House Price Prediction Using R

Siddhartha Ranjan

```r
# Loading necessary libraries
library(readr)
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(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-8

library(corrplot)

## corrplot 0.92 loaded

library(caret)

## Loading required package: lattice

library(MASS)

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

library(olsrr)

## Warning: package 'olsrr' was built under R version 4.3.2

##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:MASS':
##
##     cement

## The following object is masked from 'package:datasets':
##
##     rivers

library(glasso)

# Reading the training and test datasets
train_data <- read_csv("C:/Users/13127/Downloads/train_set.csv")

## Rows: 436 Columns: 13

## ── Column specification
────────────────────────────────────────────────────────

## Delimiter: ","
## chr (7): mainroad, guestroom, basement, hotwaterheating, airconditioning,
pr...
## dbl (6): price, area, bedrooms, bathrooms, stories, parking
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.

test_data <- read_csv("C:/Users/13127/Downloads/test_set.csv")

## Rows: 109 Columns: 13
## ── Column specification
────────────────────────────────────────────────────────

## Delimiter: ","
## chr (7): mainroad, guestroom, basement, hotwaterheating, airconditioning,
pr...
## dbl (6): price, area, bedrooms, bathrooms, stories, parking
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

Cross Checking any missing values

```
# Define a function to count missing values
count_missing_vals <- function(data, by_row = FALSE) {
    if (!by_row) {
        missing_results <- NULL
        for (i in 1:ncol(data)) {
            temp_vals <- sum(is.na(data[, i]))
            temp_df <- as.data.frame(temp_vals)
            temp_df$columns <- colnames(data)[i]
            colnames(temp_df) <- c('NAs', 'columns')
            missing_results <- rbind(missing_results, temp_df)
```

```
        }
        return(missing_results)
    } else {
        missing_results <- NULL
        for (i in 1:nrow(data)) {
            temp_vals <- sum(is.na(data[i, ]))
            temp_df <- as.data.frame(temp_vals)
            temp_df$rows <- rownames(data)[i]
            colnames(temp_df) <- c('NAs', 'rows')
            missing_results <- rbind(missing_results, temp_df)
        }
        return(missing_results)
    }
}

# Calculate missing values for train and test datasets
train_missing_vals <- count_missing_vals(train_data)
test_missing_vals <- count_missing_vals(test_data)

# Print the missing values count
train_missing_vals
```

```
##    NAs          columns
## 1   30            price
## 2   18             area
## 3   20         bedrooms
## 4   18        bathrooms
## 5   33          stories
## 6   15         mainroad
## 7   24        guestroom
## 8   19         basement
## 9   22  hotwaterheating
## 10  14  airconditioning
## 11  26          parking
## 12  29         prefarea
## 13  15 furnishingstatus
```

```
test_missing_vals
```

```
##    NAs          columns
## 1    5            price
## 2    2             area
## 3    6         bedrooms
## 4    6        bathrooms
## 5    8          stories
## 6    7         mainroad
## 7    7        guestroom
## 8    4         basement
## 9    4  hotwaterheating
## 10   8  airconditioning
```

```
## 11   3          parking
## 12   4         prefarea
## 13   6 furnishingstatus
```

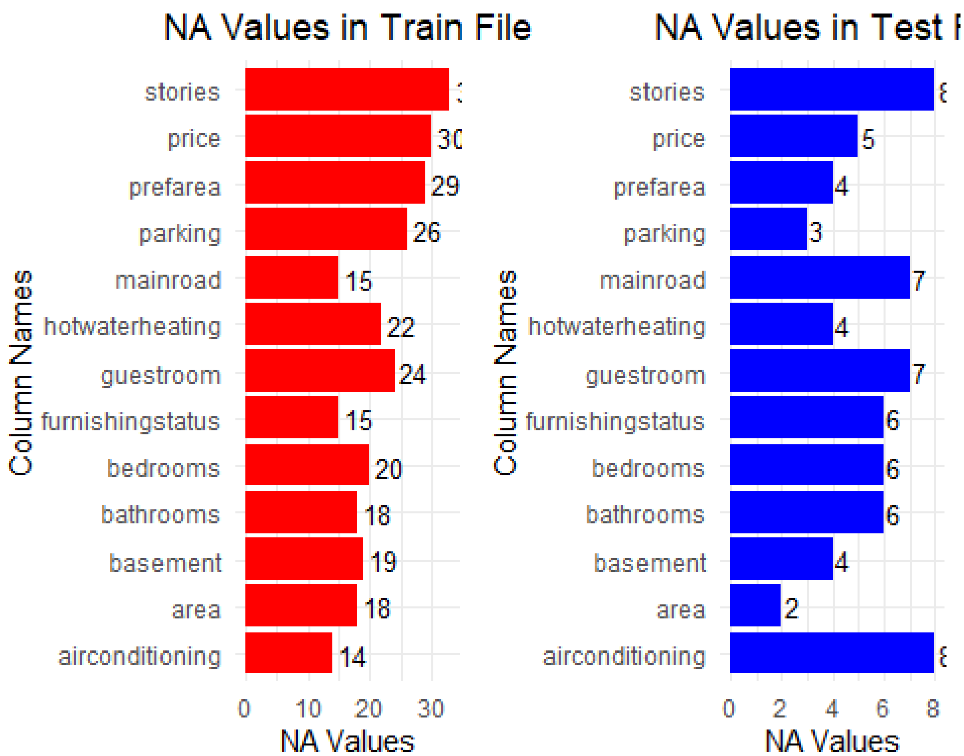
```r
library(ggplot2)
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.3.2
```

```
##
## Attaching package: 'gridExtra'
```

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

```r
# Create a function for plotting NA values
plot_na_values <- function(data, title, fill_color) {
  ggplot(data, aes(x = NAs, y = columns)) +
    geom_bar(stat = "identity", fill = fill_color) +
    labs(title = title,
         x = "NA Values",
         y = "Column Names") +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5)) +
    geom_text(aes(label = NAs), vjust = 0.5, hjust = -0.2, size = 3.5)
}

# Plot NA values in the train file
train_plot <- plot_na_values(train_missing_vals, "NA Values in Train File",
"red")

# Plot NA values in the test file
test_plot <- plot_na_values(test_missing_vals, "NA Values in Test File",
"blue")

# Display plots in a grid arrangement
grid.arrange(train_plot, test_plot, ncol = 2)
```



Handling the missing values in the train dataset

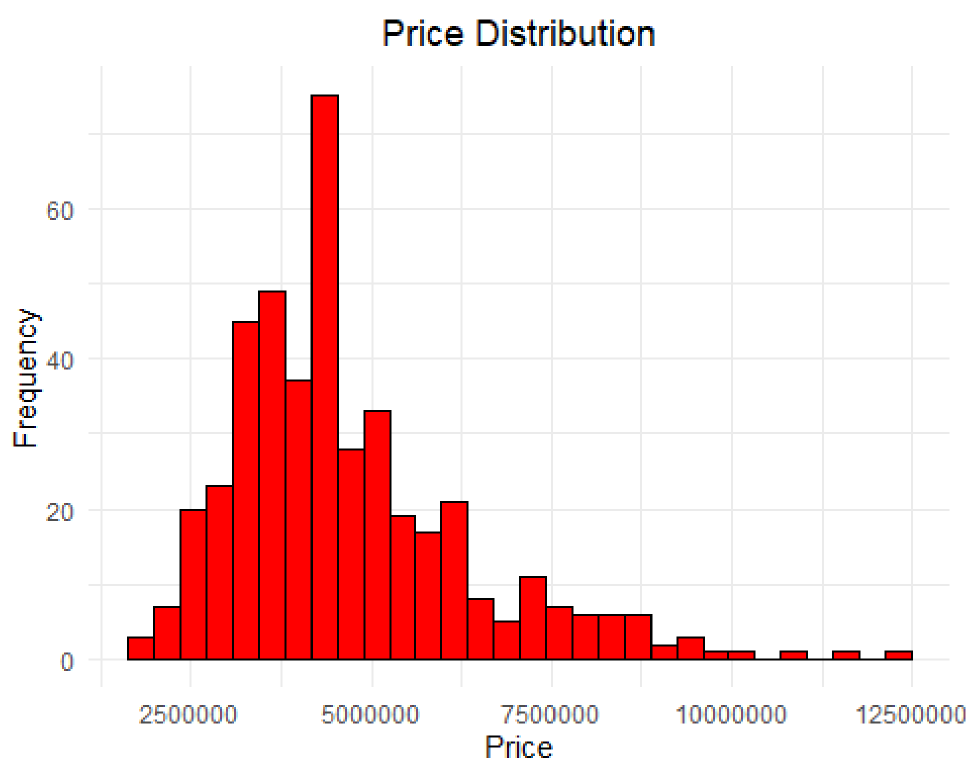
```
# List of columns with missing values and their respective counts in train
data
cols_with_null_train <- c('price', 'area', 'bedrooms', 'bathrooms',
'stories',
                          'mainroad', 'guestroom', 'basement',
'hotwaterheating',
                          'airconditioning', 'parking', 'prefarea',
'furnishingstatus')
null_counts_train <- c(33, 30, 29, 26, 24, 22, 20, 19, 18, 18, 15, 15, 14)

