```
library(magrittr) # Make all colnames lower case with no spaces
library(stringr) # String formatting and replacement
library(dplyr) # Data wrangling and manipulation
```

```
## Warning: package 'dplyr' was built under R version 4.2.2
```

```
##
## 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(readr) # Load and write csv files
library(ggplot2) # Data visualization
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:magrittr':
##
##     extract
```

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.2.3
```

```
## corrplot 0.92 loaded
```

```
# Reading csv
df_NYCSales2 <- read.csv("C:/Users/nachi/Desktop/Advanced_Analysis/CLEANED_DATA/NYC_Cleaned.csv")
```

```
summary(df_NYCSales2)
```

```
##     BOROUGH      NEIGHBORHOOD       BUILDING_CLASS_CATEGORY
##  Min.   :1.000   Length:25029       Length:25029
##  1st Qu.:3.000   Class :character   Class :character
##  Median :4.000   Mode  :character   Mode  :character
##  Mean   :3.646
##  3rd Qu.:4.000
##  Max.   :5.000
##  TAX_CLASS_AT_PRESENT     BLOCK            LOT
##  Length:25029         Min.   :    8   Min.   :   1.00
##  Class :character     1st Qu.: 3058   1st Qu.:  20.00
##  Mode  :character     Median : 5203   Median :  39.00
##                       Mean   : 5830   Mean   :  63.45
##                       3rd Qu.: 8075   3rd Qu.:  66.00
##                       Max.   :16319   Max.   :3597.00
##  BUILDING_CLASS_AT_PRESENT   ADDRESS             ZIP_CODE     RESIDENTIAL_UNITS
##  Length:25029               Length:25029       Min.   :    0   Min.   :1.000
##  Class :character           Class :character   1st Qu.:10465   1st Qu.:1.000
##  Mode  :character           Mode  :character   Median :11224   Median :2.000
##                                                Mean   :11019   Mean   :1.746
##                                                3rd Qu.:11377   3rd Qu.:2.000
##                                                Max.   :11694   Max.   :9.000
##  LAND_SQUARE_FEET GROSS_SQUARE_FEET TAX_CLASS_AT_TIME_OF_SALE   SALE_PRICE
##  Min.   :  200    Min.   :  150     Min.   :1.000             Min.   :   1162
##  1st Qu.: 2000    1st Qu.: 1320     1st Qu.:1.000             1st Qu.: 425000
##  Median : 2500    Median : 1756     Median :1.000             Median : 599000
##  Mean   : 3021    Mean   : 1975     Mean   :1.046             Mean   : 728615
##  3rd Qu.: 3782    3rd Qu.: 2360     3rd Qu.:1.000             3rd Qu.: 865000
##  Max.   :38862    Max.   :10772     Max.   :4.000             Max.   :4950000
##    YEAR_SOLD      MONTH_SOLD     price_per_unit     BUILDING_AGE
##  Min.   :2016   Min.   : 1.000   Min.   :    833   Min.   :  0.00
##  1st Qu.:2016   1st Qu.: 4.000   1st Qu.: 270000   1st Qu.: 57.00
##  Median :2017   Median : 6.000   Median : 405000   Median : 86.00
##  Mean   :2017   Mean   : 6.591   Mean   : 481238   Mean   : 75.05
##  3rd Qu.:2017   3rd Qu.:10.000   3rd Qu.: 582500   3rd Qu.: 97.00
##  Max.   :2017   Max.   :12.000   Max.   :4950000   Max.   :217.00
```

```
#remove
Final_NYCSales <- df_NYCSales2[,-c(2,3,4,6,8,9,17)]
str(Final_NYCSales)
```

```
## 'data.frame':    25029 obs. of  11 variables:
##  $ BOROUGH                  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ BLOCK                    : int  585 1942 1960 2024 2041 2042 2042 2050 2050 2051 ...
##  $ BUILDING_CLASS_AT_PRESENT: chr  "A5" "A4" "A9" "A5" ...
##  $ RESIDENTIAL_UNITS        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ LAND_SQUARE_FEET         : int  384 1549 1665 1699 1699 1488 1431 2000 1900 2000 ...
##  $ GROSS_SQUARE_FEET        : int  1152 3036 3200 3620 3536 3951 3591 3371 4122 3360 ...
```

```
##  $ TAX_CLASS_AT_TIME_OF_SALE: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ SALE_PRICE               : num  1375000 2300000 1510000 3050000 1595790 ...
##  $ YEAR_SOLD                : int  2016 2016 2017 2017 2017 2017 2017 2017 2017 2016 ...
##  $ MONTH_SOLD               : int  10 11 1 1 8 3 7 8 3 9 ...
##  $ BUILDING_AGE             : int  115 106 107 107 116 118 118 118 118 117 ...
```

```
#factorize BOROUGH, BUILDING_CLASS_AT_PRESENT, TAX_CLASS_AT_TIME_OF_SALE
#get dummies
BUILDING_CLASS_AT_PRESENT_distinct <- unique(Final_NYCSales$BUILDING_CLASS_AT_PRESENT)

print(BUILDING_CLASS_AT_PRESENT_distinct)
```

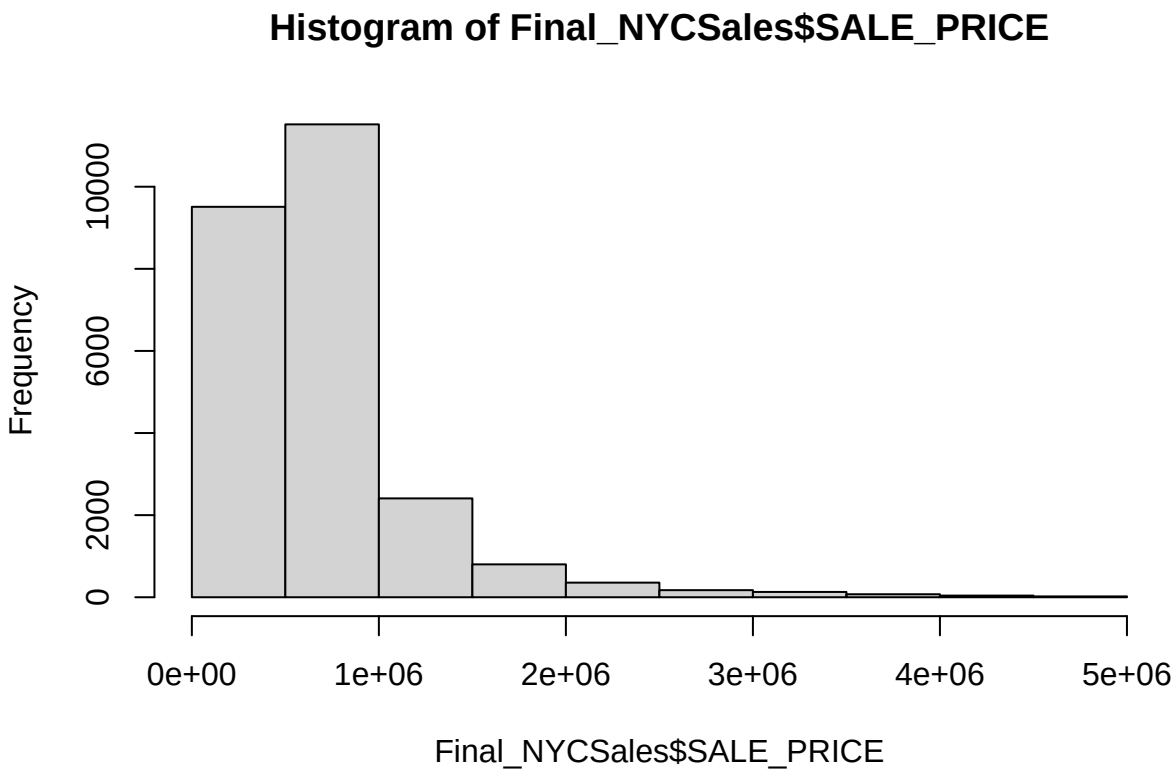
```
##  [1] "A5" "A4" "A9" "A1" "A2" "A6" "A3" "B3" "B2" "A0" "A7" "B9" "B1" "C0" "C4"
## [16] "C1" "C2" "C3" "C5" "C9" "D1" "G0" "Z0" "Z9"
```

```
#Creating dummy variables
Final_NYCSales$BUILDING_CLASS_AT_PRESENT <- as.factor(Final_NYCSales$BUILDING_CLASS_AT_PRESENT)

Final_NYCSales$BOROUGH <- as.factor(Final_NYCSales$BOROUGH)

Final_NYCSales$TAX_CLASS_AT_TIME_OF_SALE <- as.factor(Final_NYCSales$TAX_CLASS_AT_TIME_OF_SALE)

Final_NYCSales$BLOCK <- as.numeric(Final_NYCSales$BLOCK)
Final_NYCSales$RESIDENTIAL_UNITS <- as.numeric(Final_NYCSales$RESIDENTIAL_UNITS)
Final_NYCSales$GROSS_SQUARE_FEET <- as.numeric(Final_NYCSales$GROSS_SQUARE_FEET)
Final_NYCSales$LAND_SQUARE_FEET <- as.numeric(Final_NYCSales$LAND_SQUARE_FEET)
Final_NYCSales$YEAR_SOLD <- as.numeric(Final_NYCSales$YEAR_SOLD)
Final_NYCSales$MONTH_SOLD <- as.numeric(Final_NYCSales$MONTH_SOLD)
Final_NYCSales$BUILDING_AGE <- as.numeric(Final_NYCSales$BUILDING_AGE)

str(Final_NYCSales)
```

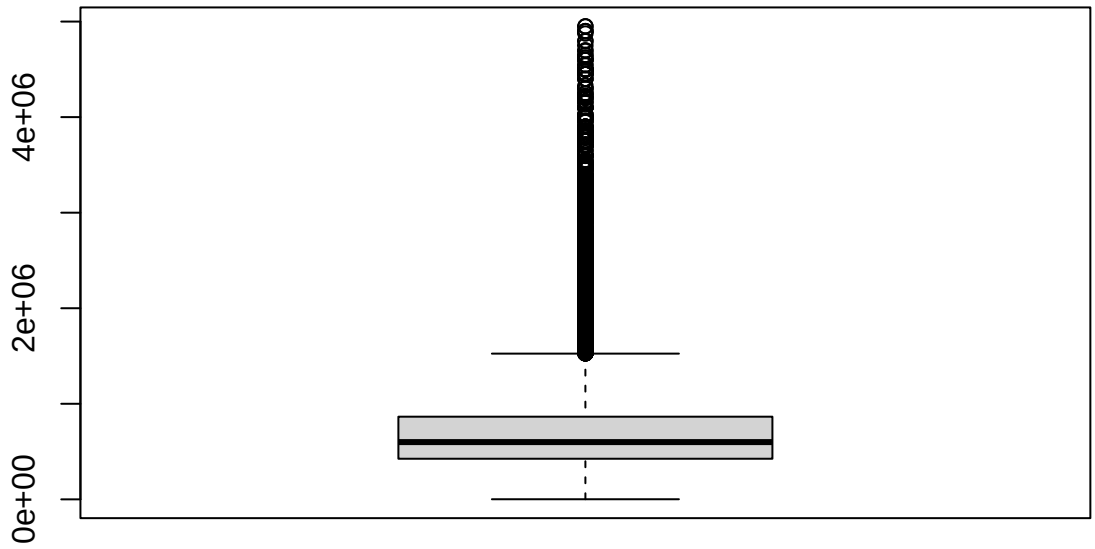
```
## 'data.frame':    25029 obs. of  11 variables:
##  $ BOROUGH                  : Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ BLOCK                    : num  585 1942 1960 2024 2041 ...
##  $ BUILDING_CLASS_AT_PRESENT: Factor w/ 24 levels "A0","A1","A2",..: 6 5 9 6 9 9 9 5 5 5 ...
##  $ RESIDENTIAL_UNITS        : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ LAND_SQUARE_FEET         : num  384 1549 1665 1699 1699 ...
##  $ GROSS_SQUARE_FEET        : num  1152 3036 3200 3620 3536 ...
##  $ TAX_CLASS_AT_TIME_OF_SALE: Factor w/ 3 levels "1","2","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SALE_PRICE               : num  1375000 2300000 1510000 3050000 1595790 ...
##  $ YEAR_SOLD                : num  2016 2016 2017 2017 2017 ...
##  $ MONTH_SOLD               : num  10 11 1 1 8 3 7 8 3 9 ...
##  $ BUILDING_AGE             : num  115 106 107 107 116 118 118 118 118 117 ...
```

Removing outliers:

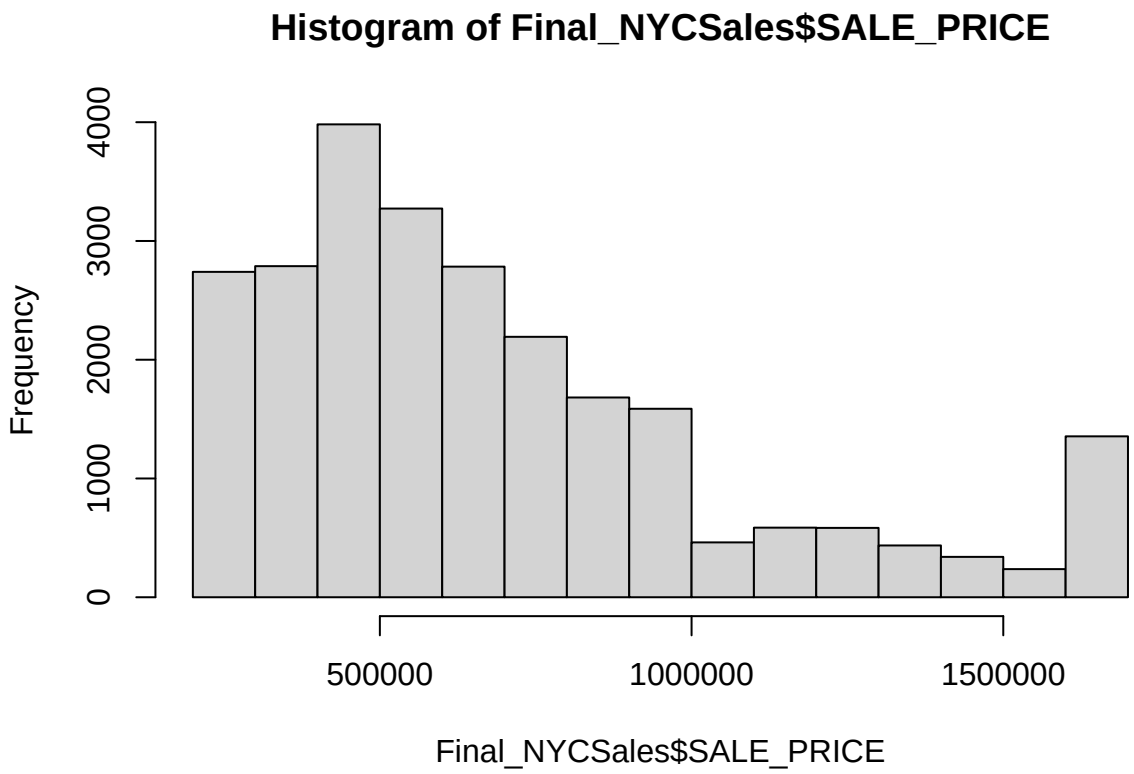
```
#Remove outliers from SALE.PRICE
# Calculate the z-scores
hist(Final_NYCSales$SALE_PRICE)
```



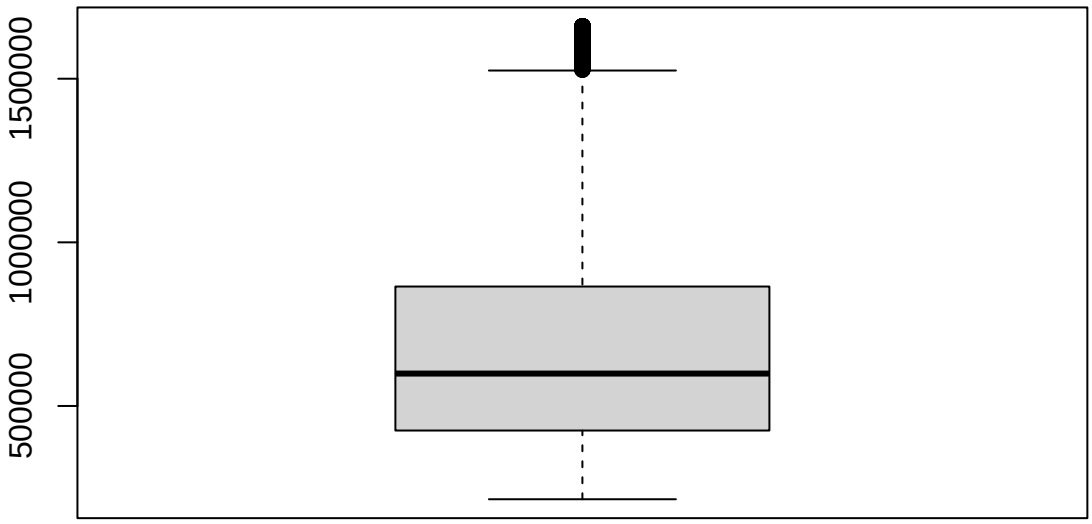

