Bike Rental Prediction Project

Description:

Problem Statement:

In bike-sharing systems, the entire process from membership to rental and return has been automated.
Using these systems, users can easily rent a bike from one location and return it to another. Hence, a bike rental company wants to understand and predict the number of bikes rented daily based on the environment and seasons.

Objective: The objective of this case is to predict bike rental counts based on environmental and seasonal settings with the help of a machine learning algorithm.

Steps to Perform:

1. Exploratory data analysis
- Load dataset and libraries

```
# Load necessary libraries
library(ggplot2)
library(dplyr)

# Load the dataset

bike_data <- read.csv(file.choose())
```

```
Console   Terminal ×   Background Jobs ×
R 4.3.2 · ~/

R version 4.3.2 (2023-10-31) -- "Eye Holes"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> library(ggplot2)
> library(dplyr)
```

# Display the first few rows of the dataset
head(bike_data)

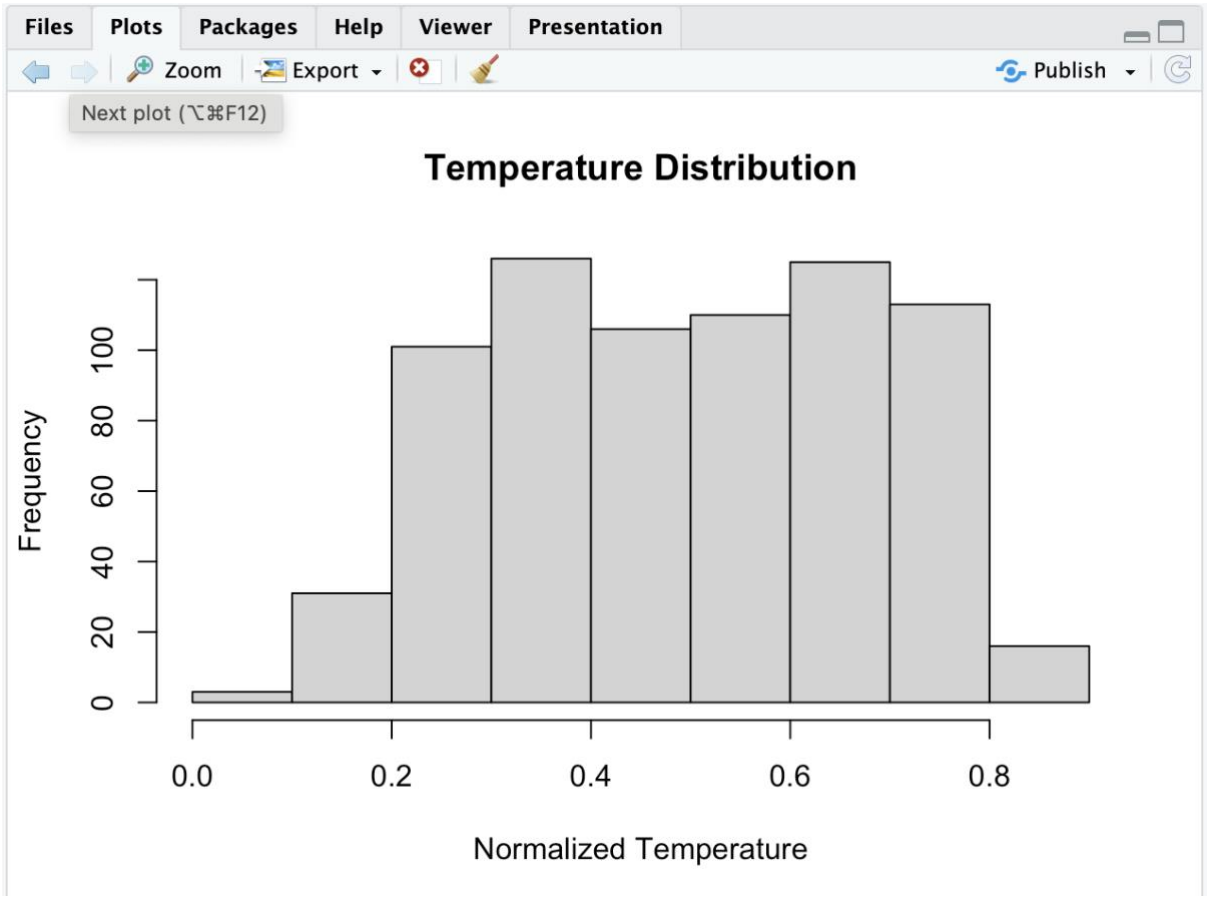
```
Console   Terminal ×   Background Jobs ×
R 4.3.2 · ~/
> # Display the first few rows of the dataset
> head(bike_data)
  instant     dteday season yr mnth holiday weekday workingday weathersit
1       1 0001-01-20      1  0    1       0       6          0          2
2       2 0002-01-20      1  0    1       0       0          0          2
3       3 0003-01-20      1  0    1       0       1          1          1
4       4 0004-01-20      1  0    1       0       2          1          1
5       5 0005-01-20      1  0    1       0       3          1          1
6       6 0006-01-20      1  0    1       0       4          1          1
      temp    atemp      hum windspeed casual registered  cnt year
1 0.344167 0.363625 0.805833 0.1604460    331        654  985 0001
2 0.363478 0.353739 0.696087 0.2485390    131        670  801 0002
3 0.196364 0.189405 0.437273 0.2483090    120       1229 1349 0003
4 0.200000 0.212122 0.590435 0.1602960    108       1454 1562 0004
5 0.226957 0.229270 0.436957 0.1869000     82       1518 1600 0005
6 0.204348 0.233209 0.518261 0.0895652     88       1518 1606 0006
>
```