# Replace NA values with appropriate imputation for each column in train data
for (i in 1:length(cols_with_null_train)) {
  col_train <- cols_with_null_train[i]
  count_train <- null_counts_train[i]

  if (is.numeric(train_data[[col_train]])) {
    # For numeric columns, impute with median
    train_data[[col_train]][is.na(train_data[[col_train]])] <-
median(train_data[[col_train]], na.rm = TRUE)
  } else {
    # For categorical columns, impute with mode
    Mode <- function(x){
      names(which.max(table(x, useNA = "no")))
    }
```

```
    train_data[is.na(train_data[[col_train]]), col_train] <-
Mode(train_data[[col_train]])
  }
}

# Check again for missing values after imputation in train data
train_missing_vals_imputed <- count_missing_vals(train_data)
train_missing_vals_imputed <- train_missing_vals_imputed %>%
    filter(NAs > 0)

# Print the remaining missing values in the train dataset after imputation
print(train_missing_vals_imputed)

## [1] NAs     columns
## <0 rows> (or 0-length row.names)

# List of columns with missing values and their respective counts in test
data
cols_with_null_test <- c('stories', 'price', 'prefarea', 'parking',
'mainroad',
                         'hotwaterheating', 'guestroom', 'furnishingstatus',
                         'bedrooms', 'bathrooms', 'basement', 'area',
'airconditioning')
null_counts_test <- c(6, 8, 3, 4, 6, 4, 6, 2, 7, 7, 5, 8, 4)

# Replace NA values with appropriate imputation for each column in test data
for (i in 1:length(cols_with_null_test)) {
  col_test <- cols_with_null_test[i]
  count_test <- null_counts_test[i]

  if (is.numeric(test_data[[col_test]])) {
    # For numeric columns, impute with median
    test_data[[col_test]][is.na(test_data[[col_test]])] <-
median(test_data[[col_test]], na.rm = TRUE)
  } else {
    # For categorical columns, impute with mode
    Mode <- function(x){
      names(which.max(table(x, useNA = "no")))
    }

    test_data[is.na(test_data[[col_test]]), col_test] <-
Mode(test_data[[col_test]])
  }
}

# Check again for missing values after imputation in test data
test_missing_vals_imputed <- count_missing_vals(test_data)
test_missing_vals_imputed <- test_missing_vals_imputed %>%
    filter(NAs > 0)
```

```
# Print the remaining missing values in the test dataset after imputation
print(test_missing_vals_imputed)

## [1] NAs     columns
## <0 rows> (or 0-length row.names)
```

Price EDA

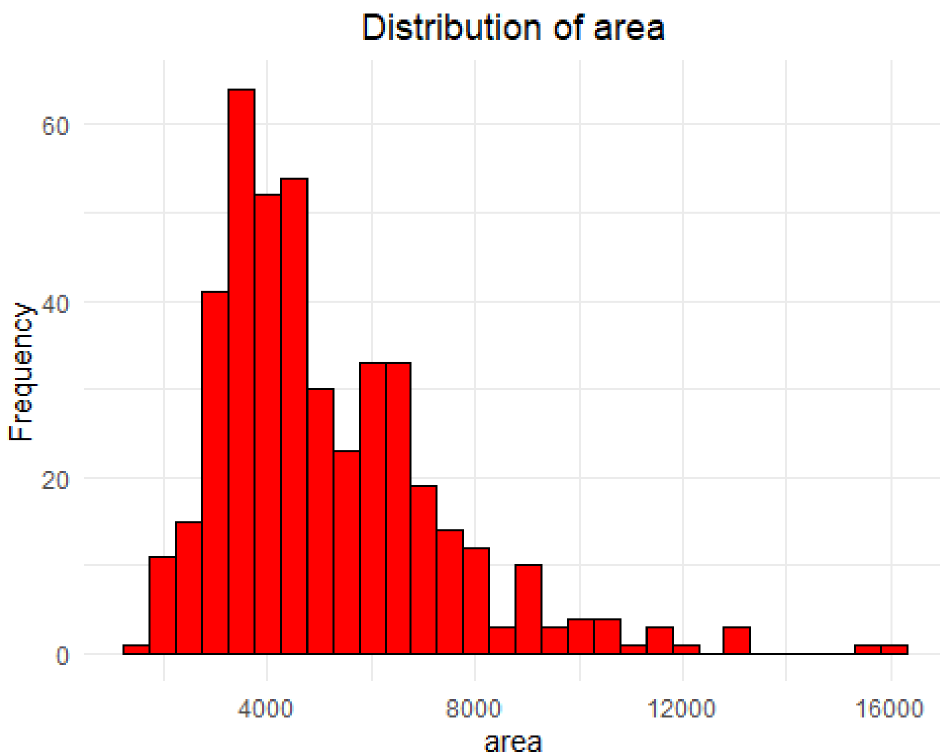
```
# Load required library
library(ggplot2)

# Visualizing the distribution of 'price' column
ggplot(train_data, aes(x = price)) +
  geom_histogram(bins = 30, fill = "red", color = "black") +
  labs(title = "Price Distribution",
       x = "Price",
       y = "Frequency") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



