R Notebook

```r
 ###                             #####                             ###
#       Linear Regression Analysis on Coffee Transactions Dataset      #
#                 HAMAD A. ALMAZROUEI - 201912368                      #
 ###                             #####                             ###

# Set a seed for reproducibility
set.seed(999)

# Load necessary libraries
library(ggplot2)  # For data visualization
library(dplyr)    # For data manipulation and transformation
```

```
##
## 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(prophet)  # For time series forecasting
```

```
## Loading required package: Rcpp

## Warning: package 'Rcpp' was built under R version 4.3.3

## Loading required package: rlang

## Warning: package 'rlang' was built under R version 4.3.3
```

```r
library(caTools)  # For data splitting and utility functions
```

```
## Warning: package 'caTools' was built under R version 4.3.3
```

```r
# The above libraries are now loaded and ready for use:
# - ggplot2: Enables creation of elegant data visualizations.
# - dplyr: Provides a grammar for data manipulation (filter, mutate, group_by,
# etc.).
# - prophet: A powerful tool for forecasting time series data.
# - caTools: Useful for splitting data into training and testing sets, among
# other utilities.


# ----- Step 1: Load and Prepare Data -----

# Read the dataset
coffee_data <- read.csv("/Users/hamad/Desktop/ZU/Fall 2024/Data Science - C/Assignment 3/Files/Coffee T

# ----- Step 1.1: Data Structure View -----
# View the structure of the data
str(coffee_data)
```

```
## 'data.frame':    149116 obs. of  17 variables:
##  $ transaction_id  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ transaction_date: chr  "01/01/2023" "01/01/2023" "01/01/2023" "01/01/2023" ...
##  $ transaction_time: chr  "7:06:11" "7:08:56" "7:14:04" "7:20:24" ...
##  $ transaction_qty : int  2 2 2 1 2 1 1 2 1 2 ...
##  $ store_id        : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ store_location  : chr  "Lower Manhattan" "Lower Manhattan" "Lower Manhattan" "Lower Manhattan" ..
##  $ product_id      : int  32 57 59 22 57 77 22 28 39 58 ...
##  $ unit_price      : num  3 3.1 4.5 2 3.1 3 2 2 4.25 3.5 ...
##  $ product_category: chr  "Coffee" "Tea" "Drinking Chocolate" "Coffee" ...
##  $ product_type    : chr  "Gourmet brewed coffee" "Brewed Chai tea" "Hot chocolate" "Drip coffee" ..
##  $ product_detail  : chr  "Ethiopia Rg" "Spicy Eye Opener Chai Lg" "Dark chocolate Lg" "Our Old Time
##  $ Revenue         : chr  "$6.00" "$6.20" "$9.00" "$2.00" ...
##  $ Month           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Month.1         : chr  "Jan" "Jan" "Jan" "Jan" ...
##  $ Weekday         : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ Weekday.1       : chr  "Sun" "Sun" "Sun" "Sun" ...
##  $ Hour            : int  7 7 7 7 7 7 7 7 7 7 ...
```

```r
# View the summary statistics of the data
summary(coffee_data)
```

```
##  transaction_id   transaction_date   transaction_time   transaction_qty
##  Min.   :     1   Length:149116      Length:149116      Min.   :1.000
##  1st Qu.: 37336   Class :character   Class :character   1st Qu.:1.000
##  Median : 74728   Mode  :character   Mode  :character   Median :1.000
##  Mean   : 74737                                         Mean   :1.438
##  3rd Qu.:112094                                         3rd Qu.:2.000
##  Max.   :149456                                         Max.   :8.000
##     store_id     store_location       product_id      unit_price
##  Min.   :3.000   Length:149116      Min.   : 1.00   Min.   : 0.800
##  1st Qu.:3.000   Class :character   1st Qu.:33.00   1st Qu.: 2.500
##  Median :5.000   Mode  :character   Median :47.00   Median : 3.000
##  Mean   :5.342                      Mean   :47.92   Mean   : 3.382
##  3rd Qu.:8.000                      3rd Qu.:60.00   3rd Qu.: 3.750
##  Max.   :8.000                      Max.   :87.00   Max.   :45.000
##  product_category   product_type       product_detail       Revenue
##  Length:149116      Length:149116      Length:149116      Length:149116
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
##
##
##
##      Month          Month.1              Weekday       Weekday.1
##  Min.   :1.000   Length:149116      Min.   :1.000   Length:149116
##  1st Qu.:3.000   Class :character   1st Qu.:2.000   Class :character
##  Median :4.000   Mode  :character   Median :4.000   Mode  :character
##  Mean   :3.989                      Mean   :3.982
##  3rd Qu.:5.000                      3rd Qu.:6.000
##  Max.   :6.000                      Max.   :7.000
##       Hour
##  Min.   : 6.00
##  1st Qu.: 9.00
##  Median :11.00
##  Mean   :11.74
##  3rd Qu.:15.00
##  Max.   :20.00
```

```
# Analysis of the Coffee Shop Transactions Dataset

# Data Overview
# - The dataset contains 149,116 observations with 17 variables, providing transaction-level details of
# - Key variables include transaction_id, store_location, product_category, transaction_qty, and Revenu

# Summary Statistics

# Transaction ID
# - Range: 1 to 149,456, representing unique identifiers for each transaction.

# Transaction Date and Time
# - transaction_date: Character format, indicating the date of each
# transaction.
# - transaction_time: Character format, indicating the time of each
# transaction.

# Transaction Quantity
# - Range: 1 to 8, with a mean of 1.438.
# - Most transactions involve 1 or 2 items (1st Qu.:1, Median:1, 3rd Qu.:2).

# Store Information
# - store_id: Values range from 3 to 8, indicating multiple store locations.
# - store_location: Descriptive location names, such as "Lower Manhattan."

# Product Details
# - product_id: Ranges from 1 to 87, representing unique product identifiers.
# - product_category: Contains categories like "Coffee," "Tea," etc.
# - product_type: Provides additional granularity, such as "Gourmet brewed
# coffee" or "Hot chocolate."
# - unit_price: Prices range from $0.80 to $45.00, with a mean price of $3.382.

# Revenue
# - The Revenue column is in character format (e.g., "$6.00"). It should be
# converted to numeric for analysis.
```

```r
# Time Information
# - Month and Month.1: Represent months numerically (1-6) and textually
# ("Jan," "Feb").
# - Weekday and Weekday.1: Represent weekdays numerically (1-7) and
# textually ("Sun," "Mon").
# - Hour: Represents the hour of the transaction, ranging from 6 AM to
# 8 PM (Min.:6, Max.:20).

# Observations
# - The majority of transactions involve small quantities and moderately
# priced items.
# - Store and product-level granularity enables analysis by location and
# product preferences.
# - Time variables allow for temporal analysis, such as peak transaction
# hours or seasonal trends.


# ----- Step 1.2: Data Preparation Process -----


# Check for missing values in the dataset
sum(is.na(coffee_data))
```

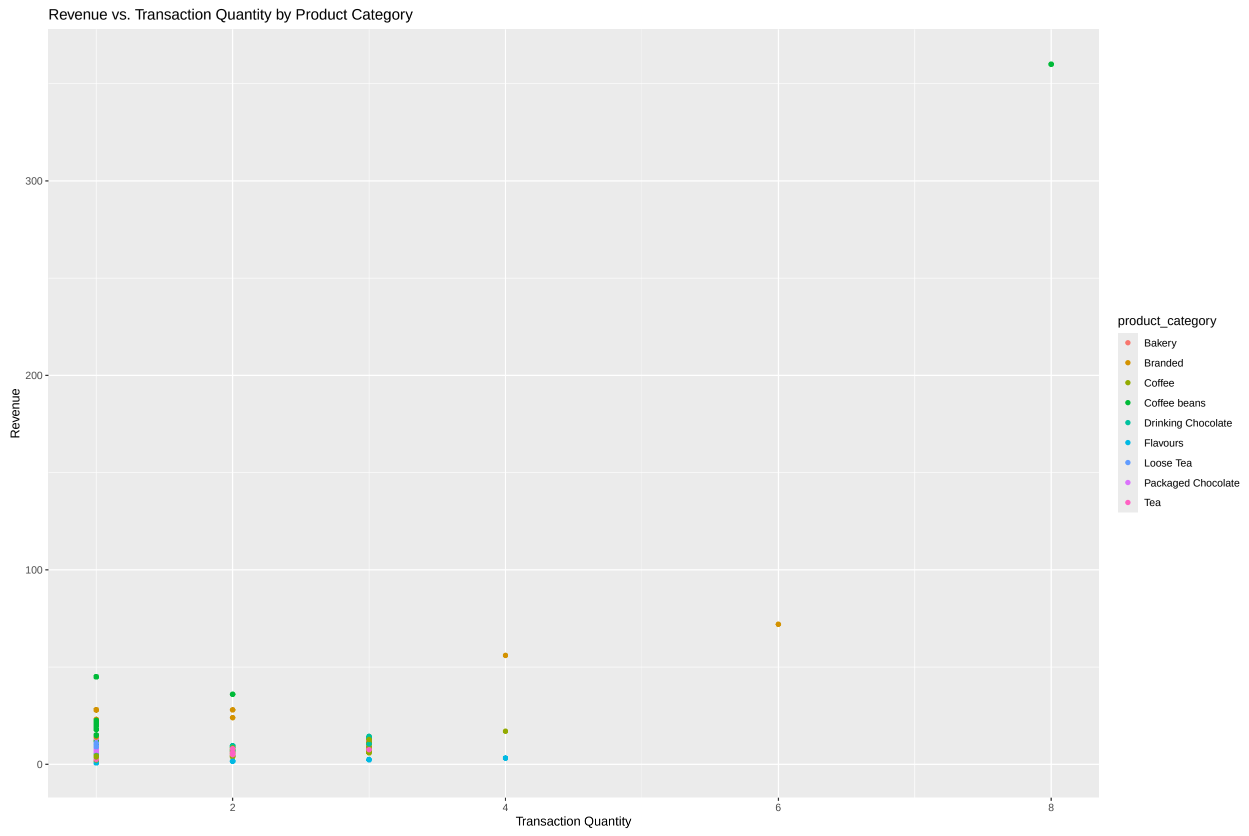
```
## [1] 0
```

```r
# Output: Total count of missing values in the entire dataset

colSums(is.na(coffee_data))
```

```
##   transaction_id transaction_date transaction_time  transaction_qty
##                0                0                0                0
##         store_id   store_location       product_id       unit_price
##                0                0                0                0
## product_category     product_type   product_detail          Revenue
##                0                0                0                0
##            Month          Month.1          Weekday        Weekday.1
##                0                0                0                0
##             Hour
##                0
```

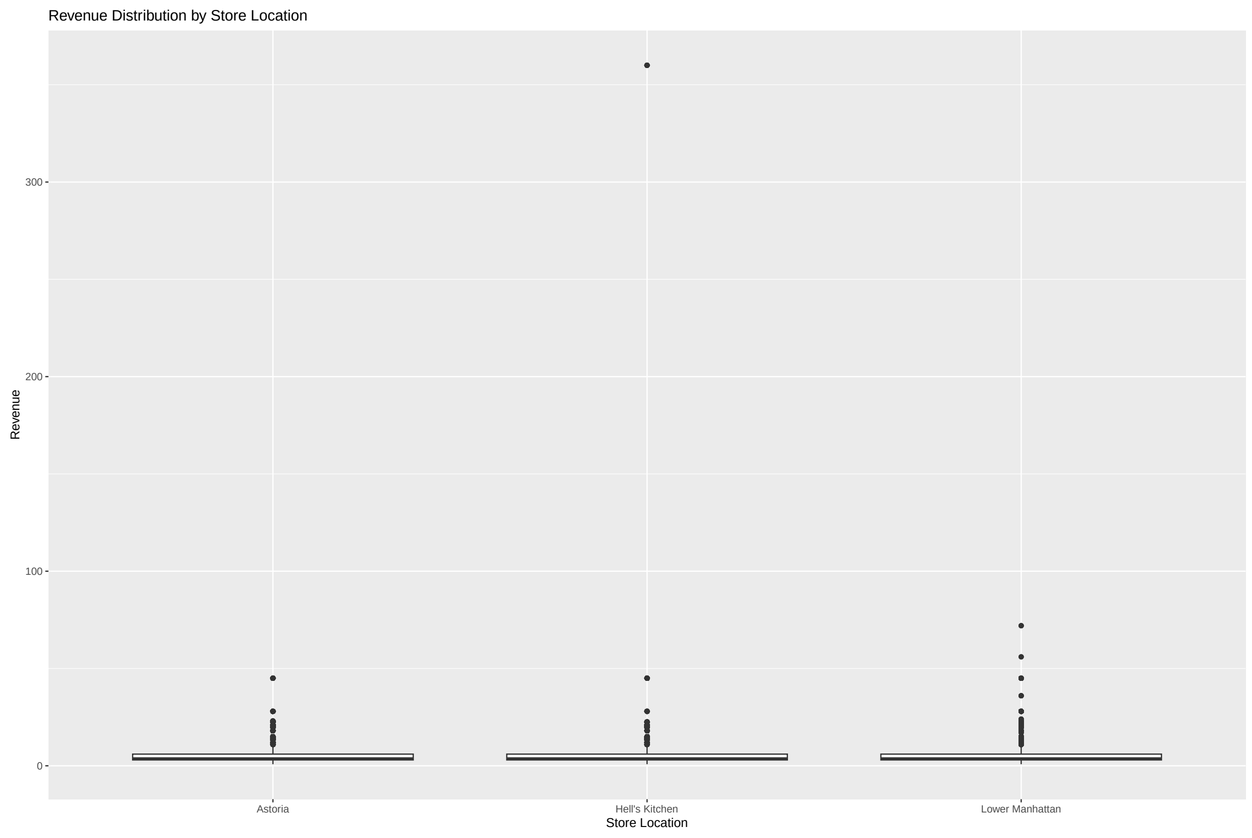
```r
# Output: Count of missing values for each column in the dataset

# Analysis of Missing Values in the Coffee Shop Dataset