```
boxplot(Final_NYCSales$SALE_PRICE)
```



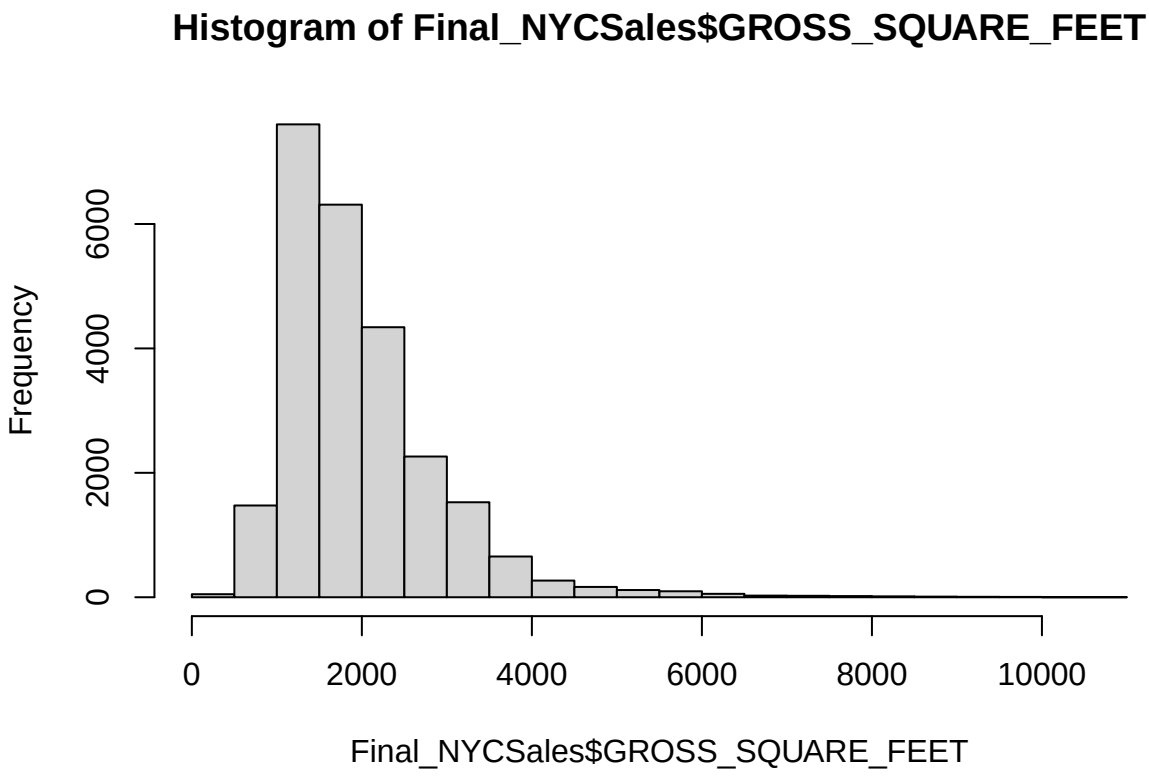

```r
z_scores <- scale(Final_NYCSales$SALE_PRICE)

# Define a threshold for outliers (e.g., 3)
threshold <- 3

# Identify outliers based on the z-scores
outliers <- abs(z_scores) > threshold

# Replace outliers with values at a specific percentile (e.g., 95th and 5th percentiles)
winsorize <- function(x, p) {
  quantiles <- quantile(x, probs = c(p, 1 - p), na.rm = TRUE)
  x[x < quantiles[1]] <- quantiles[1]
  x[x > quantiles[2]] <- quantiles[2]
  x
}

# Apply winsorization to remove outliers from both ends
Final_NYCSales$SALE_PRICE <- winsorize(Final_NYCSales$SALE_PRICE, p = 0.05)

# Alternatively, you can remove outliers by filtering the data
filtered_data <- Final_NYCSales[!outliers, ]

hist(Final_NYCSales$SALE_PRICE)
```



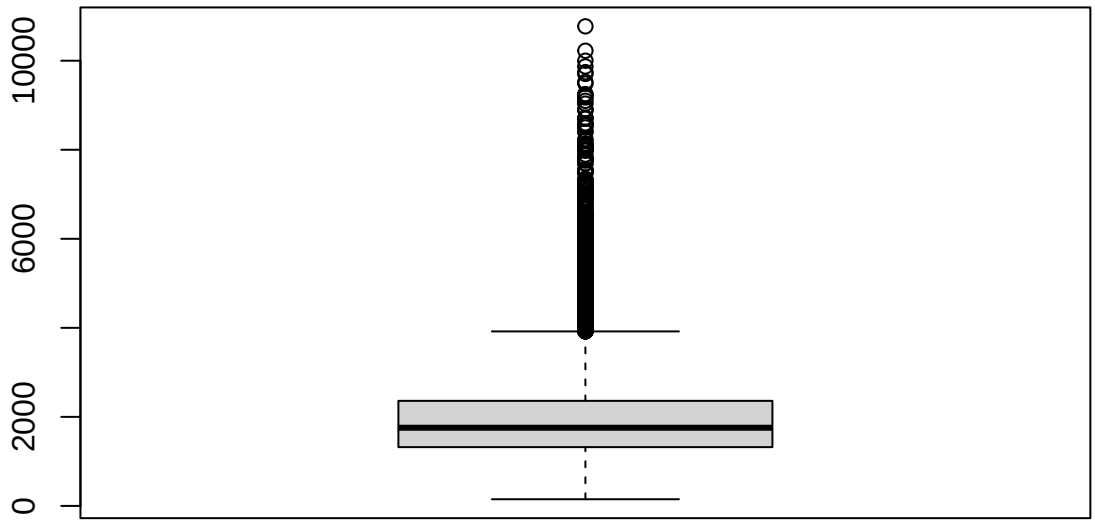

```
boxplot(Final_NYCSales$SALE_PRICE)
```



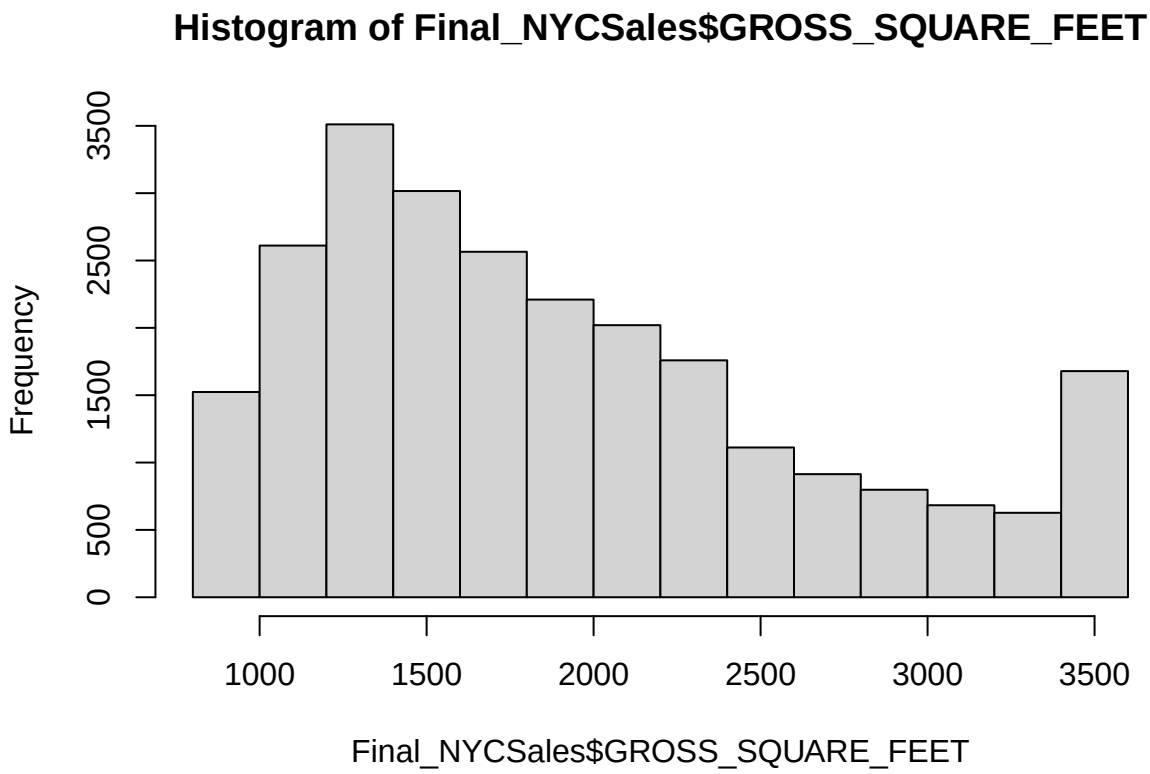

```
#Remove outliers from SALE.PRICE
# Calculate the z-scores
hist(Final_NYCSales$GROSS_SQUARE_FEET)
```



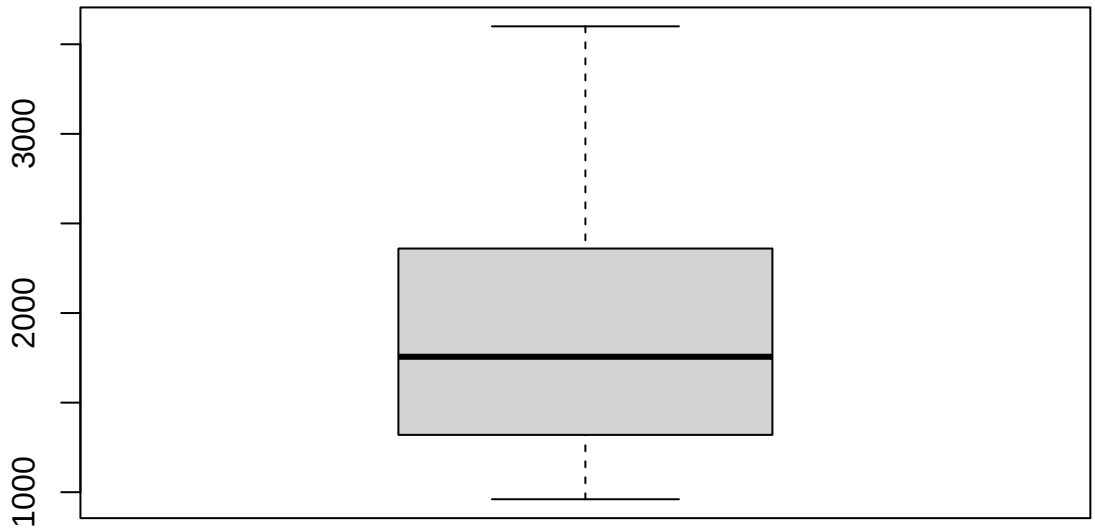

```
boxplot(Final_NYCSales$GROSS_SQUARE_FEET)
```




```r
z_scores <- scale(Final_NYCSales$GROSS_SQUARE_FEET)

# Define a threshold for outliers (e.g., 3)
threshold <- 3

# Identify outliers based on the z-scores
outliers <- abs(z_scores) > threshold

# Replace outliers with values at a specific percentile (e.g., 95th and 5th percentiles)
winsorize <- function(x, p) {
  quantiles <- quantile(x, probs = c(p, 1 - p), na.rm = TRUE)
  x[x < quantiles[1]] <- quantiles[1]
  x[x > quantiles[2]] <- quantiles[2]
  x
}

# Apply winsorization to remove outliers from both ends
Final_NYCSales$GROSS_SQUARE_FEET <- winsorize(Final_NYCSales$GROSS_SQUARE_FEET, p = 0.05)

# Alternatively, you can remove outliers by filtering the data
filtered_data <- Final_NYCSales[!outliers, ]

hist(Final_NYCSales$GROSS_SQUARE_FEET)
```




```
boxplot(Final_NYCSales$GROSS_SQUARE_FEET)
```




```
str(Final_NYCSales)
```

```
## 'data.frame':    25029 obs. of  11 variables:
##  $ BOROUGH                  : Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ BLOCK                    : num  585 1942 1960 2024 2041 ...
##  $ BUILDING_CLASS_AT_PRESENT: Factor w/ 24 levels "A0","A1","A2",..: 6 5 9 6 9 9 9 5 5 5 ...
##  $ RESIDENTIAL_UNITS        : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ LAND_SQUARE_FEET         : num  384 1549 1665 1699 1699 ...
##  $ GROSS_SQUARE_FEET        : num  1152 3036 3200 3600 3536 ...
##  $ TAX_CLASS_AT_TIME_OF_SALE: Factor w/ 3 levels "1","2","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SALE_PRICE               : num  1375000 1660000 1510000 1660000 1595790 ...
##  $ YEAR_SOLD                : num  2016 2016 2017 2017 2017 ...
##  $ MONTH_SOLD               : num  10 11 1 1 8 3 7 8 3 9 ...
##  $ BUILDING_AGE             : num  115 106 107 107 116 118 118 118 118 117 ...
```

```
#FINAL COLUMNS CHOSEN FOR REGRESSION
Final_NYCSales <- Final_NYCSales[,-c(4,5,9,10)]
Final_LDA_Data <- Final_NYCSales
str(Final_NYCSales)
```

```
## 'data.frame':    25029 obs. of  7 variables:
##  $ BOROUGH                  : Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ BLOCK                    : num  585 1942 1960 2024 2041 ...
##  $ BUILDING_CLASS_AT_PRESENT: Factor w/ 24 levels "A0","A1","A2",..: 6 5 9 6 9 9 9 5 5 5 ...
##  $ GROSS_SQUARE_FEET        : num  1152 3036 3200 3600 3536 ...
```

