```r
'install.packages("readxl")
install.packages("waterfalls")
install.packages("ggplot2")
install.packages("dplyr")
install.packages("e1071")'
library(e1071)
library(readxl)
library(dplyr)
library(ggplot2)
library(scales)
library(waterfalls)
library(caret)
library(glmnet)

### Assignment 1

## Section 1

# Read the datafile
mydata <- read_excel("ames_housing_data.xlsx")

# Print the structure of the file
str(mydata) # The datafile has 82 columns and 2930 rows

# Print the fist 6 rows of the dataset
head(mydata)

# Print the names of the columns
names(mydata)

# Get the names of the columns with missing values
columns_with_missing <- names(mydata)[colSums(is.na(mydata)) > 0]
print(columns_with_missing)

# Get the number of values missing in each of these columns
missing_counts <- colSums(is.na(mydata))
missing_counts <- missing_counts[missing_counts > 0]
print(missing_counts)

# Identify column types
continuous_cols <- names(mydata)[sapply(mydata, function(col) is.numeric(col) &&
length(unique(col)) > 10)]
discrete_cols <- names(mydata)[sapply(mydata, function(col) is.numeric(col) &&
length(unique(col)) <= 10)]
nominal_cols <- names(mydata)[sapply(mydata, function(col) is.factor(col) &&
!is.ordered(col))]
ordinal_cols <- names(mydata)[sapply(mydata, is.ordered)]
binary_cols <- names(mydata)[sapply(mydata, function(col) is.numeric(col) && all(col %in%
c(0, 1)))]


print(paste("Continuous columns:", paste(continuous_cols, collapse = ", ")))
print(paste("Discrete columns:", paste(discrete_cols, collapse = ", ")))
print(paste("Nominal columns:", paste(nominal_cols, collapse = ", ")))
print(paste("Ordinal columns:", paste(ordinal_cols, collapse = ", ")))
print(paste("Binary columns:", paste(binary_cols, collapse = ", ")))




## Section 2

# Boxplot for SalePrice
```

```r
ggplot(mydata, aes(y = SalePrice)) +
  geom_boxplot(fill = "lightblue") +
  labs(title="Boxplot of SalePrice", y="SalePrice") +
  scale_y_continuous(labels = comma)

graphics.off()
par(mar = c(5.1, 4.1, 4.1, 2.1))

# Compute Q1 and Q3
Q1 <- quantile(mydata$SalePrice, 0.25)
Q3 <- quantile(mydata$SalePrice, 0.75)

# Calculate IQR
IQR_val <- Q3 - Q1

# Define bounds
lower_bound <- Q1 - 1.5 * IQR_val
upper_bound <- Q3 + 1.5 * IQR_val

# Identify outliers below and above bounds
outliers_below <- mydata$SalePrice[which(mydata$SalePrice < lower_bound)]
outliers_above <- mydata$SalePrice[which(mydata$SalePrice > upper_bound)]

# Print results
cat("Lower Bound:", lower_bound, "\n")
cat("Upper Bound:", upper_bound, "\n")
cat("Number of outliers below lower bound:", length(outliers_below), "\n")
cat("Number of outliers above upper bound:", length(outliers_above), "\n")


# Generate the boxplot using ggplot2
ggplot(mydata, aes(y = GrLivArea)) +
  geom_boxplot(fill = "lightblue", color = "darkblue") +
  labs(title = "Boxplot of GrLivArea", y = "GrLivArea (Above Grade Living Area SqFt)") +
  theme_minimal()
par(mar = c(5.1, 4.1, 4.1, 2.1))

# Compute Q1 and Q3 for GrLivArea
Q1_GrLivArea <- quantile(mydata$GrLivArea, 0.25)
Q3_GrLivArea <- quantile(mydata$GrLivArea, 0.75)

# Calculate IQR for GrLivArea
IQR_GrLivArea <- Q3_GrLivArea - Q1_GrLivArea

# Define bounds for GrLivArea
lower_bound_GrLivArea <- Q1_GrLivArea - 1.5 * IQR_GrLivArea
upper_bound_GrLivArea <- Q3_GrLivArea + 1.5 * IQR_GrLivArea

# Identify outliers below and above bounds for GrLivArea
outliers_below_GrLivArea <- mydata$GrLivArea[which(mydata$GrLivArea <
lower_bound_GrLivArea)]
outliers_above_GrLivArea <- mydata$GrLivArea[which(mydata$GrLivArea >
upper_bound_GrLivArea)]