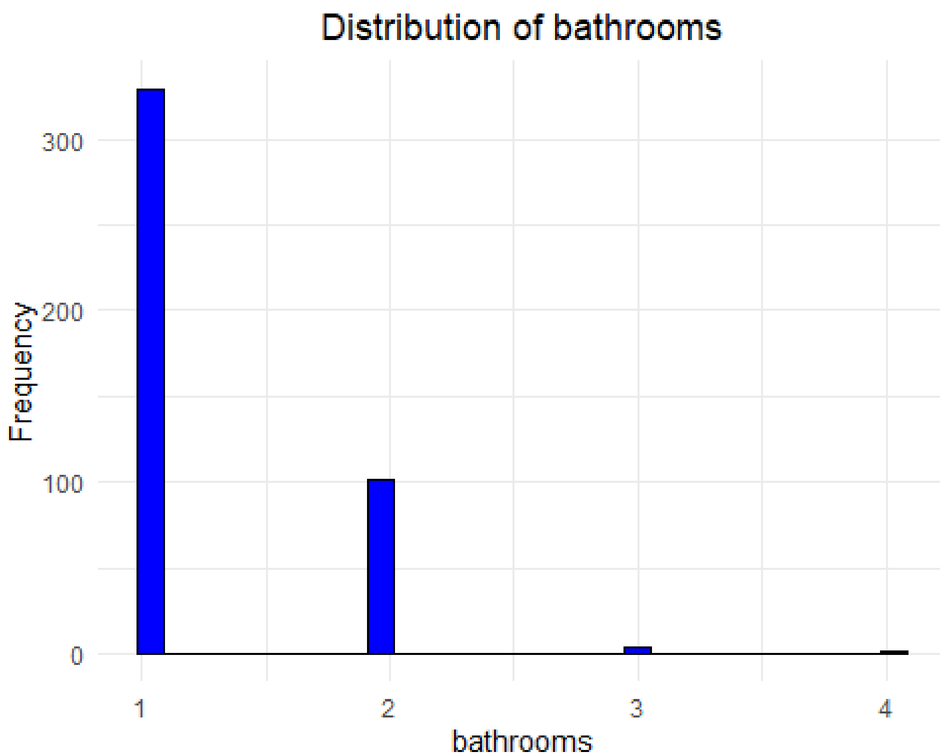

```
# Define colors for numerical and categorical columns
numerical_colors <- c("red", "green", "blue", "orange", "purple")
categorical_colors <- c("pink", "cyan", "magenta", "yellow", "grey", "brown",
"#32CD32") # Lime green as hexadecimal

# Numerical columns
numerical_cols <- c("area", "bedrooms", "bathrooms", "stories", "parking")
for(i in 1:length(numerical_cols)) {
```

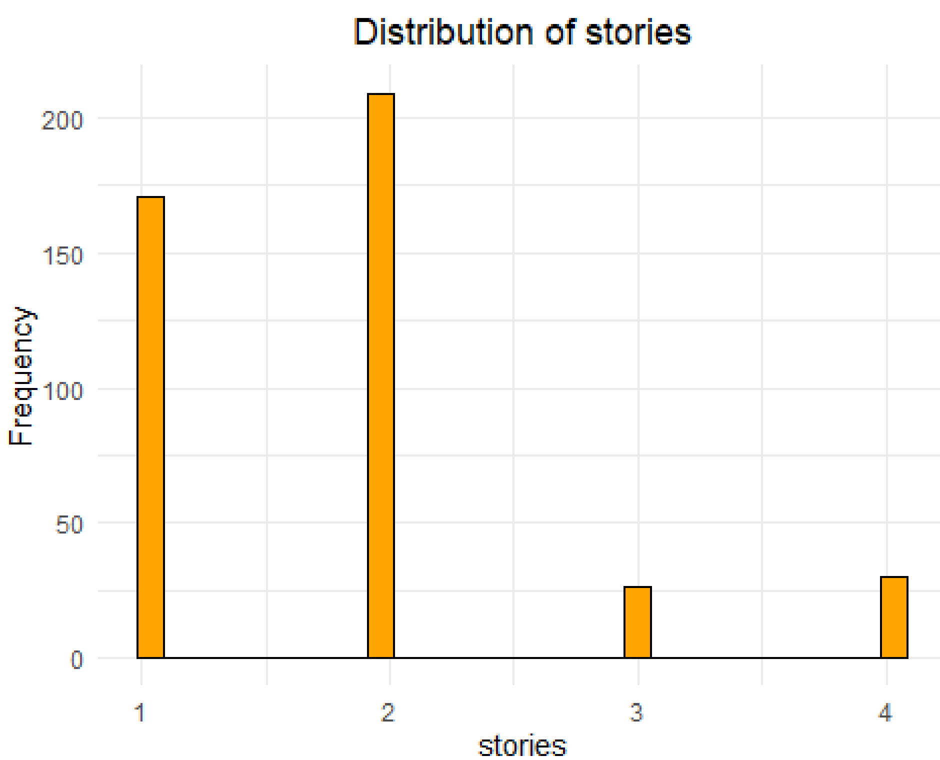
```r
  col <- numerical_cols[i]
  color <- numerical_colors[i]
  p <- ggplot(train_data, aes_string(x = col)) +
    geom_histogram(bins = 30, fill = color, color = "black") +
