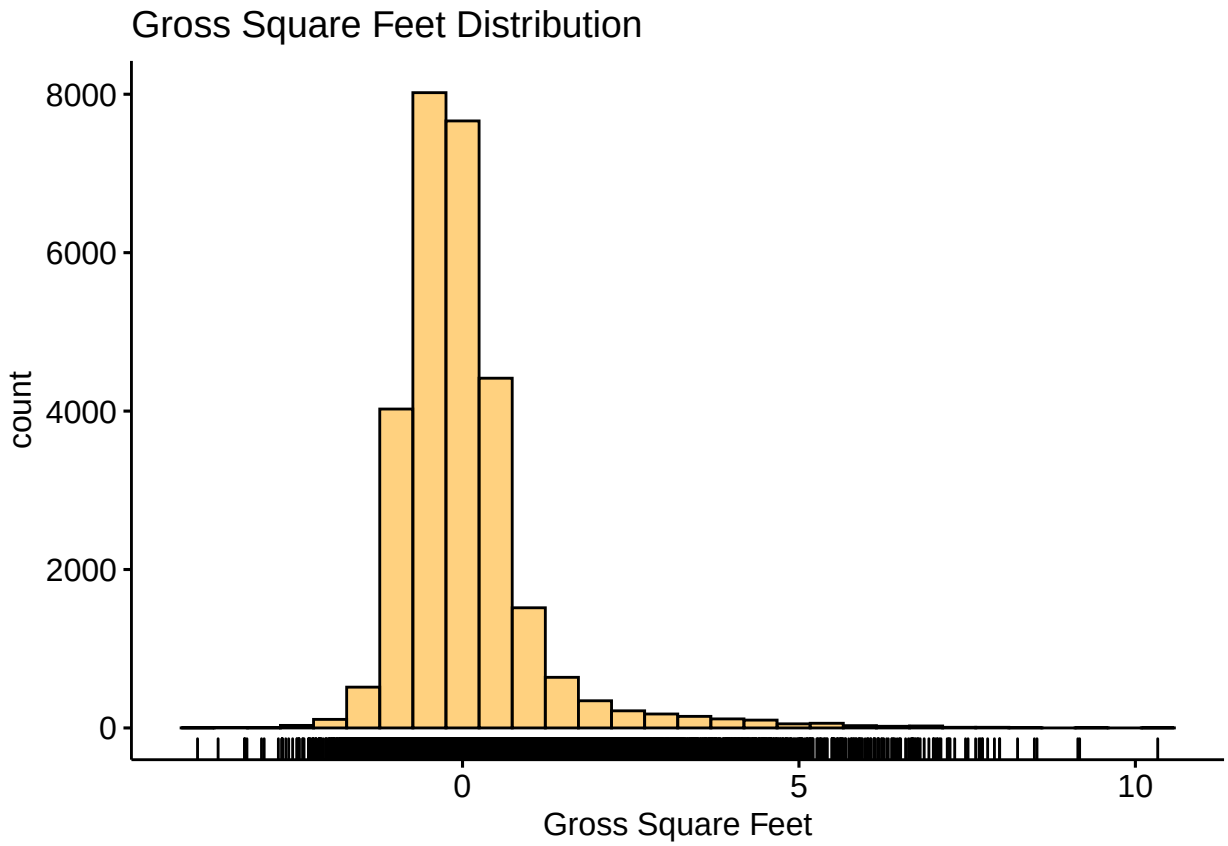

```
ggarrange(sp_bp, gsf_bp, lsf_bp, yb_bp, ncol= 2, nrow = 2)
```



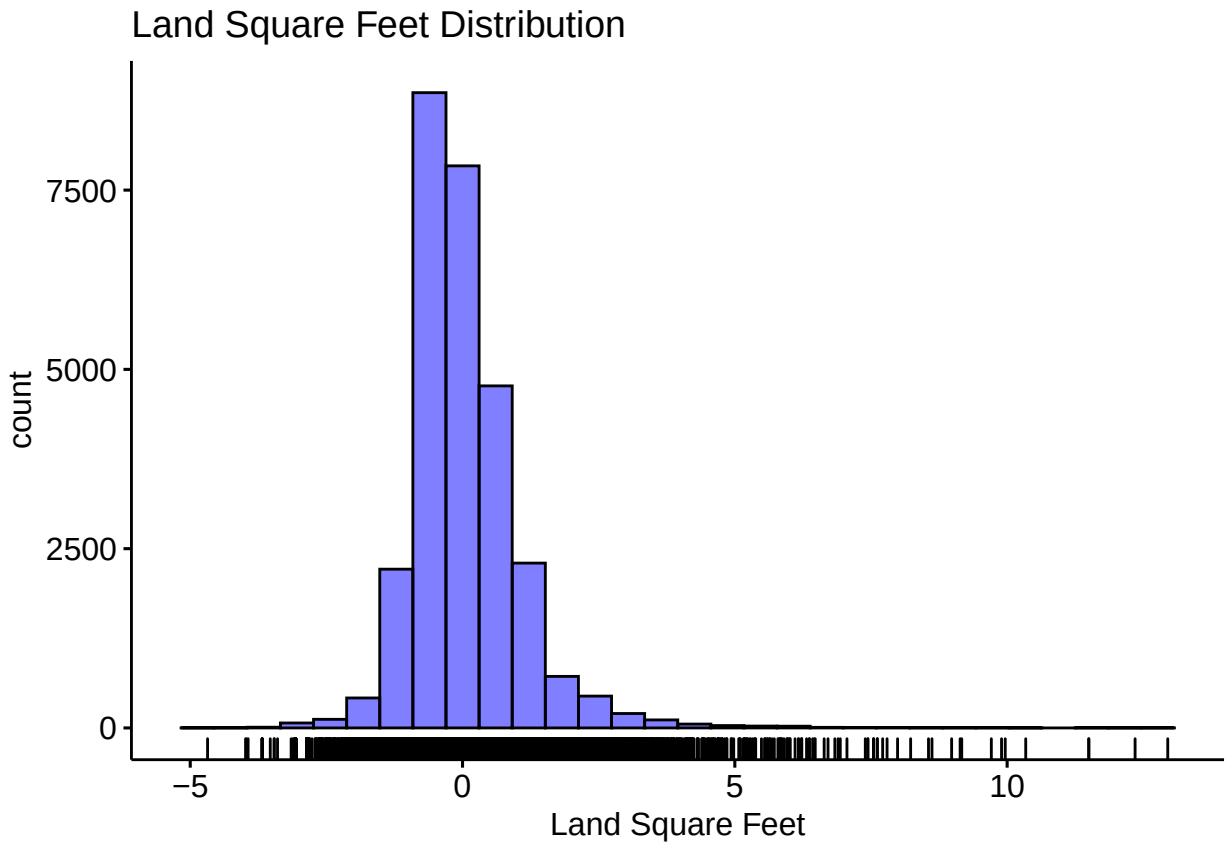

```
sp_hist3 <- gghistogram(nydata, x='sale_price', rug=TRUE,
                        color = 'black', fill = 'red',
                        xlab='Sale Price', title = 'Sale Price Distribution', bins=30)
sp_hist3
```



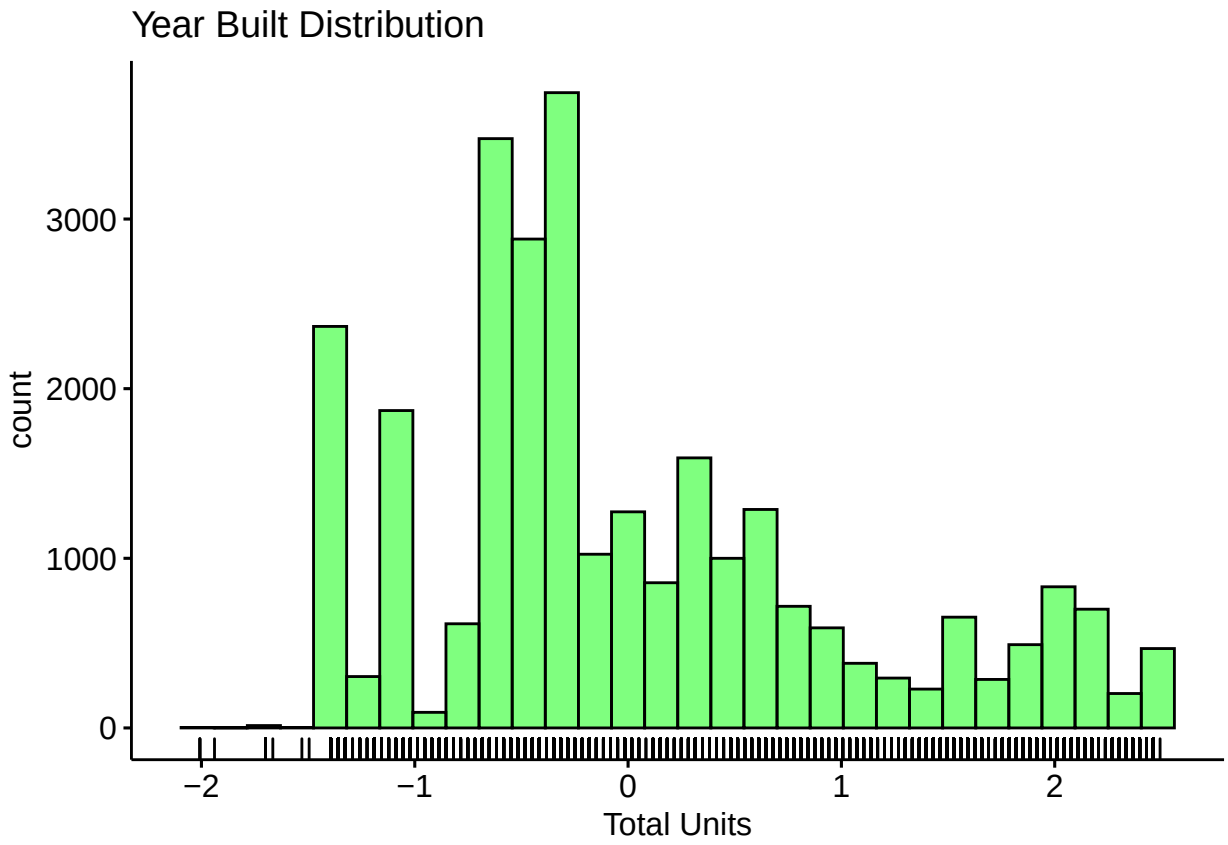

```
gsf_hist3 <- gghistogram(nydata, x='gross_square_feet', rug=TRUE,
                         color = 'black', fill = 'orange',
                         xlab='Gross Square Feet', title = 'Gross Square Feet Distribution', bins=30)
gsf_hist3
```



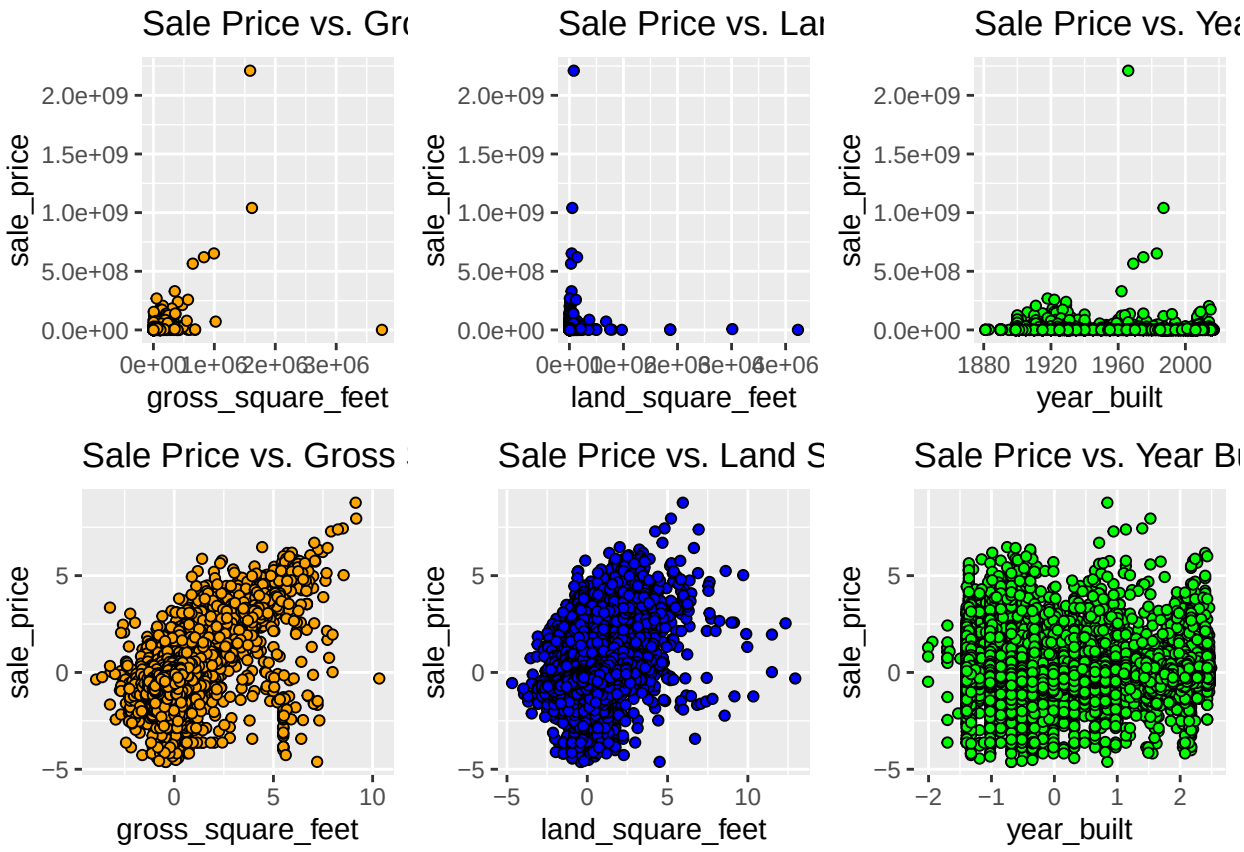

```
lsf_hist3 <- gghistogram(nydata, x='land_square_feet', rug=TRUE,
                         color = 'black', fill = 'blue',
                         xlab='Land Square Feet', title = 'Land Square Feet Distribution', bins=30)
lsf_hist3
```




```
yb_hist3 <- gghistogram(nydata, x='year_built', rug=TRUE,
                        color = 'black', fill = 'green',
                        xlab='Total Units', title = 'Year Built Distribution', bins=30)
yb_hist3
```




```
sp_gsf2 <- ggplot(nydata, aes(x=gross_square_feet, y=sale_price)) + geom_point(shape = 21, fill='orange
sp_lsf2 <- ggplot(nydata, aes(x=land_square_feet, y=sale_price))  + geom_point(shape = 21, fill='blue',
sp_yb2  <- ggplot(nydata, aes(x=year_built, y=sale_price))        + geom_point(shape = 21, fill='green'
ggarrange(sp_gsf1, sp_lsf1, sp_yb1, sp_gsf2, sp_lsf2, sp_yb2, ncol= 3, nrow = 2)
```




```
# Studying the Relation Between Sale Price and Categorical Variables
bsp <- ggplot(nydata, aes(x=factor(borough), y=sale_price, fill=factor(borough))) + geom_boxplot()
bsp = bsp + ggtitle("Sales Price Distribution for different Boroughs") + xlab('Borough') + ylab('Sale Pr
bsp = bsp + scale_color_brewer(palette="Dark2")

bcsp <- ggplot(nydata, aes(x=factor(building_class_category), y=sale_price, fill=factor(building_class_
bcsp = bcsp + ggtitle("Sales Price Distribution for different Building Classes") + xlab('Building Classe