```
##  $ TAX_CLASS_AT_TIME_OF_SALE: Factor w/ 3 levels "1","2","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SALE_PRICE               : num  1375000 1660000 1510000 1660000 1595790 ...
##  $ BUILDING_AGE             : num  115 106 107 107 116 118 118 118 118 117 ...
```

OLS MODEL

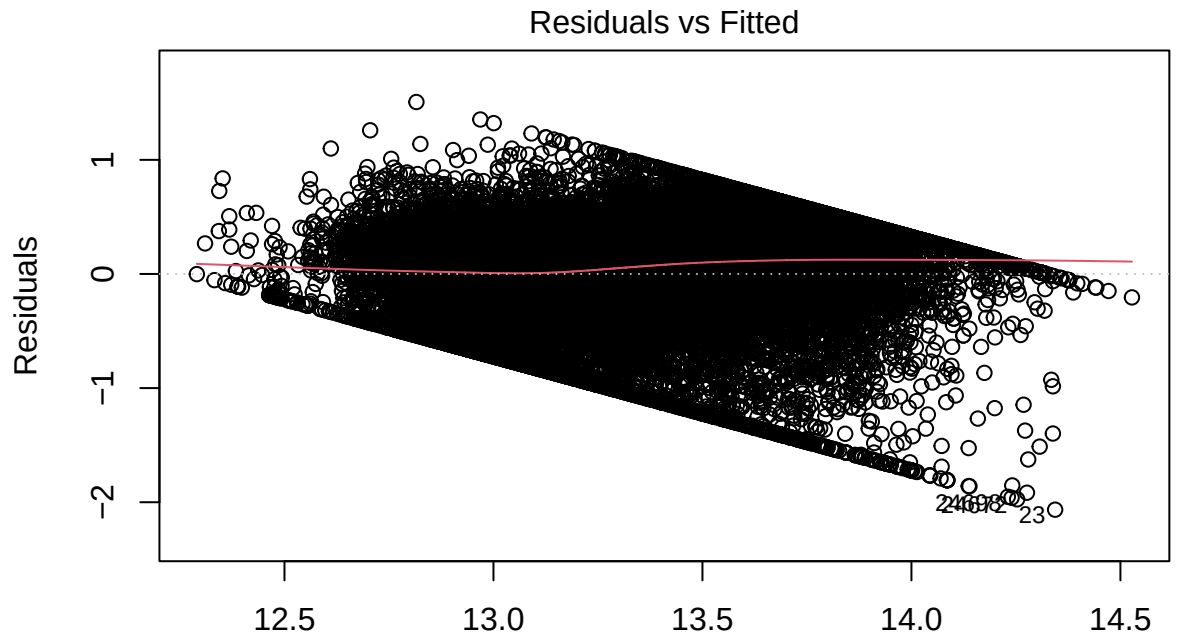
```
#Applying log transforms on SALE PRICE and GROSSSQUAREFEET
Final_NYCSales$GROSS_SQUARE_FEET <- log(Final_NYCSales$GROSS_SQUARE_FEET)
Final_NYCSales$SALE_PRICE <- log(Final_NYCSales$SALE_PRICE)

#OLS MODEL
# Fit the initial linear regression model
OLS_MODEL <- lm(SALE_PRICE ~ BOROUGH + BLOCK + GROSS_SQUARE_FEET + TAX_CLASS_AT_TIME_OF_SALE + BUILDING_
summary(OLS_MODEL)
```

```
##
## Call:
## lm(formula = SALE_PRICE ~ BOROUGH + BLOCK + GROSS_SQUARE_FEET +
##     TAX_CLASS_AT_TIME_OF_SALE + BUILDING_CLASS_AT_PRESENT + BUILDING_AGE,
##     data = Final_NYCSales)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.06546 -0.20546  0.06476  0.27462  1.50675
##
## Coefficients: (1 not defined because of singularities)
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   1.001e+01  8.822e-02 113.488  < 2e-16 ***
## BOROUGH2                     -8.141e-01  3.391e-02 -24.008  < 2e-16 ***
## BOROUGH3                     -2.306e-01  3.321e-02  -6.944 3.91e-12 ***
## BOROUGH4                     -2.727e-01  3.372e-02  -8.089 6.31e-16 ***
## BOROUGH5                     -7.117e-01  3.432e-02 -20.734  < 2e-16 ***
## BLOCK                        -4.509e-05  8.746e-07 -51.557  < 2e-16 ***
## GROSS_SQUARE_FEET             5.714e-01  1.019e-02  56.078  < 2e-16 ***
## TAX_CLASS_AT_TIME_OF_SALE2    3.680e-01  2.073e-01   1.775 0.075930 .
## TAX_CLASS_AT_TIME_OF_SALE4   -2.332e-02  4.149e-01  -0.056 0.955172
## BUILDING_CLASS_AT_PRESENTA1 -1.727e-01  2.465e-02  -7.006 2.51e-12 ***
## BUILDING_CLASS_AT_PRESENTA2 -9.076e-02  2.558e-02  -3.549 0.000388 ***
## BUILDING_CLASS_AT_PRESENTA3  1.731e-01  3.688e-02   4.695 2.68e-06 ***
## BUILDING_CLASS_AT_PRESENTA4 -1.518e-01  4.911e-02  -3.090 0.002001 **
## BUILDING_CLASS_AT_PRESENTA5 -2.756e-01  2.477e-02 -11.125  < 2e-16 ***
## BUILDING_CLASS_AT_PRESENTA6 -4.657e-01  5.384e-02  -8.651  < 2e-16 ***
## BUILDING_CLASS_AT_PRESENTA7  3.078e-01  2.940e-01   1.047 0.295148
## BUILDING_CLASS_AT_PRESENTA9 -2.348e-01  2.750e-02  -8.538  < 2e-16 ***
## BUILDING_CLASS_AT_PRESENTB1 -2.870e-01  2.579e-02 -11.129  < 2e-16 ***
## BUILDING_CLASS_AT_PRESENTB2 -2.722e-01  2.541e-02 -10.712  < 2e-16 ***
## BUILDING_CLASS_AT_PRESENTB3 -2.010e-01  2.548e-02  -7.886 3.25e-15 ***
## BUILDING_CLASS_AT_PRESENTB9 -2.725e-01  2.792e-02  -9.759  < 2e-16 ***
## BUILDING_CLASS_AT_PRESENTC0 -3.157e-01  2.664e-02 -11.849  < 2e-16 ***
## BUILDING_CLASS_AT_PRESENTC1 -4.305e-01  2.118e-01  -2.033 0.042080 *
## BUILDING_CLASS_AT_PRESENTC2 -6.070e-01  2.099e-01  -2.892 0.003836 **
## BUILDING_CLASS_AT_PRESENTC3 -6.921e-01  2.095e-01  -3.303 0.000956 ***
## BUILDING_CLASS_AT_PRESENTC4 -5.533e-01  2.254e-01  -2.455 0.014099 *
```

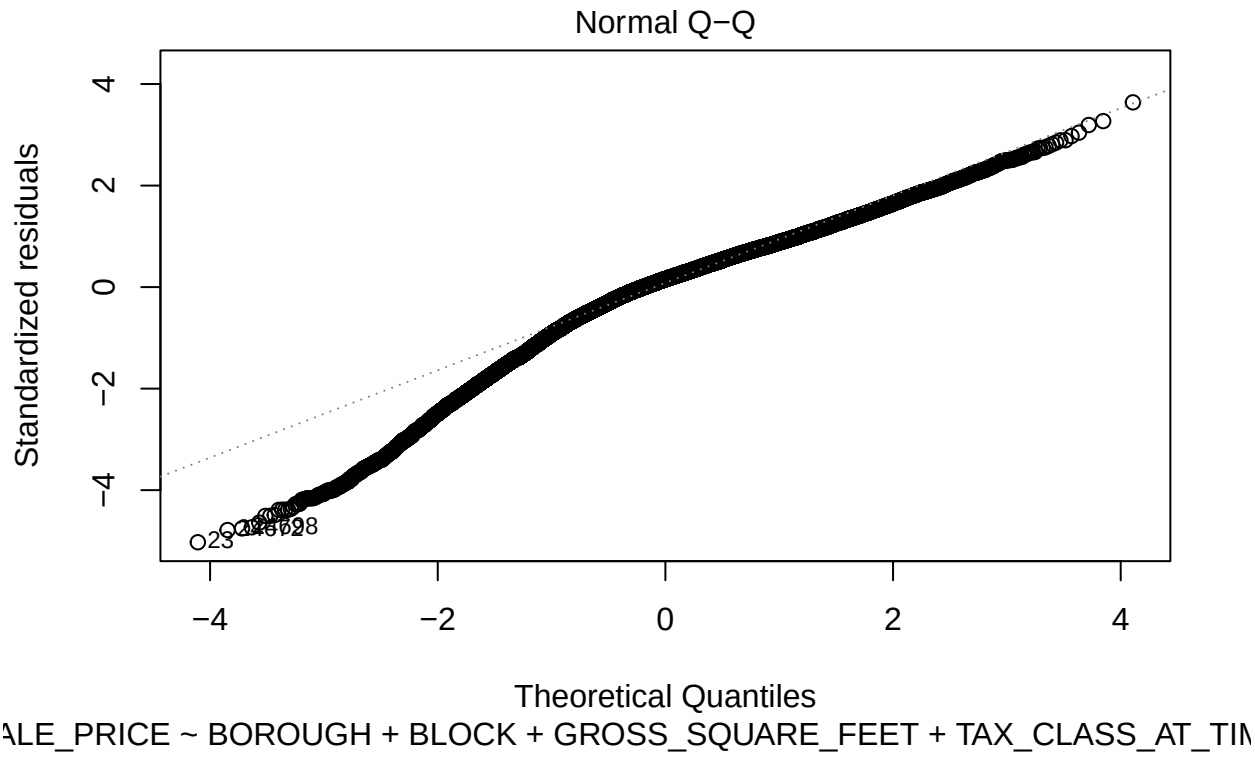
```
## BUILDING_CLASS_AT_PRESENTC5 -6.576e-01  2.195e-01  -2.997 0.002733 **
## BUILDING_CLASS_AT_PRESENTC9 -9.374e-01  2.790e-01  -3.360 0.000780 ***
## BUILDING_CLASS_AT_PRESENTD1 -2.868e-01  3.598e-01  -0.797 0.425321
## BUILDING_CLASS_AT_PRESENTG0 -5.158e-01  1.868e-01  -2.761 0.005760 **
## BUILDING_CLASS_AT_PRESENTZ0 -5.187e-01  4.149e-01  -1.250 0.211170
## BUILDING_CLASS_AT_PRESENTZ9         NA         NA      NA       NA
## BUILDING_AGE                -1.140e-03  1.072e-04 -10.640  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4141 on 24997 degrees of freedom
## Multiple R-squared:  0.3968, Adjusted R-squared:  0.396
## F-statistic: 530.4 on 31 and 24997 DF,  p-value: < 2.2e-16
```

```
plot(OLS_MODEL)
```

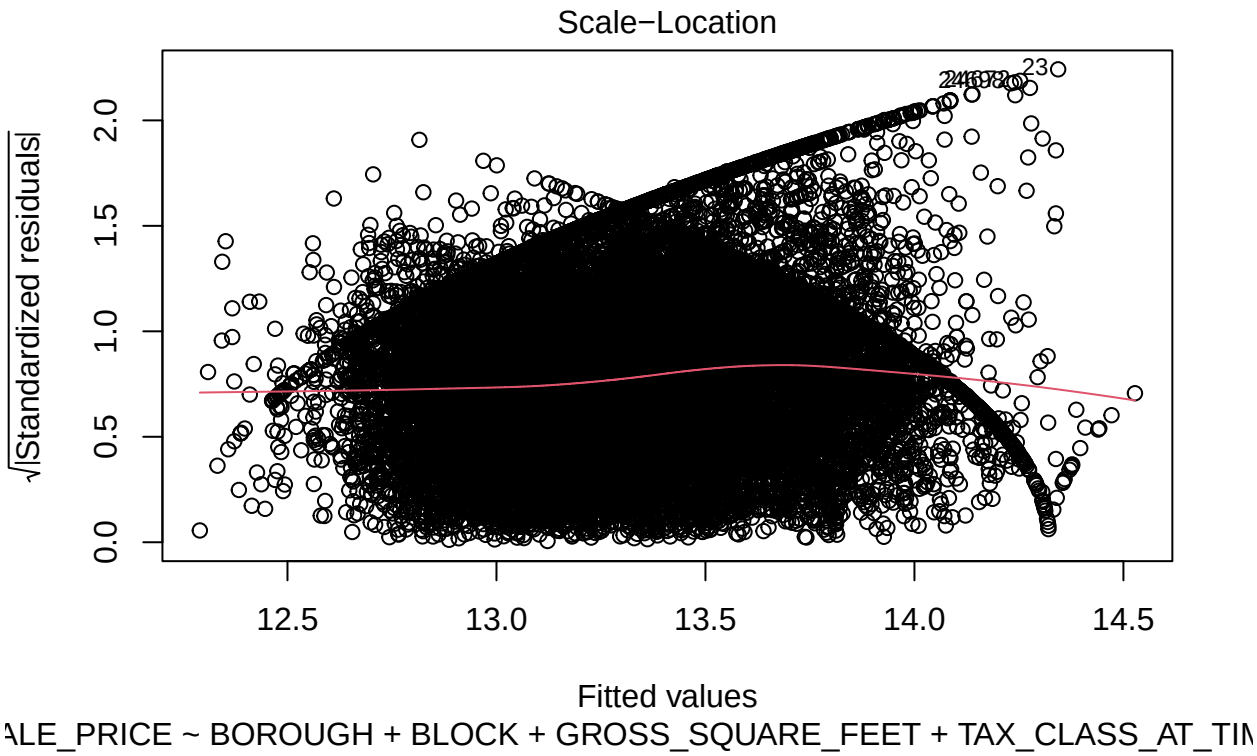
```
## Warning: not plotting observations with leverage one:
##   25027, 25029
```

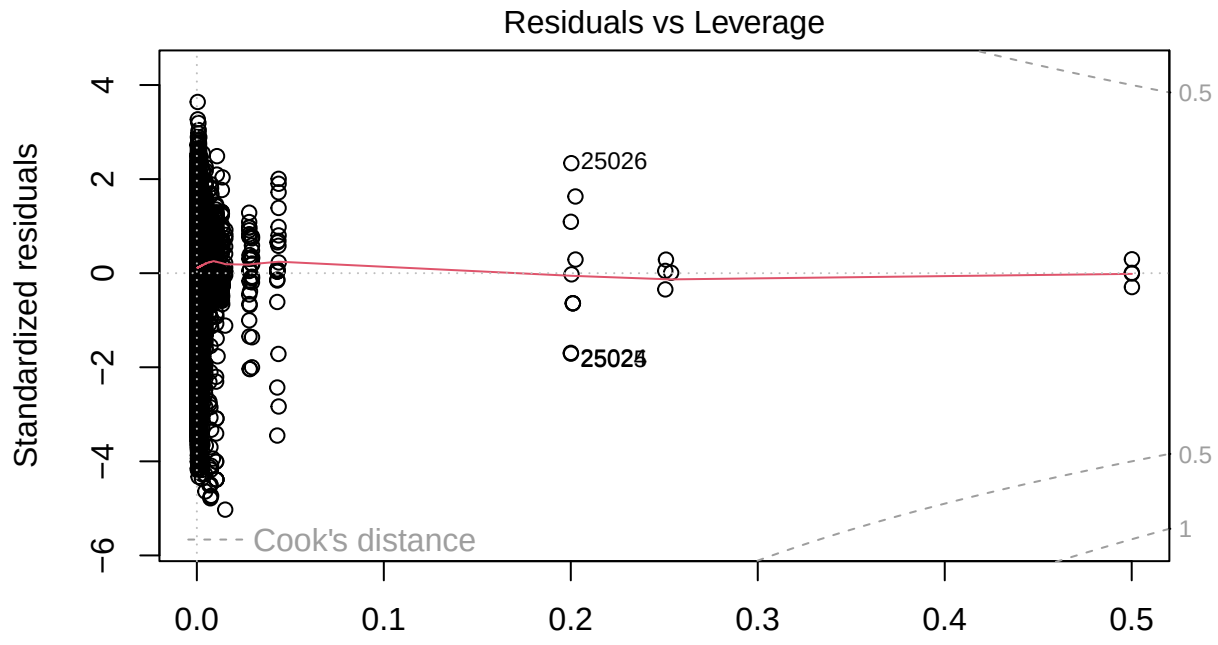


ALE_PRICE ~ BOROUGH + BLOCK + GROSS_SQUARE_FEET + TAX_CLASS_AT_TI



ALE_PRICE ~ BOROUGH + BLOCK + GROSS_SQUARE_FEET + TAX_CLASS_AT_TI


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
12.5
13.0
13.5
14.0
14.5
Fitted values
ALE_PRICE ~ BOROUGH + BLOCK + GROSS_SQUARE_FEET + TAX_CLASS_AT_TI



ALE_PRICE ~ BOROUGH + BLOCK + GROSS_SQUARE_FEET + TAX_CLASS_AT_TIM

USING DUMMY VARIABLES ON BOROUGH, BUILDING_CLASS_AT_PRESENT, TAX_CLASS_AT_TIME_OF_SALE