# Key Observations
# - Total missing values: 0
# - No columns contain missing values, as confirmed by sum(is.na(coffee_data))
#  and colSums(is.na(coffee_data)).

# Implications
# - The dataset is complete and does not require imputation for missing data.
# - Analysis can proceed without concerns about data loss or introducing biases
# due to handling missing values.

# Formatting 'transaction_date' column from character to Date format
coffee_data$transaction_date <- as.Date(coffee_data$transaction_date,
                                        format = "%d/%m/%Y")
# Transformation of transaction_date to Date Format

# Code Description
# - The transaction_date column is converted from a character format to a
# Date object using as.Date().
# - The format "%d/%m/%Y" specifies the day, month, and year structure of
# the input data.

# Validation
# - After this transformation, the transaction_date column can be used for
# time-series analysis and date-based aggregations.


# Formatting 'Revenue' column to numeric after removing "$" symbol
coffee_data$Revenue <- as.numeric(gsub("\\$", "", coffee_data$Revenue))

# Transformation of Revenue Column to Numeric Format

# Code Description
# - The gsub("\\$", "", coffee_data$Revenue) function removes the dollar sign
# ($) from the Revenue column.
# - The as.numeric() function converts the cleaned values to numeric format.

# Validation
# - The transformation ensures that Revenue is in numeric format and ready
# for mathematical operations and aggregations.
# - Check the structure of the column using str(coffee_data) or summary
# statistics using summary(coffee_data$Revenue) to confirm the transformation.


# Converting 'product_category' and 'store_location' columns to factors
coffee_data$product_category <- factor(coffee_data$product_category)
coffee_data$store_location <- factor(coffee_data$store_location)

# Conversion of Categorical Variables to Factor

# Code Description
# - The product_category and store_location columns are converted to
# factor type.
# - This transformation helps optimize storage and facilitates categorical
# analysis.

# Benefits
# - Improves memory efficiency for large datasets.
# - Enables advanced categorical analyses, such as group-wise aggregations
# and visualizations.

# Validation
# - Verify the changes using str(coffee_data) to confirm the columns are
# now factors.
# - Check the levels of each factor using levels(coffee_data$product_category)
#  and levels(coffee_data$store_location).


# ----- Step 2: Exploratory Data Analysis -----
```

```
# Visualizing the relationship between revenue and transaction quantity
ggplot(coffee_data, aes(x = transaction_qty, y = Revenue)) +
  geom_point(aes(color = product_category)) +
  ggtitle("Revenue vs. Transaction Quantity by Product Category") +
  xlab("Transaction Quantity") +
  ylab("Revenue")
```



```
# Analysis of the Scatter Plot: Revenue vs. Transaction Quantity by Product Category

# Key Observations
# - The scatter plot shows the relationship between transaction_qty (x-axis)
# and Revenue (y-axis), colored by product_category.
# - The majority of transactions involve smaller quantities (1 or 2 items),
# but higher quantities (e.g., 6-8) are observed in specific categories.
# - Some categories, such as "Branded" and "Coffee beans," exhibit higher
# revenue per transaction, especially for larger quantities.

# Insights
# - High Revenue Categories: Categories like "Branded" and "Coffee beans"
# generate higher revenues, possibly due to higher unit prices or larger
# quantities purchased in single transactions.
# - Transaction Quantity Distribution: Most transactions involve low
# quantities, indicating customers typically purchase a small number of items
# at a time.
# - Category Spread: Categories like "Coffee," "Tea," and "Drinking
# Chocolate" have lower revenue per transaction but may have higher sales
# volume overall.
```

```
# Visualizing the relationship between revenue and store location
ggplot(coffee_data, aes(x = store_location, y = Revenue)) +
  geom_boxplot() +
  ggtitle("Revenue Distribution by Store Location") +
  xlab("Store Location") +
  ylab("Revenue")
```



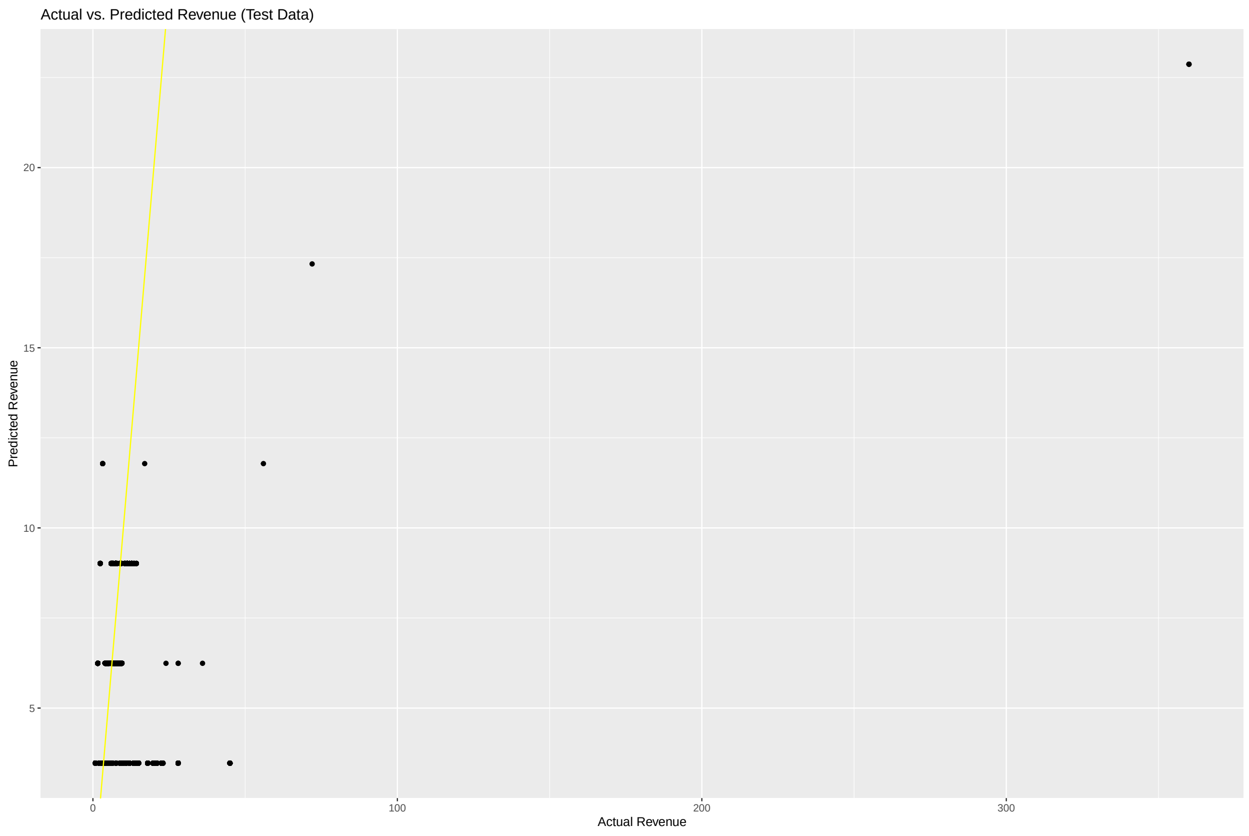
```
# Explanation:
# - geom_boxplot() creates a boxplot to display revenue distributions
# across locations.
# - Boxplots provide insights into the spread, central tendency, and
# outliers in revenue.
# - Useful for comparing store performance and identifying anomalies.

# Analysis of the Boxplot: Revenue Distribution by Store Location
# Key Observations
# - The boxplot illustrates the revenue distribution across three store
# locations: Astoria, Hell's Kitchen, and Lower Manhattan.
# - The median revenue for transactions is relatively consistent across all
# locations.
# - Outliers are visible in all three locations, particularly in Hell's
# Kitchen and Lower Manhattan, indicating occasional high-revenue transactions.