bcsp = bcsp + scale_color_brewer(palette="Dark2")

tcsp <- ggplot(nydata, aes(x=factor(tax_class_at_time_of_sale), y=sale_price, fill=factor(tax_class_at_
tcsp = tcsp + ggtitle("Sales Price Distribution for different Tax Classes") + xlab('Tax Classes') + ylab
tcsp = tcsp + scale_color_brewer(palette="Dark2")

ggarrange(bsp, tcsp, ncol= 2, nrow = 1)
```


```
nums <- c("sale_price", "gross_square_feet","land_square_feet","year_built")
cat <- c("borough","building_class_category","tax_class_at_time_of_sale")

nydata[,'borough'] <- factor(nydata$borough)
nydata[,'building_class_category'] <- factor(nydata$building_class_category)
nydata[,'tax_class_at_time_of_sale'] <- factor(nydata$tax_class_at_time_of_sale)

bcc_table = count(nydata, nydata$building_class_category)
bcc_table[order(-bcc_table$n),]
```

```
## # A tibble: 30 x 2
##    `nydata$building_class_category`     n
##    <fct>                            <int>
##  1 1                                12555
##  2 2                                 9728
##  3 3                                 2286
##  4 7                                 1723
##  5 21                                 469
##  6 15                                 319
##  7 20                                 218
##  8 8                                  195
##  9 27                                 166
## 10 28                                 166
## # ... with 20 more rows
```

```r
bcc_table[order(-bcc_table$n),][1:5,1]
```

```
## # A tibble: 5 x 1
##   `nydata$building_class_category`
##   <fct>
## 1 1
## 2 2
## 3 3
## 4 7
## 5 21
```

```r
bcc_top <- c(1,2,3,6,9,10,11,14)

nydata.main <- nydata # Creating a Checkpoint, before One-Hot-Encoding

# One Hot Encoding Categorical Variables
for(value in bcc_top){
  nydata[paste("building_class_category", value, sep = ".")] <- ifelse(nydata$building_class_category==
}

bors <- c("Manhattan","Bronx","Brooklyn","Queens","Staten_Island")
for(i in 1:5){
  nydata[paste("borough", bors[i], sep = ".")] <- ifelse(nydata$borough==i,1,0)
}

taxclass <- c(1,2,4)
for(value in taxclass){
  nydata[paste("taxclass", value, sep = ".")] <- ifelse(nydata$tax_class_at_time_of_sale==value,1,0)