- Perform data type conversion of the attributes

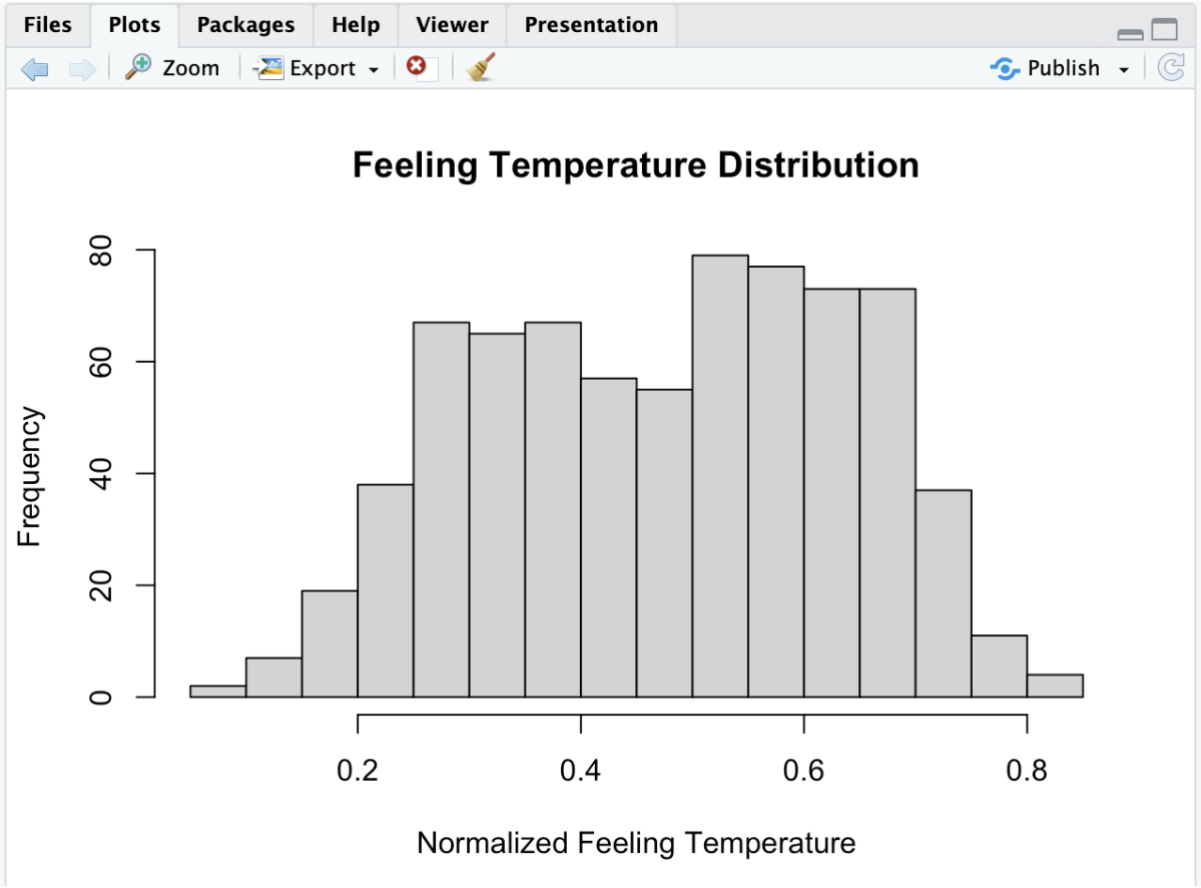
Exploratory Data Analysis (EDA)
Summary statistics
summary(bike_data)

```
> # Exploratory Data Analysis (EDA)
> ## Summary statistics
> summary(bike_data)
    instant          dteday              season           yr
 Min.   :  1.0   Min.   :0001-01-20   Min.   :1.000   Min.   :0.0000
 1st Qu.:183.5   1st Qu.:0008-08-20   1st Qu.:2.000   1st Qu.:0.0000
 Median :366.0   Median :0016-03-20   Median :3.000   Median :1.0000
 Mean   :366.0   Mean   :0016-04-01   Mean   :2.497   Mean   :0.5007
 3rd Qu.:548.5   3rd Qu.:0023-11-04   3rd Qu.:3.000   3rd Qu.:1.0000
 Max.   :731.0   Max.   :0031-12-20   Max.   :4.000   Max.   :1.0000
      mnth          holiday           weekday        workingday    weathersit
 Min.   : 1.00   Min.   :0.00000   Min.   :0.000   Min.   :0.000   1:463
 1st Qu.: 4.00   1st Qu.:0.00000   1st Qu.:1.000   1st Qu.:0.000   2:247
 Median : 7.00   Median :0.00000   Median :3.000   Median :1.000   3: 21
 Mean   : 6.52   Mean   :0.02873   Mean   :2.997   Mean   :0.684
 3rd Qu.:10.00   3rd Qu.:0.00000   3rd Qu.:5.000   3rd Qu.:1.000
 Max.   :12.00   Max.   :1.00000   Max.   :6.000   Max.   :1.000
      temp             atemp              hum          windspeed
 Min.   :0.05913   Min.   :0.07907   Min.   :0.0000   Min.   :0.02239
 1st Qu.:0.33708   1st Qu.:0.33784   1st Qu.:0.5200   1st Qu.:0.13495
 Median :0.49833   Median :0.48673   Median :0.6267   Median :0.18097
 Mean   :0.49538   Mean   :0.47435   Mean   :0.6279   Mean   :0.19049
 3rd Qu.:0.65542   3rd Qu.:0.60860   3rd Qu.:0.7302   3rd Qu.:0.23321
 Max.   :0.86167   Max.   :0.84090   Max.   :0.9725   Max.   :0.50746
     casual         registered        cnt           year
 Min.   :   2.0   Min.   :  20   Min.   :  22   Length:731
 1st Qu.: 315.5   1st Qu.:2497   1st Qu.:3152   Class :character
 Median : 713.0   Median :3662   Median :4548   Mode  :character
 Mean   : 848.2   Mean   :3656   Mean   :4504
 3rd Qu.:1096.0   3rd Qu.:4776   3rd Qu.:5956
 Max.   :3410.0   Max.   :6946   Max.   :8714
>
```