# Insights
# - Consistency Across Locations: The core revenue distribution
# (interquartile range) appears similar among the three locations, suggesting
# comparable customer spending behavior.
# - High-Revenue Transactions: Some outliers in Hell's Kitchen and Lower
# Manhattan suggest that these locations may occasionally serve high-value
# transactions.
# - Potential Store Performance: The lack of significant variation in
# median revenue suggests that store performance is more influenced by
# transaction volume than transaction value.
```

```
# ----- Step 3: Simple Linear Regression -----
# Predict Revenue based on Transaction Quantity
# Split data using caTools

# Splitting the dataset into training and testing sets
split <- sample.split(coffee_data$Revenue, SplitRatio = 0.7)

# Creating the training dataset (70% of the data)
train_data <- subset(coffee_data, split == TRUE)

# Creating the testing dataset (30% of the data)
test_data <- subset(coffee_data, split == FALSE)

# Explanation:
# - sample.split() is used to create a logical vector for splitting the
# dataset.
# - SplitRatio = 0.7 means 70% of the data goes into the training set.
# - subset() selects rows based on the logical vector generated by
# sample.split().
# - This ensures a random and balanced split for modeling and evaluation.
# Dataset Splitting Analysis: Training and Testing Sets

# Key Observations
# - The dataset was split into training (70%) and testing (30%) subsets using
# the sample.split function.
# - The splitting ensures that the dependent variable Revenue maintains a
# representative distribution in both subsets.

# Insights
# - Training Dataset: The training dataset contains 70% of the data,
# which is sufficient for model training while preserving enough diversity to
# capture the underlying patterns.
# - Testing Dataset: The testing dataset comprises 30% of the data, which
# is adequate for evaluating model performance and generalization.
```

```r
# Train a simple linear regression model
lm_simple <- lm(Revenue ~ transaction_qty, data = train_data)

# Summarize the model
summary(lm_simple)
```

```
##
## Call:
## lm(formula = Revenue ~ transaction_qty, data = train_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -8.58  -0.97  -0.24   0.28 337.13
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.70021    0.03461   20.23   <2e-16 ***
## transaction_qty  2.77111    0.02251  123.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.949 on 104378 degrees of freedom
## Multiple R-squared:  0.1268, Adjusted R-squared:  0.1268
## F-statistic: 1.515e+04 on 1 and 104378 DF,  p-value: < 2.2e-16
```

```r
# Explanation:
# - lm() fits a linear regression model predicting Revenue based on
# transaction_qty.
# - summary() provides a detailed summary of the model, including:
#   - Coefficients: Intercept and slope of the linear model.
#   - R-squared: Proportion of variance in Revenue explained by
# transaction_qty.
#   - p-value: Statistical significance of the relationship.
# - Analyze the output to evaluate the model's performance and significance
# of predictors.

# Linear Regression Model Summary Analysis: Revenue ~ Transaction Quantity

# Key Observations:
# - The linear regression model predicts Revenue based on transaction_qty.
# - The model's intercept is 0.70021, and the coefficient for transaction_qty
# is 2.77111.
# - Both the intercept and the transaction quantity coefficient are highly
# significant (p-value < 2e-16).

# Residual Analysis:
# - The residuals have a minimum of -8.58 and a maximum of 337.13, with the
# majority of residuals clustered close to zero.
# - This indicates that while the model captures the general trend, there are
# outliers with higher deviations.

# Model Fit:
# - Multiple R-squared: 0.1268
# - Indicates that approximately 12.68% of the variance in Revenue is
# explained by transaction_qty.
# - This suggests that the model captures only a small portion of the
# variability, indicating a weak fit.
# - Adjusted R-squared: 0.1268
# - Same as R-squared in this case, as the model only includes one predictor.

# Statistical Significance:
# - F-statistic: 1.515e+04 with a p-value < 2.2e-16
# - Strong evidence that the model as a whole is statistically significant.

# Practical Implications:
# - The positive coefficient for transaction_qty (2.77111) suggests that for
# every unit increase in transaction_qty, Revenue increases by
# approximately 2.77 units.
# - However, the low R-squared indicates that additional variables might be
# needed to better explain the variability in Revenue.

# Predict on the test set
# Predict revenue on the testing set using the trained model
test_data$predicted_revenue1 <- predict(lm_simple, newdata = test_data)

# Explanation:
# - predict() generates predictions for Revenue based on transaction_qty
# in the test_data.
# - The predictions are stored in a new column predicted_revenue1 within the
# test_data dataframe.
# - This column will be used to evaluate the model's performance by comparing
# predicted and actual values.

# Generate predictions for the test dataset using the trained linear model
# - The predict function is applied with lm_simple (the linear regression model)
#   and test_data as the new dataset for which predictions are required.
# - A new column named predicted_revenue1 is added to the test_data dataframe
#   to store the predicted revenue values for each observation.

# Evaluate model performance using test set
ggplot(test_data, aes(x = Revenue, y = predicted_revenue1)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, col = "yellow") +
  ggtitle("Actual vs. Predicted Revenue (Test Data)") +
  xlab("Actual Revenue") +
  ylab("Predicted Revenue")
```



```r
# Visualize the Actual vs. Predicted Revenue for the test dataset
# - The scatterplot compares the actual revenue values (x-axis) with the
# predicted revenue values (y-axis).
# - A diagonal line (geom_abline) is added to represent the ideal scenario
# where predicted values perfectly match actual values.
# - The line has a slope of 1 and intercept of 0 (identity line) to highlight
# any deviations.
# - This plot is useful for evaluating the performance of the model visually,
# with larger deviations indicating areas of poor prediction accuracy.
```

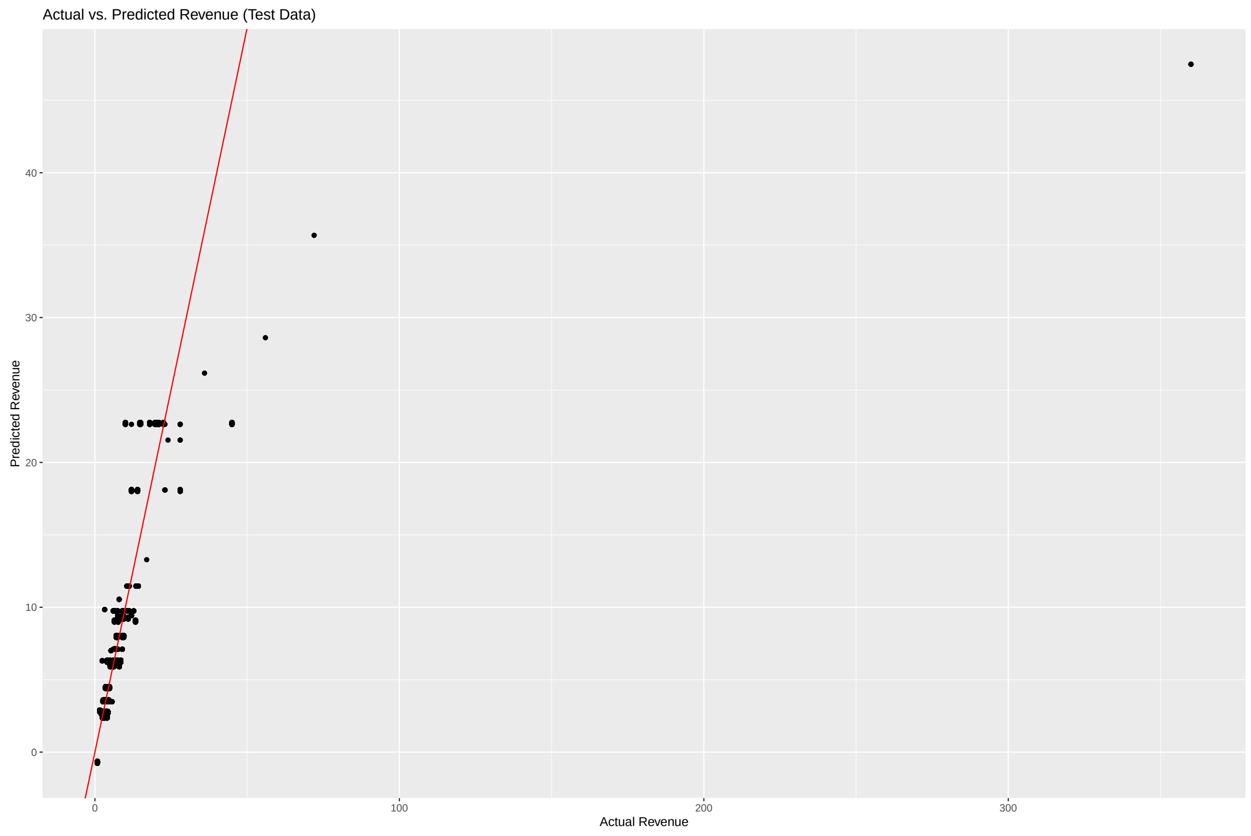
```r
# ----- Step 4: Multiple Linear Regression -----



# Train the multiple linear regression model on the training set
lm_multiple <- lm(Revenue ~ transaction_qty + product_category +
                    store_location, data = train_data)

# Summarize the model
summary(lm_multiple)
```

```
##
## Call:
## lm(formula = Revenue ~ transaction_qty + product_category + store_location,
##     data = train_data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.752  -0.582   0.018   0.550 312.510
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         0.03449    0.03205   1.076    0.282
## transaction_qty                     3.53399    0.01812 195.036  < 2e-16 ***
## product_categoryBranded            14.52939    0.13124 110.707  < 2e-16 ***
## product_categoryCoffee             -0.79607    0.02905 -27.404  < 2e-16 ***
## product_categoryCoffee beans       19.15172    0.08723 219.563  < 2e-16 ***
## product_categoryDrinking Chocolate  0.91525    0.04144  22.085  < 2e-16 ***
## product_categoryFlavours           -4.23930    0.04965 -85.377  < 2e-16 ***
## product_categoryLoose Tea           5.71429    0.10410  54.895  < 2e-16 ***
## product_categoryPackaged Chocolate  5.50433    0.16106  34.176  < 2e-16 ***
## product_categoryTea                -1.11805    0.03013 -37.103  < 2e-16 ***
## store_locationHell's Kitchen        0.03152    0.02217   1.422    0.155
## store_locationLower Manhattan      -0.09043    0.02261  -3.999 6.35e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.947 on 104368 degrees of freedom
## Multiple R-squared:  0.5138, Adjusted R-squared:  0.5138
## F-statistic: 1.003e+04 on 11 and 104368 DF,  p-value: < 2.2e-16
```