    labs(title = paste("Distribution of", col),
         x = col,
         y = "Frequency") +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))
  print(p)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

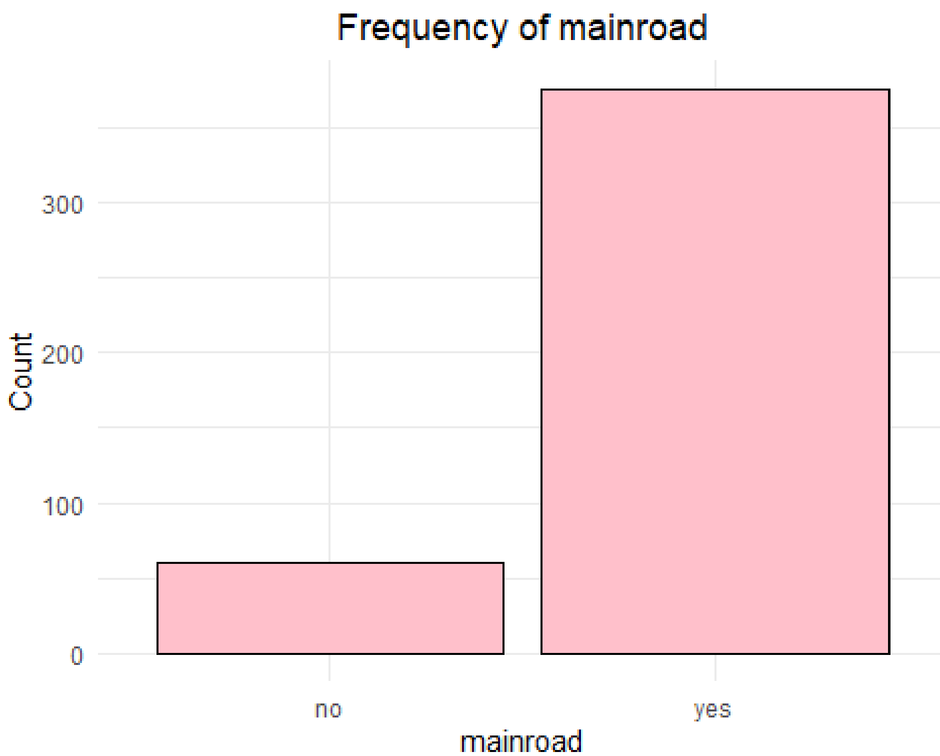

Distribution of area
Frequency
60
40
20
0
4000
8000
12000
16000
area

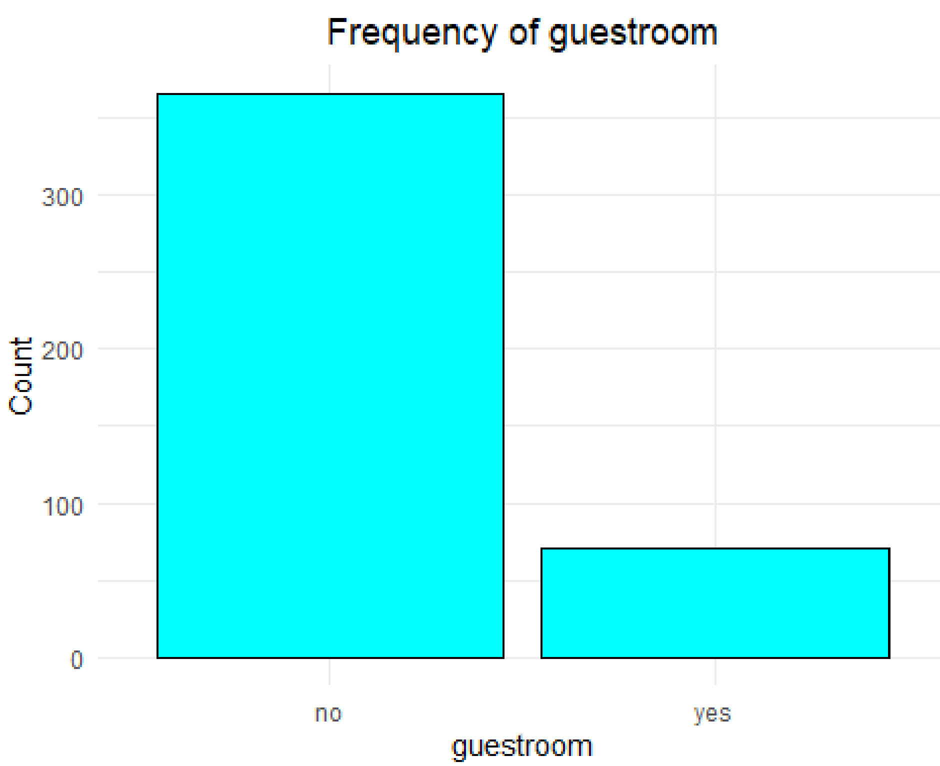

Distribution of bedrooms
Frequency
250
200
150
100
50
0
2
4
6
bedrooms

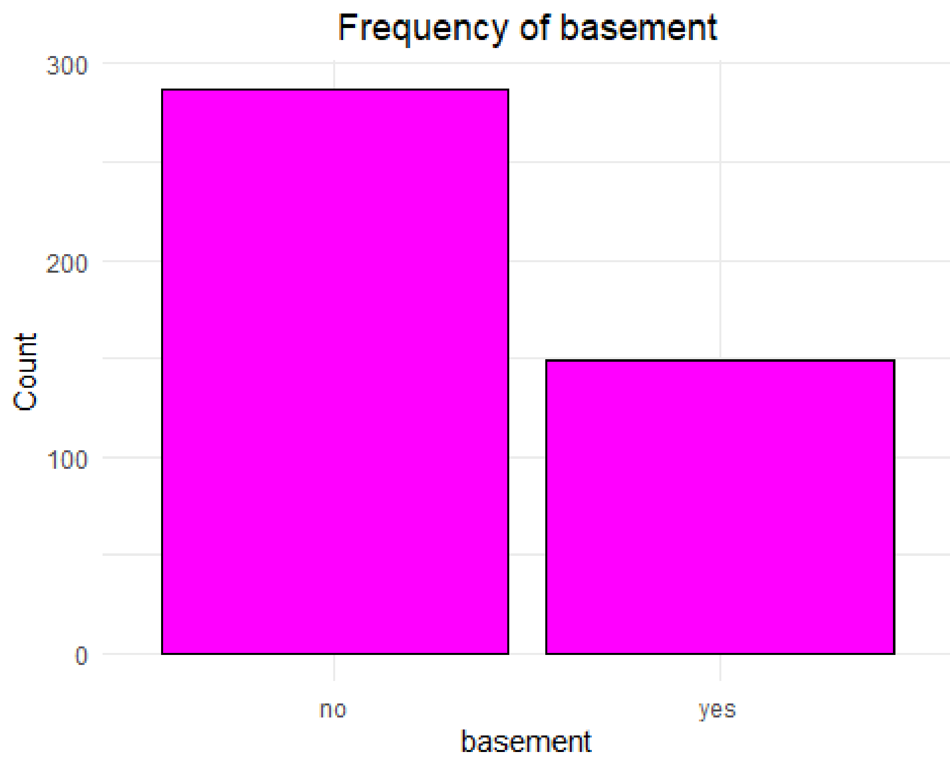

Distribution of bathrooms
300
200
100
0
Frequency
1
2
3
4
bathrooms


Distribution of stories
200
150
100
50
0
Frequency
1
2
3
4
stories

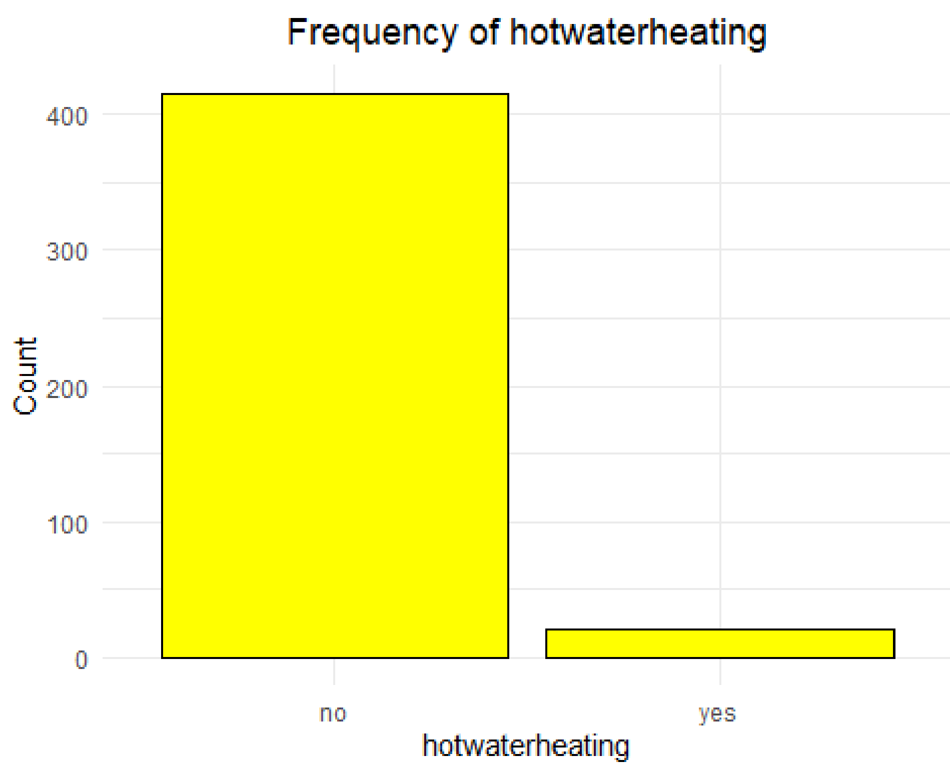