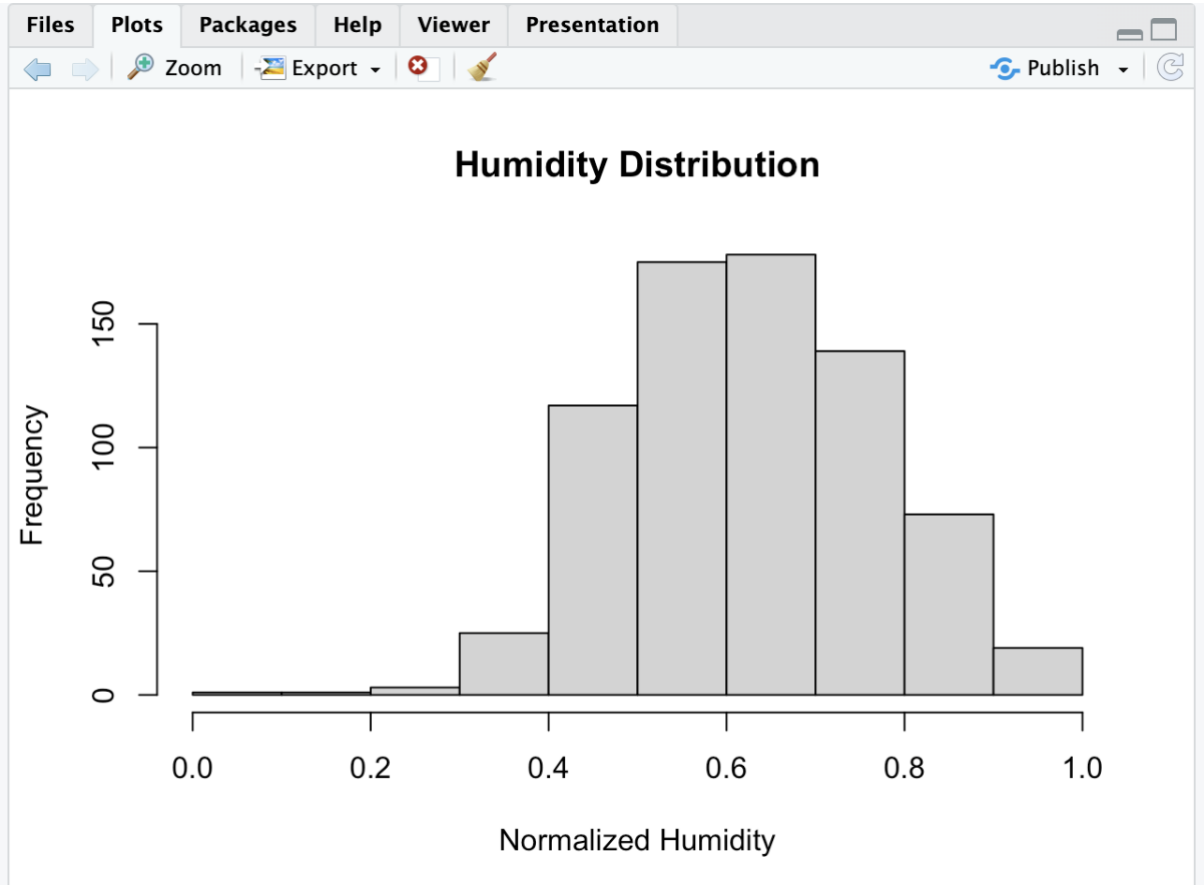
Visualize distributions of numeric variables
hist(bike_data$temp, main="Temperature Distribution", xlab="Normalized Temperature")