```
library(caret)
```

```
## Loading required package: lattice
```

```
# Identify factor variables
factor_cols <- sapply(Final_NYCSales, is.factor)

# Convert factor variables to dummy variables
dummy <- dummyVars(paste("~.", collapse = "+"), data = Final_NYCSales[, factor_cols])
dummy_data <- predict(dummy, newdata = Final_NYCSales[, factor_cols])

# Remove original factor variables from the dataset
Final_NYCSales <- Final_NYCSales[, !factor_cols]

# Add dummy variables to the dataset
Final_NYCSales <- cbind(Final_NYCSales, dummy_data)

# Print the updated dataset
head(Final_NYCSales)
```

```
##   BLOCK GROSS_SQUARE_FEET SALE_PRICE BUILDING_AGE BOROUGH.1 BOROUGH.2 BOROUGH.3
## 1   585          7.049255   14.13396          115         1         0         0
```

```
## 2  1942          8.018296   14.32233          106         1         0         0
## 3  1960          8.070906   14.22762          107         1         0         0
## 4  2024          8.188689   14.32233          107         1         0         0
## 5  2041          8.170751   14.28288          116         1         0         0
## 6  2042          8.188689   14.32233          118         1         0         0
##   BOROUGH.4 BOROUGH.5 BUILDING_CLASS_AT_PRESENT.A0 BUILDING_CLASS_AT_PRESENT.A1
## 1         0         0                            0                            0
## 2         0         0                            0                            0
## 3         0         0                            0                            0
## 4         0         0                            0                            0
## 5         0         0                            0                            0
## 6         0         0                            0                            0
##   BUILDING_CLASS_AT_PRESENT.A2 BUILDING_CLASS_AT_PRESENT.A3
## 1                            0                            0
## 2                            0                            0
## 3                            0                            0
## 4                            0                            0
## 5                            0                            0
## 6                            0                            0
##   BUILDING_CLASS_AT_PRESENT.A4 BUILDING_CLASS_AT_PRESENT.A5
## 1                            0                            1
## 2                            1                            0
## 3                            0                            0
## 4                            0                            1
## 5                            0                            0
## 6                            0                            0
##   BUILDING_CLASS_AT_PRESENT.A6 BUILDING_CLASS_AT_PRESENT.A7
## 1                            0                            0
## 2                            0                            0
## 3                            0                            0
## 4                            0                            0
## 5                            0                            0
## 6                            0                            0
##   BUILDING_CLASS_AT_PRESENT.A9 BUILDING_CLASS_AT_PRESENT.B1
## 1                            0                            0
## 2                            0                            0
## 3                            1                            0
## 4                            0                            0
## 5                            1                            0
## 6                            1                            0
##   BUILDING_CLASS_AT_PRESENT.B2 BUILDING_CLASS_AT_PRESENT.B3
## 1                            0                            0
## 2                            0                            0
## 3                            0                            0
## 4                            0                            0
## 5                            0                            0
## 6                            0                            0
##   BUILDING_CLASS_AT_PRESENT.B9 BUILDING_CLASS_AT_PRESENT.C0
## 1                            0                            0
## 2                            0                            0
## 3                            0                            0
## 4                            0                            0
## 5                            0                            0
## 6                            0                            0
```

```
##   BUILDING_CLASS_AT_PRESENT.C1 BUILDING_CLASS_AT_PRESENT.C2
## 1                            0                            0
## 2                            0                            0
## 3                            0                            0
## 4                            0                            0
## 5                            0                            0
## 6                            0                            0
##   BUILDING_CLASS_AT_PRESENT.C3 BUILDING_CLASS_AT_PRESENT.C4
## 1                            0                            0
## 2                            0                            0
## 3                            0                            0
## 4                            0                            0
## 5                            0                            0
## 6                            0                            0
##   BUILDING_CLASS_AT_PRESENT.C5 BUILDING_CLASS_AT_PRESENT.C9
## 1                            0                            0
## 2                            0                            0
## 3                            0                            0
## 4                            0                            0
## 5                            0                            0
## 6                            0                            0
##   BUILDING_CLASS_AT_PRESENT.D1 BUILDING_CLASS_AT_PRESENT.G0
## 1                            0                            0
## 2                            0                            0
## 3                            0                            0
## 4                            0                            0
## 5                            0                            0
## 6                            0                            0
##   BUILDING_CLASS_AT_PRESENT.Z0 BUILDING_CLASS_AT_PRESENT.Z9
## 1                            0                            0
## 2                            0                            0
## 3                            0                            0
## 4                            0                            0
## 5                            0                            0
## 6                            0                            0
##   TAX_CLASS_AT_TIME_OF_SALE.1 TAX_CLASS_AT_TIME_OF_SALE.2
## 1                           1                           0
## 2                           1                           0
## 3                           1                           0
## 4                           1                           0
## 5                           1                           0
## 6                           1                           0
##   TAX_CLASS_AT_TIME_OF_SALE.4
## 1                           0
## 2                           0
## 3                           0
## 4                           0
## 5                           0
## 6                           0
```

```
str(Final_NYCSales)
```

```
## 'data.frame':    25029 obs. of  36 variables:
##  $ BLOCK                        : num  585 1942 1960 2024 2041 ...
```

```
##  $ GROSS_SQUARE_FEET           : num  7.05 8.02 8.07 8.19 8.17 ...
##  $ SALE_PRICE                  : num  14.1 14.3 14.2 14.3 14.3 ...
##  $ BUILDING_AGE                : num  115 106 107 107 116 118 118 118 118 117 ...
##  $ BOROUGH.1                   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ BOROUGH.2                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BOROUGH.3                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BOROUGH.4                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BOROUGH.5                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.A0: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.A1: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.A2: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.A3: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.A4: num  0 1 0 0 0 0 0 1 1 1 ...
##  $ BUILDING_CLASS_AT_PRESENT.A5: num  1 0 0 1 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.A6: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.A7: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.A9: num  0 0 1 0 1 1 1 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.B1: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.B2: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.B3: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.B9: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.C0: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.C1: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.C2: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.C3: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.C4: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.C5: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.C9: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.D1: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.G0: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.Z0: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BUILDING_CLASS_AT_PRESENT.Z9: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ TAX_CLASS_AT_TIME_OF_SALE.1 : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ TAX_CLASS_AT_TIME_OF_SALE.2 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ TAX_CLASS_AT_TIME_OF_SALE.4 : num  0 0 0 0 0 0 0 0 0 0 ...
```

RIDGE REGRESSION

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.2.3
```

```
## Loading required package: Matrix
```

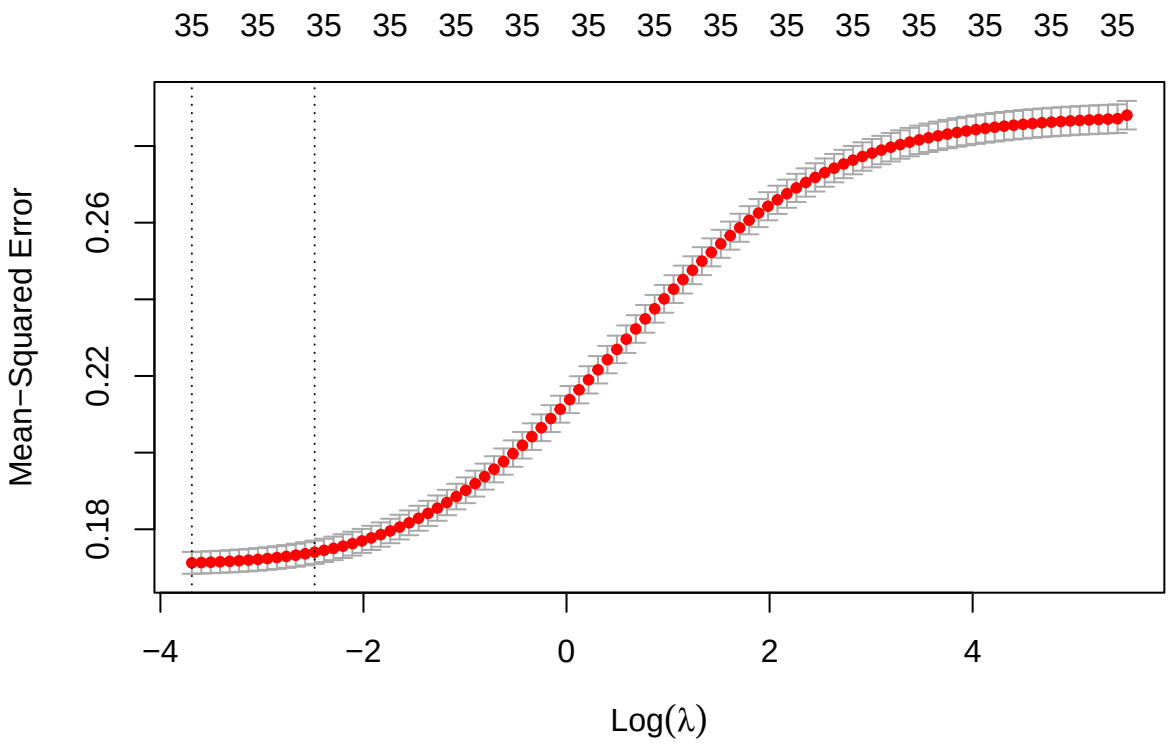
```
## Warning: package 'Matrix' was built under R version 4.2.2
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-7
```

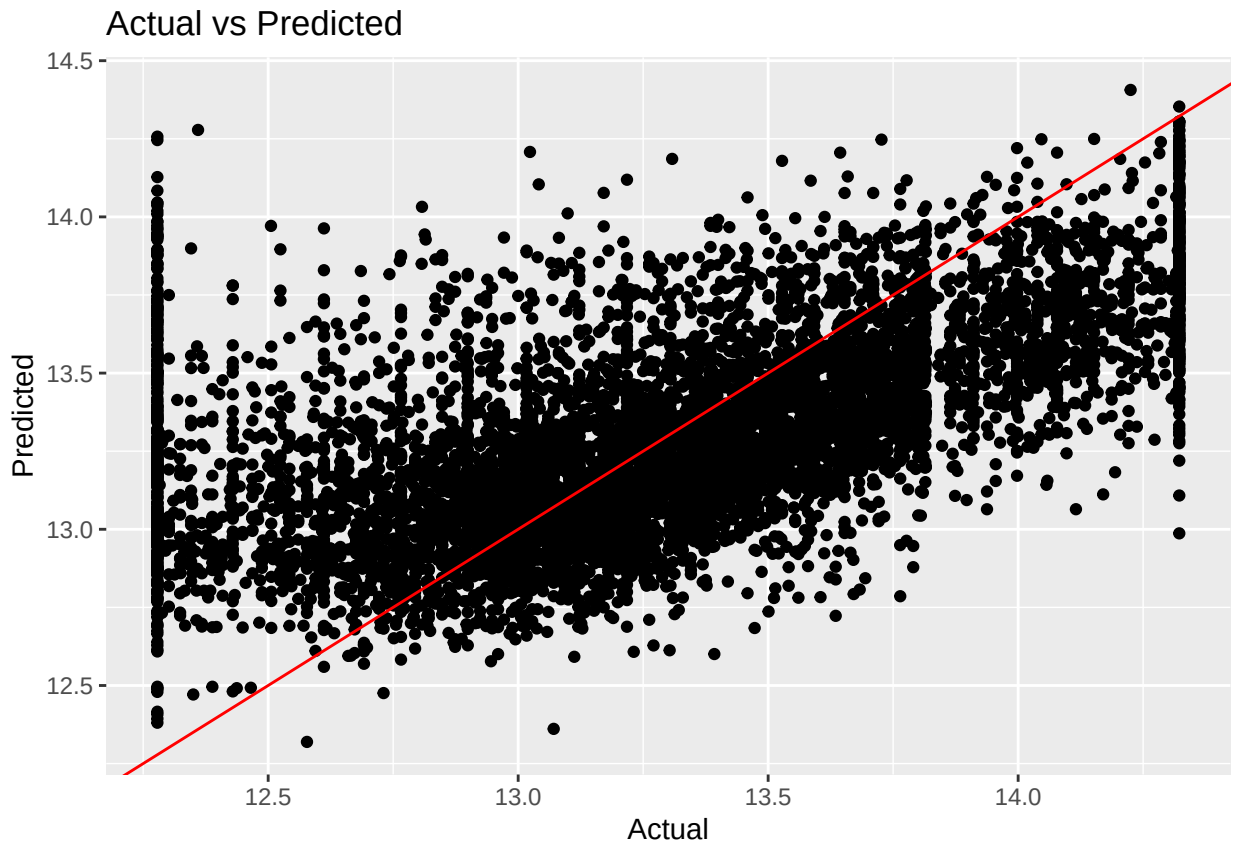
```
library(ggplot2)

# Split the data into training and test sets
set.seed(123)  # Set a seed for reproducibility
train_indices <- sample(1:nrow(Final_NYCSales), 0.7 * nrow(Final_NYCSales))  # 70% for training
train_data <- Final_NYCSales[train_indices, ]
test_data <- Final_NYCSales[-train_indices, ]

# Prepare the training data
train_x <- train_data[, -3]  # Features (excluding target variable)
train_y <- train_data[, 3]   # Target variable

# Perform ridge regression with cross-validation
ridge_model <- cv.glmnet(x = as.matrix(train_x), y = train_y, alpha = 0, nfolds = 5)
plot(ridge_model)
```