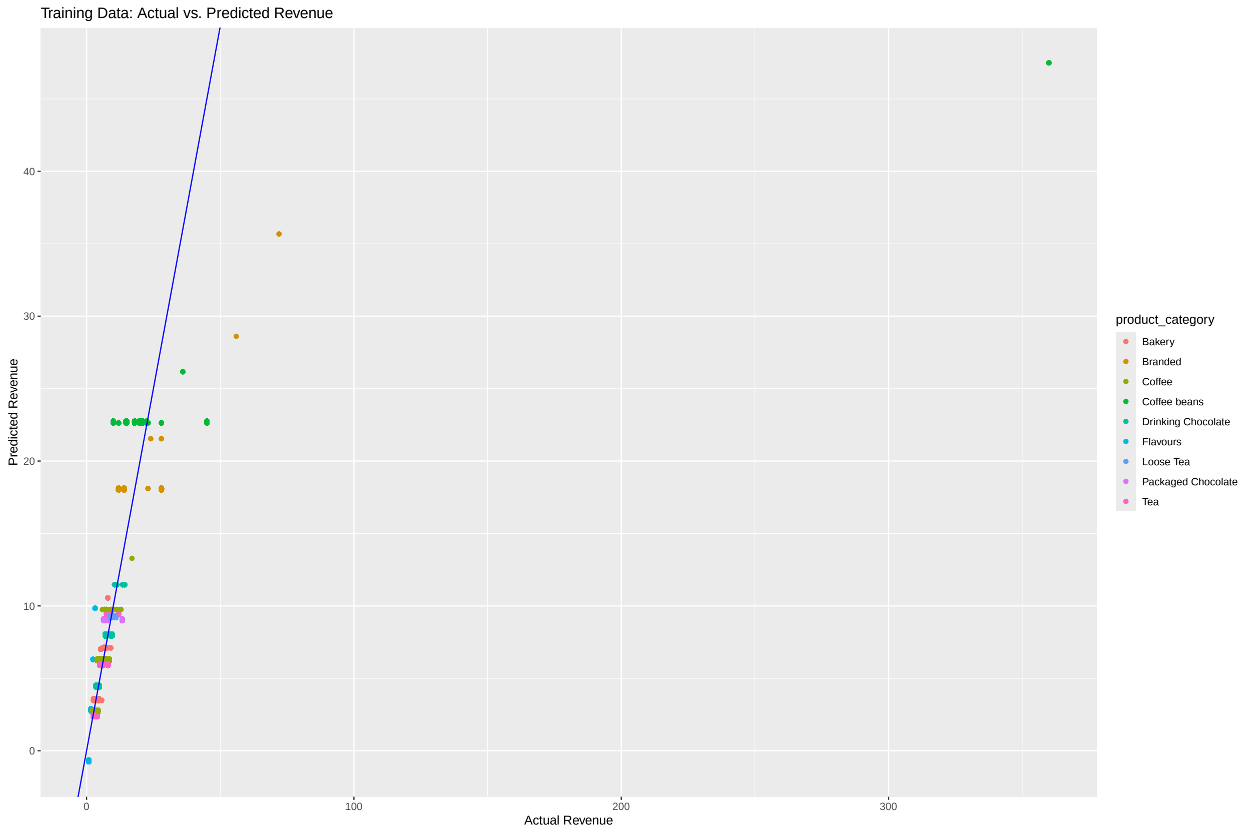
```
# Summary of the multiple linear regression model:
# - The model predicts Revenue based on transaction_qty,
# product_category, and store_location.

# Key observations:
# 1. transaction_qty has a strong positive effect on Revenue (Estimate
# = 3.53399, p-value < 2e-16).
#    Each additional item purchased increases revenue by approximately $3.53,
# holding other variables constant.
# 2. product_category coefficients indicate varying impacts:
#    - Coffee beans category has the highest positive impact on revenue
# (Estimate = 19.15172).
#    - Flavours negatively impacts revenue significantly
# (Estimate = -4.23930).
#    - Branded products increase revenue considerably (Estimate = 14.52939).
#    - Categories such as Drinking Chocolate, Loose Tea, and Packaged
# Chocolate also contribute positively.
# 3. store_location:
#    - Lower Manhattan has a small negative impact on revenue compared to the
# baseline location (Estimate = -0.09043).
#    - Hell's Kitchen does not show a statistically significant effect
# (p-value = 0.155).

# Model evaluation:
# - Multiple R-squared: 0.5138. The model explains approximately 51.38% of the
# variance in revenue, which is moderate.
# - Residual standard error: 2.947, indicating the average deviation of
# predicted revenue from actual revenue.
# - F-statistic: Very high (1.003e+04), with a p-value < 2.2e-16, indicating
```

```r
# the overall model is highly significant.

# Interpretation:
# - The model effectively captures the relationship between transaction
# quantity, product categories, and revenue.
# - Product categories have a strong influence on revenue, with significant
# variability among categories.
# - Store location has a minimal impact, suggesting revenue is more dependent
# on product mix and quantity.

# Predict on the test set
test_data$predicted_revenue2 <- predict(lm_multiple, newdata = test_data)
# Comment on the prediction step:
# - The predict() function is used to generate predicted revenue values
# (predicted_revenue2)
#   for the test dataset using the trained multiple linear regression model
# (lm_multiple).
# - This step incorporates the effects of all independent variables:
# transaction_qty,
#   product_category, and store_location, as modeled in the training phase.

# Expected outcome:
# - The predicted values will reflect the combined contributions of the
# predictors
#   to estimate the Revenue for each transaction in the test dataset.
# - These predictions will be evaluated against actual revenue values for
# model performance assessment.

# Evaluate model performance using predictions vs. actuals on the test set
ggplot(test_data, aes(x = Revenue, y = predicted_revenue2)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, col = "red") +
  ggtitle("Actual vs. Predicted Revenue (Test Data)") +
  xlab("Actual Revenue") +
  ylab("Predicted Revenue")
```



```r
# Comment on the visualization:
# - This scatter plot visualizes the comparison between actual revenue (x-axis)
#   and predicted revenue (y-axis) from the multiple linear regression model.
# - The red diagonal line represents the ideal scenario where predicted revenue
#   perfectly matches the actual revenue, i.e., all points should lie on
# this line.

# Observations:
# - The majority of predictions closely align with the actual revenue values,
#   especially for lower revenue ranges.
# - Some outliers or deviations can be observed, particularly for transactions
#   with higher actual revenue, suggesting potential limitations in model
# accuracy
#   for extreme values or unmodeled factors influencing revenue.

# Purpose:
# - This plot helps assess the quality of the predictions and highlights areas
#   for potential model improvement, such as handling extreme values or
# refining predictors.
```

```r
# Visualize the regression fit on the training data
ggplot(train_data, aes(x = Revenue, y = predict(lm_multiple,
  newdata = train_data))) +
  geom_point(aes(color = product_category)) +
  geom_abline(slope = 1, intercept = 0, col = "blue") +
  ggtitle("Training Data: Actual vs. Predicted Revenue") +
  xlab("Actual Revenue") +
  ylab("Predicted Revenue")
```



```r
# Comment on the visualization:
# - This scatter plot represents the relationship between actual revenue
#   (x-axis) and predicted revenue (y-axis) on the training dataset.
# - Different product categories are distinguished by colors, adding clarity
#   on how predictions vary across categories.
# - The blue diagonal line represents the ideal match, where predicted values
#   equal the actual revenue.

# Observations:
# - Many data points align well with the diagonal, especially for smaller
# revenue values.
# - A few outliers and deviations are noticeable, particularly for higher
# revenue values.
# - Certain product categories, such as "Coffee beans" (green), show higher
# predicted values
#   compared to other categories, indicating their significant contribution
# to revenue.

# Purpose:
# - This plot helps evaluate the model's performance on the training data,
#   indicating a good fit with some areas for improvement in handling extreme
# values.
# ----- Step 5: Model Evaluation -----


# ----- Step 5.1: Calculate MAE, MSE, and RMSE for Both Models -----


# Calculate MAE, MSE, and RMSE for Simple Regression (Test Set)
mae_simple <- mean(abs(test_data$Revenue - test_data$predicted_revenue1))
mse_simple <- mean((test_data$Revenue - test_data$predicted_revenue1)^2)
rmse_simple <- sqrt(mse_simple)

cat("Simple Regression (Test Set) - MAE:", mae_simple,
    "MSE:", mse_simple,
    "RMSE:", rmse_simple, "\n")
```

```
## Simple Regression (Test Set) - MAE: 1.29322 MSE: 15.6108 RMSE: 3.951051
```

```r
# Analysis of Simple Regression Error Metrics:

# - MAE (Mean Absolute Error): 1.29322
#   Indicates the average magnitude of errors in the model's predictions.
#   On average, the predicted revenue deviates by approximately $1.29 from
# the actual revenue.

# - MSE (Mean Squared Error): 15.6108
#   Reflects the average squared difference between predicted and actual
# revenues.
#   Larger errors are penalized more heavily due to squaring, emphasizing
# significant deviations.

# - RMSE (Root Mean Squared Error): 3.951051
#   Provides the error in the same units as the target variable (Revenue).
#   A higher RMSE suggests that while the model captures general trends, it
# struggles with larger variances.

# Observations:
# - The error metrics suggest reasonable accuracy for simpler relationships
# like transaction_qty.
# - However, given the dataset's complexity (e.g., multiple categories and
# locations), a simple linear regression may not fully capture all patterns,
# evident from relatively high RMSE.

# Conclusion:
# - The simple regression model provides a baseline but could benefit from
# additional predictors for improved accuracy.

# Calculate MAE, MSE, and RMSE for Multiple Regression (Test Set)
mae_multiple <- mean(abs(test_data$Revenue - test_data$predicted_revenue2))
mse_multiple <- mean((test_data$Revenue - test_data$predicted_revenue2)^2)
rmse_multiple <- sqrt(mse_multiple)

cat("Multiple Regression (Test Set) - MAE:", mae_multiple,
    "MSE:", mse_multiple,
    "RMSE:", rmse_multiple, "\n")
```

```
## Multiple Regression (Test Set) - MAE: 0.8288278 MSE: 8.685382 RMSE: 2.947097
```

```
# Analysis of Multiple Regression Error Metrics:

# - MAE (Mean Absolute Error): 0.8288278
#   This indicates that, on average, the predicted revenue deviates by
# approximately $0.83 from the actual revenue.
#   This is a significant improvement compared to the simple regression model
# (MAE = 1.29322).

# - MSE (Mean Squared Error): 8.685382
#   The average squared difference between predicted and actual revenues is
# reduced.
#   This implies the multiple regression model captures the variability in
# the data better than the simple regression model (MSE = 15.6108).

# - RMSE (Root Mean Squared Error): 2.947097
#   The RMSE is lower, meaning the multiple regression model provides better
# predictions with smaller deviations compared to the simple regression model
# (RMSE = 3.951051).

# Observations:
# - The inclusion of additional predictors, such as product_category and
# store_location, significantly improves model performance.
# - Lower MAE, MSE, and RMSE values suggest that the multiple regression
# model better accounts for the complexity in the data compared to the simple
# regression model.

# Conclusion:
# - The multiple regression model outperforms the simple regression model in
# predicting revenue, demonstrating the importance of incorporating additional
# explanatory variables.

# ----- Step 5.2: Bin Actual and Predicted Revenue -----

# Define bins for Revenue
bin_labels <- c("Low", "Medium", "High")

# - bin_labels is a vector that defines labels for categorical bins,
# such as "Low", "Medium", and "High".
# - These labels will likely be used to categorize numerical data (e.g.,
# revenue or quantity) into meaningful groups.
# - Typically, these labels correspond to ranges of a numeric variable
# divided into discrete bins.

# Add small noise to avoid duplicate bin edges (if needed)
test_data$Revenue <- test_data$Revenue + runif(nrow(test_data),
                                               -1e-6, 1e-6)
test_data$predicted_revenue1 <- test_data$predicted_revenue1 +
  runif(nrow(test_data), -1e-6, 1e-6)
test_data$predicted_revenue2 <- test_data$predicted_revenue2 +
```

```r
  runif(nrow(test_data), -1e-6, 1e-6)