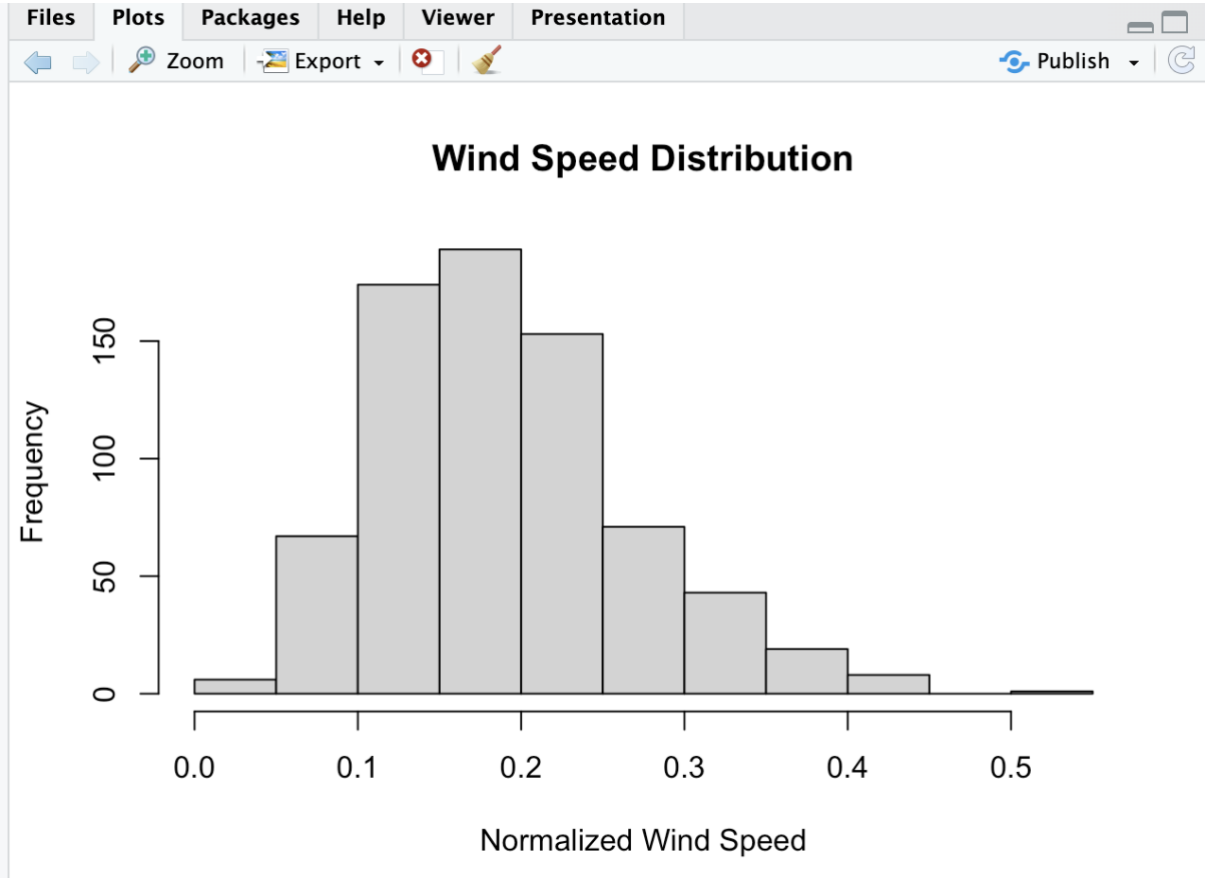
hist(bike_data$atemp, main="Feeling Temperature Distribution", xlab="Normalized Feeling Temperature")



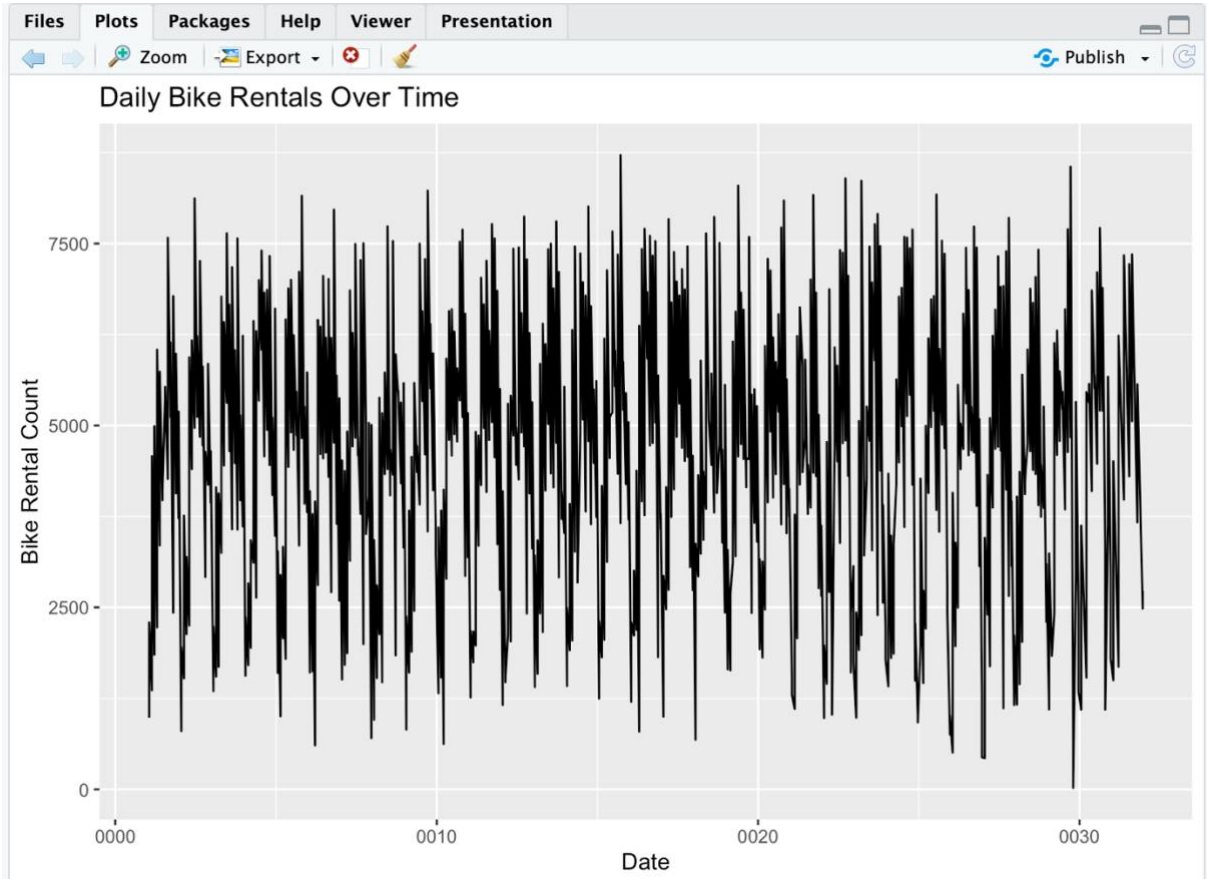
hist(bike_data$hum, main="Humidity Distribution", xlab="Normalized Humidity")



hist(bike_data$windspeed, main="Wind Speed Distribution", xlab="Normalized Wind Speed")