# Print results for GrLivArea
cat("Lower Bound for GrLivArea:", lower_bound_GrLivArea, "\n")
cat("Upper Bound for GrLivArea:", upper_bound_GrLivArea, "\n")
cat("Number of outliers below lower bound for GrLivArea:",
length(outliers_below_GrLivArea), "\n")
cat("Number of outliers above upper bound for GrLivArea:",
length(outliers_above_GrLivArea), "\n")



# Generate the scatter plot using ggplot2
```

```
ggplot(mydata, aes(x = BldgType, y = SalePrice)) +
  geom_point(aes(color = BldgType), size = 3) +
  labs(title = "Scatter plot of SalePrice by BldgType",
       x = "Building Type",
       y = "Sale Price") +
  scale_y_continuous(labels = comma) +
  theme_minimal() +
  theme(legend.position = "none")


# Scatter plot of SalePrice by SaleCondition
ggplot(mydata, aes(x = SaleCondition, y = SalePrice)) +
  geom_point(aes(color = SaleCondition), size = 3) +
  labs(title = "Scatter plot of SalePrice by SaleCondition",
       x = "Sale Condition",
       y = "Sale Price") +
  scale_y_continuous(labels = scales::comma) +
  theme_minimal() +
  theme(legend.position = "none")


# Scatter plot of SalePrice by Functional
ggplot(mydata, aes(x = Functional, y = SalePrice)) +
  geom_point(aes(color = Functional), size = 3) +
  labs(title = "Scatter plot of SalePrice by Functional",
       x = "Home Functionality",
       y = "Sale Price") +
  scale_y_continuous(labels = scales::comma) +
  theme_minimal() +
  theme(legend.position = "none")


# Scatter plot of SalePrice by Zoning
ggplot(mydata, aes(x = Zoning, y = SalePrice)) +
  geom_point(aes(color = Zoning), size = 3) +
  labs(title = "Scatter plot of SalePrice by Zoning",
       x = "Zoning Classification",
       y = "Sale Price") +
  scale_y_continuous(labels = scales::comma) +
  theme_minimal() +
  theme(legend.position = "none")
graphics.off()
par(mar = c(5.1, 4.1, 4.1, 2.1))
# Plot the Waterfall
start_count <- nrow(mydata)
# Open a graphics device
plot.new()


# Apply filters
condition1 <- mydata[mydata$Zoning %in% c("RH", "RL", "RM"), ]
condition2 <- condition1[condition1$SaleCondition == "Normal", ]
condition3 <- condition2[condition2$BldgType == "1Fam", ]
condition4 <- condition3[condition3$Functional == "Typ", ]
condition5 <- condition4[condition4$SalePrice <= 339500, ]
condition6 <- condition5[condition5$GrLivArea <= 2667, ]

# Calculate the number of rows dropped at each step
drops <- c(
  start_count,
  nrow(condition1) - nrow(mydata),
  nrow(condition2) - nrow(condition1),
  nrow(condition3) - nrow(condition2),
  nrow(condition4) - nrow(condition3),
  nrow(condition5) - nrow(condition4),
```

```
  nrow(condition6) - nrow(condition5)
)

# Create labels for each condition
labels <- c(
  "Initial Count", "After Zoning Filter", "After Sale Condition Filter",
  "After Building Type Filter", "After Functionality Filter",
  "After Price Constraint", "After Living Area Constraint"
)

# Plot the waterfall chart
waterfall(values = drops, labels = labels)

# Add axis labels
axis(side = 1, at = 1:length(labels), labels = labels)


# Create a dataframe with the filtered data
# Apply filters
typical_homes <- subset(mydata,
                        Zoning %in% c("RH", "RL", "RM") &
                          SaleCondition == "Normal" &
                          BldgType == "1Fam" &
                          Functional == "Typ" &
                          SalePrice <= 339500 &
                          GrLivArea <= 2667)

# Display the size of the filtered dataframe
size <- dim(typical_homes)
print(paste("The filtered dataframe has", size[1], "rows and", size[2], "columns."))



