Untitled

2023-11-18

```r
# Load required library
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
```

```r
library(readr)
library(ggplot2)
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.3.1

##
## Attaching package: 'lubridate'

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

```r
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
# Load the dataset "raw_sales"
data_sales <- read_csv("/Users/amyou/Desktop/ADS 506/ADS 506 Final Project/raw_sales.csv")
```

```
## Rows: 29580 Columns: 5

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): propertyType
## dbl  (3): postcode, price, bedrooms
## dttm (1): datesold
##
## 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.
```

```r
# Transform the "datesold" column to ensure it is in date format
data_sales$datesold <- as.Date(data_sales$datesold, format = "%m/%d/%Y %H:%M")
```

```r
# Create a new column "YearMonth" with the first day of each month
data_sales <- data_sales %>%
  mutate(YearMonth = floor_date(datesold, unit = "month"))

# Aggregate data into monthly time index
monthly_data <- data_sales %>%
  group_by(YearMonth) %>%
  summarise(
    TotalPrice = sum(price),
    AvgBedrooms = mean(bedrooms)
  )

# Create a new column "Quarter" to represent the quarter
data_sales <- data_sales %>%
  mutate(Quarter = quarter(datesold, with_year = TRUE))

# Aggregate data into quarterly time index
quarterly_data <- data_sales %>%
  group_by(Quarter) %>%
  summarise(
    TotalPrice = sum(price),
    AvgBedrooms = mean(bedrooms)
  )

# View the first few rows of the aggregated data
head(monthly_data)
```

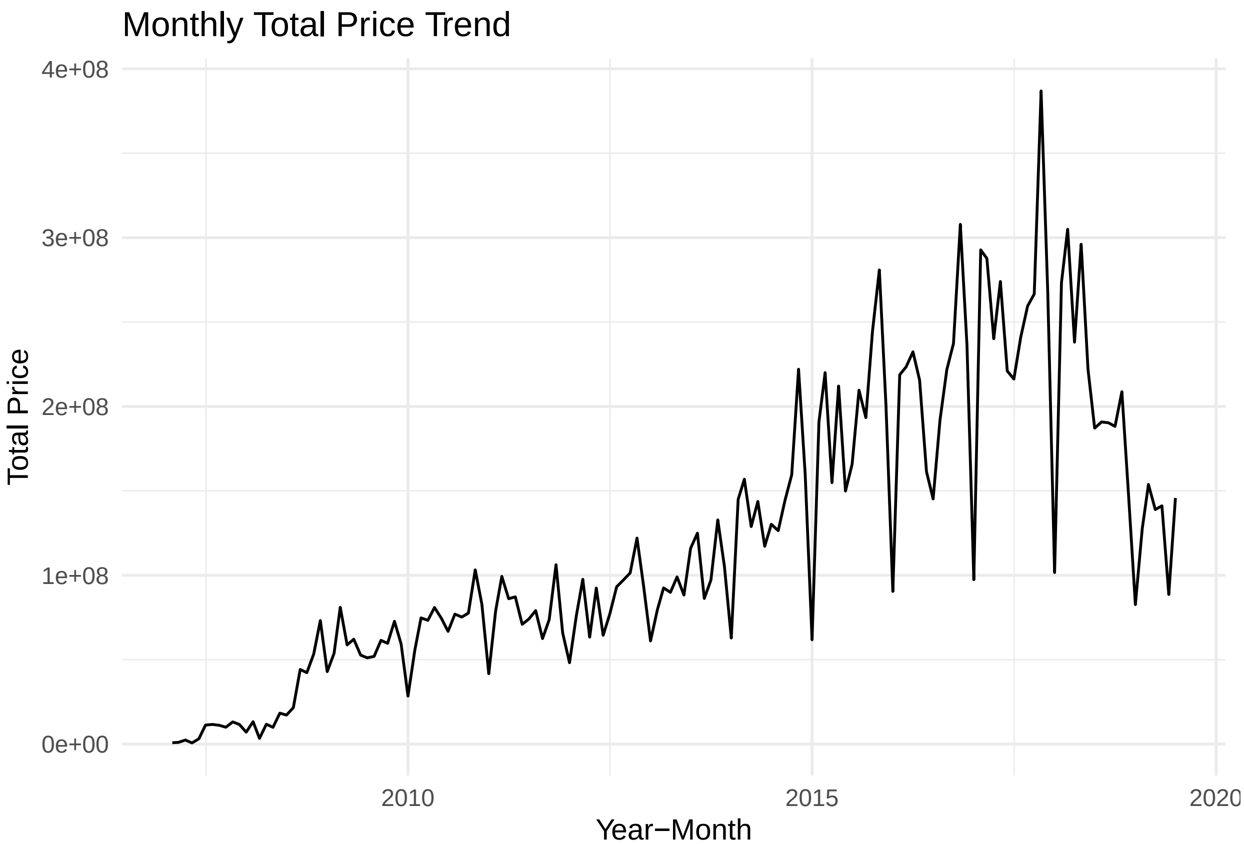
```
## # A tibble: 6 x 3
##   YearMonth  TotalPrice AvgBedrooms
##   <date>          <dbl>       <dbl>
## 1 2007-02-01     815000        3.5
## 2 2007-03-01    1018000        3.33
## 3 2007-04-01    2394000        3.67
## 4 2007-05-01     679000        3
## 5 2007-06-01    3122000        3.33
## 6 2007-07-01   11249500        3.26
```

```r
head(quarterly_data)
```

```
## # A tibble: 6 x 3
##   Quarter TotalPrice AvgBedrooms
##     <dbl>      <dbl>       <dbl>
## 1   2007.    1833000        3.4
## 2   2007.    6195000        3.36
## 3   2007.   34016000        3.32
## 4   2007.   34745450        3.25
## 5   2008.   23720000        3.47
## 6   2008.   40007150        3.41
```

```r
# Plot Monthly Total Price Trend
ggplot(monthly_data, aes(x = YearMonth, y = TotalPrice)) +
  geom_line() +
  labs(
    title = "Monthly Total Price Trend",
```

```
    x = "Year-Month",
    y = "Total Price"
  ) +
  theme_minimal()
```




```
# Check for missing values in monthly_data
sum(is.na(monthly_data$TotalPrice))
```

```
## [1] 0
```

```
sum(is.na(monthly_data$YearMonth))
```

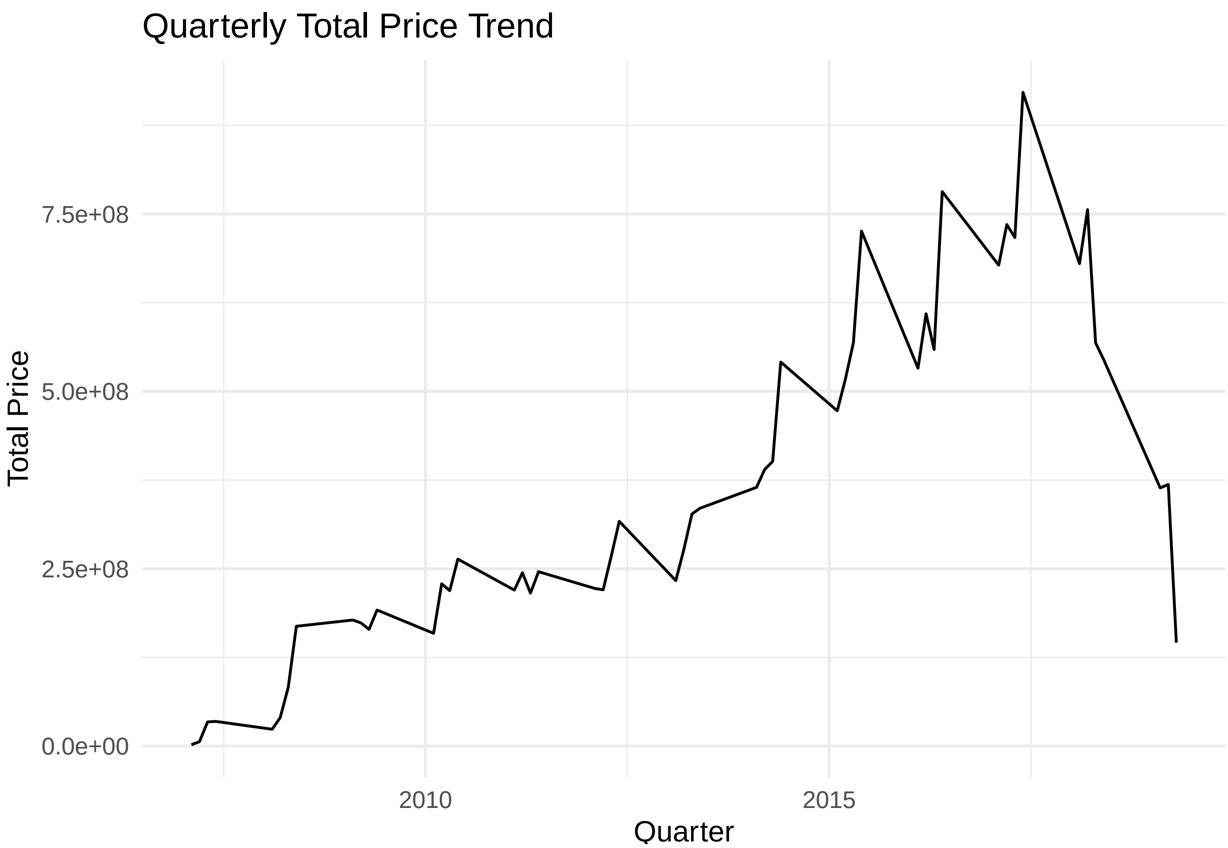
```
## [1] 0
```