Visualize bike rentals over time
bike_data$dteday <- as.Date(bike_data$dteday) # Convert dteday to Date object
ggplot(bike_data, aes(x=dteday, y=cnt)) + geom_line() + labs(title="Daily Bike Rentals Over Time", x="Date", y="Bike Rental Count")



Data Type Conversion
Convert 'season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit' to factor as they are categorical variables
bike_data$season <- as.factor(bike_data$season)
bike_data$yr <- as.factor(bike_data$yr)
bike_data$mnth <- as.factor(bike_data$mnth)
bike_data$holiday <- as.factor(bike_data$holiday)
bike_data$weekday <- as.factor(bike_data$weekday)
bike_data$workingday <- as.factor(bike_data$workingday)
bike_data$weathersit <- as.factor(bike_data$weathersit)

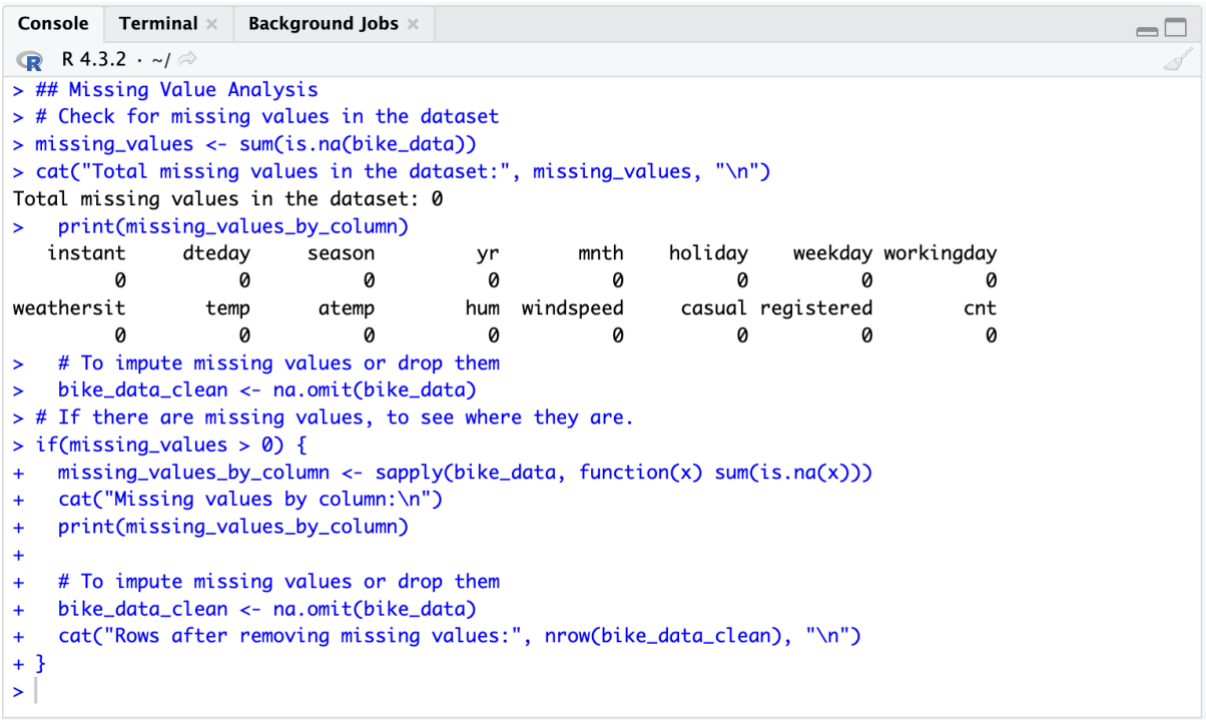
```
> # Data Type Conversion
> ## Convert 'season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit' to factor as they
are categorical variables
> bike_data$season <- as.factor(bike_data$season)
> bike_data$yr <- as.factor(bike_data$yr)
> bike_data$mnth <- as.factor(bike_data$mnth)
> bike_data$holiday <- as.factor(bike_data$holiday)
> bike_data$weekday <- as.factor(bike_data$weekday)
> bike_data$workingday <- as.factor(bike_data$workingday)
> bike_data$weathersit <- as.factor(bike_data$weathersit)
>
```

# Check the structure of the data after conversions
str(bike_data)

```
Console  Terminal  Background Jobs
R 4.3.2 · ~/
> # Check the structure of the data after conversions
> str(bike_data)
'data.frame':   731 obs. of  17 variables:
 $ instant   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ dteday    : Date, format: "0001-01-20" "0002-01-20" "0003-01-20" ...
 $ season    : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
 $ yr        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ mnth      : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ weekday   : Factor w/ 7 levels "0","1","2","3",..: 7 1 2 3 4 5 6 7 1 2 ...
 $ workingday: Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 1 1 2 ...
 $ weathersit: Factor w/ 3 levels "1","2","3": 2 2 1 1 1 1 2 2 1 1 ...
 $ temp      : num  0.344 0.363 0.196 0.2 0.227 ...
 $ atemp     : num  0.364 0.354 0.189 0.212 0.229 ...
 $ hum       : num  0.806 0.696 0.437 0.59 0.437 ...
 $ windspeed : num  0.16 0.249 0.248 0.16 0.187 ...
 $ casual    : int  331 131 120 108 82 88 148 68 54 41 ...
 $ registered: int  654 670 1229 1454 1518 1518 1362 891 768 1280 ...
 $ cnt       : int  985 801 1349 1562 1600 1606 1510 959 822 1321 ...
 $ year      : chr  "0001" "0002" "0003" "0004" ...
>
```