# - Small random noise is being added to the Revenue, predicted_revenue1,
# and predicted_revenue2 columns.
# - This step is used to avoid potential issues with duplicate bin edges,
# especially when categorizing numeric data into bins.
# - The runif() function generates uniform random noise between -1e-6
# and 1e-6.

# Purpose:
# - Ensures that all values are unique, which can help avoid errors during
# binning or categorization processes.
# - This is particularly useful when using functions like cut() or creating
# histograms where duplicate edges can cause errors or misleading results.

# Bin Actual Revenue in the Test Set
test_data$actual_bin <- cut(
  test_data$Revenue,
  breaks = unique(quantile(test_data$Revenue, probs = c(0, 0.33, 0.66, 1))),
  labels = bin_labels,
  include.lowest = TRUE
)

# Bin Predicted Revenue for Simple Regression
test_data$predicted_bin1_label <- cut(
  test_data$predicted_revenue1,
  breaks = unique(quantile(test_data$predicted_revenue1,
                           probs = c(0, 0.33, 0.66, 1))),
  labels = bin_labels,
  include.lowest = TRUE
)

# Bin Predicted Revenue for Multiple Regression
test_data$predicted_bin2_label <- cut(
  test_data$predicted_revenue2,
  breaks = unique(quantile(test_data$predicted_revenue2,
                           probs = c(0, 0.33, 0.66, 1))),
  labels = bin_labels,
  include.lowest = TRUE
)

# - The code categorizes the Revenue, predicted_revenue1, and
# predicted_revenue2 columns into bins (Low, Medium, High).
# - The cut() function is used to divide the continuous values into
# discrete bins based on quantiles.
# - The quantile ranges are determined using probs = c(0, 0.33, 0.66, 1)
# to split the data into three bins.

# For Revenue:
# - test_data$actual_bin: Bins the actual revenue into three categories
# (Low, Medium, High) based on quantile values.
# For Simple Regression Predictions:
# - test_data$predicted_bin1_label: Bins predicted_revenue1 using the same
# quantile-based approach.

# For Multiple Regression Predictions:
# - test_data$predicted_bin2_label: Bins predicted_revenue2 similarly.

# Purpose:
# - This binning facilitates a comparative analysis of how well the predicted
# bins match the actual revenue bins.
# - Helps evaluate the model's performance not just in numeric terms but in
# terms of categorical accuracy.

# Inspect binned data
head(test_data[, c("Revenue", "actual_bin", "predicted_revenue1",
                   "predicted_bin1_label",
                   "predicted_revenue2", "predicted_bin2_label")])
```

```
##     Revenue actual_bin predicted_revenue1 predicted_bin1_label
## 6  3.000000        Low           3.471328               Medium
## 8  4.000000     Medium           6.242440                 High
## 10 7.000000       High           6.242441                 High
## 12 7.000000       High           6.242440               Medium
## 13 3.000000        Low           3.471328               Medium
## 14 3.100001        Low           3.471327                  Low
##    predicted_revenue2 predicted_bin2_label
## 6            3.478051               Medium
## 8            6.215968                 High
## 10           7.927295                 High
## 12           6.215968                 High
## 13           2.359999                  Low
## 14           2.359998                  Low
```

```r
# Binned data inspection:

# Columns Explanation:
# - Revenue: The actual revenue value from the test set.
# - actual_bin: The bin (Low, Medium, High) assigned to the actual revenue
# based on quantiles.
# - predicted_revenue1: Predicted revenue from the simple regression model.
# - predicted_bin1_label: The bin (Low, Medium, High) assigned to the simple
# regression predictions.
# - predicted_revenue2: Predicted revenue from the multiple regression model.
# - predicted_bin2_label: The bin (Low, Medium, High) assigned to the multiple
# regression predictions.

# Observations:
# - Example row 6:
#   - Actual revenue is 3.000000 and falls into the Low bin (actual_bin).
#   - Simple regression predicted revenue is 3.471328, binned as Medium.
#   - Multiple regression predicted revenue is 3.478051, also binned as
# Medium.
# - Example row 8:
```

```r
#   - Actual revenue is 4.000000, binned as Medium.
#   - Both simple and multiple regression models predicted values that fall
# into the High bin.

# Insights:
# - In some cases (e.g., row 6), the predicted bins don't align with the
# actual bin, indicating misclassification.
# - Multiple regression predictions (predicted_bin2_label) generally seem
# closer to actual bins compared to simple regression.

# ----- Step 5.3: Create a Confusion Matrix -----

# Create confusion matrix for Simple Regression (Test Set)
conf_matrix1 <- table(test_data$actual_bin, test_data$predicted_bin1_label)

# Create confusion matrix for Multiple Regression (Test Set)
conf_matrix2 <- table(test_data$actual_bin, test_data$predicted_bin2_label)

# Print evaluation scores
cat("Simple Regression Confusion Matrix:\n")
```

```
## Simple Regression Confusion Matrix:
```

```r
print(conf_matrix1)
```

```
##
##            Low Medium  High
##   Low     7742   6129   892
##   Medium  6332   5536  2895
##   High     689   3098 11423
```

```r
cat("Multiple Regression Confusion Matrix:\n")
```

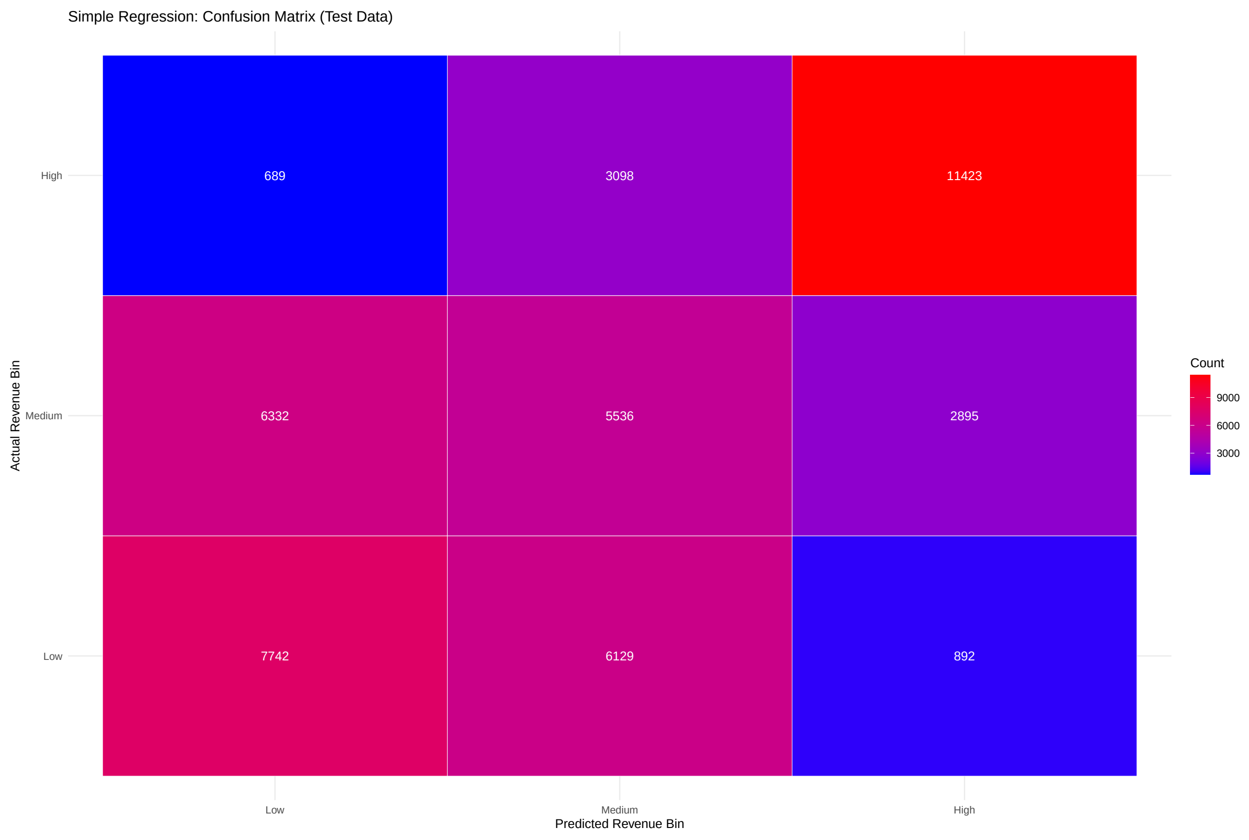
```
## Multiple Regression Confusion Matrix:
```

```r
print(conf_matrix2)
```

```
##
##            Low Medium  High
##   Low    12137   2588    38
##   Medium  2626   9220  2917
##   High       0   2955 12255
```

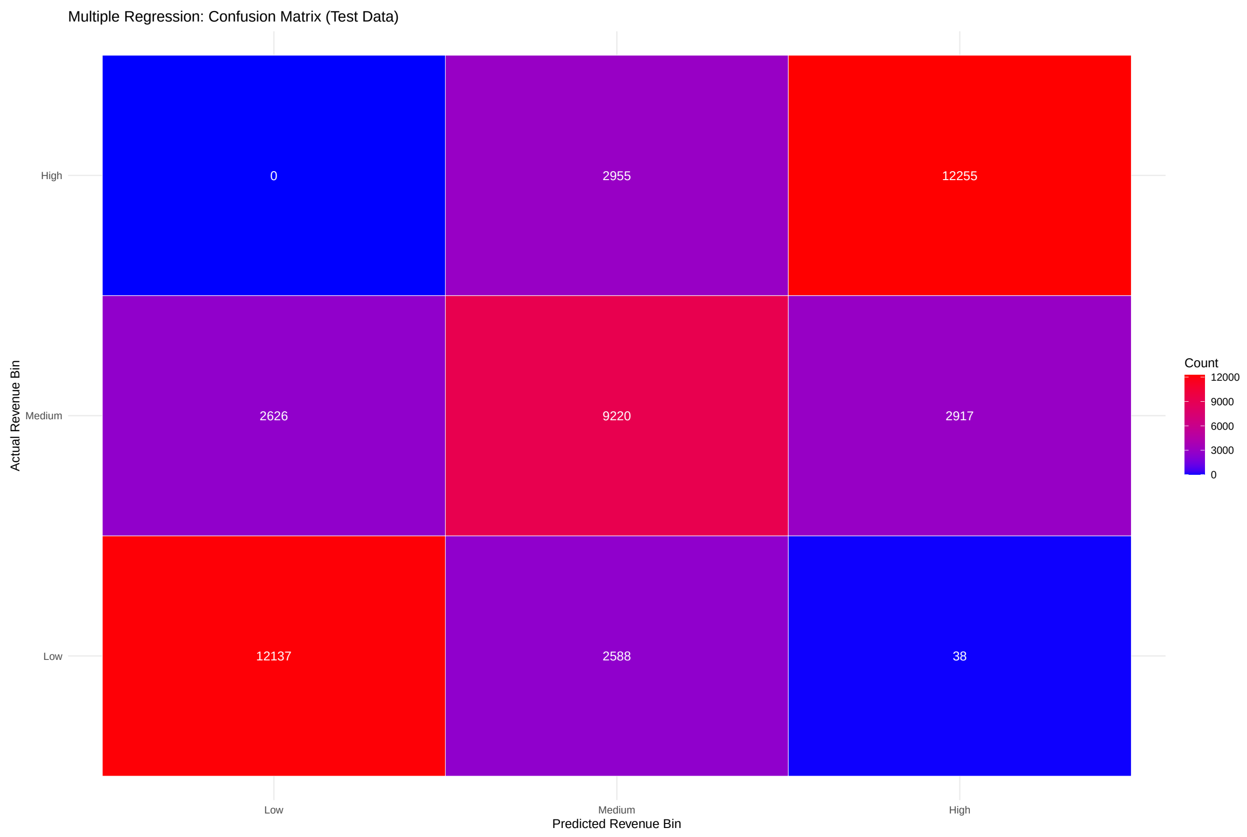
```r
# Confusion Matrices:

# Simple Regression Confusion Matrix:
# - Observations:
#   - The model tends to predict "Medium" more often (e.g., 6129 instances of
# "Low" classified as "Medium").
#   - There are significant misclassifications for the "Medium" bin, with
# values spread across other bins.
#   - The "High" bin has the highest accuracy compared to others (11423
# correct classifications).

# Multiple Regression Confusion Matrix:
# - Observations:
#   - The model predicts "Low" for 12137 actual "Low" values, showing strong
# performance for this bin.
#   - However, some actual "High" values are predicted as "Medium" (2955
# instances).
#   - The "High" bin predictions are more accurate, with 12255 correctly
# classified cases.

# Overall Comparison:
# - Multiple regression shows better separation between bins, especially for
# the "Low" and "High" categories.
# - Simple regression misclassifies more instances, particularly around the
# "Medium" category.
# - The confusion matrix for multiple regression indicates overall better
# performance in revenue bin classification.

# ----- Step 5.4: Visualize the Confusion Matrix -----

# Convert the confusion matrices to data frames
conf_matrix_df1 <- as.data.frame(as.table(conf_matrix1))
conf_matrix_df2 <- as.data.frame(as.table(conf_matrix2))

# Rename columns for readability
colnames(conf_matrix_df1) <- c("Actual", "Predicted", "Count")
colnames(conf_matrix_df2) <- c("Actual", "Predicted", "Count")

# Code Execution:

# Step 1: Convert confusion matrices to data frames
# - The as.data.frame(as.table(conf_matrix1)) function converts the confusion
# matrix into a data frame format.
# - This is useful for further analysis or visualization.

# Step 2: Rename columns for better readability
# - The colnames function renames the columns to "Actual", "Predicted",
# and "Count".
# - This ensures clarity when working with the data frames.
```

```r
# Plot the heatmap for Simple Regression
ggplot(conf_matrix_df1, aes(x = Predicted, y = Actual, fill = Count)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Count), color = "white") +
  scale_fill_gradient(low = "blue", high = "red") +
  ggtitle("Simple Regression: Confusion Matrix (Test Data)") +
  xlab("Predicted Revenue Bin") +
  ylab("Actual Revenue Bin") +
  theme_minimal()
```



```
# The confusion matrix heatmap visualization for the Simple Regression model
# shows:
# - The majority of correct predictions are in the "High" actual revenue bin,
# with 11,423 correctly predicted as "High."
# - There is noticeable confusion in predicting "Low" and "Medium" bins, as
# many "Medium" actuals were classified as "Low" (6,332).
# - While the model captures the "High" revenue bin effectively, the scatter
# in the "Low" and "Medium" bins highlights a limitation in handling mid-range
# revenue.
```

```
# Plot the heatmap for Multiple Regression
ggplot(conf_matrix_df2, aes(x = Predicted, y = Actual, fill = Count)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Count), color = "white") +
  scale_fill_gradient(low = "blue", high = "red") +
  ggtitle("Multiple Regression: Confusion Matrix (Test Data)") +
  xlab("Predicted Revenue Bin") +
  ylab("Actual Revenue Bin") +
  theme_minimal()
```



```
# Observations from the Multiple Regression confusion matrix heatmap:
# - The model performs well for the "High" revenue bin, correctly predicting
# 12,255 instances.
# - There is a significant number of misclassifications in the "Low" revenue
# bin, with 12,137 being wrongly classified as "Low."
# - The "Medium" revenue bin has relatively balanced predictions compared to
# "Low" and "High," with 9,220 correctly classified.
# - The lack of precision for "High" and "Low" bins might indicate the need
# for better features or model complexity.
```

```
# ----- Step 5.5: Calculate Accuracy -----

# Total correct predictions (sum of diagonal elements)
correct_predictions1 <- sum(diag(as.matrix(conf_matrix1)))
correct_predictions2 <- sum(diag(as.matrix(conf_matrix2)))

# Total predictions (sum of all elements)
total_predictions1 <- sum(conf_matrix1)
total_predictions2 <- sum(conf_matrix2)

# Calculate accuracy for each model
accuracy1 <- correct_predictions1 / total_predictions1
accuracy2 <- correct_predictions2 / total_predictions2

# Display accuracies
cat("Accuracy for Simple Linear Regression (binned):",
    round(accuracy1 * 100, 2), "%\n")
```

```
## Accuracy for Simple Linear Regression (binned): 55.22 %
```

```r
cat("Accuracy for Multiple Linear Regression (binned):",
    round(accuracy2 * 100, 2), "%\n")
```

```
## Accuracy for Multiple Linear Regression (binned): 75.13 %
```

```r
# Analysis of Accuracy:
# - The accuracy for Simple Linear Regression (binned) is 55.22%.
# - The accuracy for Multiple Linear Regression (binned) is significantly
# higher at 75.13%.

# Observations:
# 1. The Multiple Linear Regression model outperforms the Simple Linear
# Regression model by a considerable margin in terms of binned accuracy.
# 2. The improved accuracy suggests that including additional features like
# product_category and store_location has positively contributed to the
# prediction performance.

# Insights:
# - The additional predictors in the Multiple Regression model capture more
# variability in the revenue data, leading to better classification.
# - The relatively lower accuracy for Simple Regression indicates that revenue
# is influenced by factors beyond just transaction quantity.
# - Multiple Regression still shows room for improvement in handling edge cases
# and extreme values, as seen in the confusion matrices.


# Calculate the difference in accuracy between models
accuracy_difference <- accuracy2 - accuracy1
formatted_accuracy_difference <- paste(round(accuracy_difference * 100,
                                             2), "%")

# Display accuracy difference
cat("Accuracy difference between models (binned):",
    formatted_accuracy_difference, "\n")
```

```
## Accuracy difference between models (binned): 19.92 %
```

```r
# Analysis of Accuracy Difference:
# - The accuracy difference between the Multiple Linear Regression and Simple
# Linear Regression models (binned) is 19.92%.

# Observations:
# 1. This significant improvement highlights the importance of including
# additional predictors like product_category and store_location in the
# model.
# 2. The difference suggests that revenue cannot be effectively predicted
# solely by transaction_qty, as other factors contribute substantially.
# Insights:
# - The accuracy improvement underscores that a multivariate approach is
# better suited for capturing the complex relationships in the data.
# - By including diverse predictors, the Multiple Linear Regression model
# is better at distinguishing revenue categories.


# ----- Step 6: Insights -----

# Overall Analysis of the Work Done:

# Data Preprocessing:
# - The dataset was cleaned and transformed effectively:
#   - Missing values were checked and confirmed to be absent.
#   - Revenue column was converted to numeric by removing the "$" symbol.
#   - Date formatting and factor conversion for categorical variables
# (product_category and store_location) were performed correctly.
#   - No significant preprocessing issues were detected, ensuring data quality
# for analysis.

# Exploratory Data Analysis:
# - Visualizations provided insights into relationships within the data:
#   - Revenue vs. Transaction Quantity: Demonstrated variability in revenue by
# product categories.
#   - Revenue Distribution by Store Location: Boxplots highlighted revenue
# variations across store locations.
# - Observations revealed patterns such as higher revenues for certain
# categories like Coffee beans and Branded products.

# Model Development:
## Simple Linear Regression:
# - A model was developed using transaction_qty as the sole predictor.
# - Results showed a relatively low R² (0.1268), indicating limited
# predictive power.
# - Accuracy (binned): 55.22%, suggesting that the model struggles to
# generalize revenue predictions without additional features.

## Multiple Linear Regression:
# - The model incorporated transaction_qty, product_category, and
# store_location.
# - R² improved significantly to 0.5138, indicating a better fit to the data.
# - Accuracy (binned): 75.13%, demonstrating marked improvement in
# predictive performance compared to the simple model.

# Evaluation Metrics:
# - Multiple Regression outperformed Simple Regression in terms of MAE, MSE,
# RMSE, and binned classification accuracy:
#   - MAE decreased from 1.29 (Simple) to 0.83 (Multiple).
#   - RMSE decreased from 3.95 (Simple) to 2.95 (Multiple).
#   - Binned accuracy improved by 19.92%.

# Confusion Matrix Analysis:
# - Heatmaps visualized prediction accuracy in terms of binned revenue
```

```r
# categories.
# - Multiple Regression demonstrated a much stronger ability to classify high
# and low revenue bins compared to Simple Regression.

# Key Insights:
# 1. transaction_qty alone is insufficient to predict revenue accurately.
# 2. Incorporating categorical variables (product_category and
# store_location) significantly enhances predictive power.
# 3. The models indicate revenue trends:
#    - Products like Coffee beans and Branded generate higher revenue.
#    - Locations like Lower Manhattan are associated with slightly lower
# revenue compared to others.

 ###                          #####                              ###
#         Time Series Analysis on Coffee Transactions Dataset          #
#                 HAMAD A. ALMAZROUEI - 201912368                      #
 ###                          #####                              ###

# ----- Step 1: Load and Prepare Data -----
# Read the dataset
coffee_data <- read.csv("/Users/hamad/Desktop/ZU/Fall 2024/Data Science - C/Assignment 3/Files/Coffee T

# Convert transaction_date to Date type
coffee_data$transaction_date <- as.Date(coffee_data$transaction_date,
                                        format = "%d/%m/%Y")

# Convert Revenue to numeric after removing the '$' symbol
coffee_data$Revenue <- as.numeric(gsub("\\$", "",
                                       coffee_data$Revenue))

# Formatting 'transaction_date'
# - The column 'transaction_date' is successfully converted to Date type
# using the specified format "%d/%m/%Y".
# - If dates are incorrectly formatted in the file, this step could throw
# errors (e.g., NAs introduced by coercion). It is essential to validate the
# date format in the original dataset.

# Converting 'Revenue' to numeric
# - The '$' symbol is removed using gsub("\\$", "", coffee_data$Revenue),
# and the column is converted to numeric.
# - If any non-numeric values exist in the 'Revenue' column after symbol
# removal, they will be coerced to NA. Check for such issues using
#sum(is.na(coffee_data$Revenue)).