```r
# Categorical columns
categorical_cols <- c("mainroad", "guestroom", "basement", "hotwaterheating",
"airconditioning", "prefarea", "furnishingstatus")
for(i in 1:length(categorical_cols)) {
  col <- categorical_cols[i]
  color <- categorical_colors[i]
  p <- ggplot(train_data, aes_string(x = col)) +
    geom_bar(fill = color, color = "black") +
    labs(title = paste("Frequency of", col),
         x = col,
         y = "Count") +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))
  print(p)
}
```


Frequency of mainroad
Count
300
200
100
0
no
yes
mainroad

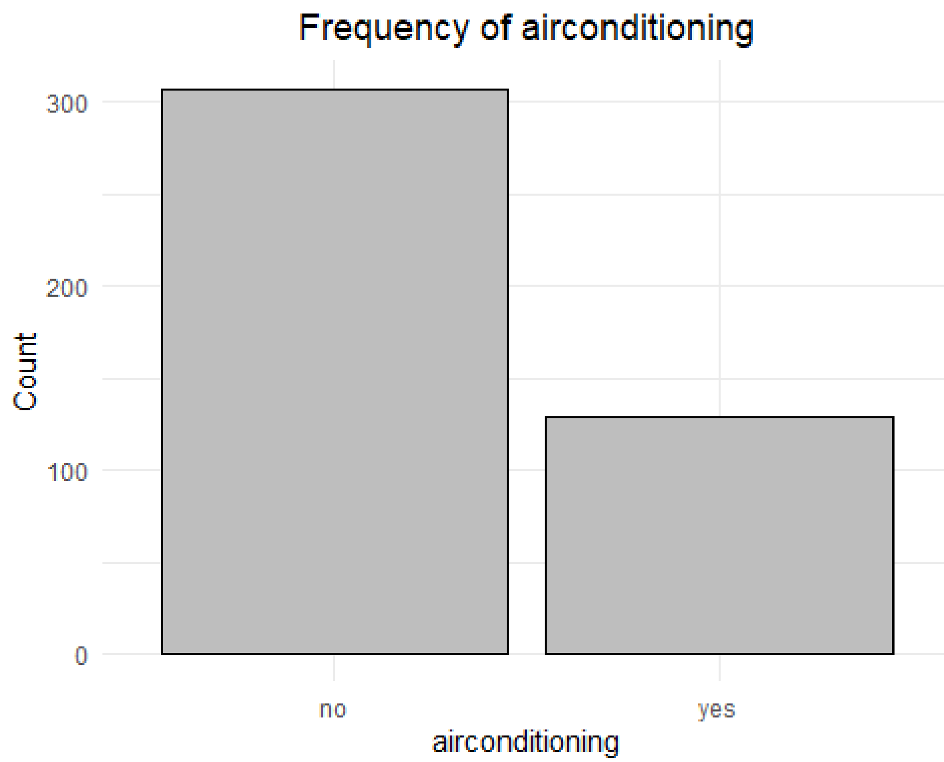

Frequency of guestroom
Count
300
200
100
0
no
yes
guestroom

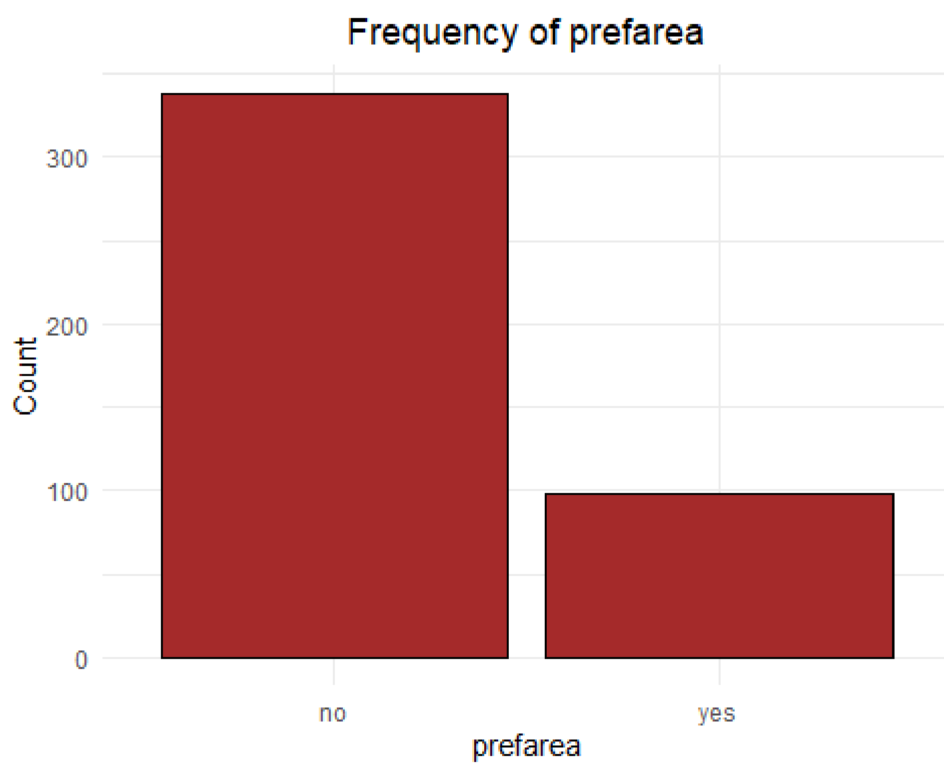


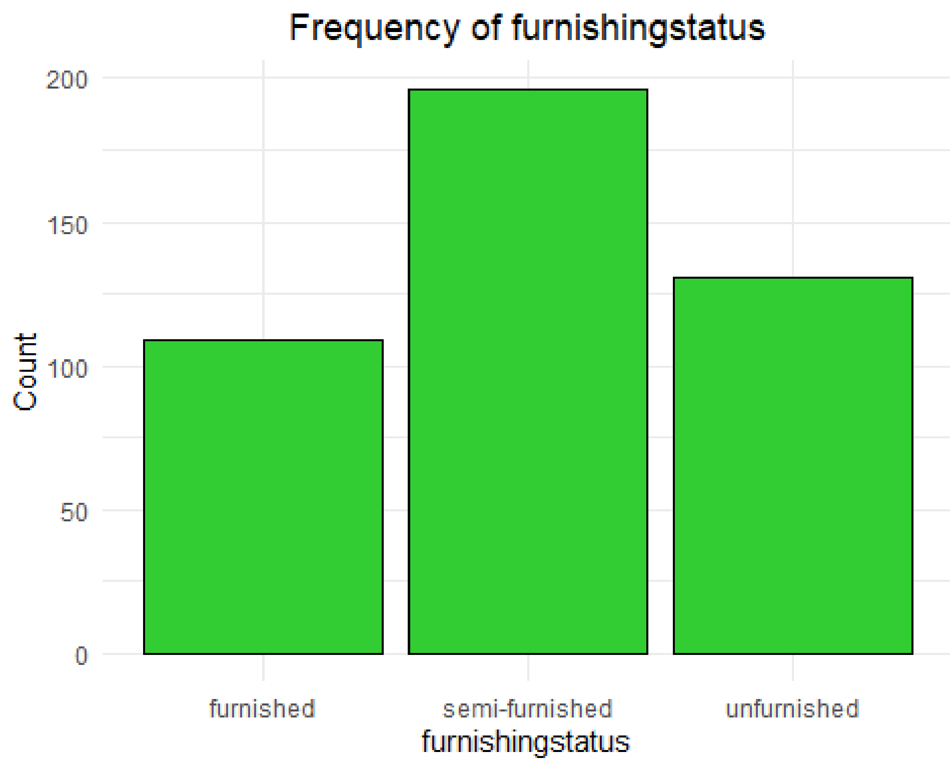
Frequency of basement
300
200
100
0
Count
no
yes
basement



Frequency of hotwaterheating
400
300
200
100
0
Count
no
yes
hotwaterheating


Frequency of airconditioning
300
200
100
0
Count
no
yes
airconditioning


Frequency of prefarea
300
200
100
0
Count
no
yes
prefarea