## Section 3

# Get the names of the columns with missing values
columns_with_missing <- names(typical_homes)[colSums(is.na(typical_homes)) > 0]
print(columns_with_missing)

# Get the number of values missing in each of these columns
missing_counts <- colSums(is.na(typical_homes))
missing_counts <- missing_counts[missing_counts > 0]
print(missing_counts)

# Counting duplicate rows in typical_homes dataframe
duplicate_count <- sum(duplicated(typical_homes))
print(paste("Number of duplicate rows:", duplicate_count))

# Removing duplicate rows
typical_homes_unique <- unique(typical_homes)

# Checking YearBuilt and YearRemodel consistency
typical_homes <- typical_homes[!(typical_homes$YearBuilt > typical_homes$YearRemodel),]

# Filtering rows with negative SalePrice
negative_saleprice_rows <- typical_homes[typical_homes$SalePrice < 0, ]
# Viewing the rows with negative SalePrice values
print(negative_saleprice_rows)

# Histogram for SalePrice
hist(typical_homes$SalePrice, main="Histogram for SalePrice", xlab="SalePrice")

# Scatter plot for SalePrice vs. GrLivArea
plot(typical_homes$GrLivArea, typical_homes$SalePrice, main="Scatterplot", xlab="Living
Area", ylab="SalePrice")
```

```
graphics.off()
par(mar = c(5.1, 4.1, 4.1, 2.1))

# Select variables
cat('_____________________NEW_____________')
continuous_discrete_vars <- c("LotArea", "YearBuilt", "TotalBsmtSF", "GrLivArea",
"PoolArea", "YrSold", "YearRemodel",
                              "LotFrontage")
nominal_ordinal_vars <- c("Fence", "Utilities", "HouseStyle", "RoofStyle", "Street",
"Alley",
                          "Neighborhood", "Condition1", "Condition2", "OverallQual",
"OverallCond", "SaleCondition",
                          "GarageType")


# Create a list of selected variables
selected_vars <- c(
  "LotArea", "YearBuilt", "TotalBsmtSF", "GrLivArea", "PoolArea", "YrSold", "YearRemodel",
"LotFrontage",
  "Fence", "Utilities", "HouseStyle", "RoofStyle", "Street", "Alley", "Neighborhood",
  "Condition1", "Condition2", "OverallQual", "OverallCond", "SaleCondition", "GarageType"
)

# Subset the dataset with the selected variables
selected_data <- typical_homes[selected_vars]

# Summarize the data using table() for nominal and ordinal variables
nominal_ordinal_summaries <- lapply(selected_data, function(col) {
  if (is.factor(col)) {
    table(col)
  } else {
    NULL
  }
})

# Summarize the data using summary() for continuous and discrete variables
continuous_discrete_summaries <- lapply(selected_data, function(col) {
  if (is.numeric(col)) {
    summary(col)
  } else {
    NULL
  }
})

# Print the summaries
for (i in seq_along(nominal_ordinal_summaries)) {
  if (!is.null(nominal_ordinal_summaries[[i]])) {
    cat(paste("Summary for", names(nominal_ordinal_summaries)[i]), "\n")
    print(nominal_ordinal_summaries[[i]])
  }
}

for (i in seq_along(continuous_discrete_summaries)) {
  if (!is.null(continuous_discrete_summaries[[i]])) {
    cat(paste("Summary for", names(continuous_discrete_summaries)[i]), "\n")
    print(continuous_discrete_summaries[[i]])
  }
}


## Section 4


# Plots for Continuous Variables
```

```
# Create a new data frame with selected variables
selected_vars <- c("LotArea", "FirstFlrSF", "TotalBsmtSF", "GrLivArea", "SalePrice")