```
library(zoo)  # For the na.locf function
```

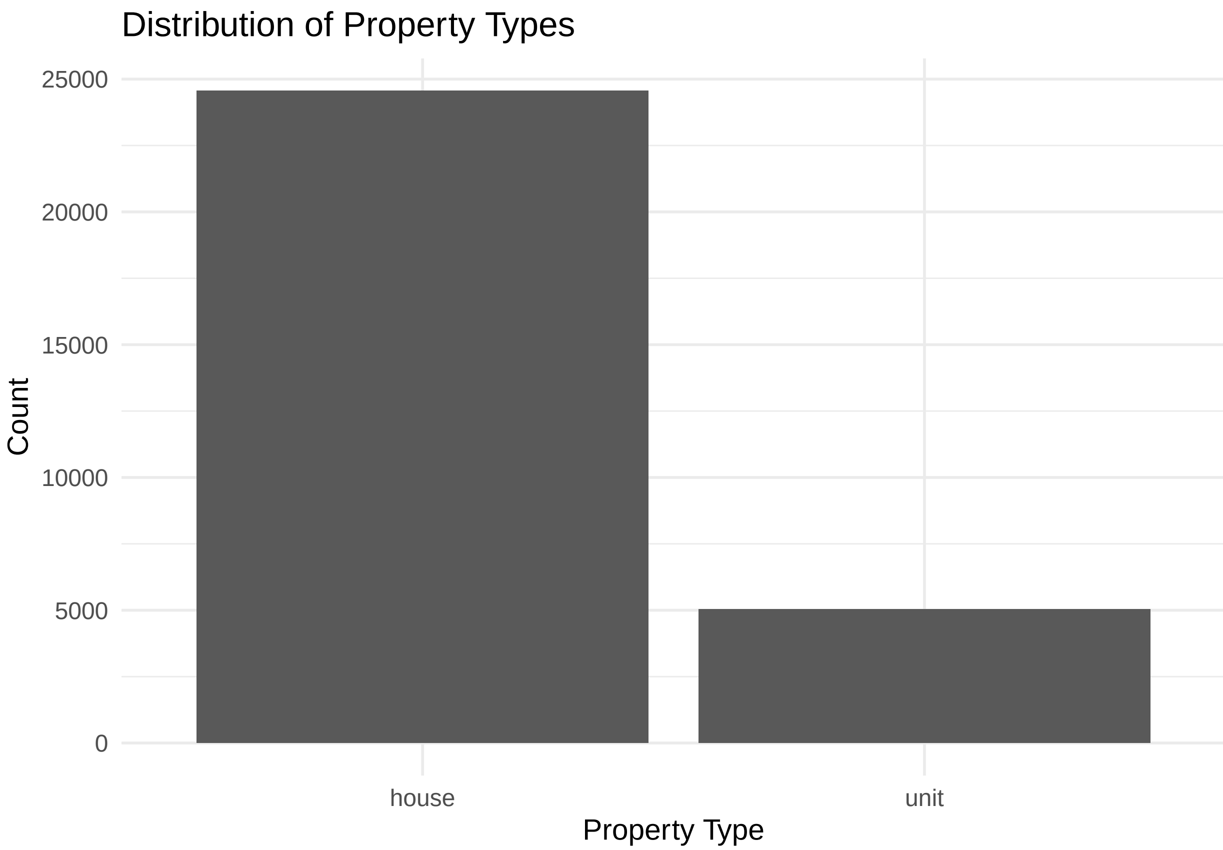
```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
monthly_data$YearMonth <- zoo::na.locf(monthly_data$YearMonth)

# Plot Quarterly Total Price Trend
ggplot(quarterly_data, aes(x = Quarter, y = TotalPrice)) +
  geom_line() +
  labs(
```

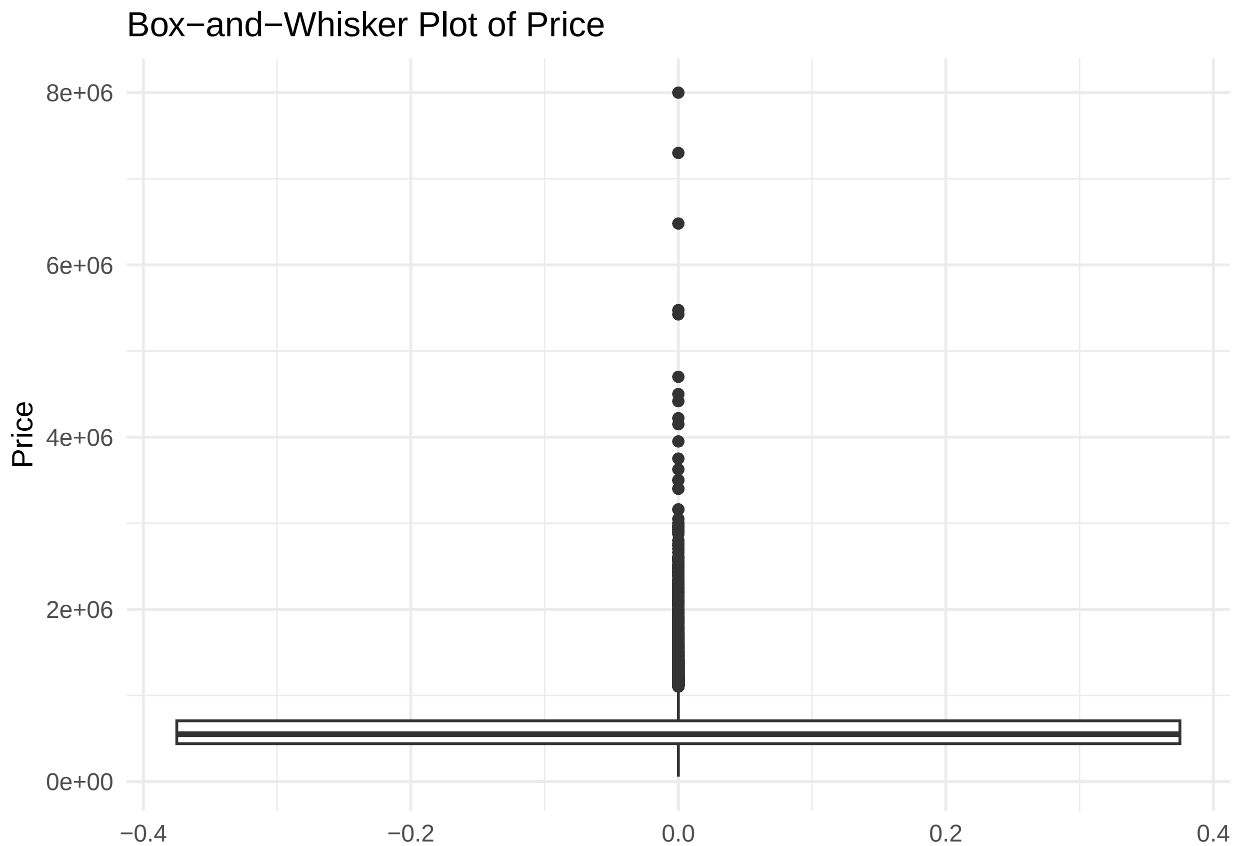
```
    title = "Quarterly Total Price Trend",
    x = "Quarter",
    y = "Total Price"
  ) +
  theme_minimal()
```