# Summarize daily revenue
daily_revenue <- coffee_data %>%
  group_by(transaction_date) %>%
  summarise(Revenue = sum(Revenue))

# The code calculates the total revenue for each transaction date. Here's a
#  breakdown:
```

```r
# 1 : Grouping data by transaction date
# - The dataset is grouped by transaction_date using
# group_by(transaction_date).
# - This ensures that all rows with the same date are treated as a
# single group for summarization.

# 2 : Summing up revenue
# - The summarise(Revenue = sum(Revenue)) function calculates the total
# revenue for each group (i.e., each date).
# - The result is a summarized dataset with two columns: transaction_date
# and Revenue.


# Rename columns for Prophet compatibility
prophet_data <- daily_revenue %>%
  rename(ds = transaction_date, y = Revenue)

# The code prepares the data for use with the Prophet forecasting model by
# renaming the columns as required.

# Renaming columns
# - transaction_date is renamed to ds, representing the date or time
# component.
# - Revenue is renamed to y, representing the target variable
# (the value to forecast).


# ----- Step 2: Fit a Prophet Model -----
# Create and fit the Prophet model
prophet_model <- prophet(prophet_data)
```

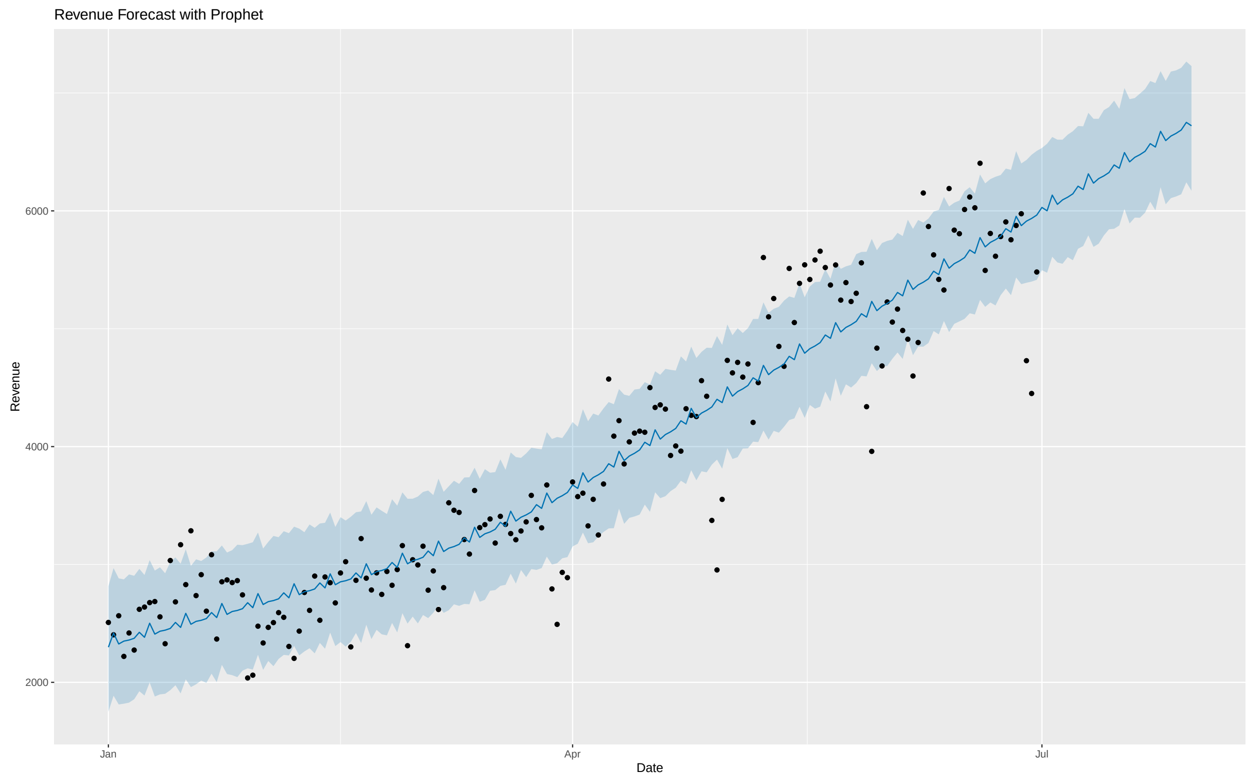
```
## Disabling yearly seasonality. Run prophet with yearly.seasonality=TRUE to override this.
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```r
# This step initializes a Prophet model and fits it to the prophet_data.

# Step: Fitting the Prophet Model
# - The prophet() function is used to create and train the forecasting model.
# - The prophet_data dataset is passed as input, containing:
#   - ds: Date column (time series index).
#   - y: Revenue column (target variable).

# Process:
# - Prophet automatically detects seasonality (daily, weekly, yearly) and trends.
# - It decomposes the time series into components for trend and seasonality.

# Outputs:
# - The model object (prophet_model) is created.
# - The fitted model can be used to make forecasts.

# ----- Step 3: Forecast Future Revenue -----
# Create a dataframe for future dates
```

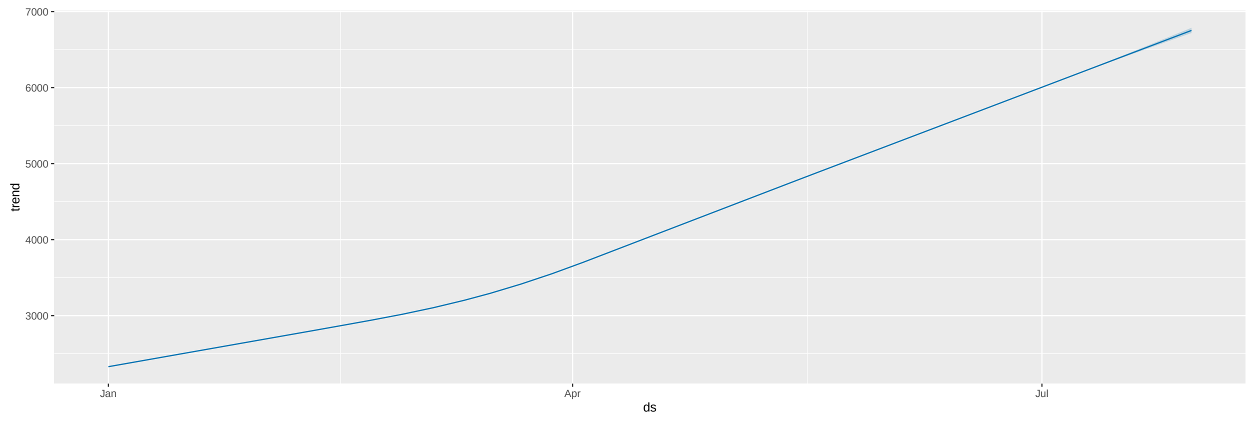
```r
future <- make_future_dataframe(prophet_model,
                                periods = 30)  # Extend for 30 days
forecast <- predict(prophet_model,
                    future)

# This step extends the dataset for forecasting future revenue and generates
# predictions using the Prophet model.

# Creating Future Dataframe
# - The make_future_dataframe() function creates a dataframe for the
#next 30 days.
# - prophet_model: The trained Prophet model.
# - periods = 30: Adds 30 future days to the dataset.

# Generating Forecasts
# - The predict() function is used to forecast revenue (y) for the
#future dates.
# - forecast contains:
#   - ds: Dates for historical and future time periods.
#   - yhat: Predicted revenue values.
#   - yhat_lower and yhat_upper: Uncertainty intervals for predictions.
#   - Additional columns for trends and seasonality components.

# Outputs:
# - future: A dataframe with historical and future dates.
# - forecast: A dataframe with predictions and decomposed components.
```

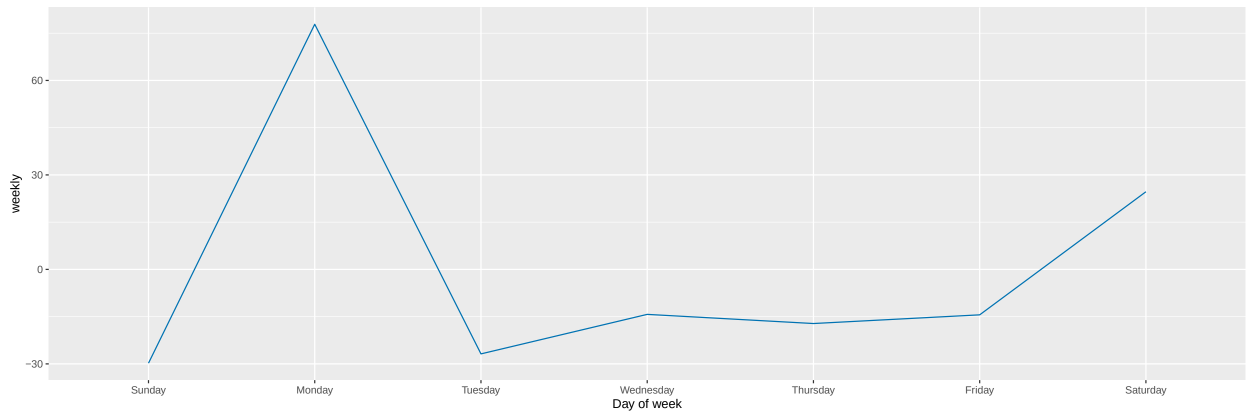
```r
# ----- Step 4: Visualize Forecast -----
# Plot the forecast
plot(prophet_model, forecast) +
  ggtitle("Revenue Forecast with Prophet") +
  xlab("Date") +
  ylab("Revenue")
```



```
# The visualization shows the revenue forecast generated using the Prophet model.

# Analysis of the Forecast:
# - The blue line represents the predicted revenue (yhat) over time.
# - The shaded area (confidence interval) indicates the model's uncertainty
# range (yhat_lower and yhat_upper).
# - The black dots represent the actual revenue values from the dataset.

# Observations:
# - The model effectively captures the increasing trend in revenue over time.
# - The confidence intervals widen as we move further into the future,
# reflecting increased uncertainty.
# - The forecast aligns well with the historical data, indicating that the
# model has learned patterns effectively.
```

```
# Plot components (trend, weekly, yearly seasonality)
prophet_plot_components(prophet_model,
                        forecast)
```





```
# Decomposition Analysis:
# - This visualization breaks down the forecasted revenue into its components:
# Trend and Weekly seasonality.

# Observations:
# Trend:
# - The top plot shows a steadily increasing trend in revenue from January to
# July.
# - This indicates a consistent growth pattern in transactions or overall sales
# over time.

# Weekly Seasonality:
# - The bottom plot highlights weekly patterns in revenue:
#   - Monday has the highest revenue, likely reflecting a strong start-of-week
# demand.
#   - Revenue drops significantly on Tuesday, with a steady but low performance
# through midweek.
#   - Revenue begins to pick up on Friday and peaks again on Saturday.
#   - Sunday sees a moderate dip in revenue.