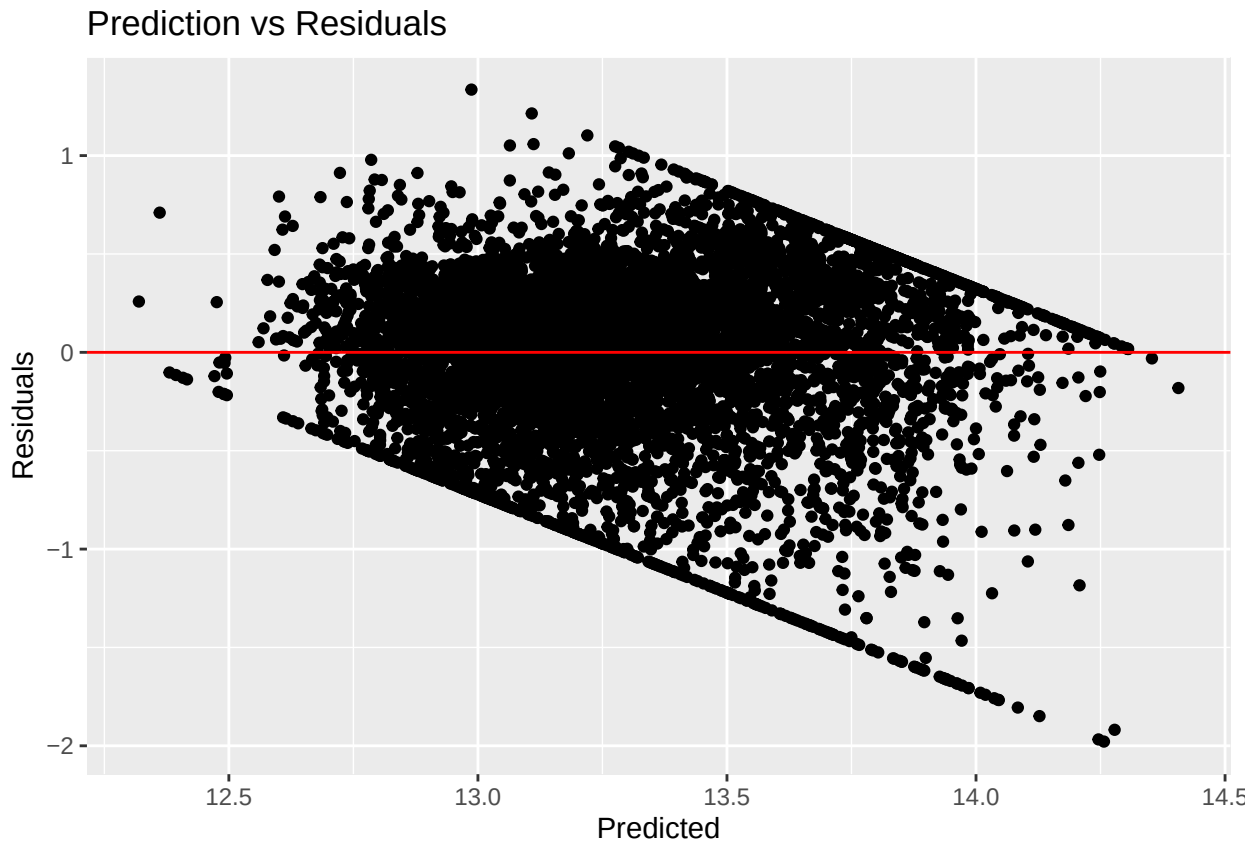

```
# Find the optimal lambda value
best_lambda <- ridge_model$lambda.min
best_lambda
```

```
## [1] 0.02495994
```

```r
# Train the ridge regression model with the optimal lambda
final_model <- glmnet(x = as.matrix(train_x), y = train_y, alpha = 0, lambda = best_lambda)


# Prepare the test data
test_x <- test_data[, -3]  # Features (excluding target variable)
test_y <- test_data[, 3]   # Target variable

# Predict using the trained model
predictions <- predict(final_model, newx = as.matrix(test_x))

# Store predictions in a separate object
predicted_values <- as.vector(predictions)

# Create a data frame for plotting
plot_data <- data.frame(Actual = test_y, Predicted = predicted_values)

# Plot actual vs predicted
ggplot(plot_data, aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(x = "Actual", y = "Predicted") +
  ggtitle("Actual vs Predicted")
```



```r
# Calculate residuals
residuals <- test_y - predicted_values

# Plot prediction vs residuals
ggplot(plot_data, aes(x = Predicted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  labs(x = "Predicted", y = "Residuals") +
  ggtitle("Prediction vs Residuals")
```




```r
# Evaluate the model
mse <- mean((predicted_values - test_y)^2)
rmseRidge <- sqrt(mse)

# Calculate R-squared
y_mean <- mean(test_y)  # Mean of the actual target variable
ss_total <- sum((test_y - y_mean)^2)  # Total sum of squares
ss_residual <- sum((test_y - predicted_values)^2)  # Residual sum of squares
r_squared <- 1 - (ss_residual / ss_total)

# Calculate adjusted R-squared
n <- nrow(test_data)  # Number of observations
p <- ncol(test_x)  # Number of predictors
adj_r_squared <- 1 - (1 - r_squared) * ((n - 1) / (n - p - 1))
```

```r
# Print the results
print(paste("RMSE:", rmseRidge))
```

```
## [1] "RMSE: 0.417293689493776"
```

```r
print(paste("R-squared:", r_squared))
```

```
## [1] "R-squared: 0.364829192699975"
```

```r
print(paste("Adjusted R-squared:", adj_r_squared))
```

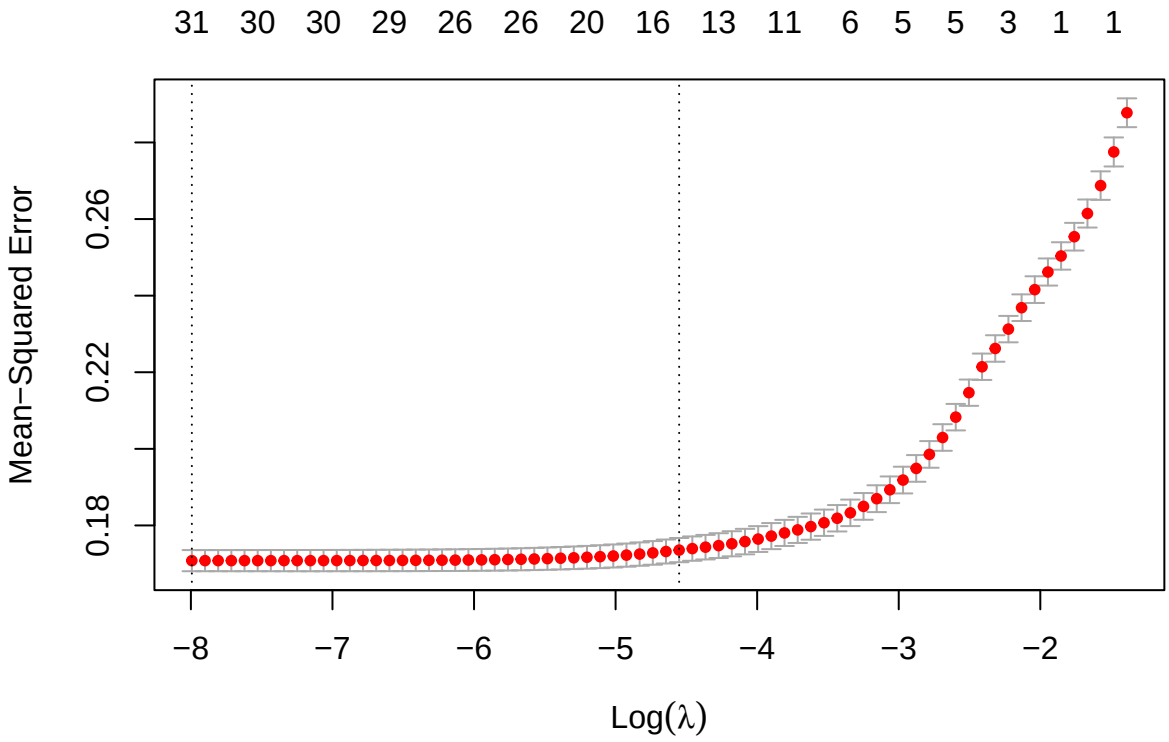
```
## [1] "Adjusted R-squared: 0.361854352842421"
```

LASSO REGRESSION

```r
#LASSO regression
library(glmnet)
library(ggplot2)

# Split the data into training and test sets
set.seed(123)  # Set a seed for reproducibility
train_indices <- sample(1:nrow(Final_NYCSales), 0.7 * nrow(Final_NYCSales))  # 70% for training
train_data <- Final_NYCSales[train_indices, ]
test_data <- Final_NYCSales[-train_indices, ]

# Prepare the training data
train_x <- train_data[, -3]  # Features (excluding target variable)
train_y <- train_data[, 3]   # Target variable

# Perform Lasso regression with cross-validation
lasso_model <- cv.glmnet(x = as.matrix(train_x), y = train_y, alpha = 1, nfolds = 5)
lasso_model
```

```
##
## Call:  cv.glmnet(x = as.matrix(train_x), y = train_y, nfolds = 5, alpha = 1)
##
## Measure: Mean-Squared Error
##
##       Lambda Index Measure       SE Nonzero
## min 0.000338    72  0.1708 0.002759      31
## 1se 0.010556    35  0.1736 0.003161      15
```

```r
plot(lasso_model)
```




```
# Find the optimal lambda value
best_lambda <- lasso_model$lambda.min

# Train the Lasso regression model with the optimal lambda
final_model <- glmnet(x = as.matrix(train_x), y = train_y, alpha = 1, lambda = best_lambda)
summary(final_model)
```

```
##           Length Class     Mode
## a0         1     -none-    numeric
## beta      35     dgCMatrix S4
## df         1     -none-    numeric
## dim        2     -none-    numeric
## lambda     1     -none-    numeric
## dev.ratio  1     -none-    numeric
## nulldev    1     -none-    numeric
## npasses    1     -none-    numeric
## jerr       1     -none-    numeric
## offset     1     -none-    logical
## call       5     -none-    call
## nobs       1     -none-    numeric
```

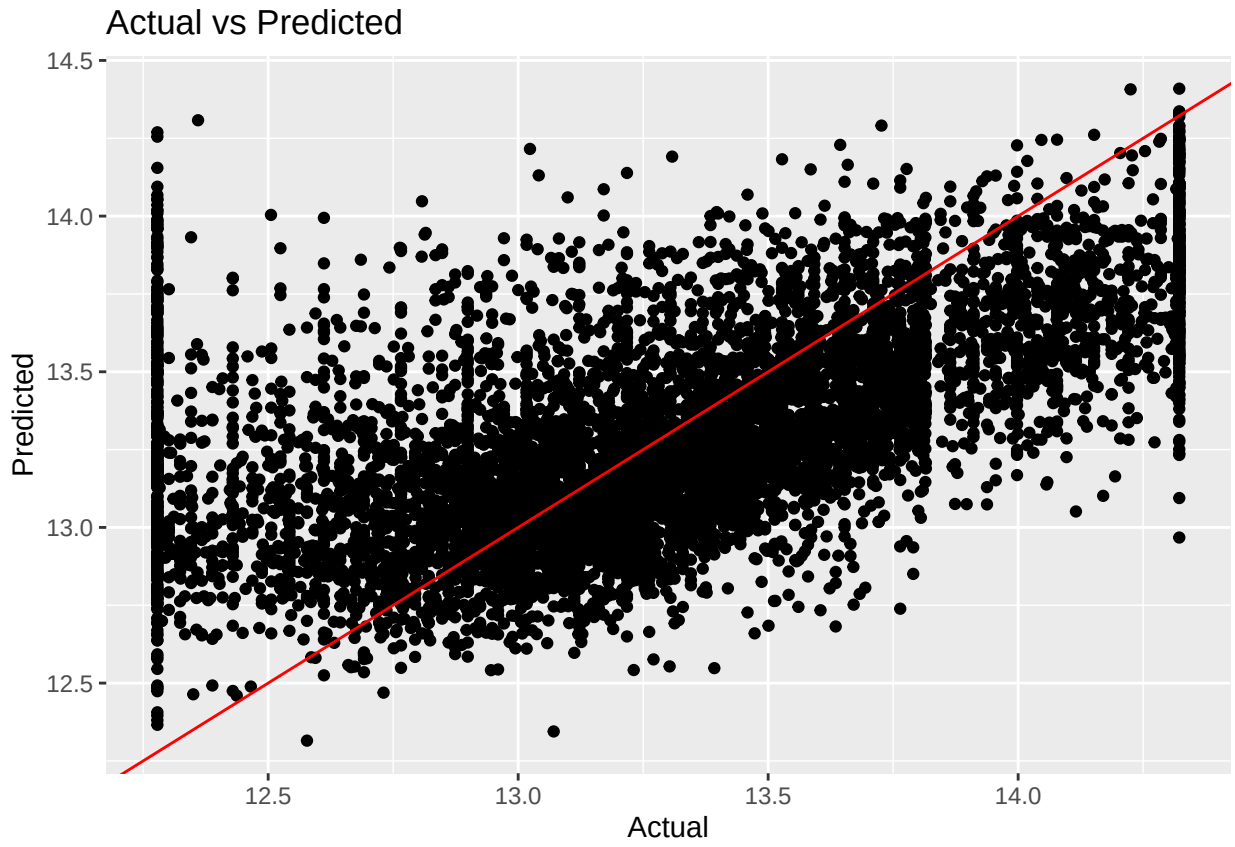
```
#Variable effectiveness
# Get the coefficients of the selected variables
lasso_coefficients <- coef(final_model)
```

```r
# Create a data frame to store the variable names and coefficients
variable_summary <- data.frame(
  Variable = rownames(lasso_coefficients),
  Coefficient = lasso_coefficients[,1]
)

# Sort the variables by the absolute value of the coefficients
variable_summary <- variable_summary[order(abs(variable_summary$Coefficient), decreasing = TRUE), ]

# Print the summary of variable effectiveness
print(variable_summary)
```