}

nydata <- select(nydata, -cat)
cnames <- colnames(nydata)
types <- sapply(nydata, class)


nydata <- select(nydata, -c('sale_date'))

# Begin Modeling
set.seed(11)

nydata.subset <- sample(nrow(nydata), nrow(nydata)*0.7) # Subsetting Data to Split into Train/Test
nytrain <- nydata[nydata.subset,] # Train Data
nytest  <- nydata[-nydata.subset,] # Test Data

# Linear Regression Model #

lm.full <- lm(sale_price~.,data = nytrain)
summary(lm.full) # Note the Insignificant Variables
```

```
##
## Call:
## lm(formula = sale_price ~ ., data = nytrain)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1962 -0.2604  0.0849  0.3606  5.9161
##
## Coefficients: (4 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  0.004583   0.029992   0.153   0.8785
## land_square_feet             0.096038   0.006620  14.508  < 2e-16 ***
## gross_square_feet            0.472173   0.008841  53.405  < 2e-16 ***
## year_built                  -0.011443   0.005801  -1.973   0.0485 *
## building_class_category.1   -0.002531   0.226785  -0.011   0.9911
## building_class_category.2   -0.061049   0.226849  -0.269   0.7878
## building_class_category.3   -0.032259   0.227462  -0.142   0.8872
## building_class_category.6   -0.070329   0.339950  -0.207   0.8361
## building_class_category.9   -2.658300   0.271896  -9.777  < 2e-16 ***
## building_class_category.10 -4.957982   0.168579 -29.410  < 2e-16 ***
## building_class_category.11         NA         NA      NA       NA
## building_class_category.14         NA         NA      NA       NA
## borough.Manhattan            1.438615   0.037855  38.003  < 2e-16 ***
## borough.Bronx               -0.142417   0.021037  -6.770 1.33e-11 ***
## borough.Brooklyn             0.471904   0.018999  24.838  < 2e-16 ***
## borough.Queens               0.255332   0.015972  15.986  < 2e-16 ***
## borough.Staten_Island              NA         NA      NA       NA
## taxclass.1                  -0.264586   0.228107  -1.160   0.2461
## taxclass.2                  -0.121633   0.030281  -4.017 5.92e-05 ***
## taxclass.4                         NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7155 on 19756 degrees of freedom
## Multiple R-squared:  0.4917, Adjusted R-squared:  0.4913
## F-statistic:  1274 on 15 and 19756 DF,  p-value: < 2.2e-16
```