```
# Process train data
train_data_numeric <- train_data %>% select_if(is.numeric)
train_data_cat <- train_data %>% select_if(~!is.numeric(.))
train_data_cat_numeric <- train_data_cat %>%
mutate_all(funs(as.integer(as.factor(.))))

## Warning: `funs()` was deprecated in dplyr 0.8.0.
## ℹ Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

combined_train_data <- cbind(train_data_cat_numeric, train_data_numeric)

# Process test data
test_data_numeric <- test_data %>% select_if(is.numeric)
test_data_cat <- test_data %>% select_if(~!is.numeric(.))
test_data_cat_numeric <- test_data_cat %>%
mutate_all(funs(as.integer(as.factor(.))))

## Warning: `funs()` was deprecated in dplyr 0.8.0.
## ℹ Please use a list of either functions or lambdas:
```

```
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
combined_test_data <- cbind(test_data_cat_numeric, test_data_numeric)

# You can view the first few rows of the transformed datasets
head(combined_train_data)
```

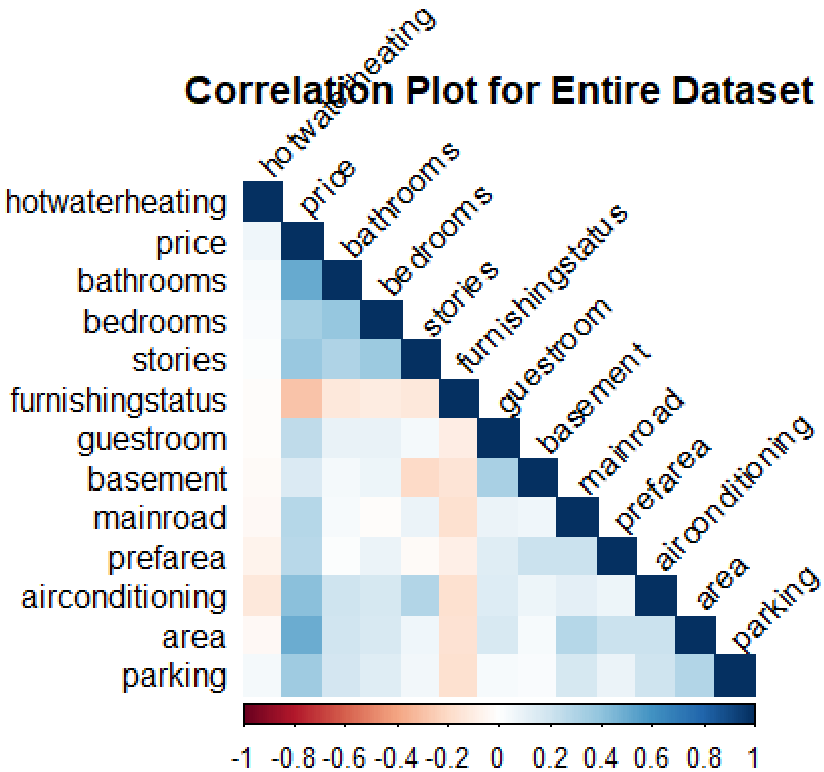
```
##   mainroad guestroom basement hotwaterheating airconditioning prefarea
## 1        2         1        1               1               2        1
## 2        2         1        2               1               2        1
## 3        2         1        2               1               2        1
## 4        2         1        2               1               1        2
## 5        2         1        1               1               1        1
## 6        2         1        2               1               1        1
##   furnishingstatus   price area bedrooms bathrooms stories parking
## 1                1 7525000 6000        3         2       4       1
## 2                2 6300000 7200        3         2       1       3
## 3                1 3920000 3816        2         1       1       2
## 4                3 4252500 2610        3         1       2       0
## 5                3 3010000 3750        3         1       2       0
## 6                2 4620000 5010        3         1       2       0
```

```
head(combined_test_data)
```

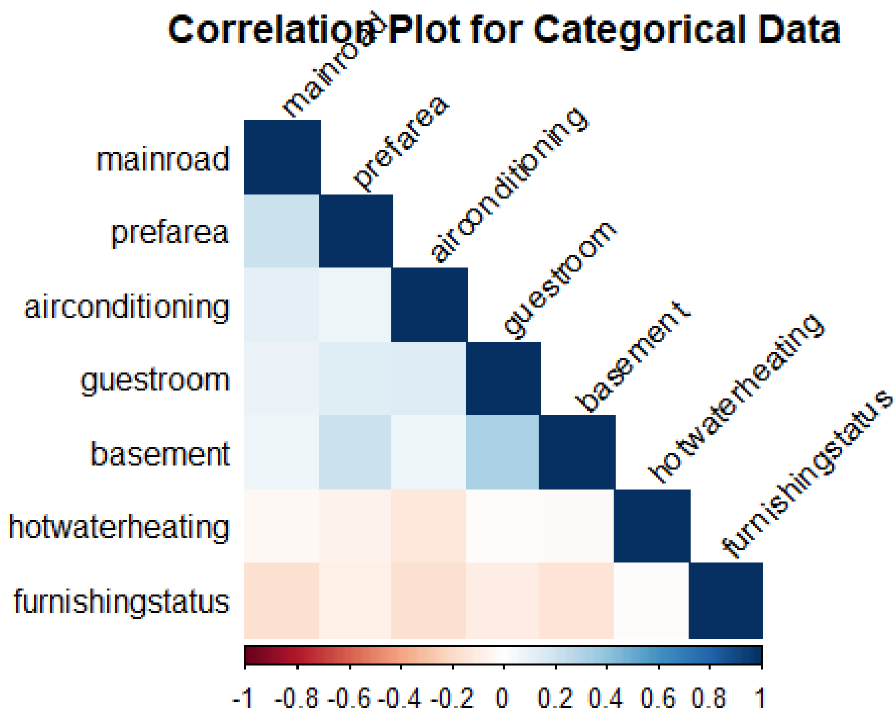
```
##   mainroad guestroom basement hotwaterheating airconditioning prefarea
## 1        1         1        2               1               1        1
## 2        2         1        1               1               2        2
## 3        2         1        1               1               1        1
## 4        2         1        1               1               2        1
## 5        2         1        1               1               1        1
## 6        2         1        1               1               1        1
##   furnishingstatus   price area bedrooms bathrooms stories parking
## 1                3 4060000 5900        4         2       2       1
## 2                1 6650000 6500        3         2       3       0
## 3                2 3710000 4040        2         1       1       0
## 4                2 6440000 5000        3         1       2       0
## 5                1 2800000 3960        3         1       1       0
## 6                3 4900000 6720        3         1       1       0
```

```
# Generating correlation plots for train data
# Correlation plot for entire dataset
corrplot(cor(combined_train_data, use = "complete.obs"), method = "color",
order = "hclust",
```

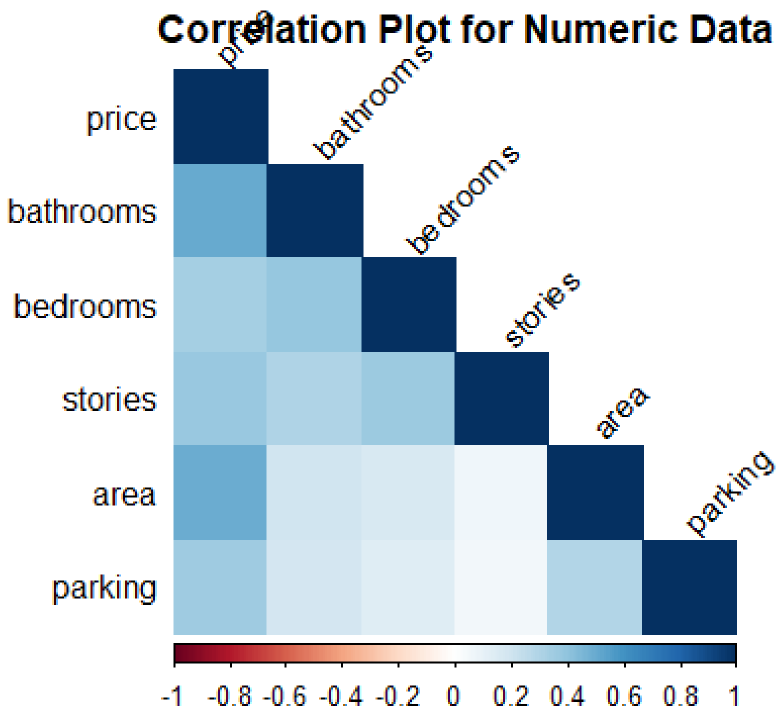
```
         tl.col = "black", tl.srt = 45, type = "lower")
title("Correlation Plot for Entire Dataset")
```




```
# Correlation plot for categorical data
corrplot(cor(train_data_cat_numeric, use = "complete.obs"), method = "color",
order = "hclust",
         tl.col = "black", tl.srt = 45, type = "lower")
title("Correlation Plot for Categorical Data")
```