selected_data <- typical_homes[selected_vars]

# Generate plots for each selected variable
plots <- lapply(selected_vars, function(var) {
  # Generate histograms for selected continuous variables
  if (var != "SalePrice") {
    hist_plot <- ggplot(selected_data, aes(x = .data[[var]])) +
      geom_histogram(binwidth = 500) +  # Adjust binwidth as needed +
      labs(title = paste("Histogram of", var))
  } else {
    hist_plot <- NULL
  }

  # Generate box plots for selected continuous variables
  box_plot <- ggplot(selected_data, aes(x = 1, y = .data[[var]])) +
    geom_boxplot() +
    labs(title = paste("Box Plot of", var)) +
    theme(axis.title.x=element_blank(),
          axis.text.x=element_blank(),
          axis.ticks.x=element_blank())

  # Generate scatterplots with LOESS smoothers
  scatter_plot <- ggplot(selected_data, aes(x = .data[[var]], y = SalePrice)) +
    geom_point() +
    geom_smooth(method = "loess", se = FALSE, color = "blue") +
    labs(title = paste("Scatterplot of", var, "vs. SalePrice"))

  # Arrange plots in a grid (you can adjust the layout as needed)
  library(gridExtra)
  if (var != "SalePrice") {
    grid.arrange(hist_plot, box_plot, scatter_plot, ncol = 3)
  } else {
    grid.arrange(box_plot, scatter_plot, ncol = 2)
  }
})

# Print or visualize the plots (you can modify as needed)
for (plot in plots) {
  print(plot)
}
graphics.off()
par(mar = c(5.1, 4.1, 4.1, 2.1))

# Print the plots for SalePrice
# Create a new data frame with SalePrice variable
sale_price_data <- typical_homes["SalePrice"]

# Set options to prevent scientific notation in labels
options(scipen = 999)

# Set up a multi-panel layout
par(mfrow = c(1, 2))  # 1 row, 2 columns

# Create a boxplot for SalePrice in blue
boxplot(sale_price_data, col = "lightblue", main = "SalePrice Boxplot", ylab =
"SalePrice")

# Create a histogram for SalePrice in blue
hist(sale_price_data$SalePrice, col = "lightblue", main = "SalePrice Histogram", xlab =
"SalePrice")
```

```r
# Identify the outliers and print the statistical summary
# Create a new data frame with selected variables including SalePrice
selected_vars <- c("SalePrice", "LotArea", "FirstFlrSF", "TotalBsmtSF", "GrLivArea")
selected_data <- typical_homes[selected_vars]

# Function to calculate outlier limits and count outliers
calculate_outliers <- function(var) {
  # Calculate quartiles
  q1 <- quantile(var, 0.25, na.rm = TRUE)
  q3 <- quantile(var, 0.75, na.rm = TRUE)

  # Calculate the interquartile range (IQR)
  iqr <- q3 - q1

  # Calculate lower and upper bounds for outliers
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr

  # Identify outliers
  outliers_below <- var[var < lower_bound]
  outliers_above <- var[var > upper_bound]

  return(list(
    Lower_Bound = lower_bound,
    Upper_Bound = upper_bound,
    Num_Outliers_Below = length(outliers_below),
    Num_Outliers_Above = length(outliers_above)
  ))
}

# Calculate outliers for each selected variable
outliers_info <- lapply(selected_data, calculate_outliers)

# Print the outlier limits, number of outliers, and statistical summary for each variable
for (i in seq_along(outliers_info)) {
  var_name <- names(outliers_info)[i]
  cat("\nVariable:", var_name, "\n")
  cat("Lower Bound:", outliers_info[[i]]$Lower_Bound, "\n")