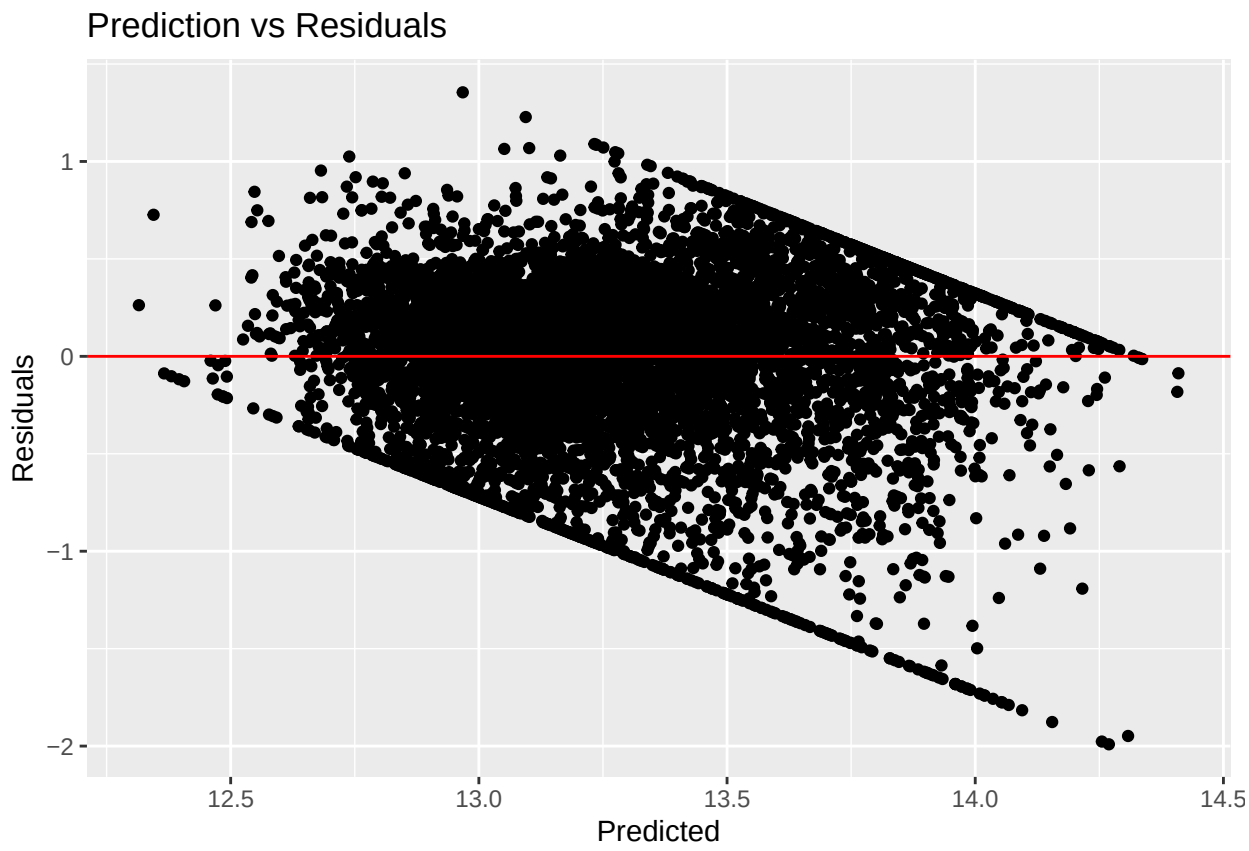
```
##                                                 Variable   Coefficient
## (Intercept)                                   (Intercept)  9.182729e+00
## GROSS_SQUARE_FEET                       GROSS_SQUARE_FEET  5.869721e-01
## BUILDING_CLASS_AT_PRESENT.A7 BUILDING_CLASS_AT_PRESENT.A7  5.205271e-01
## BOROUGH.1                                       BOROUGH.1  4.627687e-01
## BUILDING_CLASS_AT_PRESENT.A3 BUILDING_CLASS_AT_PRESENT.A3  4.509386e-01
## BOROUGH.2                                       BOROUGH.2 -3.586869e-01
## BUILDING_CLASS_AT_PRESENT.C9 BUILDING_CLASS_AT_PRESENT.C9 -2.959927e-01
## BUILDING_CLASS_AT_PRESENT.D1 BUILDING_CLASS_AT_PRESENT.D1  2.933057e-01
## BOROUGH.5                                       BOROUGH.5 -2.618110e-01
## BUILDING_CLASS_AT_PRESENT.A0 BUILDING_CLASS_AT_PRESENT.A0  2.560440e-01
## BUILDING_CLASS_AT_PRESENT.G0 BUILDING_CLASS_AT_PRESENT.G0 -2.281638e-01
## BOROUGH.3                                       BOROUGH.3  2.269955e-01
## BUILDING_CLASS_AT_PRESENT.Z0 BUILDING_CLASS_AT_PRESENT.Z0 -2.132064e-01
## BUILDING_CLASS_AT_PRESENT.Z9 BUILDING_CLASS_AT_PRESENT.Z9  2.015755e-01
## BUILDING_CLASS_AT_PRESENT.C1 BUILDING_CLASS_AT_PRESENT.C1  1.998883e-01
## BUILDING_CLASS_AT_PRESENT.A6 BUILDING_CLASS_AT_PRESENT.A6 -1.923464e-01
## BOROUGH.4                                       BOROUGH.4  1.843386e-01
## BUILDING_CLASS_AT_PRESENT.A2 BUILDING_CLASS_AT_PRESENT.A2  1.619847e-01
## BUILDING_CLASS_AT_PRESENT.A1 BUILDING_CLASS_AT_PRESENT.A1  8.306862e-02
## BUILDING_CLASS_AT_PRESENT.A4 BUILDING_CLASS_AT_PRESENT.A4  6.891389e-02
## BUILDING_CLASS_AT_PRESENT.C3 BUILDING_CLASS_AT_PRESENT.C3 -6.054936e-02
## BUILDING_CLASS_AT_PRESENT.B3 BUILDING_CLASS_AT_PRESENT.B3  5.624166e-02
## BUILDING_CLASS_AT_PRESENT.C0 BUILDING_CLASS_AT_PRESENT.C0 -4.646233e-02
## BUILDING_CLASS_AT_PRESENT.B1 BUILDING_CLASS_AT_PRESENT.B1 -2.801061e-02
## BUILDING_CLASS_AT_PRESENT.A9 BUILDING_CLASS_AT_PRESENT.A9  2.556338e-02
## BUILDING_CLASS_AT_PRESENT.C5 BUILDING_CLASS_AT_PRESENT.C5 -1.925524e-02
## BUILDING_CLASS_AT_PRESENT.B2 BUILDING_CLASS_AT_PRESENT.B2 -1.771880e-02
## BUILDING_CLASS_AT_PRESENT.C2 BUILDING_CLASS_AT_PRESENT.C2  1.660814e-02
## BUILDING_CLASS_AT_PRESENT.B9 BUILDING_CLASS_AT_PRESENT.B9 -1.636450e-02
## BUILDING_CLASS_AT_PRESENT.A5 BUILDING_CLASS_AT_PRESENT.A5 -1.594998e-02
## BUILDING_CLASS_AT_PRESENT.C4 BUILDING_CLASS_AT_PRESENT.C4  6.480429e-03
## BUILDING_AGE                                 BUILDING_AGE -1.169838e-03
## TAX_CLASS_AT_TIME_OF_SALE.1   TAX_CLASS_AT_TIME_OF_SALE.1 -1.316929e-04
## BLOCK                                           BLOCK -4.495577e-05
## TAX_CLASS_AT_TIME_OF_SALE.2   TAX_CLASS_AT_TIME_OF_SALE.2  0.000000e+00
## TAX_CLASS_AT_TIME_OF_SALE.4   TAX_CLASS_AT_TIME_OF_SALE.4  0.000000e+00
```

```r
# Prepare the test data
test_x <- test_data[, -3]  # Features (excluding target variable)
test_y <- test_data[, 3]   # Target variable
```

```r
# Predict using the trained model
predictions <- predict(final_model, newx = as.matrix(test_x))

# Store predictions in a separate object
predicted_values <- as.vector(predictions)

# Create a data frame for plotting
plot_data <- data.frame(Actual = test_y, Predicted = predicted_values)

# Plot actual vs predicted
ggplot(plot_data, aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(x = "Actual", y = "Predicted") +
  ggtitle("Actual vs Predicted")
```




```r
# Calculate residuals
residuals <- test_y - predicted_values

# Plot prediction vs residuals
ggplot(plot_data, aes(x = Predicted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  labs(x = "Predicted", y = "Residuals") +
  ggtitle("Prediction vs Residuals")
```




```
# Evaluate the model
mse <- mean((predicted_values - test_y)^2)
rmseLasso <- sqrt(mse)

# Calculate R-squared
y_mean <- mean(test_y)  # Mean of the actual target variable
ss_total <- sum((test_y - y_mean)^2)  # Total sum of squares
ss_residual <- sum((test_y - predicted_values)^2)  # Residual sum of squares
r_squared <- 1 - (ss_residual / ss_total)

# Calculate adjusted R-squared
n <- nrow(test_data)  # Number of observations
p <- ncol(test_x)  # Number of predictors
adj_r_squared <- 1 - (1 - r_squared) * ((n - 1) / (n - p - 1))

# Print the results
print(paste("RMSE:", rmseLasso))
```

```
## [1] "RMSE: 0.417539523681832"
```

```
print(paste("R-squared:", r_squared))
```

```
## [1] "R-squared: 0.364080594301084"
```

```r
print(paste("Adjusted R-squared:", adj_r_squared))
```

```
## [1] "Adjusted R-squared: 0.361102248362444"
```

With an RMSE of 0.417 and an adjusted R-squared of 0.362, as well as an R-squared of 0.365, the model appears to have a moderate level of accuracy and goodness of fit.

The RMSE indicates the average difference between the predicted values and the actual values, with a lower value indicating better predictive performance. In this case, an RMSE of 0.417 suggests that, on average, the model's predictions have an error of approximately 0.417 units.

The adjusted R-squared value of 0.362 suggests that the model explains about 36.2% of the variance in the dependent variable after accounting for the number of predictors in the model. This indicates that the model captures a moderate amount of the variability in the data.

The R-squared value of 0.365 indicates that the model explains approximately 36.5% of the variance in the dependent variable, without adjusting for the number of predictors. While this value is slightly higher than the adjusted R-squared, it should be interpreted with caution as it doesn't account for the complexity of the model.

Overall, an adjusted R-squared of 0.362 suggests that the model has some ability to explain the variation in the data, but there is still room for improvement.

CA

```r
max(Final_LDA_Data$SALE_PRICE)
```

```
## [1] 1660000
```

```r
min(Final_LDA_Data$SALE_PRICE)
```

```
## [1] 215000
```

```r
# Define the number of categories and the range of values
num_categories <- 5
min_price <- min(Final_LDA_Data$SALE_PRICE)
max_price <- max(Final_LDA_Data$SALE_PRICE)

# Calculate the bin size
bin_size <- (max_price - min_price) / num_categories

# Create breaks and labels for binning
breaks <- seq(min_price, max_price, bin_size)
labels <- c("low", "mid1", "mid2", "mid3", "high")

# Create a new categorical variable for binned SALE_PRICE
Final_LDA_Data$SALE_PRICE_BIN <- cut(Final_LDA_Data$SALE_PRICE,
                                     breaks = breaks,
                                     labels = labels,
                                     include.lowest = TRUE)

# View the updated dataset with the binned SALE_PRICE variable
head(Final_LDA_Data)
```

```
##   BOROUGH BLOCK BUILDING_CLASS_AT_PRESENT GROSS_SQUARE_FEET
## 1       1   585                       A5              1152
## 2       1  1942                       A4              3036
## 3       1  1960                       A9              3200
## 4       1  2024                       A5              3600
## 5       1  2041                       A9              3536
## 6       1  2042                       A9              3600
##   TAX_CLASS_AT_TIME_OF_SALE SALE_PRICE BUILDING_AGE SALE_PRICE_BIN
## 1                         1    1375000          115           high
## 2                         1    1660000          106           high
## 3                         1    1510000          107           high
## 4                         1    1660000          107           high
## 5                         1    1595790          116           high
## 6                         1    1660000          118           high
```

```
NYC_CA_DATA <- Final_LDA_Data[, c("SALE_PRICE_BIN", "BUILDING_CLASS_AT_PRESENT","TAX_CLASS_AT_TIME_OF_SA

head(NYC_CA_DATA)
```

```
##   SALE_PRICE_BIN BUILDING_CLASS_AT_PRESENT TAX_CLASS_AT_TIME_OF_SALE BOROUGH
## 1           high                       A5                         1       1
## 2           high                       A4                         1       1
## 3           high                       A9                         1       1
## 4           high                       A5                         1       1
## 5           high                       A9                         1       1
## 6           high                       A9                         1       1
```

```
# Load the required library
library(ca)
```

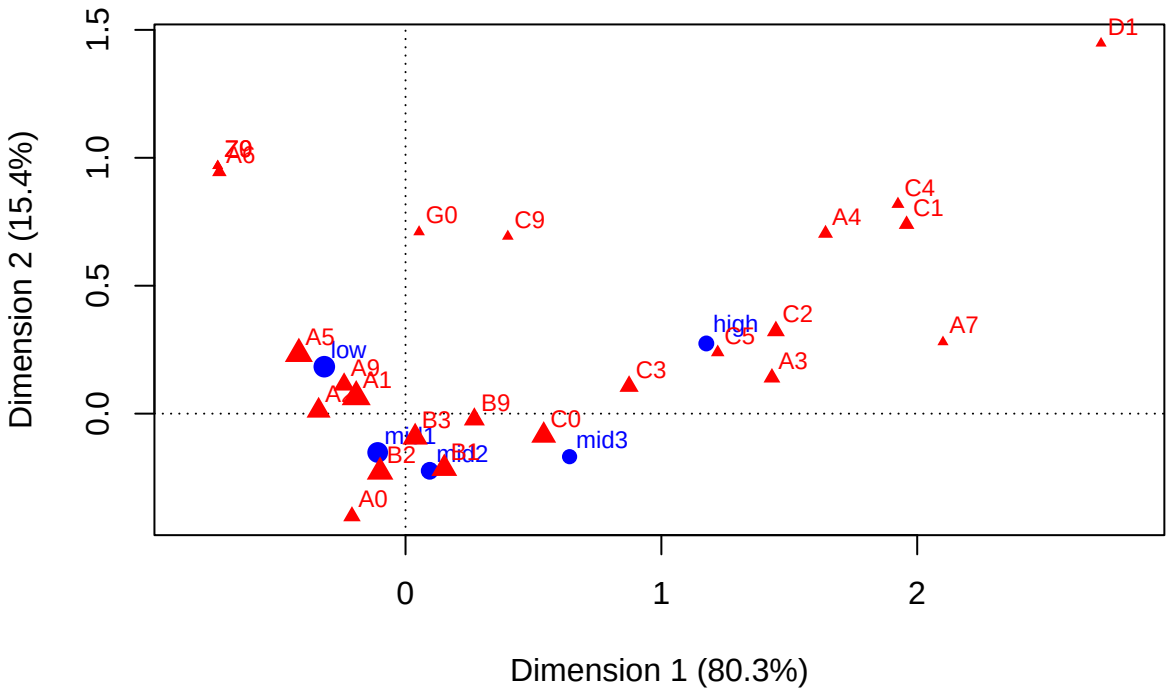
```
## Warning: package 'ca' was built under R version 4.2.3
```

```
# Create a contingency table from the two categorical columns
contingency_table <- table(NYC_CA_DATA$SALE_PRICE_BIN, NYC_CA_DATA$BUILDING_CLASS_AT_PRESENT)

# Perform Correspondence Analysis
ca_result <- ca(contingency_table)

# Summary of the Correspondence Analysis
summary(ca_result)
```

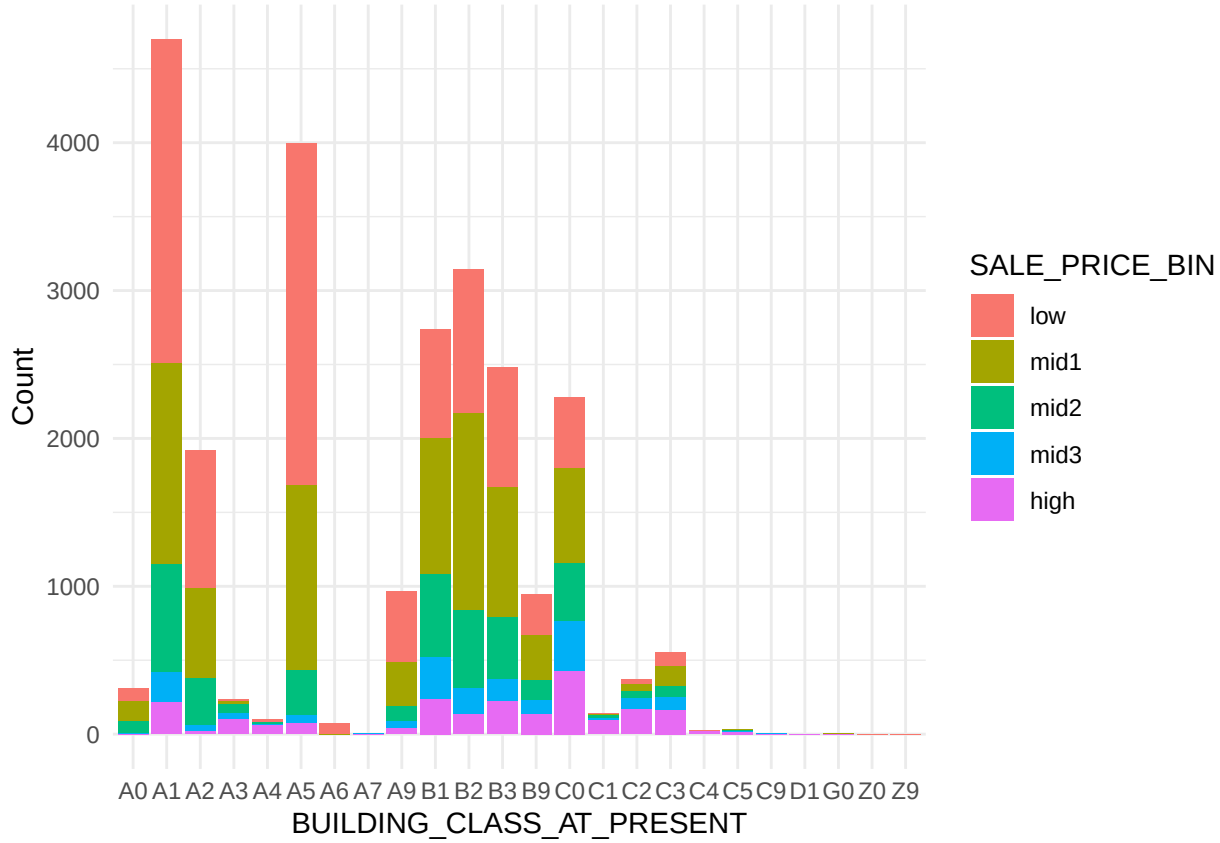
```
##
## Principal inertias (eigenvalues):
##
##  dim    value      %   cum%   scree plot
##  1      0.187036  80.3  80.3  ********************
##  2      0.035913  15.4  95.7  ****
##  3      0.005787   2.5  98.2  *
##  4      0.004236   1.8 100.0
##         -------- -----
##  Total: 0.232972 100.0
##
```

```
##
## Rows:
##     name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
## 1 |  low |  381  994  221 | -317 745 205 |  184 249 357 |
## 2 | mid1 |  317  821   58 | -109 279  20 | -152 542 203 |
## 3 | mid2 |  152  766   50 |   95 116   7 | -223 649 211 |
## 4 | mid3 |   65  880  140 |  641 824 144 | -168  57  51 |
## 5 | high |   84  995  531 | 1176 943 624 |  274  51 177 |
##
## Columns:
##      name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
## 1  |   A0 |   12  795   14 | -209 170   3 | -401 625  55 |
## 2  |   A1 |  188  873   38 | -193 782  37 |   66  91  23 |
## 3  |   A2 |   77  949   40 | -340 948  47 |   12   1   0 |
## 4  |   A3 |    9  967   86 | 1432 958 103 |  141   9   5 |
## 5  |   A4 |    4  959   56 | 1642 810  56 |  704 149  54 |
## 6  |   A5 |  160  982  159 | -417 748 148 |  234 235 243 |
## 7  |   A6 |    3  914   20 | -728 340   8 |  945 574  74 |
## 8  |   A7 |    0  776    2 | 2101 763   2 |  280  14   0 |
## 9  |   A9 |   39  932   13 | -240 761  12 |  114 172  14 |
## 10 |   B1 |  109  926   35 |  152 309  13 | -214 617 140 |
## 11 |   B2 |  126  846   39 | -100 136   7 | -228 711 182 |
## 12 |   B3 |   99  672    6 |   38  96   1 |  -92 576  24 |
## 13 |   B9 |   38  955   12 |  269 949  15 |  -22   7   1 |
## 14 |   C0 |   91  962  121 |  540 938 142 |  -85  23  18 |
## 15 |   C1 |    6  967  108 | 1959 846 114 |  740 121  85 |
## 16 |   C2 |   15  999  141 | 1448 951 167 |  323  47  43 |
## 17 |   C3 |   22  990   74 |  874 976  90 |  107  15   7 |
## 18 |   C4 |    1  960   19 | 1925 813  19 |  819 147  18 |
## 19 |   C5 |    1  957   10 | 1220 921  12 |  240  36   2 |
## 20 |   C9 |    0  621    1 |  400 155   0 |  693 466   3 |
## 21 |   D1 |    0  874    4 | 2719 681   3 | 1447 193   5 |
## 22 |   G0 |    0  931    0 |   53   5   0 |  710 926   3 |
## 23 |   Z0 |    0  909    0 | -734 332   0 |  968 577   1 |
## 24 |   Z9 |    0  909    0 | -734 332   0 |  968 577   1 |
```