```
# Additional Analysis and Visualization

# Example: Histogram of property types
ggplot(data_sales, aes(x = propertyType)) +
  geom_bar() +
  labs(
    title = "Distribution of Property Types",
    x = "Property Type",
    y = "Count"
  ) +
  theme_minimal()
```




```r
# Create a box-and-whisker plot for the "price" column
ggplot(data_sales, aes(y = price)) +
  geom_boxplot() +
  labs(
    title = "Box-and-Whisker Plot of Price",
    y = "Price"
  ) +
  theme_minimal()
```




```r
# EDA

# Highlight potential outliers
# Calculate the lower and upper bounds for potential outliers
q1 <- quantile(data_sales$price, 0.25)
q3 <- quantile(data_sales$price, 0.75)
iqr <- q3 - q1
lower_bound <- q1 - 1.5 * iqr
upper_bound <- q3 + 1.5 * iqr

# Identify potential outliers
outliers <- data_sales[data_sales$price < lower_bound | data_sales$price > upper_bound,]

# Print the identified outliers
cat("Identified Outliers:\n")
```

```
## Identified Outliers:
```

```r
print(outliers)
```

```
## # A tibble: 1,374 x 7
##    datesold   postcode   price propertyType bedrooms YearMonth  Quarter
##    <date>        <dbl>   <dbl> <chr>           <dbl> <date>       <dbl>
##  1 2007-04-30     2606 1530000 house               4 2007-04-01   2007.
##  2 2007-07-21     2603 1780000 house               4 2007-07-01   2007.
##  3 2007-09-21     2603 1460000 house               5 2007-09-01   2007.
##  4 2007-12-13     2612 1105000 house               4 2007-12-01   2007.
##  5 2008-06-02     2603 1180000 house               4 2008-06-01   2008.
```

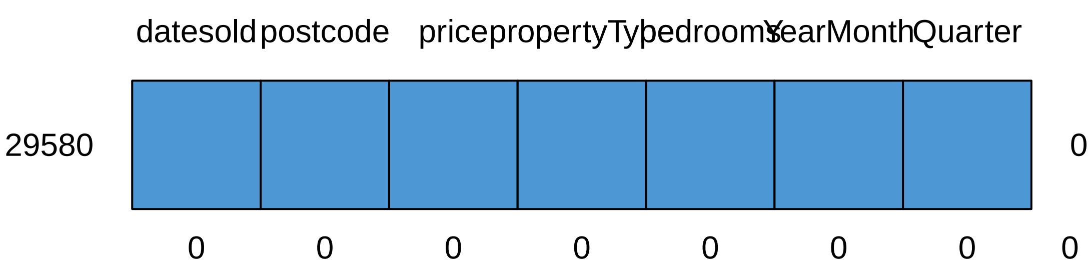
```
##   6 2008-06-30     2607 1165000 house              5 2008-06-01   2008.
##   7 2008-07-18     2602 1300000 house              4 2008-07-01   2008.
##   8 2008-09-02     2600 1380000 house              5 2008-09-01   2008.
##   9 2008-09-04     2913 1150000 house              5 2008-09-01   2008.
##  10 2008-10-27     2602 1120000 house              4 2008-10-01   2008.
## # i 1,364 more rows
```

```r
# Load required libraries
library(mice)
```

```
##
## Attaching package: 'mice'
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
# Create a missing data pattern plot
md.pattern(data_sales)
```

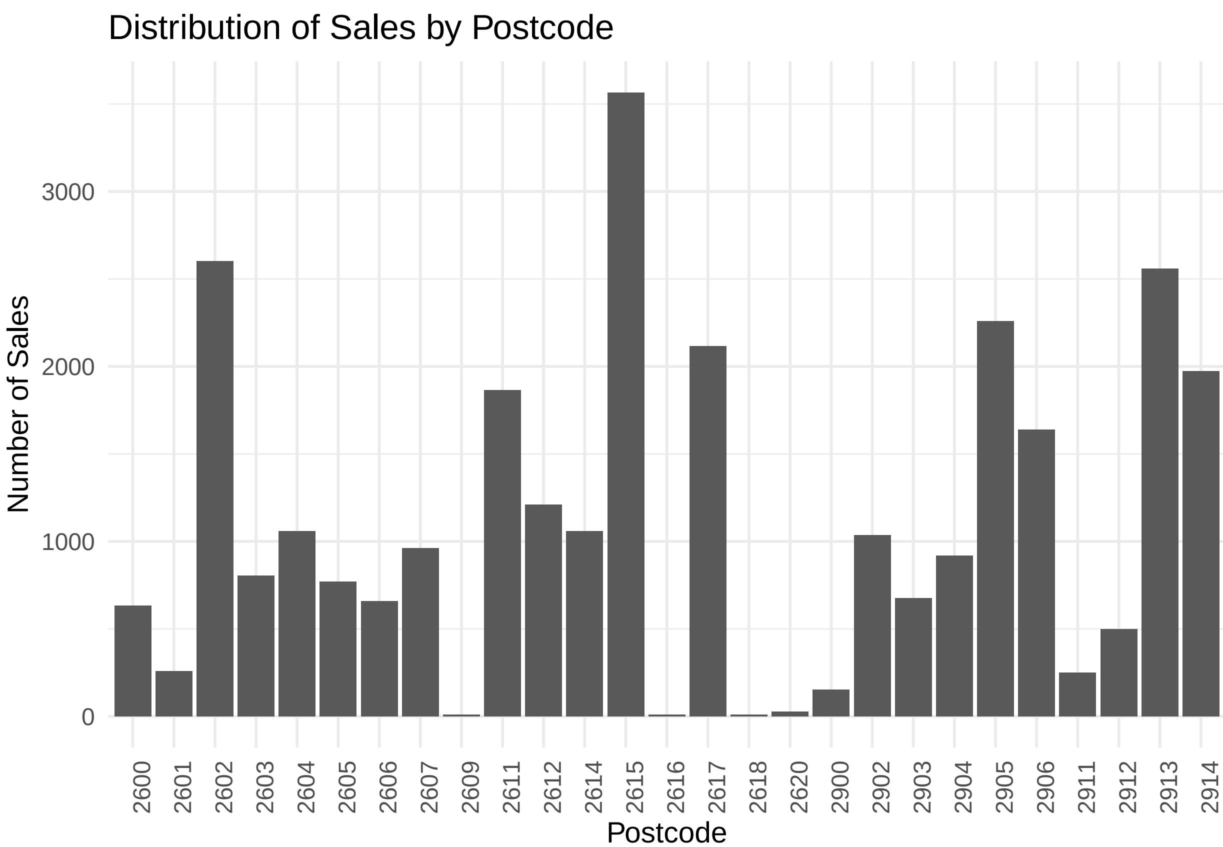
```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```