```
lm.null <- lm(sale_price~1,data = nytrain)
summary(lm.null)
```

```
##
## Call:
## lm(formula = sale_price ~ 1, data = nytrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6229 -0.5052 -0.1043  0.3550  7.4396
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.381e-05  7.135e-03  -0.005    0.996
##
## Residual standard error: 1.003 on 19771 degrees of freedom
```

```
# Conduct and Compare stepwise-forward/backward regression using BIC
```

```r
# Setting k = log(n), where n is the number of samples, changes criterion from AIC to BIC
lm.stepfor <- step(lm.null,scope = list(lower=formula(lm.null),
                                        upper=formula(lm.full)),
                   direction = 'forward', trace = 0, k = log(nrow(nytrain)))
# Note that the insignificant variables from 'lm.full' will not be included
# But you will still have to remove the insiginificant variables--using the summary function
lm.stepfor$anova
```

```
##                              Step Df    Deviance Resid. Df Resid. Dev
## 1                                 NA          NA     19771   19899.29
## 2           + gross_square_feet -1 7588.702980     19770   12310.59
## 3           + borough.Manhattan -1  736.931593     19769   11573.65
## 4  + building_class_category.10 -1  449.197071     19768   11124.46
## 5               + borough.Bronx -1  418.899046     19767   10705.56
## 6       + borough.Staten_Island -1  237.181853     19766   10468.38
## 7                   + taxclass.1 -1   77.081748     19765   10391.29
## 8            + land_square_feet -1   71.679296     19764   10319.61
## 9              + borough.Queens -1  134.102076     19763   10185.51
## 10  + building_class_category.9 -1   51.091058     19762   10134.42
## 11  + building_class_category.2 -1    9.473587     19761   10124.95
## 12                  + taxclass.4 -1    7.302431     19760   10117.65
##            AIC
## 1     136.7721
## 2   -9348.3345
## 3  -10558.9325
## 4  -11331.7210
## 5  -12080.7372
## 6  -12513.8197
## 7  -12650.0534
## 8  -12777.0215
## 9  -13025.7481
## 10 -13115.2830
## 11 -13123.8824
## 12 -13128.2557
```