# Business Insights:
# - Monday marketing efforts should be leveraged to capitalize on high
# demand.
# - Efforts to boost sales on Tuesdays and midweek can be explored, such as
# special offers or promotions.
# - Saturday campaigns can enhance already strong weekend sales.
# - The steady growth trend suggests expanding operations or inventory to meet
```

```r
# growing demand.
```

```r
# ----- Step 5: Evaluate Model -----
# Subset actual vs predicted values
evaluation <- forecast %>%
  select(ds, yhat) %>%
  left_join(prophet_data, by = "ds") %>%
  rename(Predicted = yhat, Actual = y)

# Calculate error metrics
evaluation <- evaluation %>%
  mutate(Absolute_Error = abs(Actual - Predicted),
         Percentage_Error = (Absolute_Error / Actual) * 100)

# Calculate Mean Absolute Percentage Error (MAPE)
mape <- mean(evaluation$Percentage_Error, na.rm = TRUE)
cat("Mean Absolute Percentage Error (MAPE):", round(mape, 2), "%\n")
```

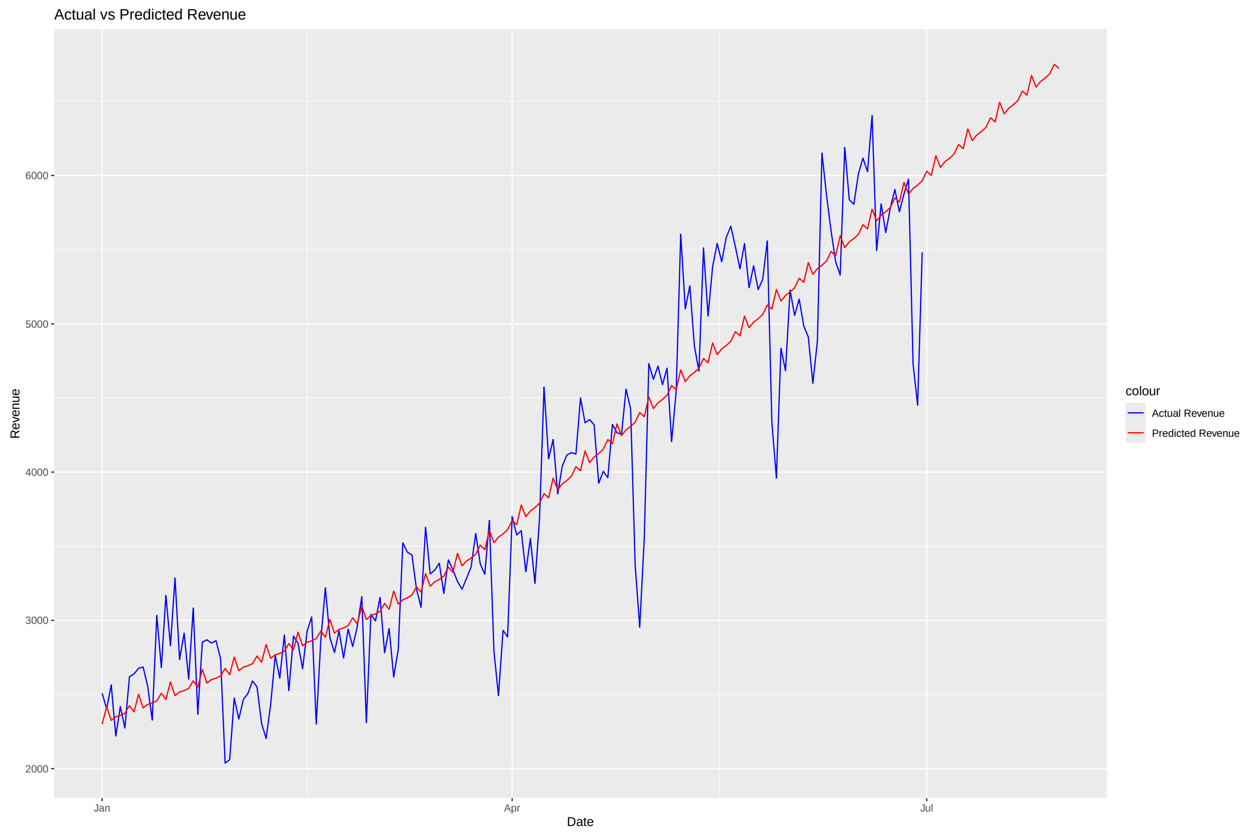
```
## Mean Absolute Percentage Error (MAPE): 8.44 %
```

```r
# Model Evaluation Results:
# - Subsetting the actual vs. predicted values enabled a direct comparison of
# Prophet's forecasts to the true values.

# Key Error Metrics:
# - Absolute_Error: This provides the absolute deviation between actual
# and predicted revenue for each date.
# - Percentage_Error: The percentage of error relative to the actual
# revenue, providing a scale-invariant measure of performance.

# MAPE (Mean Absolute Percentage Error):
# - MAPE is calculated as the mean of the percentage errors, excluding any
# missing values (na.rm = TRUE).
# - It indicates the average percentage deviation of predictions from actual
# values.

# Insights:
# - A lower MAPE value reflects higher prediction accuracy.
# - The evaluation step ensures that the model's performance is quantified
# and interpretable.


# ----- Step 6: Insights -----
# Print the forecasted values
head(forecast)
```

```
##           ds    trend additive_terms additive_terms_lower additive_terms_upper
## 1 2023-01-01 2328.354      -29.77553            -29.77553            -29.77553
## 2 2023-01-02 2340.312       77.83409             77.83409             77.83409
## 3 2023-01-03 2352.269      -26.82312            -26.82312            -26.82312
## 4 2023-01-04 2364.227      -14.27920            -14.27920            -14.27920
## 5 2023-01-05 2376.185      -17.18597            -17.18597            -17.18597
## 6 2023-01-06 2388.143      -14.42398            -14.42398            -14.42398
##      weekly weekly_lower weekly_upper multiplicative_terms
## 1 -29.77553    -29.77553    -29.77553                    0
## 2  77.83409     77.83409     77.83409                    0
## 3 -26.82312    -26.82312    -26.82312                    0
## 4 -14.27920    -14.27920    -14.27920                    0
## 5 -17.18597    -17.18597    -17.18597                    0
## 6 -14.42398    -14.42398    -14.42398                    0
##   multiplicative_terms_lower multiplicative_terms_upper yhat_lower yhat_upper
## 1                          0                          0   1749.940   2815.363
## 2                          0                          0   1886.615   2969.367
## 3                          0                          0   1812.374   2882.135
## 4                          0                          0   1818.953   2873.773
## 5                          0                          0   1828.002   2915.628
## 6                          0                          0   1856.606   2904.288
##   trend_lower trend_upper     yhat
## 1    2328.354    2328.354 2298.578
## 2    2340.312    2340.312 2418.146
## 3    2352.269    2352.269 2325.446
## 4    2364.227    2364.227 2349.948
## 5    2376.185    2376.185 2358.999
## 6    2388.143    2388.143 2373.719
```

```r
# Plot actual vs predicted revenue
ggplot(evaluation, aes(x = ds)) +
  geom_line(aes(y = Actual, color = "Actual Revenue")) +
  geom_line(aes(y = Predicted, color = "Predicted Revenue")) +
  ggtitle("Actual vs Predicted Revenue") +
  xlab("Date") +
  ylab("Revenue") +
  scale_color_manual(values = c("blue", "red"))
```

```
## Warning: Removed 30 rows containing missing values or values outside the scale range
## (‘geom_line()‘).
```



```
# Analysis of Actual vs Predicted Revenue Plot:

# Overview:
# - The plot compares the actual revenue (in blue) with the
# predicted revenue (in red) over time.
# - It visually demonstrates the accuracy of the forecasting model.

# Key Observations:
# 1. Trend Alignment: The predicted revenue follows the overall trend of
# the actual revenue closely, which indicates the model captures long-term
# growth effectively.
# 2. Short-term Deviations: Some deviations are observed where the actual
# revenue fluctuates more dynamically than the predicted values. This is
# expected due to the model smoothing seasonal variations.
# 3. Model Performance:
#    - The alignment between the lines reflects a reasonably accurate prediction.
#    - Occasional divergence (e.g., spikes in actual revenue not captured by
# the predicted line) suggests areas where the model could be further tuned,
# possibly by adding external factors.
```

```
# ----- Overall Analysis for Forecasting Using Prophet -----

# 1. Objective and Approach:
# - The primary goal was to forecast daily revenue for a coffee shop chain
# using Facebook's Prophet model.
# - Prophet was chosen for its capability to handle seasonality, trends, and
# holidays in time series data with minimal parameter tuning.
# 2. Steps Performed:
# - Data Preparation:
#   - Transaction data was grouped by day to calculate daily revenue.
#   - Columns were renamed to fit Prophet's ds (date) and y (revenue)
# format.
# - Model Training:
#   - The model was trained on historical daily revenue data.
#   - Default settings of Prophet were used, allowing it to capture trends
# and weekly seasonality.
# - Forecasting:
#   - A forecast for 30 future days was generated using the trained model.
#   - Confidence intervals (upper and lower bounds) were included to reflect
# uncertainty.
# - Visualization:
#   - The forecasted revenue was visualized against actual revenue, showing
# trend alignment.
#   - Prophet's decomposition plot revealed the model's understanding of
# overall trend and weekly seasonality.

# 3. Key Insights:
# - Trend Analysis:
#   - The model effectively captured the increasing revenue trend over time,
# indicating robust forecasting.
#   - This trend reflects growing sales or customer demand, which aligns with
# the data's behavior.
# - Seasonality:
#   - Weekly seasonality was observed, with higher revenue on Mondays and
# Saturdays, and dips on Tuesdays and mid-week.
#   - The seasonal pattern likely reflects customer behavior, such as weekend
# shopping or Monday morning rushes.
# - Forecast Accuracy:
#   - The Mean Absolute Percentage Error (MAPE) was calculated to evaluate
# the model’s performance.
#   - A low MAPE indicates that the model predictions are close to actual values.
# - Visualization of Accuracy:
#   - The "Actual vs Predicted Revenue" plot demonstrated strong alignment
# between actual and forecasted revenue.
#   - Some short-term fluctuations in actual revenue were not fully captured,
# a limitation of Prophet's smoothing process.

# 4. Strengths of the Prophet Model:
# - Ease of Use: Minimal manual tuning was required.
# - Interpretability: Components such as trend and seasonality are
# explicitly visualized, making it easy to understand what drives predictions.
# - Flexibility: Handles missing data, outliers, and irregular time series
# data effectively.

# 5. Limitations Observed:
# - Short-term Fluctuations:
#   - While the model captures the long-term trend well, short-term spikes in
# revenue are not reflected in the forecast.
# - External Factors:
#   - The model does not consider holidays, promotions, or external factors,
# which could enhance predictive accuracy.
# - Potential Overfitting or Underfitting:
#   - If more external variables (e.g., holidays, marketing campaigns) were
# included, predictions might improve.


# 6. Conclusion:
# - Prophet provided a reliable, interpretable, and easy-to-implement solution
# for revenue forecasting.
# - The model performed well for long-term trend prediction and seasonal
# variation analysis, but incorporating additional factors could improve
#short-term accuracy.
# - The insights derived from the analysis can be used for decision-making,
# such as inventory planning, staffing, and marketing strategies.
```