```
# Correlation plot for numeric data
corrplot(cor(train_data_numeric, use = "complete.obs"), method = "color",
order = "hclust",
         tl.col = "black", tl.srt = 45, type = "lower")
title("Correlation Plot for Numeric Data")
```




```
set.seed(100)
index <- sample(1:nrow(train_data), 0.75*nrow(train_data))
df_train <- train_data[index,]
df_test <- train_data[-index,]

# Create initial linear model
fit <- lm(log10(price) ~ . , data = df_train)
summary(fit)

##
## Call:
## lm(formula = log10(price) ~ ., data = df_train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.287712 -0.057045  0.007576  0.063891  0.301730
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     6.284e+00  3.152e-02 199.374  < 2e-16 ***
## area                            1.836e-05  2.671e-06   6.875 3.37e-11 ***
## bedrooms                        1.220e-02  8.795e-03   1.387 0.166522
## bathrooms                       7.112e-02  1.271e-02   5.597 4.78e-08 ***
## stories                         3.660e-02  7.955e-03   4.601 6.12e-06 ***
## mainroadyes                     3.289e-02  1.722e-02   1.910 0.057012 .
## guestroomyes                    2.450e-02  1.533e-02   1.598 0.111043
## basementyes                     3.851e-02  1.252e-02   3.077 0.002278 **
```

```
## hotwaterheatingyes              6.478e-02  2.579e-02   2.511 0.012526 *
## airconditioningyes              4.755e-02  1.287e-02   3.693 0.000261 ***
## parking                         1.847e-02  6.799e-03   2.716 0.006973 **
## prefareayes                     5.574e-02  1.317e-02   4.232 3.05e-05 ***
## furnishingstatussemi-furnished -5.572e-03  1.352e-02  -0.412 0.680513
## furnishingstatusunfurnished    -4.027e-02  1.525e-02  -2.640 0.008698 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0961 on 313 degrees of freedom
## Multiple R-squared:  0.587,  Adjusted R-squared:  0.5698
## F-statistic: 34.22 on 13 and 313 DF,  p-value: < 2.2e-16
```

```r
# Identifying significant variables
t <- summary(fit)$coefficients[,4] < 0.05
name <- names(which(t == TRUE)) # Filter variables with p-value < 0.05
print("Significant variables:")
```

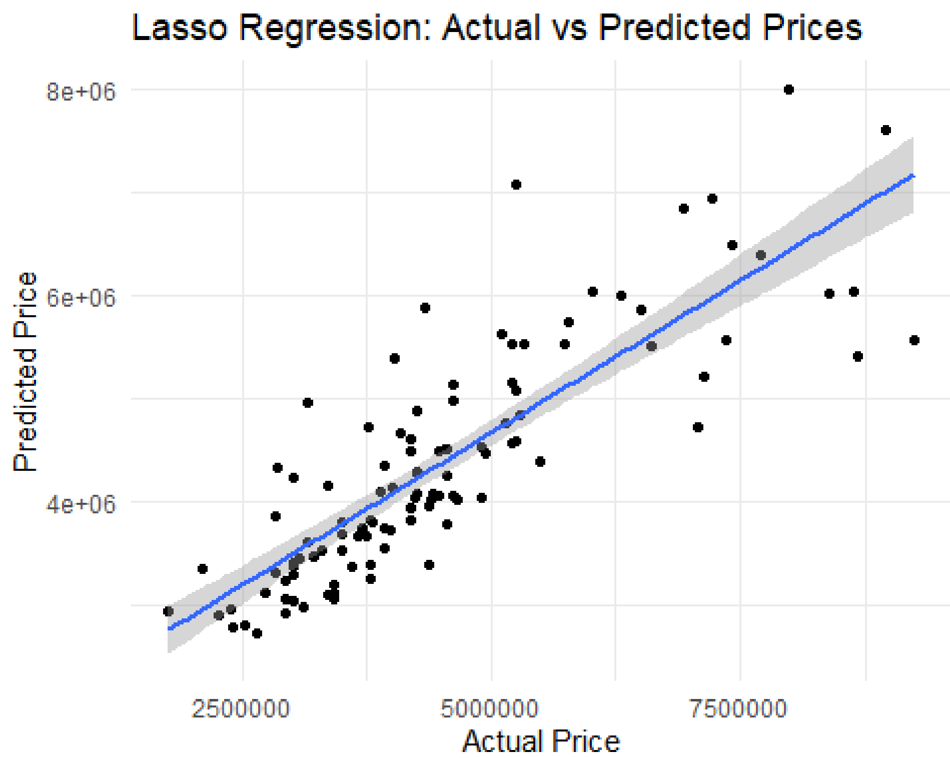
```
## [1] "Significant variables:"
```

```r
print(name)
```

```
##  [1] "(Intercept)"                 "area"
##  [3] "bathrooms"                   "stories"
##  [5] "basementyes"                 "hotwaterheatingyes"
##  [7] "airconditioningyes"          "parking"
##  [9] "prefareayes"                 "furnishingstatusunfurnished"
```

```r
# Refit model using only significant variables
if (all(name %in% colnames(df_train))) {
    fit_name <- lm(log10(price) ~ . , data = df_train[, name])
    summary(fit_name)

    # Prediction using the new model
    pred <- predict(fit_name, df_test)
    result_lm1 <- data.frame(cbind(Actual_Values = df_test$price,
                                   Predicted_Values = 10^(pred)))
    rmse = sqrt(mean(fit_name$residuals^2))
    mae = mean(fit_name$residuals^2)
    error = data.frame('RMSE' = rmse, 'MAE' = mae,
                       'R-Squared' = summary(fit_name)$r.squared)
    print(rmse)
    print(mae)
    print(error)

    # Plotting Actual vs Predicted Values
    ggplot(df_test, aes(x = log10(price), y = log10(pred)))+
      geom_point()+
      geom_smooth()+
      theme_minimal()+
      labs(title = "Actual Values vs Predicted Values",
```

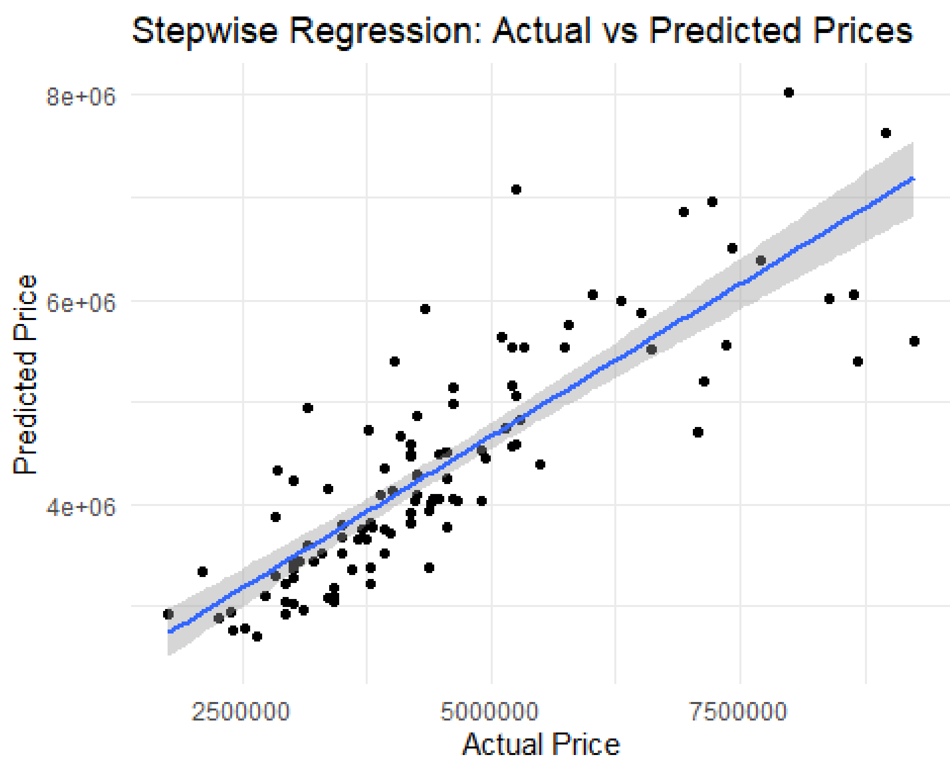
```r
           x = "Sale Price",
           y = "Predicted Sale Price")+
      theme(plot.title = element_text(hjust = 0.5, vjust = 0.5))