```r
lm.stepback <- step(lm.full, direction = 'backward', trace = 0,k = log(nrow(nytrain)))
lm.stepback$anova
```

```
##                            Step Df     Deviance Resid. Df Resid. Dev
## 1                               NA           NA     19756   10114.72
## 2                - taxclass.4  0 0.000000e+00     19756   10114.72
## 3      - borough.Staten_Island  0 0.000000e+00     19756   10114.72
## 4 - building_class_category.14  0 0.000000e+00     19756   10114.72
## 5 - building_class_category.11  0 0.000000e+00     19756   10114.72
## 6  - building_class_category.1  1 6.378134e-05     19757   10114.72
## 7  - building_class_category.6  1 3.666639e-02     19758   10114.76
## 8  - building_class_category.3  1 9.691151e-01     19759   10115.72
## 9                - year_built  1 1.921095e+00     19760   10117.65
##         AIC
## 1 -13094.41
## 2 -13094.41
## 3 -13094.41
```

```
## 4 -13094.41
## 5 -13094.41
## 6 -13104.30
## 7 -13114.12
## 8 -13122.12
## 9 -13128.26
```

```
lmfor_bic <- min(lm.stepfor$anova['AIC'])
lmback_bic <- min(lm.stepback$anova['AIC'])
# Forward BIC < Backward BIC
lmfor_bic < lmback_bic # The stepwise-backward selection leads to the minimum BIC
```

```
## [1] FALSE
```

```
# Note that the insignificant variables from 'lm.full' will not be included
# But you will still have to remove the insiginificant variables--using the summary function
summary(lm.stepback)
```

```
##
## Call:
## lm(formula = sale_price ~ land_square_feet + gross_square_feet +
##     building_class_category.2 + building_class_category.9 + building_class_category.10 +
##     borough.Manhattan + borough.Bronx + borough.Brooklyn + borough.Queens +
##     taxclass.1 + taxclass.2, data = nytrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2068 -0.2603  0.0842  0.3607  5.9192
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -0.001857   0.029037  -0.064  0.94901