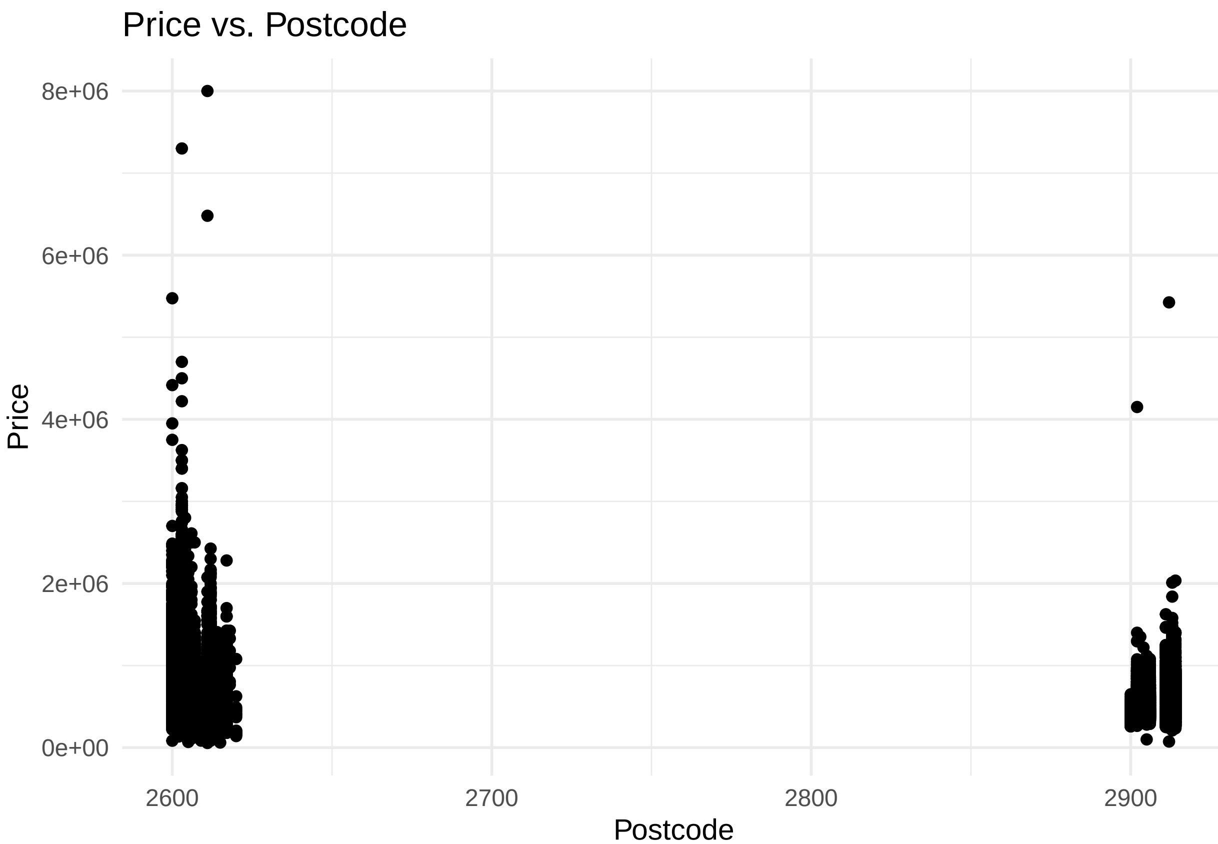

```
##       datesold postcode price propertyType bedrooms YearMonth Quarter
## 29580        1        1     1            1        1         1       1 0
##              0        0     0            0        0         0       0 0
```

```r
# Summary statistics for "price"
summary_price <- summary(data_sales$price)
mean_price <- mean(data_sales$price)
median_price <- median(data_sales$price)
min_price <- min(data_sales$price)
max_price <- max(data_sales$price)
sd_price <- sd(data_sales$price)

# Summary statistics for "bedrooms"
summary_bedrooms <- summary(data_sales$bedrooms)
mean_bedrooms <- mean(data_sales$bedrooms)
median_bedrooms <- median(data_sales$bedrooms)
min_bedrooms <- min(data_sales$bedrooms)
```

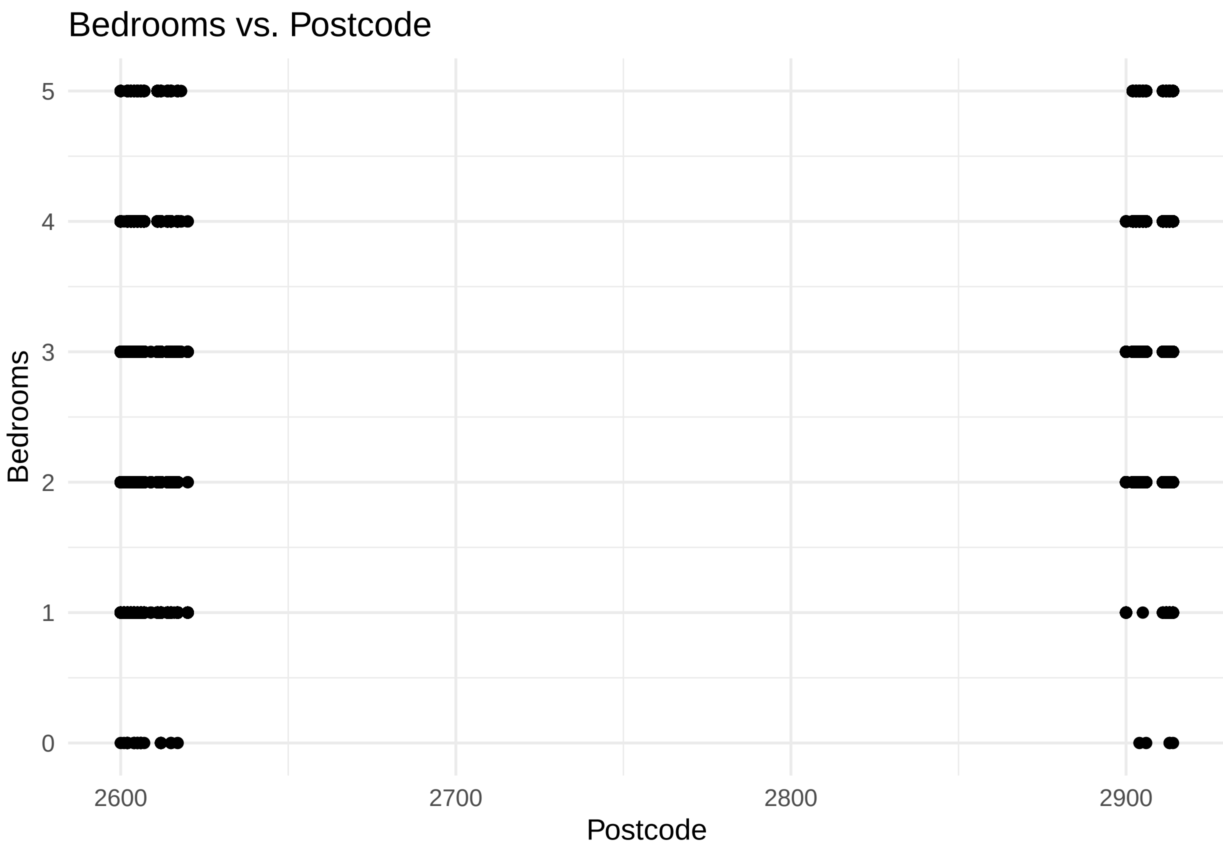
```r
max_bedrooms <- max(data_sales$bedrooms)
sd_bedrooms <- sd(data_sales$bedrooms)

# Create a bar plot to show the distribution of sales by "postcode"
postcode_counts <- table(data_sales$postcode)

ggplot(data = as.data.frame(postcode_counts), aes(x = Var1, y = Freq)) +
  geom_bar(stat = "identity") +
  labs(
    title = "Distribution of Sales by Postcode",
    x = "Postcode",
    y = "Number of Sales"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
# Create scatterplots to explore location-specific trends
ggplot(data_sales, aes(x = postcode, y = price)) +
  geom_point() +
  labs(
    title = "Price vs. Postcode",
    x = "Postcode",
    y = "Price"
  ) +
  theme_minimal()
```




```r
ggplot(data_sales, aes(x = postcode, y = bedrooms)) +
  geom_point() +
  labs(
    title = "Bedrooms vs. Postcode",
    x = "Postcode",
    y = "Bedrooms"
  ) +
  theme_minimal()
```