    # Plotting Residuals
    plot(residuals(fit_name))
} else {
    print("One or more variables in 'name' are not in 'df_train'")
}

## [1] "One or more variables in 'name' are not in 'df_train'"

# Lasso Regression
set.seed(100)
index <- sample(1:nrow(train_data), 0.75*nrow(train_data))
df_train <- train_data[index,]
df_test <- train_data[-index,]

# Creating model matrix for train and test data
x_train <- model.matrix(price ~ ., df_train)[,-1]
y_train <- log10(df_train$price)

x_test <- model.matrix(price ~ ., df_test)[,-1]

# Cross-validation for lambda selection
set.seed(2021)
cv_model <- cv.glmnet(x_train, y_train, alpha = 1)
lambda_optimal <- cv_model$lambda.min

# Fitting Lasso Model
model_lasso <- glmnet(x_train, y_train, alpha = 1, lambda = lambda_optimal)

# Predicting on test data and converting predictions back from log scale
predicted_lasso <- predict(model_lasso, s = lambda_optimal, newx = x_test)
predicted_lasso <- 10^predicted_lasso

# Actual vs Predicted: Creating dataframe
actual_vs_predicted_lasso <- data.frame(Actual = df_test$price, Predicted =
as.vector(predicted_lasso))

# Plotting Actual vs Predicted Prices
ggplot(actual_vs_predicted_lasso, aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_smooth(method = 'lm') +
  labs(title = "Lasso Regression: Actual vs Predicted Prices", x = "Actual
Price", y = "Predicted Price") +
  theme_minimal()

## `geom_smooth()` using formula = 'y ~ x'
```




```r
# Load necessary library
library(MASS)

# Stepwise Regression using stepAIC
set.seed(100)
initial_model <- lm(log10(price) ~ ., data = df_train)
stepwise_model_aic <- stepAIC(initial_model, direction = "both")

## Start:  AIC=-1518.24
## log10(price) ~ area + bedrooms + bathrooms + stories + mainroad +
##     guestroom + basement + hotwaterheating + airconditioning +
##     parking + prefarea + furnishingstatus
##
##                    Df Sum of Sq    RSS     AIC
## <none>                          2.8904 -1518.2
## - bedrooms          1   0.01776 2.9082 -1518.2
## - guestroom         1   0.02358 2.9140 -1517.6
## - mainroad          1   0.03370 2.9241 -1516.5
## - hotwaterheating   1   0.05825 2.9487 -1513.7
## - furnishingstatus  2   0.08489 2.9753 -1512.8
## - parking           1   0.06812 2.9586 -1512.6
## - basement          1   0.08742 2.9779 -1510.5
## - airconditioning   1   0.12595 3.0164 -1506.3
## - prefarea          1   0.16539 3.0558 -1502.0
## - stories           1   0.19547 3.0859 -1498.8
## - bathrooms         1   0.28925 3.1797 -1489.0
## - area              1   0.43648 3.3269 -1474.2
```

```r
# Summary of the stepwise model
summary(stepwise_model_aic)

##
## Call:
## lm(formula = log10(price) ~ area + bedrooms + bathrooms + stories +
##     mainroad + guestroom + basement + hotwaterheating + airconditioning +
##     parking + prefarea + furnishingstatus, data = df_train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.287712 -0.057045  0.007576  0.063891  0.301730
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     6.284e+00  3.152e-02 199.374  < 2e-16 ***
## area                            1.836e-05  2.671e-06   6.875 3.37e-11 ***
## bedrooms                        1.220e-02  8.795e-03   1.387 0.166522
## bathrooms                       7.112e-02  1.271e-02   5.597 4.78e-08 ***
## stories                         3.660e-02  7.955e-03   4.601 6.12e-06 ***
## mainroadyes                     3.289e-02  1.722e-02   1.910 0.057012 .
## guestroomyes                    2.450e-02  1.533e-02   1.598 0.111043
## basementyes                     3.851e-02  1.252e-02   3.077 0.002278 **
## hotwaterheatingyes              6.478e-02  2.579e-02   2.511 0.012526 *
## airconditioningyes              4.755e-02  1.287e-02   3.693 0.000261 ***
## parking                         1.847e-02  6.799e-03   2.716 0.006973 **
## prefareayes                     5.574e-02  1.317e-02   4.232 3.05e-05 ***
## furnishingstatussemi-furnished -5.572e-03  1.352e-02  -0.412 0.680513
## furnishingstatusunfurnished    -4.027e-02  1.525e-02  -2.640 0.008698 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0961 on 313 degrees of freedom
## Multiple R-squared:  0.587,  Adjusted R-squared:  0.5698
## F-statistic: 34.22 on 13 and 313 DF,  p-value: < 2.2e-16

# Prediction using the stepwise model
predicted_stepwise <- predict(stepwise_model_aic, df_test)
predicted_stepwise <- 10^predicted_stepwise

# Actual vs Predicted
actual_vs_predicted_stepwise <- data.frame(Actual = df_test$price, Predicted
= as.vector(predicted_stepwise))

# Plotting Actual vs Predicted Prices
ggplot(actual_vs_predicted_stepwise, aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_smooth(method = 'lm') +
  labs(title = "Stepwise Regression: Actual vs Predicted Prices", x = "Actual
```

```
Price", y = "Predicted Price") +
  theme_minimal()

## `geom_smooth()` using formula = 'y ~ x'
```




```
# Calculating RMSE and MAE
rmse_stepwise <- sqrt(mean((log10(df_test$price) -
log10(predicted_stepwise))^2))
mae_stepwise <- mean(abs(log10(df_test$price) - log10(predicted_stepwise)))

# Displaying error metrics
error_stepwise <- data.frame(RMSE = rmse_stepwise, MAE = mae_stepwise)
print(error_stepwise)

##         RMSE        MAE
## 1 0.07714765 0.05586812

# Install Metrics package (if not already installed)
if (!require(Metrics)) {
  install.packages("Metrics", dependencies = TRUE)
  library(Metrics)
}

## Loading required package: Metrics

## Warning: package 'Metrics' was built under R version 4.3.2
```

```
##
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':
##
##     precision, recall

# Load necessary library
library(Metrics)

# Fit a linear regression model
model <- lm(log10(price) ~ ., data = df_train)

# Make predictions on the test set
predictions <- predict(model, newdata = df_test)

# Calculate RMSE
rmse_val <- rmse(df_test$price, 10^predictions)  # Converting predictions
back from log scale

# Calculate MAE
mae_val <- mae(df_test$price, 10^predictions)  # Converting predictions back
from log scale

# Extract R-Squared and Adjusted R-Squared
r_squared_val <- summary(model)$r.squared
adjusted_r_squared_val <- summary(model)$adj.r.squared

# Display the Metrics
cat("RMSE:", rmse_val, "\n")

## RMSE: 897386.5

cat("MAE:", mae_val, "\n")

## MAE: 590237.9

cat("R-Squared:", r_squared_val, "\n")

## R-Squared: 0.586972

cat("Adjusted R-Squared:", adjusted_r_squared_val, "\n")

## Adjusted R-Squared: 0.5698174
```