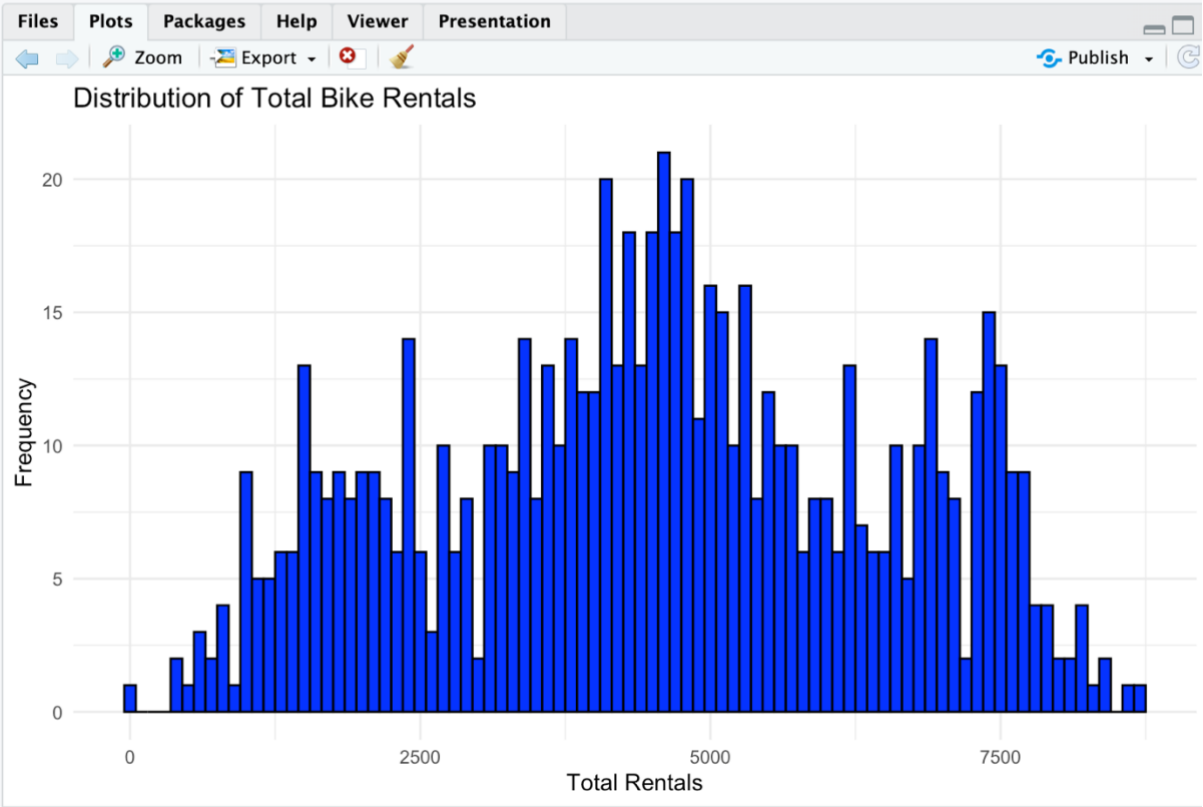
- Carry out the missing value analysis

```
## Missing Value Analysis
# Check for missing values in the dataset
missing_values <- sum(is.na(bike_data))
cat("Total missing values in the dataset:", missing_values, "\n")

# If there are missing values, to see where they are.
if(missing_values > 0) {
  missing_values_by_column <- sapply(bike_data, function(x) sum(is.na(x)))
  cat("Missing values by column:\n")
  print(missing_values_by_column)

  # To impute missing values or drop them
  bike_data_clean <- na.omit(bike_data)
  cat("Rows after removing missing values:", nrow(bike_data_clean), "\n")
}
```



```
# Visualizing the distribution of total bike rentals
library(ggplot2)
ggplot(bike_data, aes(x = cnt)) +
  geom_histogram(binwidth = 100, fill = "blue", color = "black") +
  theme_minimal() +
  labs(title = "Distribution of Total Bike Rentals", x = "Total Rentals", y = "Frequency")
```

```
> # Visualizing the distribution of total bike rentals
> library(ggplot2)
> ggplot(bike_data, aes(x = cnt)) +
+   geom_histogram(binwidth = 100, fill = "blue", color = "black") +
+   theme_minimal() +
+   labs(title = "Distribution of Total Bike Rentals", x = "Total Rentals", y = "Frequency")
>
```