  cat("Upper Bound:", outliers_info[[i]]$Upper_Bound, "\n")
  cat("Number of Outliers Below:", outliers_info[[i]]$Num_Outliers_Below, "\n")
  cat("Number of Outliers Above:", outliers_info[[i]]$Num_Outliers_Above, "\n")

  # Print statistical summary with labels
  cat("Statistical Summary:", "\n")
  summary_values <- summary(selected_data[[var_name]])
  cat(paste(names(summary_values), summary_values, sep = ": "), "\n")
}



# Plots for Categorical variables


# Categorical variables of interest
categorical_vars <- c("OverallQual", "OverallCond", "SaleCondition", "GarageType",
"Fence", "HouseStyle")

# Initialize a list to store plots
all_plots <- list()

# Loop through each categorical variable and create plots
for (var in categorical_vars) {
  # Create a bar chart
  bar_chart <- ggplot(typical_homes, aes(x = reorder(.data[[var]], -SalePrice), fill =
.data[[var]])) +
```

```r
    geom_bar() +
    labs(title = paste("Bar Chart of", var), x = var, y = "Count") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  # Create a count plot
  count_plot <- ggplot(typical_homes, aes(x = .data[[var]], fill = .data[[var]])) +
    geom_bar() +
    labs(title = paste("Count Plot of", var), x = var, y = "Count") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  # Create a box plot
  box_plot <- ggplot(typical_homes, aes(x = .data[[var]], y = SalePrice, fill =
.data[[var]])) +
    geom_boxplot() +
    labs(title = paste("Box Plot of", var), x = var, y = "Sale Price") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  # Arrange the plots in a single panel
  combined_plot <- grid.arrange(bar_chart, count_plot, box_plot, ncol = 3)

  # Store the combined plot in the list
  all_plots[[var]] <- combined_plot
}


# Print the combined plots for all categorical variables
for (var in categorical_vars) {
  print(all_plots[[var]])
}
graphics.off()
par(mar = c(5.1, 4.1, 4.1, 2.1))

## Section 5


# Create scatterplots for SalePrice and log(SalePrice) against "LotArea," "FirstFlrSF,"
and "GrLivArea"
variables_to_plot <- c("LotArea", "FirstFlrSF", "GrLivArea")
# Assuming 'SalePrice' is a column in your dataframe 'typical_homes'
typical_homes$log_SalePrice <- log(typical_homes$SalePrice)


# Loop through the predictor variables and create scatterplots
for (var in variables_to_plot) {
  # Scatterplot for SalePrice
  plot_saleprice <- ggplot(typical_homes, aes_string(x = var, y = "SalePrice")) +
    geom_point(color = "blue") +
    labs(title = paste("Scatterplot of SalePrice vs.", var),
         x = var, y = "SalePrice")

  # Scatterplot for log(SalePrice)
  plot_log_saleprice <- ggplot(typical_homes, aes_string(x = var, y = "log_SalePrice")) +
    geom_point(color = "red") +
    labs(title = paste("Scatterplot of log(SalePrice) vs.", var),
         x = var, y = "log(SalePrice)")

  # Display both scatterplots in a grid
  grid.arrange(plot_saleprice, plot_log_saleprice, ncol = 2)
}

par(mar = c(5.1, 4.1, 4.1, 2.1))
```

```
# Computing the Skewness

# Compute skewness for SalePrice
skewness_saleprice <- skewness(typical_homes$SalePrice, na.rm = TRUE)

# Compute skewness for log(SalePrice)
skewness_log_saleprice <- skewness(log(typical_homes$SalePrice), na.rm = TRUE)

# Print the skewness values
cat("Skewness of SalePrice:", skewness_saleprice, "\n")