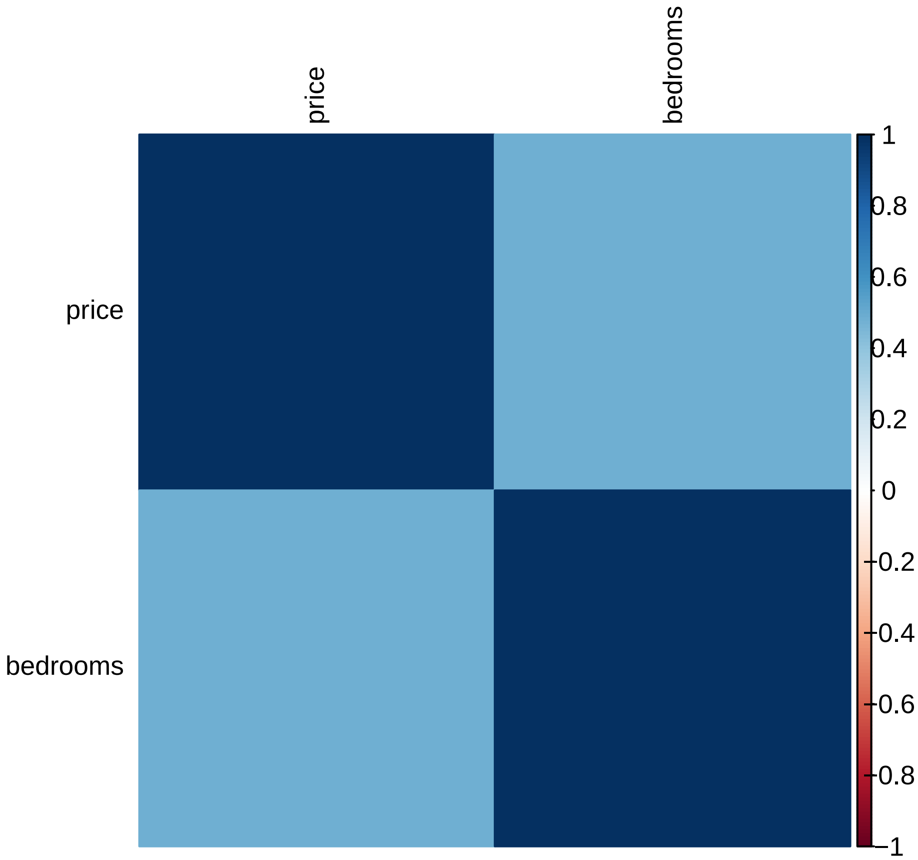

```r
# Calculate the Pearson correlation between "price" and "bedrooms"
correlation_price_bedrooms <- cor(data_sales$price, data_sales$bedrooms)

# Print the correlation coefficient
cat("Pearson's Correlation between Price and Bedrooms: ", correlation_price_bedrooms, "\n")
```

```
## Pearson's Correlation between Price and Bedrooms:  0.4842117
```

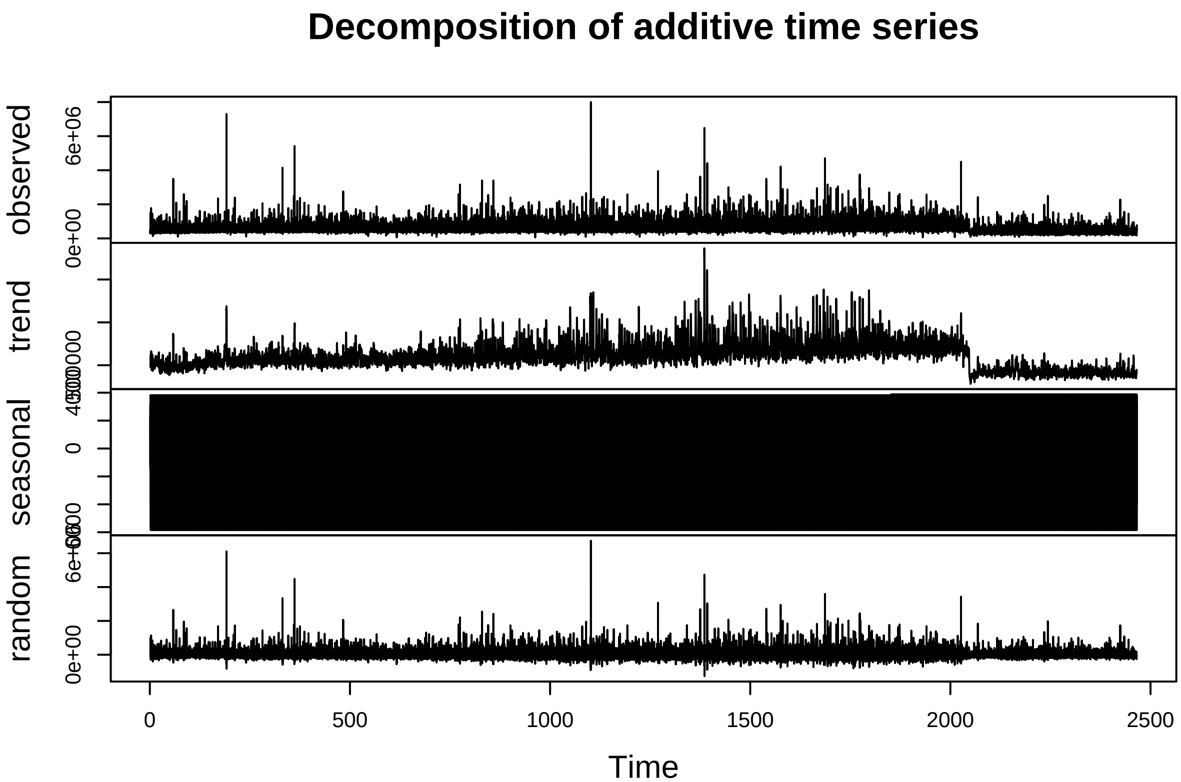
```r
# Install required library
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
# Select the variables for correlation plot
selected_vars <- c("price", "bedrooms")

# Calculate the correlation matrix for these variables
correlation_matrix <- cor(data_sales[, selected_vars])

# create correlation plot
corrplot(correlation_matrix, method = "color", tl.cex = 0.8, tl.col = "black")
```



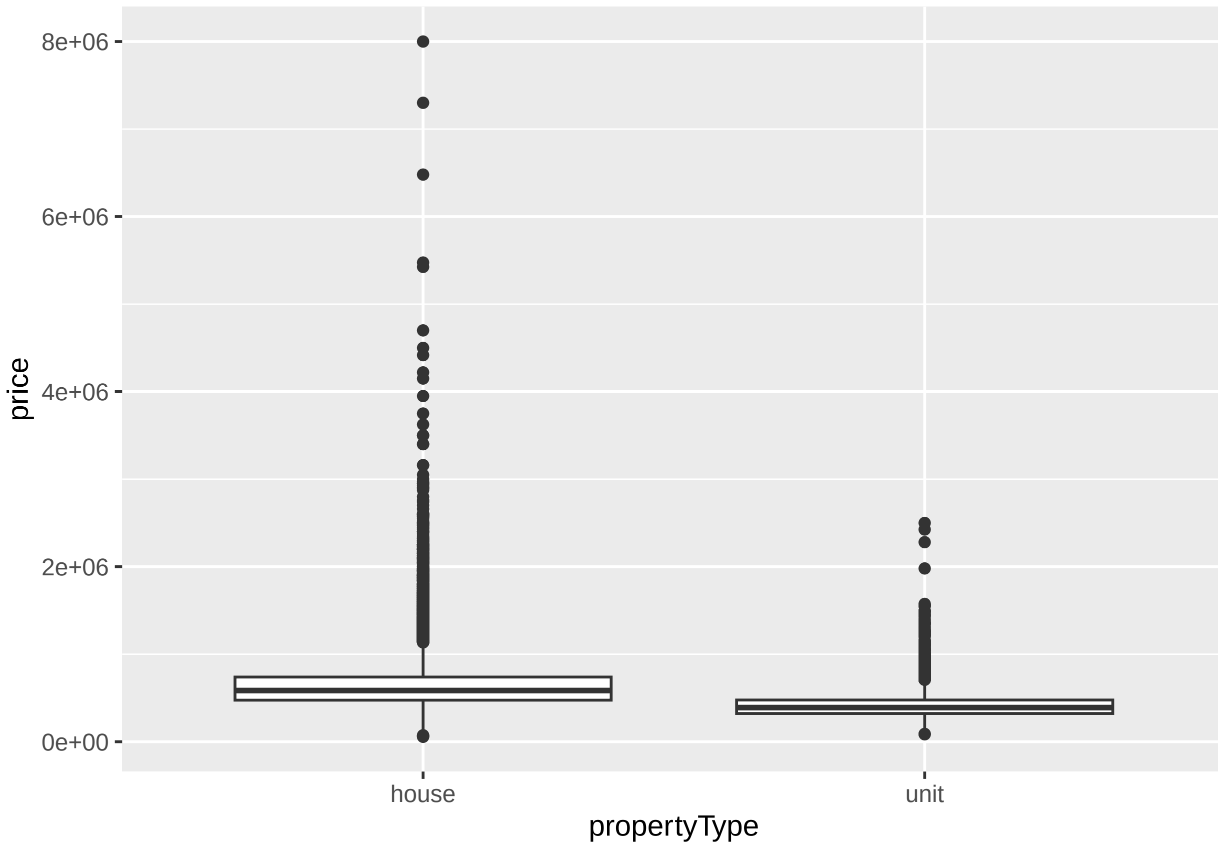

```r
# Decomposing a time series into its components, to include random, seasonal, trend and observed, can p
# Create a time series object
ts_data <- ts(data_sales$price, frequency = 12)  # Set frequency to 12 for monthly data

# Decompose the time series
decomposed_ts <- decompose(ts_data)

# Plot the decomposed components
plot(decomposed_ts)
```




```r
colSums(is.na(data_sales))
```

```
##      datesold     postcode        price propertyType     bedrooms    YearMonth
##             0            0            0            0            0            0
##       Quarter
##             0
```

```r
library(ggplot2)
ggplot(data_sales, aes(x = propertyType, y = price)) +
  geom_boxplot()
```