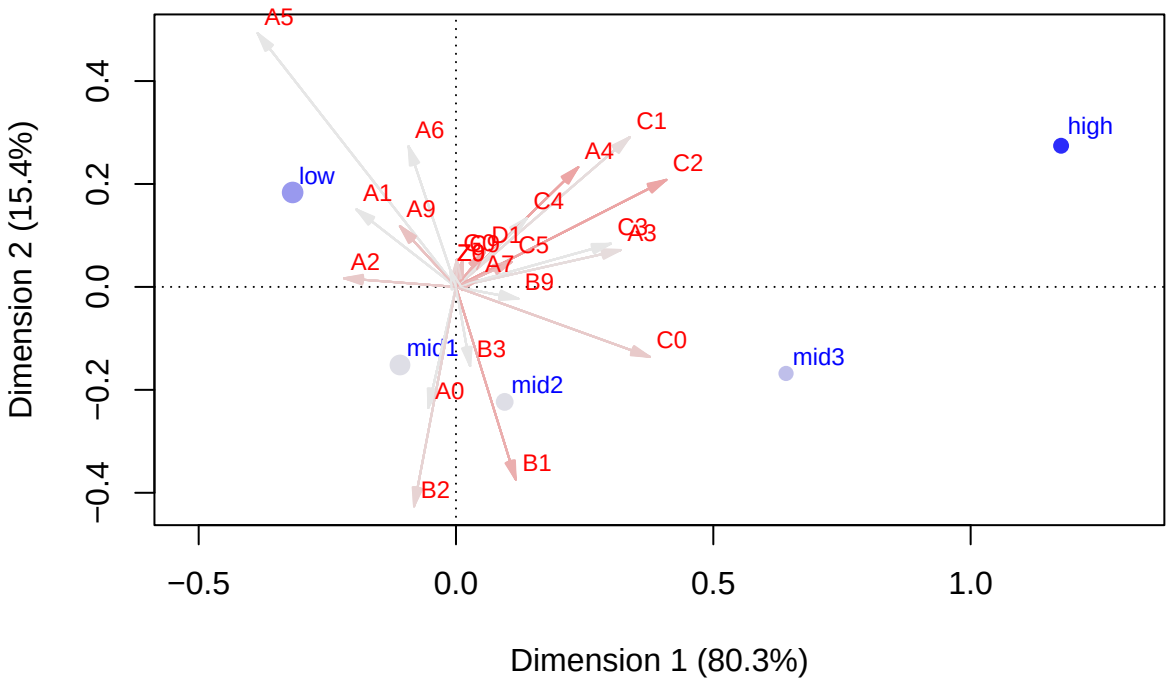
```r
# Plot the Correspondence Analysis
plot(ca_result,mass = TRUE)
```




```
ggplot(NYC_CA_DATA, aes(x = BUILDING_CLASS_AT_PRESENT, fill = SALE_PRICE_BIN)) +
  geom_bar() +
  labs(x = "BUILDING_CLASS_AT_PRESENT", y = "Count", fill = "SALE_PRICE_BIN") +
  theme_minimal()
```




```
# Access the coordinates of the categories
category_coordinates <- ca_result$colcoord
# Access the contributions of the categories
category_contributions <- ca_result$colcontrib

plot(ca_result, mass = TRUE,contrib = "absolute", map = "rowgreen" ,arrows =
c(F,T))
```



Making 5 bins

Category 1 (low): SALE_PRICE <= 215000 + (1 * 288000)

Category 2 (mid1) : 215000 + (1 * 288000) < SALE_PRICE <= 215000 + (2 * 288000) Category 3 (mid2): 215000 + (2 * 288000) < SALE_PRICE <= 215000 + (3 * 288000) Category 4 (mid3): 215000 + (3 * 288000) < SALE_PRICE <= 215000 + (4 * 288000) Category 5 (high): SALE_PRICE > 215000 + (4 * 288000)

The building classes represented by the codes you provided are typically used in the New York City Property Sales dataset. Here's a breakdown of what each code represents:

- A0: One-family dwellings (attached or semi-detached) - generally, homes with one unit and a shared wall with another unit.
- A1: One-family dwellings (semi-detached) - homes with one unit and a separate wall from another unit.
- A2: Two-family dwellings (attached) - homes with two units and a shared wall with another unit.
- A3: Multi-family walk-up apartments (3 or more units, not including co-ops) - typically low-rise buildings with multiple units.
- A4: Multi-family elevator apartments (3 or more units, not including co-ops) - typically mid-rise or high-rise buildings with multiple units and an elevator.
- A5: Cooperative units (apartment buildings) - units in buildings that are collectively owned by shareholders.

- A6: Condominium units (apartment buildings) - individually owned units in buildings with shared common areas.
- A7: Office buildings - buildings primarily used for commercial office space.
- A9: Open space (non-buildable land) - areas without any structures or buildings.
- B1: Store buildings (primarily retail) - buildings used for commercial retail purposes.
- B2: Store buildings (mixed use) - buildings used for a combination of residential and commercial purposes.
- B3: Store buildings (primarily residential) - buildings with residential units above commercial spaces.
- B9: Other stores (hotel, garage, etc.) - buildings used for various purposes, such as hotels or garages.
- C0-C9: Walk-up apartments (mostly 3 to 6 stories) - buildings with walk-up apartments, typically low-rise.
- D1: Elevator apartments (mostly 7 or more stories) - buildings with apartments and an elevator, typically mid-rise or high-rise.
- G0: Miscellaneous mixed-use buildings - buildings used for a combination of different purposes, such as residential and commercial.
- Z0: Open space with a residential building - areas with open space and a residential structure.
- Z9: Miscellaneous structures of any zoning class - miscellaneous structures that do not fit into any specific category.

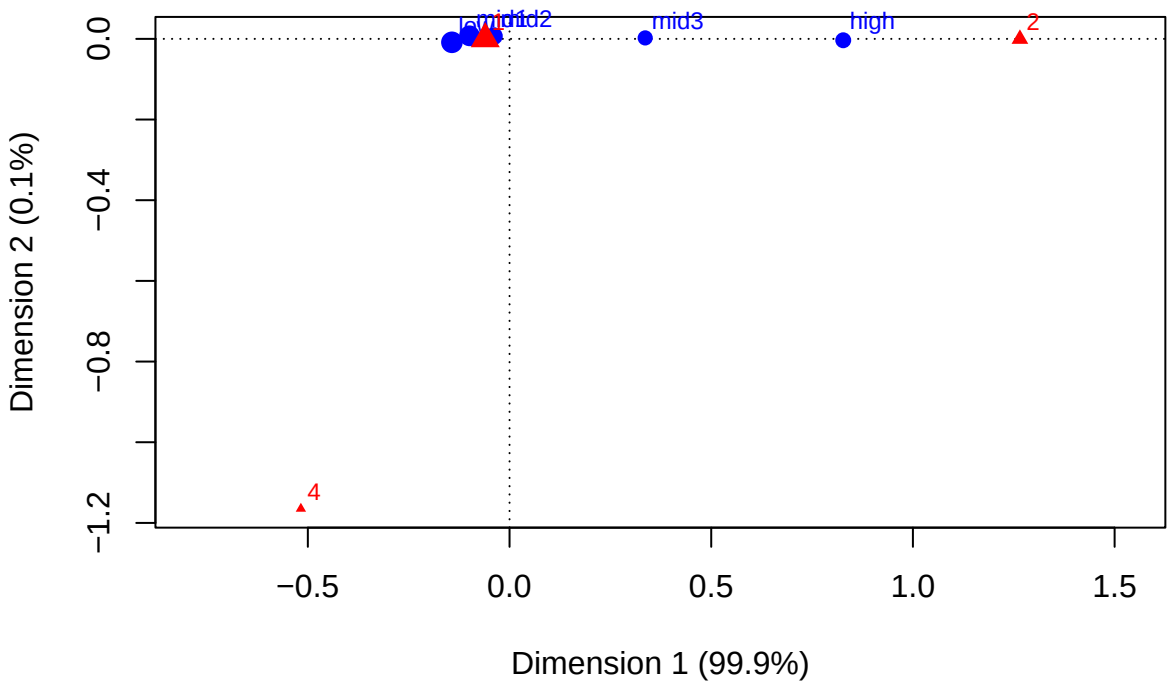
CA WITH TAX_CLASS_AT_TIME_OF_SALE

```
# Load the required library
library(ca)

# Create a contingency table from the two categorical columns
contingency_table <- table(NYC_CA_DATA$SALE_PRICE_BIN, NYC_CA_DATA$TAX_CLASS_AT_TIME_OF_SALE)

# Perform Correspondence Analysis
ca_result <- ca(contingency_table)

# Summary of the Correspondence Analysis
summary(ca_result)
```

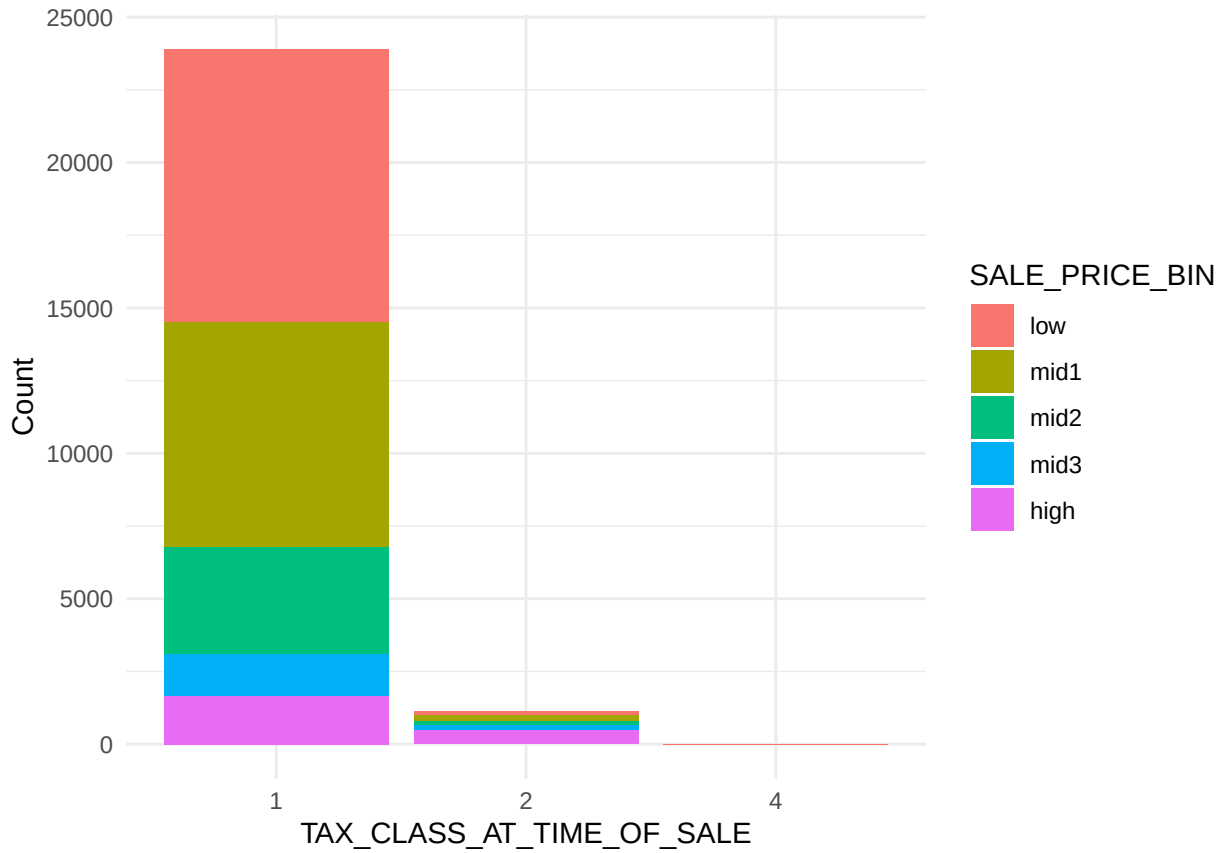
```
##
## Principal inertias (eigenvalues):
##
##  dim    value      %   cum%   scree plot
##  1      0.076264  99.9  99.9  *************************
##  2      5.4e-050   0.1 100.0
##         -------- -----
##  Total: 0.076318 100.0
##
##
## Rows:
##     name   mass  qlt  inr    k=1  cor ctr    k=2 cor ctr
## 1 |  low |  381 1000  102 | -143  996 102 |   -9   4 517 |
```

```
## 2 | mid1 |  317 1000   41 |  -99  994  41 |    7   6 329 |
## 3 | mid2 |  152 1000    3 |  -39  970   3 |    7  30 129 |
## 4 | mid3 |   65 1000   97 |  336 1000  97 |    2   0   7 |
## 5 | high |   84 1000  757 |  827 1000 757 |   -3   0  19 |
##
## Columns:
##      name   mass  qlt  inr    k=1  cor ctr     k=2 cor  ctr
## 1 |    1 |  954 1000   45 |  -60 1000  45 |     0   0    0 |
## 2 |    2 |   45 1000  954 | 1265 1000 954 |     0   0    0 |
## 3 |    4 |    0 1000    1 | -517  165   0 | -1165 835 1000 |
```