## land_square_feet            0.099154   0.006435  15.408  < 2e-16 ***
## gross_square_feet           0.465128   0.008131  57.206  < 2e-16 ***
## building_class_category.2  -0.050716   0.011566  -4.385 1.17e-05 ***
## building_class_category.9  -2.661511   0.271902  -9.788  < 2e-16 ***
## building_class_category.10 -4.948888   0.168438 -29.381  < 2e-16 ***
## borough.Manhattan           1.464126   0.035896  40.788  < 2e-16 ***
## borough.Bronx              -0.133517   0.019967  -6.687 2.34e-11 ***
## borough.Brooklyn            0.485196   0.017011  28.522  < 2e-16 ***
## borough.Queens              0.264710   0.014928  17.733  < 2e-16 ***
## taxclass.1                 -0.277114   0.027990  -9.901  < 2e-16 ***
## taxclass.2                 -0.113629   0.030089  -3.776  0.00016 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7156 on 19760 degrees of freedom
## Multiple R-squared:  0.4916, Adjusted R-squared:  0.4913
## F-statistic:  1737 on 11 and 19760 DF,  p-value: < 2.2e-16
```

```
# Note that all variables have a significant P-Value at an alpha of 0.001
lmstep.var <- names(coef(lm.stepback))[2:12]
```

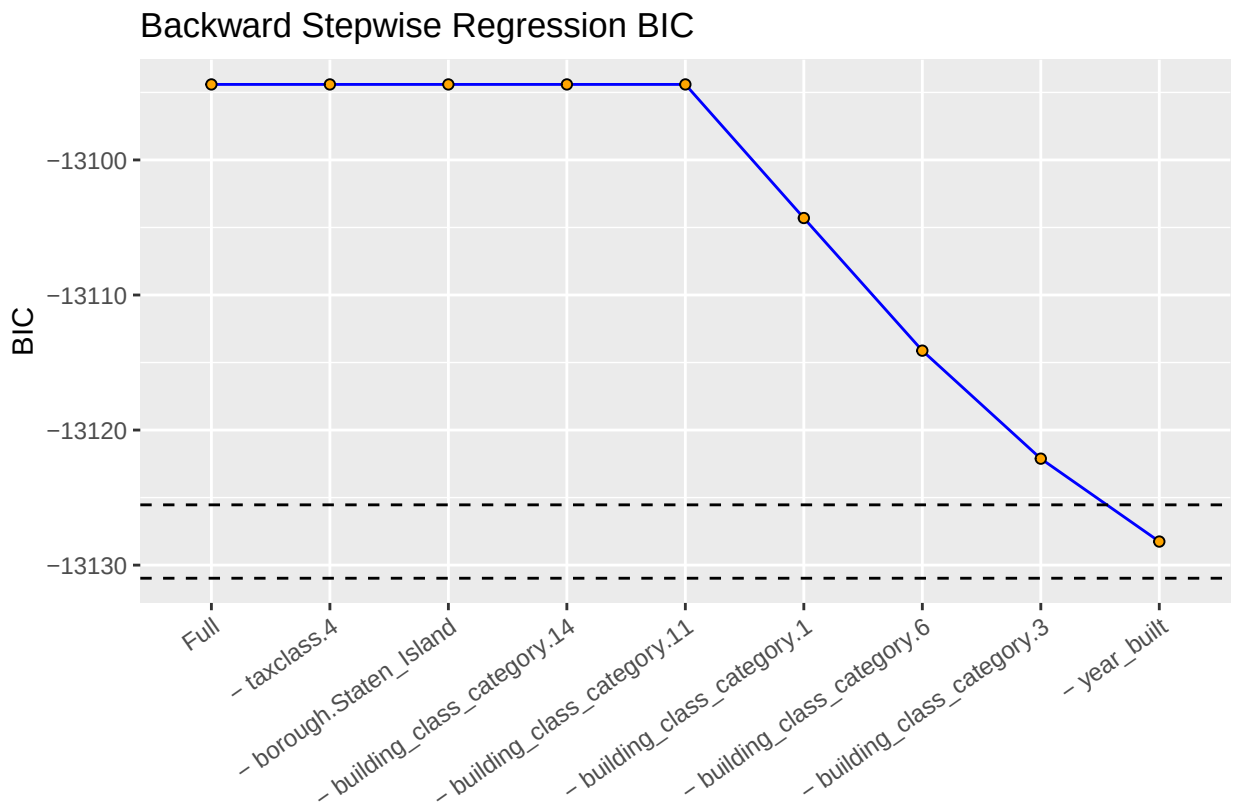
```
lmstep.form <- lm.stepback$terms[[3]]

# Final Model
fmla <- as.formula(paste("sale_price ~ ", paste(lmstep.var, collapse = "+")))
lm.final <- lm(fmla, data = nytrain)
# Note the model is identical to 'lm.stepback' because we didn't have to remove any variables

lm_train.rmse <- sqrt(mean((lm.final$fitted.values - nytrain$sale_price)**2))
paste("The Training RMSE for the Linear Model chosen via Backward Stepwise Selection is: ", round(lm_tra
```

```
## [1] "The Training RMSE for the Linear Model chosen via Backward Stepwise Selection is:  0.7153"
```

```
lm.predict <- predict(lm.final, nytest)
lm_test.rmse <- sqrt(mean((lm.predict - nytest$sale_price)**2))
paste("The Test RMSE for the Linear Model chosen via Backward Stepwise Selection is: ", round(lm_test.rm
```

```
## [1] "The Test RMSE for the Linear Model chosen via Backward Stepwise Selection is:  0.6848"
```

```
# Plotting BIC
anv <- lm.stepback$anova
vars <- c("Full", anv$Step[2:9])