cat("Skewness of log(SalePrice):", skewness_log_saleprice, "\n")


# Comparing Linear Regression Models
# Set a random seed for reproducibility
set.seed(123)

# Split the data into training (70%) and testing (30%) sets
n <- nrow(typical_homes)
train_indices <- sample(1:n, 0.7 * n)  # 70% for training
train_data <- typical_homes[train_indices, ]
test_data <- typical_homes[-train_indices, ]

# Create linear regression models for SalePrice and log(SalePrice) using training data
lm_model_saleprice_train <- lm(SalePrice ~ LotArea + FirstFlrSF + GrLivArea, data =
train_data)
lm_model_log_saleprice_train <- lm(log(SalePrice) ~ LotArea + FirstFlrSF + GrLivArea, data
= train_data)

# Make predictions on the test data
predictions_saleprice <- predict(lm_model_saleprice_train, newdata = test_data)
predictions_log_saleprice <- exp(predict(lm_model_log_saleprice_train, newdata =
test_data))

# Calculate RMSE for SalePrice on the test data
rmse_saleprice <- sqrt(mean((test_data$SalePrice - predictions_saleprice)^2))

# Calculate RMSE for log(SalePrice) on the test data
rmse_log_saleprice <- sqrt(mean((log(test_data$SalePrice) -
log(predictions_log_saleprice))^2))

# Print the RMSE values for both models
cat("RMSE for SalePrice on Test Data:", rmse_saleprice, "\n")
cat("RMSE for log(SalePrice) on Test Data:", rmse_log_saleprice, "\n")


# Calculate RMSE for SalePrice units
rmse_saleprice <- exp(rmse_log_saleprice) - 1

cat("RMSE for SalePrice on Test Data:", rmse_saleprice, "\n")

# _____________________________________________________
# _____________________________________________________


### Assignment 3


# Question 12
# Creating multiple regression model using the explanatory variables from the first set
model_3 <- lm(SalePrice ~ FirstFlrSF + SecondFlrSF + GrLivArea + TotalBsmtSF, data=mydata)
summary(model_3)


##### Hnadling the missing values
```

```
colSums(is.na(mydata[, c("FirstFlrSF", "SecondFlrSF", "GrLivArea", "TotalBsmtSF",
                         "LotFrontage", "LotArea", "BsmtFinSF1", "BsmtFinSF2",
                         "BsmtUnfSF", "MasVnrArea", "SalePrice")]))

# Delete rows where TotalBsmtSF is NA (only one row)
mydata <- mydata[!is.na(mydata$TotalBsmtSF), ]

# Replace 490 missing values in LotFrontage with its mean (Every house is ought to be
connected to
# the street, replacing the missing values with the mean of linear feet connected to
street)
mydata$LotFrontage[is.na(mydata$LotFrontage)] <- mean(mydata$LotFrontage, na.rm = TRUE)

# Replace missing values in MasVnrArea with 0 (Replacng 23 Missing Masonry veneer area 0)
mydata$MasVnrArea[is.na(mydata$MasVnrArea)] <- 0

# Checking the missing values
colSums(is.na(mydata[, c("FirstFlrSF", "SecondFlrSF", "GrLivArea", "TotalBsmtSF",
                         "LotFrontage", "LotArea", "BsmtFinSF1", "BsmtFinSF2",
                         "BsmtUnfSF", "MasVnrArea", "SalePrice")]))

# Question 13
# Creating multiple regression model with this combined set of explanatory variables
model_4 <- lm(SalePrice ~ FirstFlrSF + SecondFlrSF + GrLivArea + TotalBsmtSF +
                LotFrontage + LotArea + BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF + MasVnrArea,
              data=mydata)
summary(model_4)

# Checking is Model_3 is nested within Model_4
anova(model_3, model_4)
```