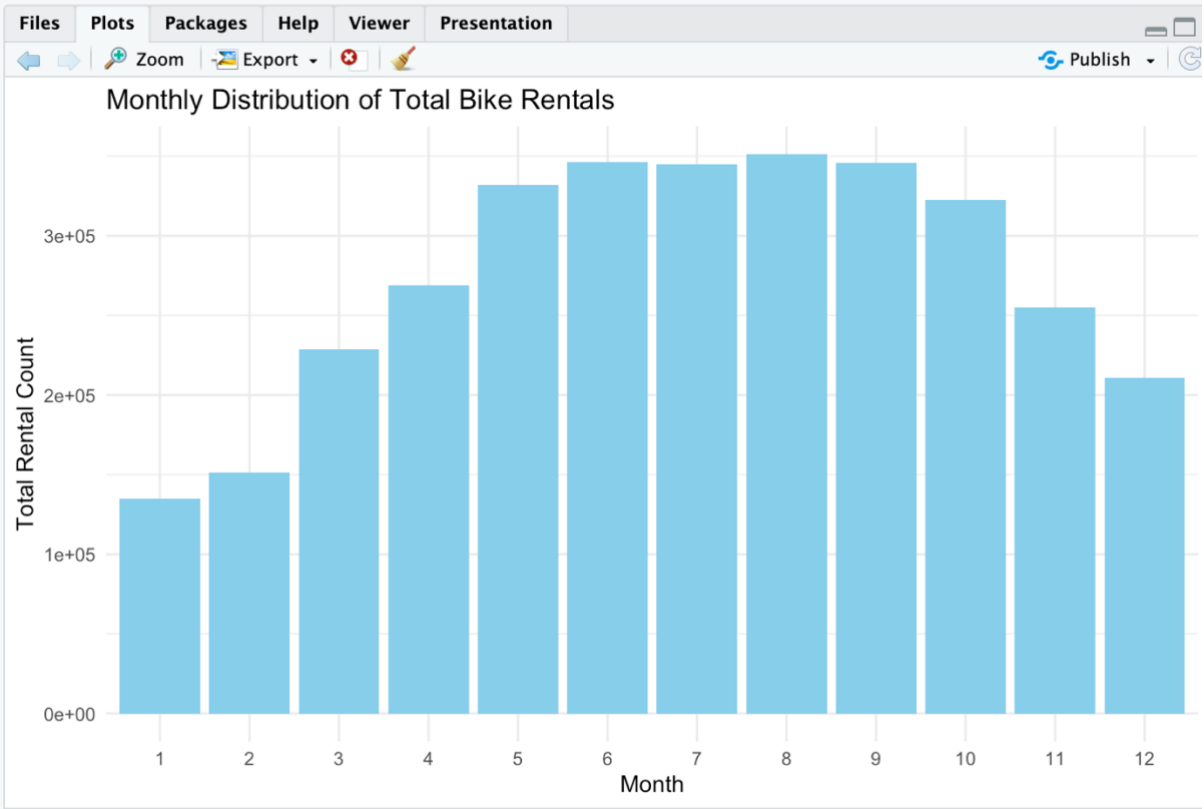


2. Attributes distributions and trends

- Plot monthly distribution of the total number of bikes rented

```
# Plot monthly distribution of the total number of bikes rented
monthly_rental <- bike_data %>%
  group_by(mnth) %>%
  summarise(total_rental = sum(cnt))

# Plot the monthly distribution
ggplot(monthly_rental, aes(x = factor(mnth), y = total_rental)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Monthly Distribution of Total Bike Rentals",
       x = "Month",
       y = "Total Rental Count") +
  theme_minimal()
```

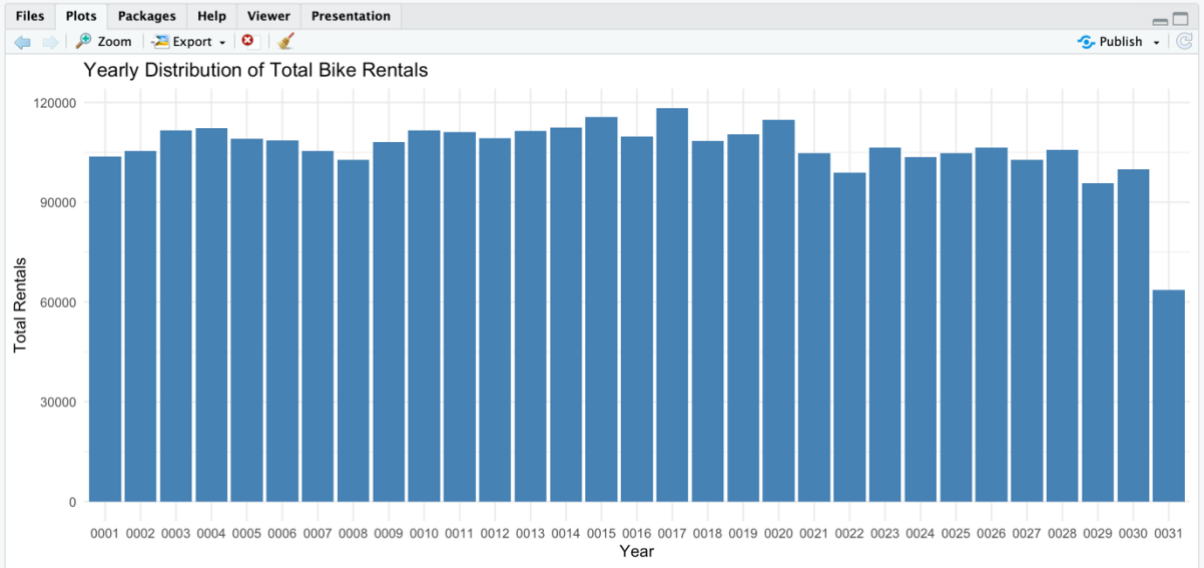
```
> # Plot monthly distribution of the total number of bikes rented
> monthly_rental <- bike_data %>%
+   group_by(mnth) %>%
+   summarise(total_rental = sum(cnt))
> # Plot the monthly distribution
> ggplot(monthly_rental, aes(x = factor(mnth), y = total_rental)) +
+   geom_bar(stat = "identity", fill = "skyblue") +
+   labs(title = "Monthly Distribution of Total Bike Rentals",
+        x = "Month",
+        y = "Total Rental Count") +
+   theme_minimal()
>
```