Below, we filter out the outliers based on interquartile ranges using the standard principle of 1.5 times the lowest and highest quartile. We then display the number of rows with outliers to see how many there are:

```
Q1 <- quantile(data_sales$price, 0.25)
Q3 <- quantile(data_sales$price, 0.75)
IQR <- IQR(data_sales$price)
outliers <- subset(data_sales, price < (Q1 - 1.5 * IQR) | price > (Q3 + 1.5 * IQR))
nrow(outliers)
```

```
## [1] 1374
```

Below: we show the total amount of rows:

```
nrow(data_sales)
```

```
## [1] 29580
```

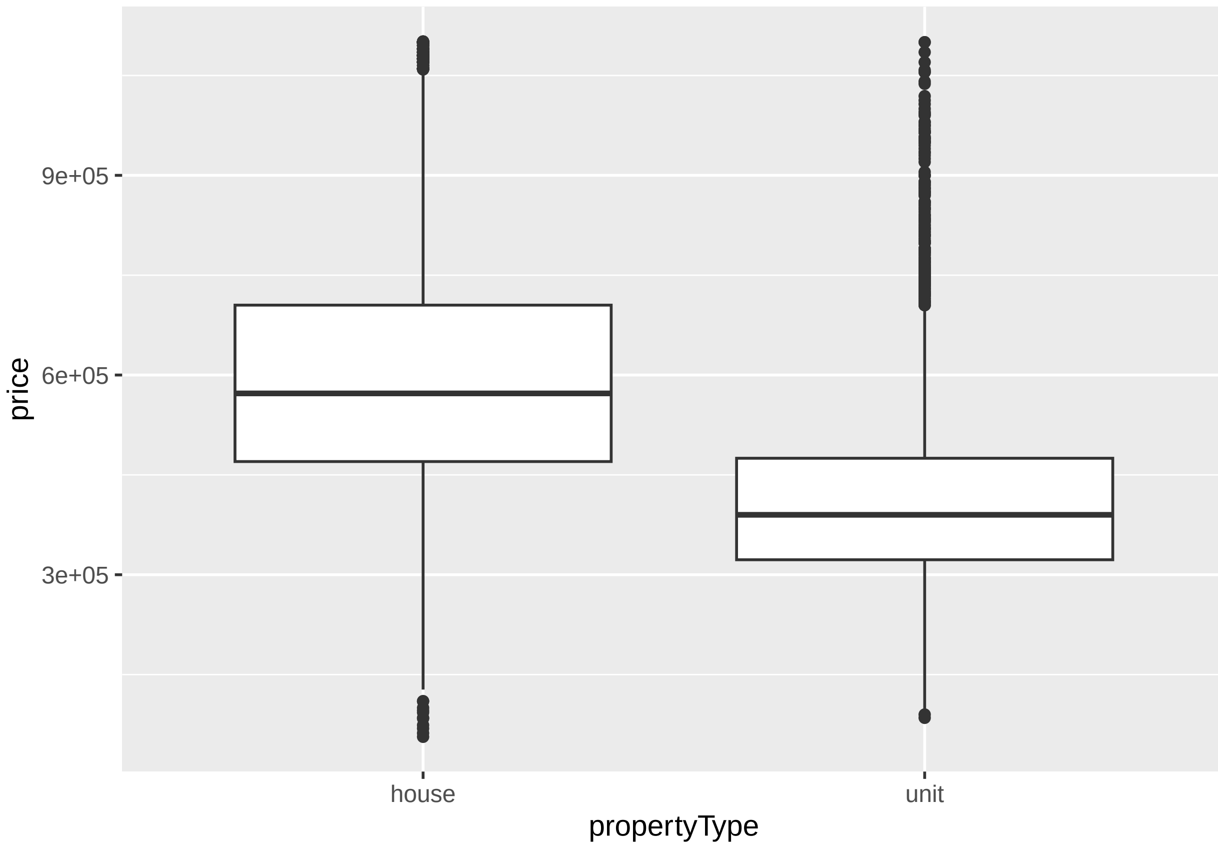
Next, we remove outliers:

```
clean_data <- subset(data_sales, price >= (Q1 - 1.5 * IQR) & price <= (Q3 + 1.5 * IQR))
nrow(clean_data)
```

```
## [1] 28206
```

Next we show the boxplots of prices after removing outliers

```
library(ggplot2)
ggplot(clean_data, aes(x = propertyType, y = price)) +
  geom_boxplot()
```



Next, we perform data splitting with an 80/20 approach

```r
library(readr)
library(caret)
```

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

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats 1.0.0     v tibble  3.2.1
## v purrr   1.0.1     v tidyr   1.3.0
## v stringr 1.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x mice::filter() masks dplyr::filter(), stats::filter()
## x dplyr::lag()   masks stats::lag()
## x purrr::lift()  masks caret::lift()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
# Splitting the data into training and testing sets
set.seed(123)  # Setting seed for reproducibility
splitIndex <- createDataPartition(clean_data$price, p = 0.8, list = FALSE)

# Creating training and testing datasets
train_data <- clean_data[splitIndex, ]
test_data <- clean_data[-splitIndex, ]

# Displaying the number of rows in training and testing
```

```r
nrow(train_data)
```

```
## [1] 22565
```

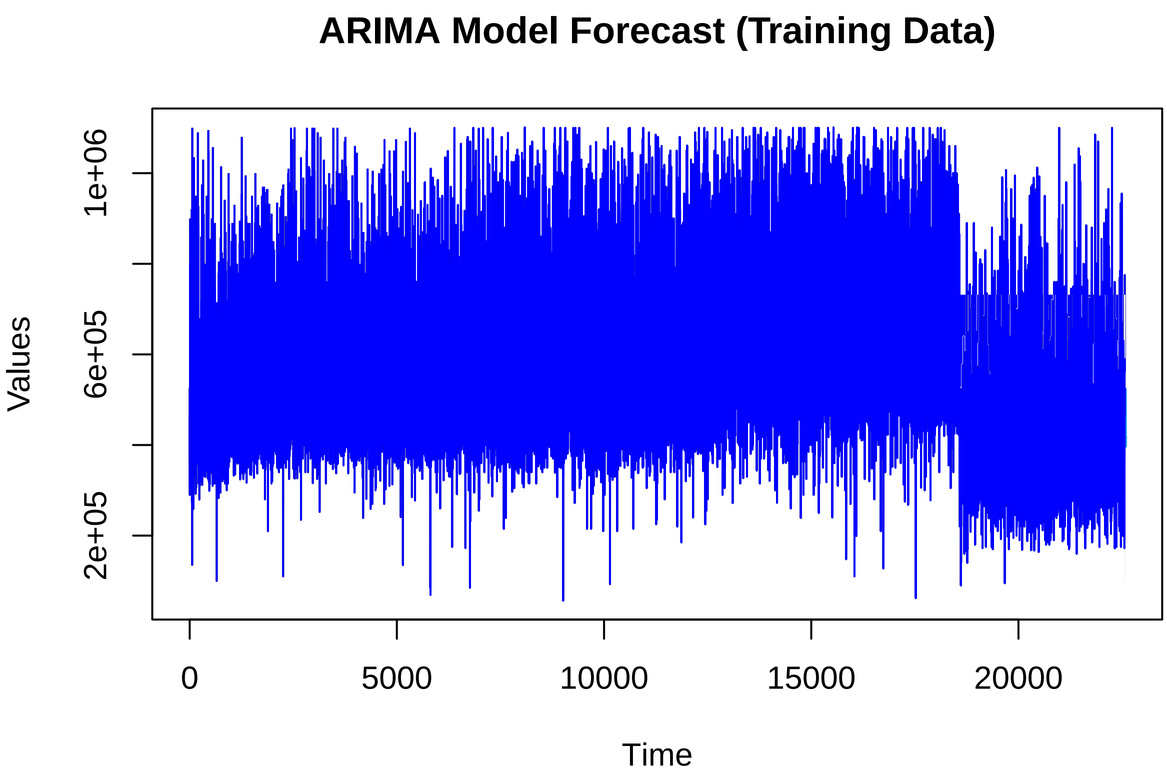
```r
nrow(test_data)
```