anv[,'ind'] <- vars
bic <- anv$AIC
bic.min <- min(bic)
bic.std <- sd(bic)

bics <- ggplot(anv, aes(x = factor(ind, level = vars) , y=AIC, group = 1)) + geom_line(color='blue') + g
bics <- bics + theme(axis.text.x = element_text(angle = 35, hjust = 1)) + labs(x = "", y = "BIC")
bics <- bics + geom_hline(yintercept = bic.min + 0.2*bic.std, linetype = 'dashed', color = 'black', size
bics <- bics + geom_hline(yintercept = bic.min - 0.2*bic.std, linetype = 'dashed', color = 'black', size
bics
```




```
# Ridge Regression
library(glmnet)
nytrain.mat <- model.matrix(fmla, data = nytrain)
nytest.mat <- model.matrix(fmla, data = nytest)
grid <- 10**seq(2, -2, length = 100)

ridge.fit <- glmnet(nytrain.mat, nytrain$sale_price, alpha = 0, lambda = grid, thresh = 1e-12)
ridge.cv <- cv.glmnet(nytrain.mat, nytrain$sale_price, alpha = 0, lambda = grid, thresh = 1e-12)
lambda.min <- ridge.cv$lambda.min
lambda.min
```

```
## [1] 0.01
```

```
ridge.preds <- predict(ridge.fit, s = lambda.min, newx = nytest.mat)
ridge_test.rmse <- sqrt(mean((nytest$sale_price - ridge.preds)**2))
paste("The Test RMSE for the Ridge Regression Model is: ", round(ridge_test.rmse,4))
```

```
## [1] "The Test RMSE for the Ridge Regression Model is:  0.685"
```

```
# Lasso Regression
nytrain.mat <- model.matrix(fmla, data = nytrain)
nytest.mat <- model.matrix(fmla, data = nytest)
grid <- 10**seq(2, -2, length = 100)
```

```
lasso.fit <- glmnet(nytrain.mat, nytrain$sale_price, alpha = 1, lambda = grid, thresh = 1e-12)
lasso.cv <- cv.glmnet(nytrain.mat, nytrain$sale_price, alpha = 1, lambda = grid, thresh = 1e-12)
lambda.min <- lasso.cv$lambda.min
lambda.min
```

```
## [1] 0.01
```

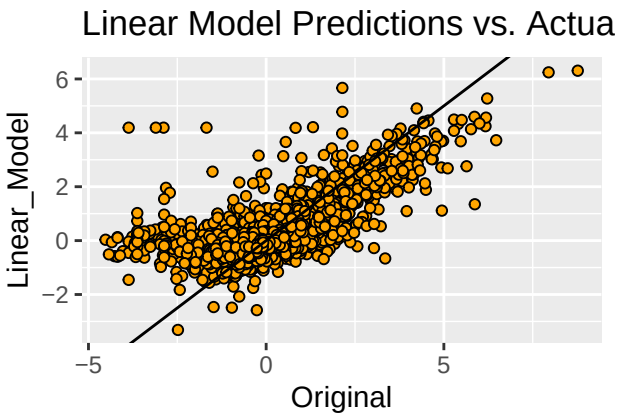
```
lasso.preds <- predict(lasso.fit, s = lambda.min, newx = nytest.mat)
lasso_test.rmse <- sqrt(mean((nytest$sale_price - lasso.preds)**2))
paste("The Test RMSE for the Lasso Regression Model is: ", round(lasso_test.rmse,4))
```

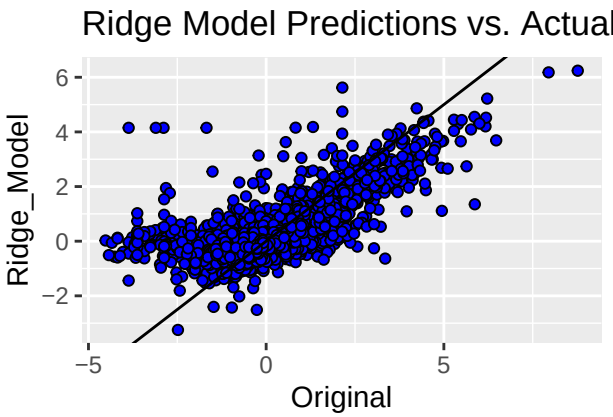
```
## [1] "The Test RMSE for the Lasso Regression Model is:  0.6877"
```

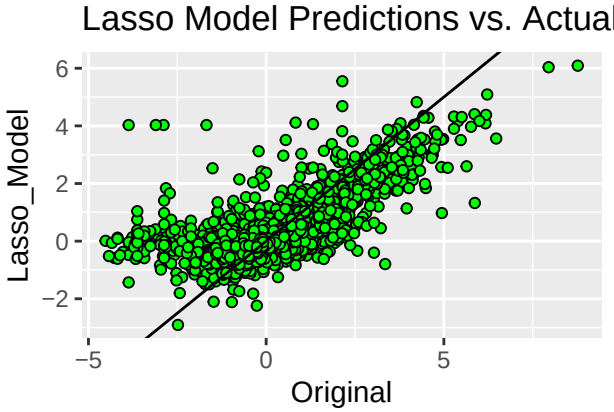
```