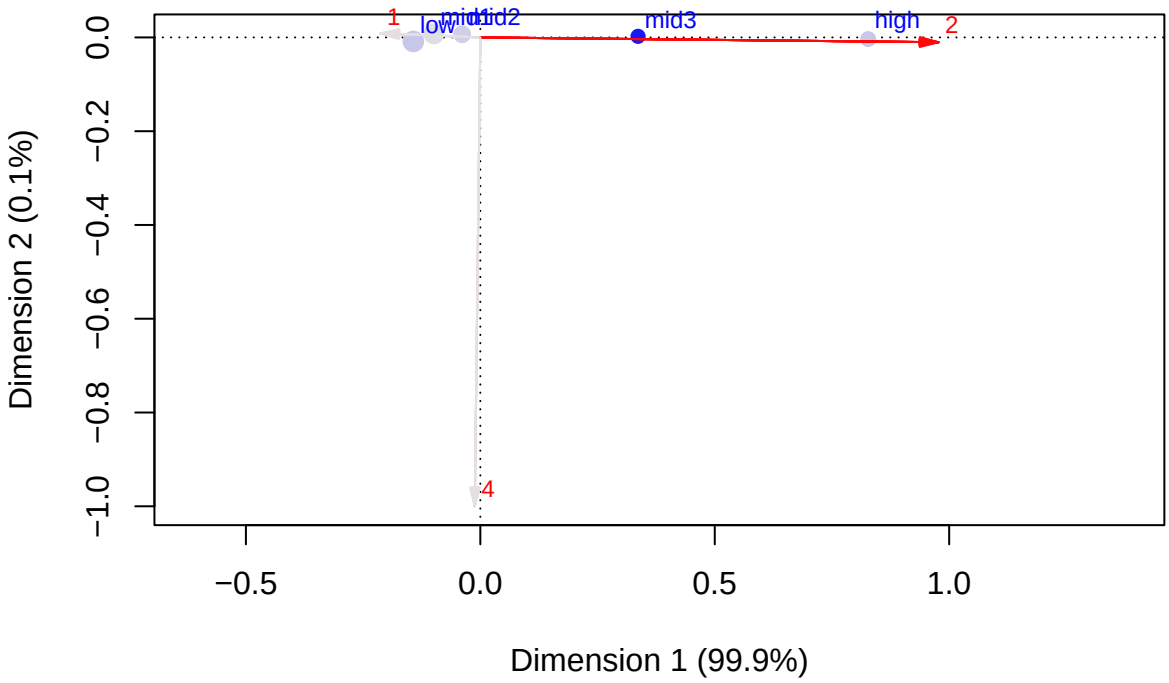
```
# Plot the Correspondence Analysis
plot(ca_result,mass = TRUE)
```




```
ggplot(NYC_CA_DATA, aes(x = TAX_CLASS_AT_TIME_OF_SALE, fill = SALE_PRICE_BIN)) +
  geom_bar() +
  labs(x = "TAX_CLASS_AT_TIME_OF_SALE", y = "Count", fill = "SALE_PRICE_BIN") +
  theme_minimal()
```




```
# Access the coordinates of the categories
category_coordinates <- ca_result$colcoord
# Access the contributions of the categories
category_contributions <- ca_result$colcontrib

plot(ca_result, mass = TRUE,contrib = "absolute", map = "rowgreen" ,arrows =
c(F,T))
```



TAX CLASSES In the context of New York City property taxation, the tax classes 1, 2, 3, and 4 represent different types of properties and their respective tax assessment rules. Here's an overview of what each tax class represents:

1. Tax Class 1: This tax class is applied to properties that are primarily residential, including one- to three-unit properties (such as single-family homes, duplexes, and triplexes), as well as certain condominiums and cooperatives. Tax Class 1 properties generally receive the lowest property tax rates.

2. Tax Class 2: This tax class is applied to residential properties that are considered multiple-dwelling properties. It includes apartment buildings with three or more units, as well as rental cooperatives and some condominiums. Tax Class 2 properties are subject to higher tax rates compared to Tax Class 1 properties.

3. Tax Class 3: This tax class is applied to utility company-owned properties, such as power plants, transmission lines, and telecommunication facilities. These properties are assessed based on specific tax rules related to their utility use.

4. Tax Class 4: This tax class is applied to commercial properties, including office buildings, retail stores, hotels, and other commercial establishments. It also includes certain mixed-use properties that have a significant portion of commercial space. Tax Class 4 properties generally have higher tax rates compared to residential properties.

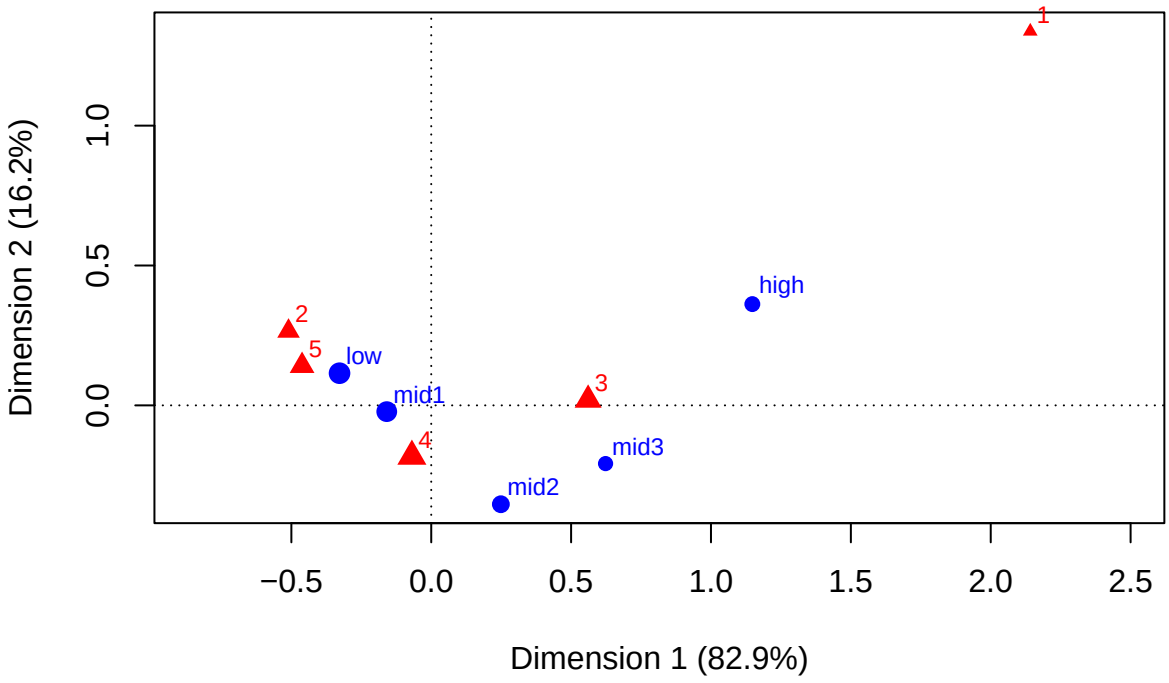
It's important to note that the specific definitions and rules for tax classes may vary depending on the municipality and jurisdiction. The above descriptions provide a general understanding of the tax classes as they relate to New York City. For more accurate and detailed information, it's recommended to refer to the specific local tax laws and regulations.

CA with BOROUGH

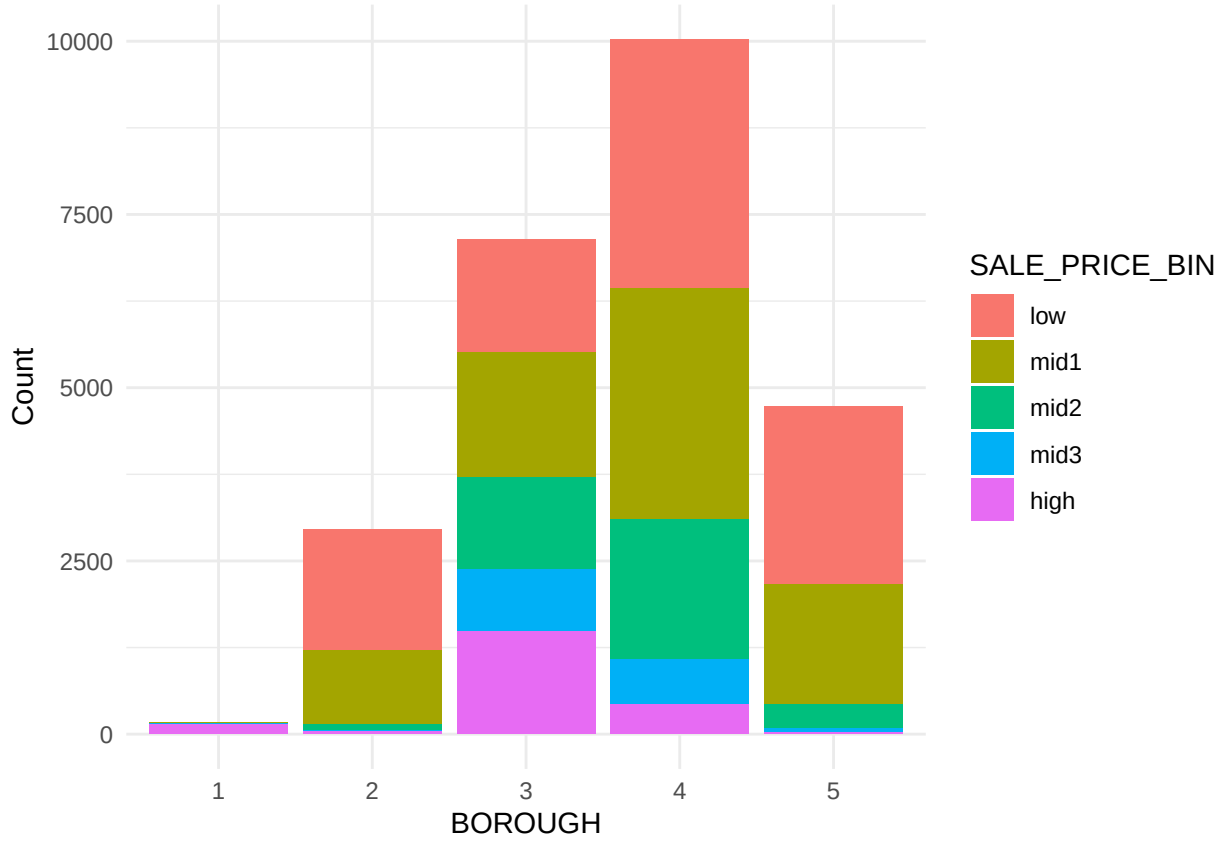
```r
# Load the required library
library(ca)

# Create a contingency table from the two categorical columns
contingency_table <- table(NYC_CA_DATA$SALE_PRICE_BIN, NYC_CA_DATA$BOROUGH)

# Perform Correspondence Analysis
ca_result <- ca(contingency_table)

# Summary of the Correspondence Analysis
summary(ca_result)
```

```
##
## Principal inertias (eigenvalues):
##
##  dim    value      %   cum%   scree plot
##  1      0.194921  82.9  82.9  ********************
##  2      0.038046  16.2  99.1  ****
##  3      0.002176   0.9 100.0
##  4      2.9e-050   0.0 100.0
##         -------- -----
##  Total: 0.235173 100.0
##
##
## Rows:
##     name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
## 1 |  low |  381 1000  196 | -328 890 210 |  115 109 132 |
## 2 | mid1 |  317  998   35 | -159 978  41 |  -22  19   4 |
## 3 | mid2 |  152  983  123 |  249 326  48 | -353 657 498 |
## 4 | mid3 |   65  947  127 |  623 852 130 | -208  95  74 |
## 5 | high |   84  999  520 | 1148 909 570 |  362  90 291 |
##
## Columns:
##     name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
## 1 |    1 |    7  977  195 | 2141 703 165 | 1337 274 330 |
## 2 |    2 |  118  996  167 | -510 784 158 |  265 212 218 |
## 3 |    3 |  285  994  384 |  561 993 461 |   20   1   3 |
## 4 |    4 |  401  972   67 |  -69 122  10 | -182 850 349 |
## 5 |    5 |  189 1000  187 | -462 914 207 |  141  86  99 |
```

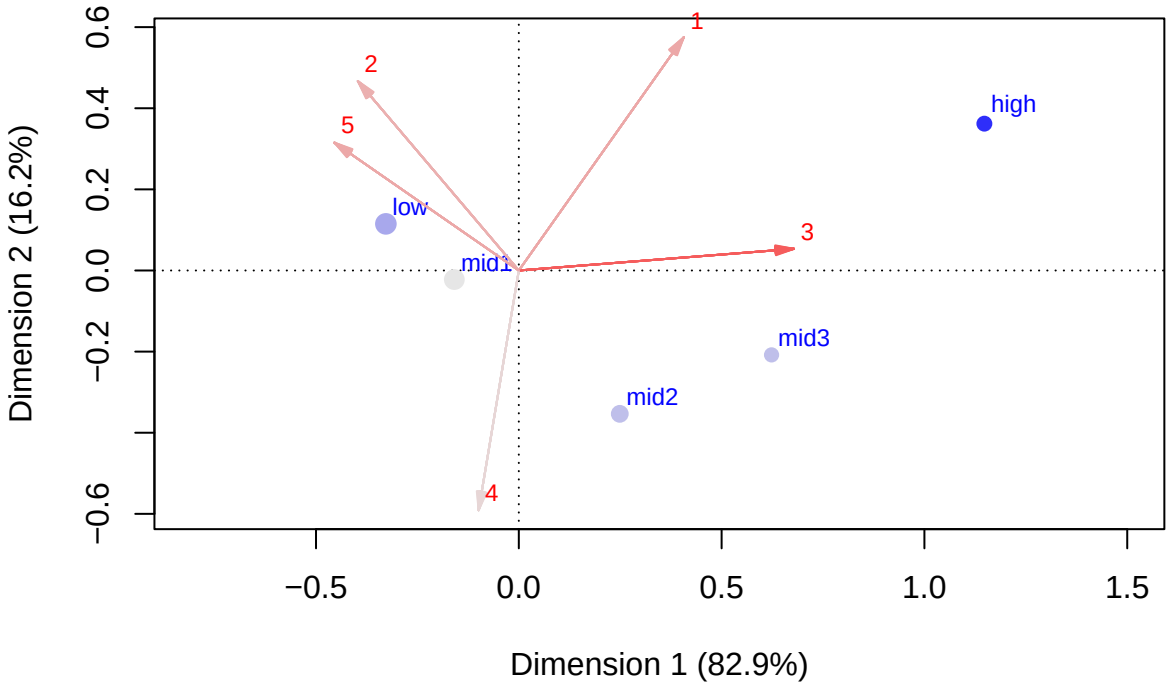
```r
# Plot the Correspondence Analysis
plot(ca_result,mass = TRUE)
```




```
ggplot(NYC_CA_DATA, aes(x = BOROUGH, fill = SALE_PRICE_BIN)) +
  geom_bar() +
  labs(x = "BOROUGH", y = "Count", fill = "SALE_PRICE_BIN") +
  theme_minimal()
```




```
# Access the coordinates of the categories
category_coordinates <- ca_result$colcoord
# Access the contributions of the categories
category_contributions <- ca_result$colcontrib

plot(ca_result, mass = TRUE,contrib = "absolute", map = "rowgreen" ,arrows =
c(F,T))
```



In the context of New York City, the borough numbers 1, 2, 3, 4, and 5 represent the five boroughs that make up the city. Each borough is a distinct administrative division with its own characteristics and local government. Here's a breakdown of what each borough represents:

1. Manhattan (Borough 1): Manhattan is the most densely populated borough and is located at the center of New York City. It is known for its iconic skyline, financial district, cultural attractions, and diverse neighborhoods.

2. Bronx (Borough 2): The Bronx is located north of Manhattan and is the only borough that is primarily situated on the mainland of the United States. It is known for its rich history, vibrant culture, and attractions such as the Bronx Zoo and Yankee Stadium.

3. Brooklyn (Borough 3): Brooklyn is located on the western end of Long Island and is known for its diverse neighborhoods, cultural institutions, and lively arts and music scene. It is the most populous borough in New York City.

4. Queens (Borough 4): Queens is located on Long Island, east of Manhattan, and is the largest borough in terms of land area. It is known for its diverse population, international cuisine, and attractions such as Flushing Meadows-Corona Park and Citi Field.

5. Staten Island (Borough 5): Staten Island is located in the southwestern part of the city and is separated from the rest of the city by the waters of the New York Harbor. It is known for its suburban feel, natural landscapes, and attractions such as the Staten Island Ferry and the Staten Island Zoo.

The borough numbers are commonly used to identify and differentiate between the different parts of New York City.