```
## [1] 5641
```

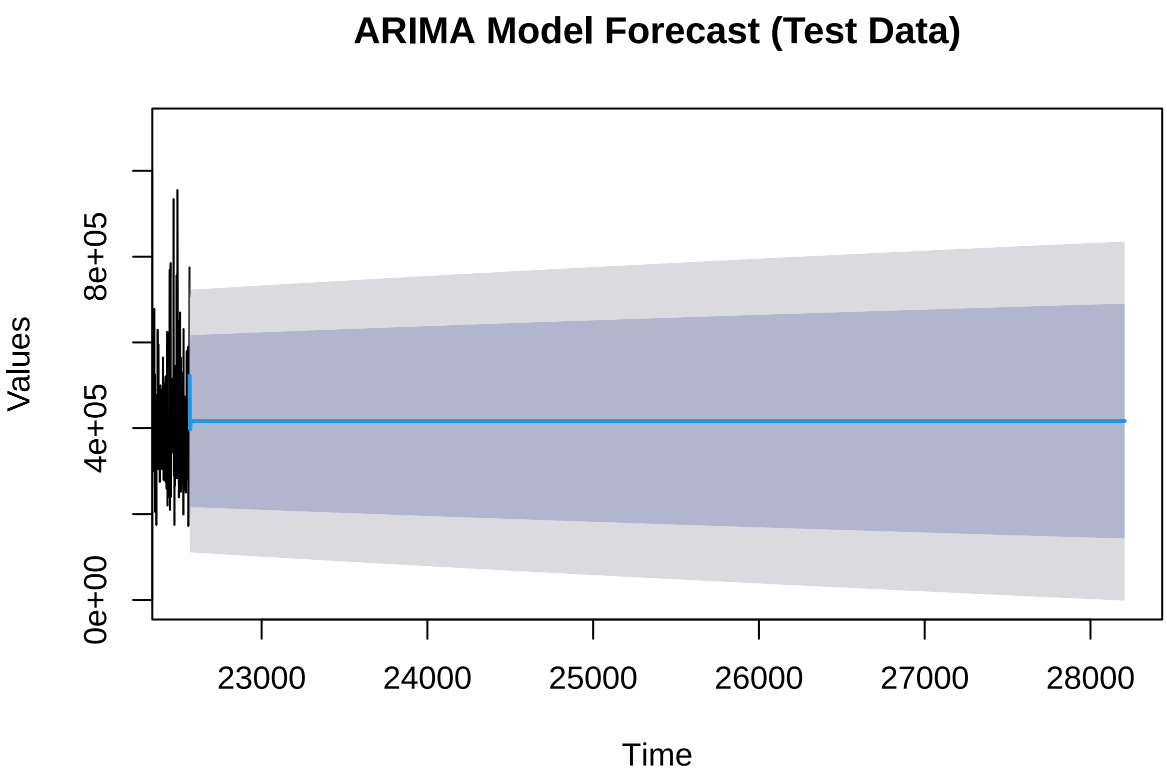
```r
# Create a time series object
ts_train <- ts(train_data$price, frequency = 1)

# Fit an ARIMA model using auto.arima
arima <- auto.arima(ts_train)

# Print the model summary
print(summary(arima))
```

```
## Series: ts_train
## ARIMA(2,1,3)
##
## Coefficients:
##          ar1      ar2      ma1    ma2      ma3
##       0.8867  -0.2968  -1.5558  0.656  -0.0948
## s.e.  0.0664   0.0242   0.0669  0.067   0.0209
##
## sigma^2 = 2.178e+10:  log likelihood = -300573.9
## AIC=601159.7   AICc=601159.7   BIC=601207.9
##
## Training set error measures:
##                     ME   RMSE      MAE       MPE     MAPE      MASE
## Training set -359.8651 147548 110949.7 -7.357757 21.80914 0.9528801
##                      ACF1
## Training set -0.0002277707
```

```r
# Plot the fitted values and observed values on the training data
plot(forecast(arima), main = "ARIMA Model Forecast (Training Data)", xlab = "Time", ylab = "Values", xli
lines(ts_train, col = "blue")
```



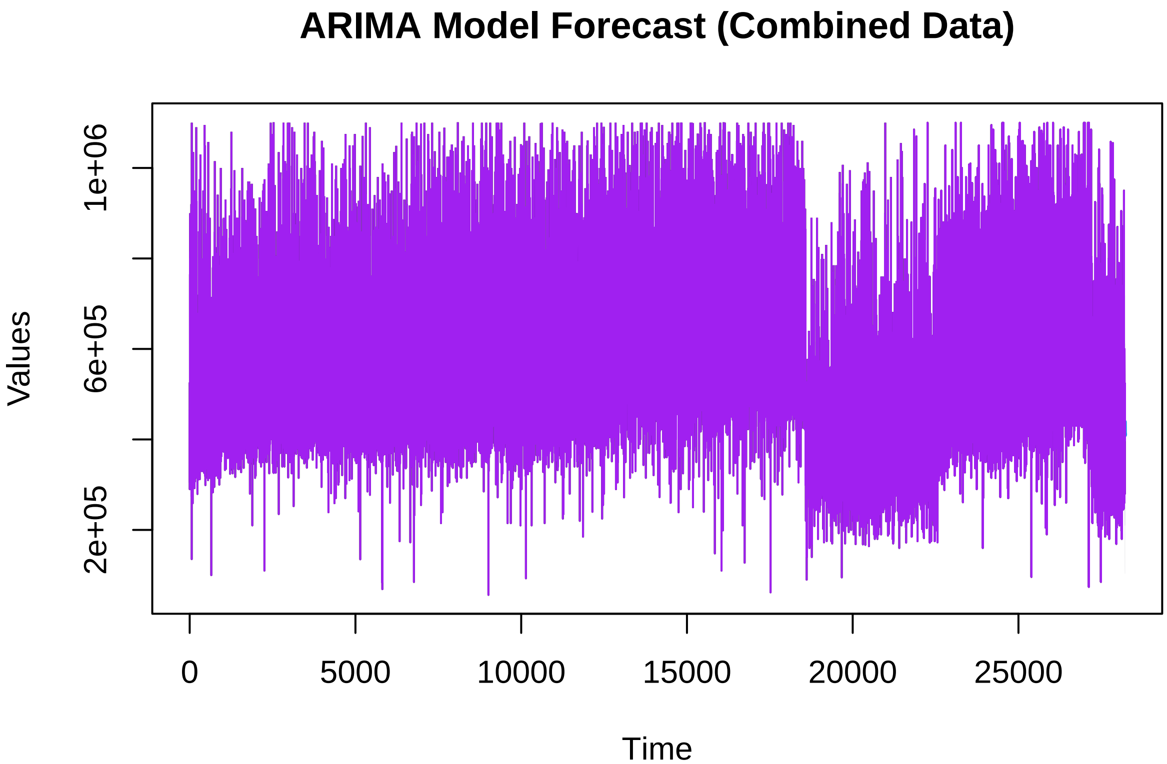

```
# Forecast on the test data
forecast_values <- forecast(arima, h = nrow(test_data))

ts_test <- ts(test_data$price, frequency = 1)