# Random Forest
rf.fit <- randomForest(fmla, data = nytrain, ntree = 500, importance = TRUE)
rf.preds <- predict(rf.fit, nytest)
rf_test.rmse <- sqrt(mean((rf.preds - nytest$sale_price)**2))
paste("The Test RMSE for the Random Forest Regression Model is: ", round(rf_test.rmse,4))
```

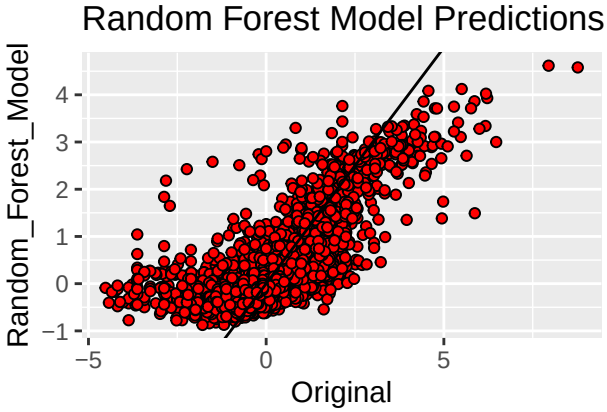
```
## [1] "The Test RMSE for the Random Forest Regression Model is:  0.6717"
```

```
# Final Plot
sales_price <- cbind(nytest$sale_price, lm.predict, ridge.preds, lasso.preds, rf.preds)
sales_price <- as.data.frame(sales_price)
names(sales_price) <- c("Original", "Linear_Model", "Ridge_Model", "Lasso_Model", "Random_Forest_Model")

res1 <- ggplot(sales_price, aes(x = Original, y = Linear_Model))
res1 = res1 + geom_point(shape = 21, fill='orange', color = 'black') + ggtitle('Linear Model Predictions
res1 = res1 + geom_abline(intercept = 0, slope = 1, linetype = 'solid', color = 'black', size = 0.5)

res2 <- ggplot(sales_price, aes(x = Original, y = Ridge_Model))
res2 = res2 + geom_point(shape = 21, fill='blue', color = 'black') + ggtitle('Ridge Model Predictions vs
res2 = res2 + geom_abline(intercept = 0, slope = 1, linetype = 'solid', color = 'black', size = 0.5)

res3 <- ggplot(sales_price, aes(x = Original, y = Lasso_Model))
res3 = res3 + geom_point(shape = 21, fill='green', color = 'black') + ggtitle('Lasso Model Predictions
res3 = res3 + geom_abline(intercept = 0, slope = 1, linetype = 'solid', color = 'black', size = 0.5)

res4 <- ggplot(sales_price, aes(x = Original, y = Random_Forest_Model))
res4 = res4 + geom_point(shape = 21, fill='red', color = 'black') + ggtitle('Random Forest Model Predict
res4 = res4 + geom_abline(intercept = 0, slope = 1, linetype = 'solid', color = 'black', size = 0.5)
ggarrange(res1, res2, res3, res4, ncol= 2, nrow = 2)
```


Linear Model Predictions vs. Actua
Linear_Model
Original
6
4
2
0
−2
−5
0
5


Ridge Model Predictions vs. Actual
Ridge_Model
Original
6
4
2
0
−2
−5
0
5


Lasso Model Predictions vs. Actual
Lasso_Model
Original
6
4
2
0
−2
−5
0
5


Random Forest Model Predictions
Random_Forest_Model
Original
4
3
2
1
0
−1
−5
0
5