# Plot the forecasted values and observed values on the test data
plot(forecast_values, main = "ARIMA Model Forecast (Test Data)", xlab = "Time", ylab = "Values", xlim =
lines(ts_test, col = "blue")
```



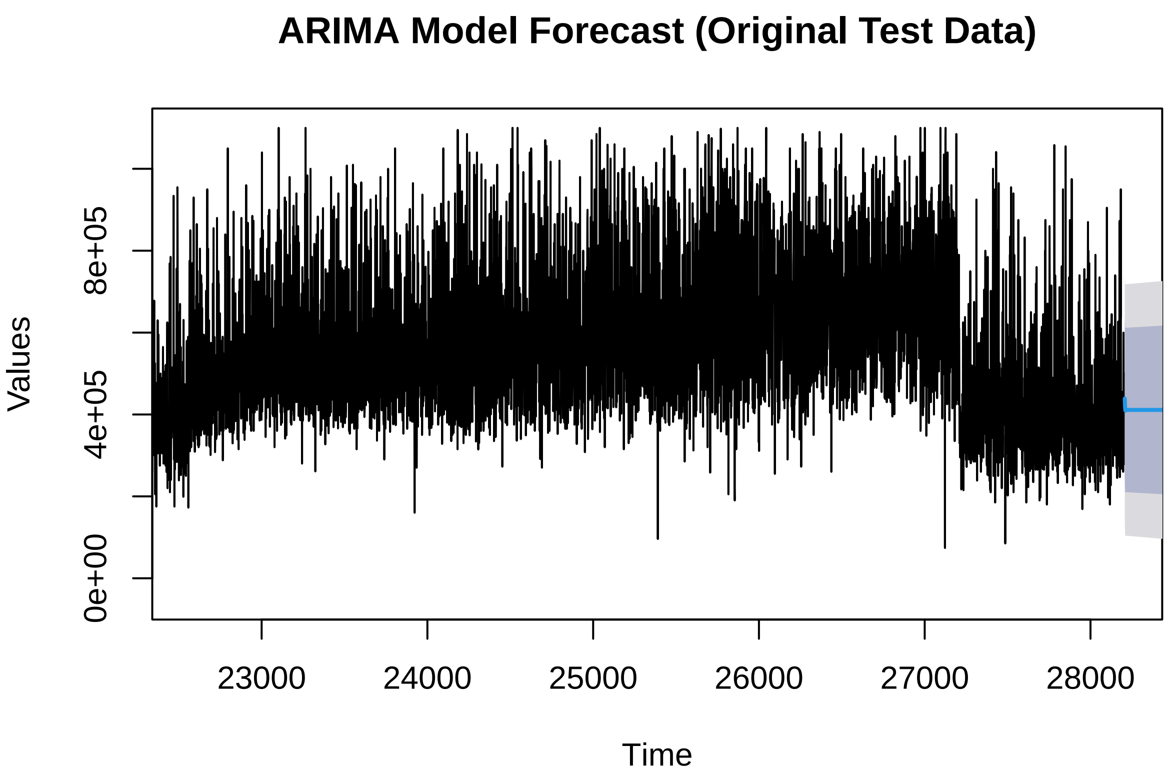

```
ts_combined <- ts(c(train_data$price, test_data$price), frequency = 1)

# Retrain the ARIMA model on the combined data
arima_model_combined <- auto.arima(ts_combined)

# Print the model summary
print(summary(arima_model_combined))
```

```
## Series: ts_combined
## ARIMA(2,1,5)
##
## Coefficients:
##          ar1      ar2      ma1     ma2      ma3      ma4     ma5
##       0.8445  -0.2370  -1.5656  0.6465  -0.0812  -0.0089  0.0153
## s.e.  0.5455   0.4691   0.5457  0.8622   0.2310   0.0438  0.0468
##
## sigma^2 = 2.262e+10:  log likelihood = -376250.5
## AIC=752517.1   AICc=752517.1   BIC=752583.1
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE    MASE
## Training set -248.5512 150367.4 115040.9 -7.500898 22.40456 0.91822
##                      ACF1
## Training set 0.0001438191
```

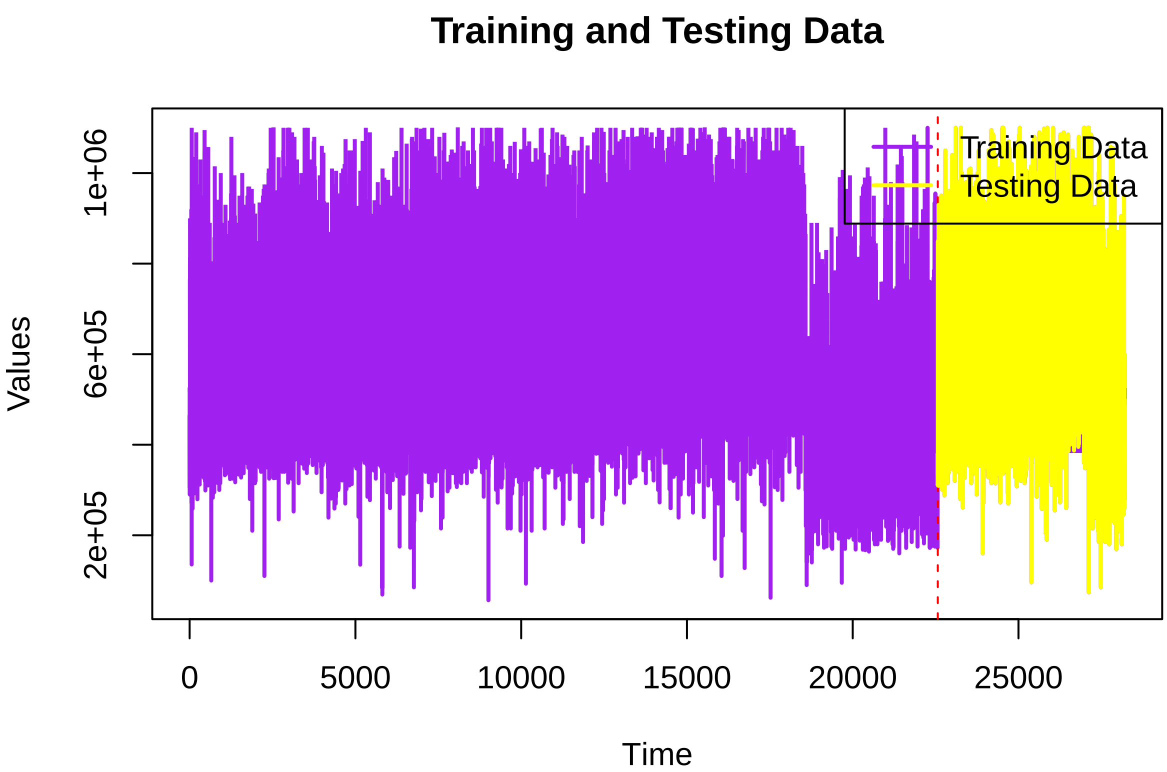
```
# Plot the fitted values and observed values on the combined data
plot(forecast(arima_model_combined), main = "ARIMA Model Forecast (Combined Data)", xlab = "Time", ylab
lines(ts_combined, col = "purple")
```




```
# Forecast on the original test data
forecast_values_combined <- forecast(arima_model_combined, h = nrow(test_data))

# Plot the forecasted values and observed values on the original test data
plot(forecast_values_combined, main = "ARIMA Model Forecast (Original Test Data)", xlab = "Time", ylab =
lines(ts_test, col = "purple")
```




```
plot(ts_combined, type = "l", col = "purple", lwd = 2, main = "Training and Testing Data", xlab = "Time

# Add a vertical line to indicate the separation between training and testing data
abline(v = length(ts_train) + 0.5, col = "red", lty = 2)

# Add points for the testing data on the same graph
lines(length(ts_train) + 1:length(ts_test), test_data$price, col = "yellow", lwd = 2)

# Add a legend
legend("topright", legend = c("Training Data", "Testing Data"), col = c("purple", "yellow"), lty = c(1,
```




```r
# Forecast on the original test data
forecast_combined <- forecast(arima_model_combined, h = nrow(test_data))

# Extract the forecasted values
forecast_values <- forecast_combined$mean

# Calculate Mean Absolute Percentage Error (MAPE)
mape <- mean(abs(test_data$price - forecast_values) / test_data$price) * 100

# Print the MAPE
cat("Mean Absolute Percentage Error (MAPE):", mape, "%\n")
```

```
## Mean Absolute Percentage Error (MAPE): 28.27479 %
```