- Plot yearly distribution of the total number of bikes rented

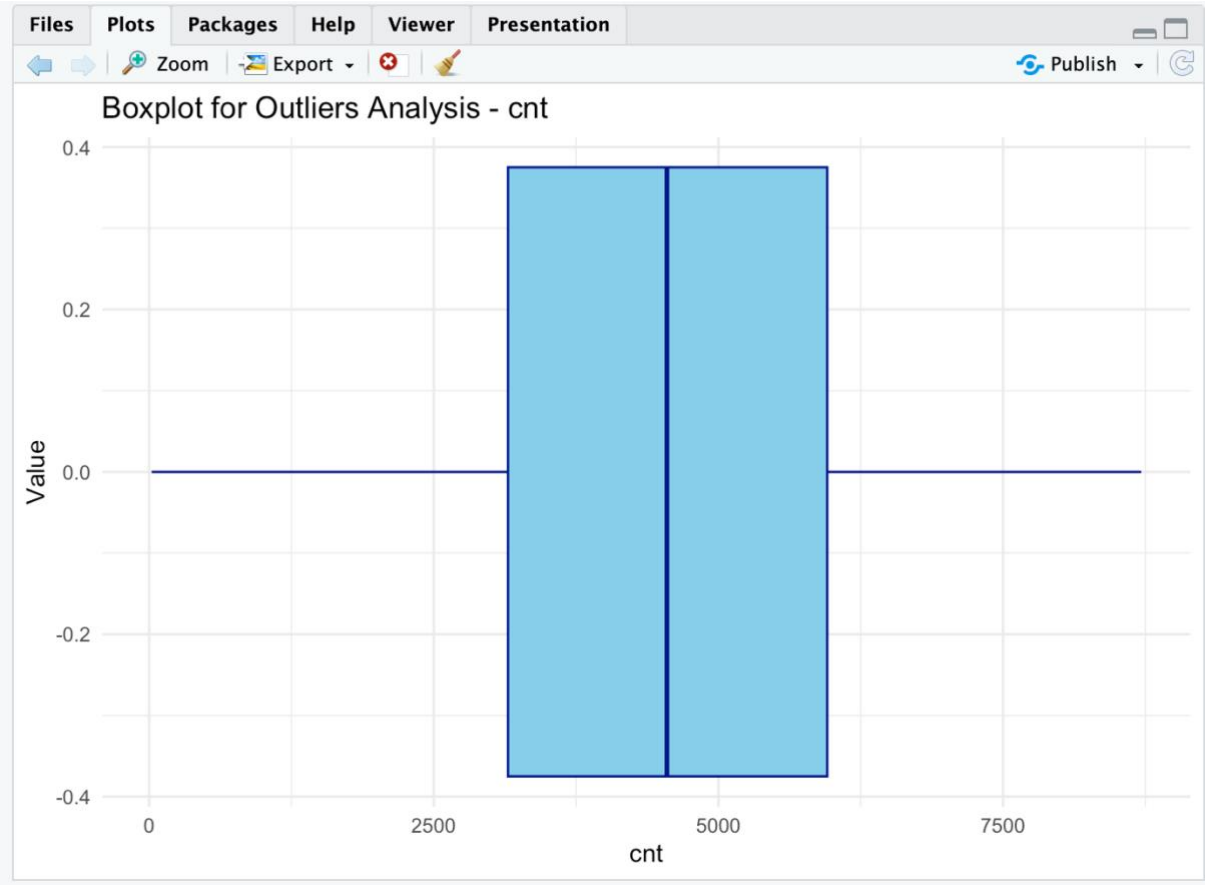
```
# Convert the 'dteday' column to Date type
bike_data$dteday <- as.Date(bike_data$dteday, format="%Y-%m-%d")

# Extract year from the 'dteday' column
bike_data$year <- format(bike_data$dteday, "%Y")

# Summarize total bikes rented by year
yearly_data <- bike_data %>%
  group_by(year) %>%
  summarise(total_rentals = sum(cnt))

# Plotting the yearly distribution of total bike rentals
ggplot(yearly_data, aes(x=year, y=total_rentals)) +
  geom_bar(stat="identity", fill="steelblue") +
  theme_minimal() +
  labs(title="Yearly Distribution of Total Bike Rentals",
       x="Year",
       y="Total Rentals")
```

```
> # Convert the 'dteday' column to Date type
> bike_data$dteday <- as.Date(bike_data$dteday, format="%Y-%m-%d")
> # Extract year from the 'dteday' column
> bike_data$year <- format(bike_data$dteday, "%Y")
> # Summarize total bikes rented by year
> yearly_data <- bike_data %>%
+   group_by(year) %>%
+   summarise(total_rentals = sum(cnt))
> # Plotting the yearly distribution of total bike rentals
> ggplot(yearly_data, aes(x=year, y=total_rentals)) +
+   geom_bar(stat="identity", fill="steelblue") +
+   theme_minimal() +
+   labs(title="Yearly Distribution of Total Bike Rentals",
+        x="Year",
+        y="Total Rentals")
>
```



- Plot boxplot for outliers analysis

```
# Function to plot boxplots for outlier analysis
plot_outliers <- function(data, feature_name) {
  # Using ggplot2 to create a boxplot to visualize outliers
  ggplot(data, aes_string(x = feature_name)) +
    geom_boxplot(fill = "skyblue", color = "darkblue") +
    labs(title = paste("Boxplot for Outliers Analysis -", feature_name),
         x = feature_name,
         y = "Value") +
    theme_minimal()
}

# Attributes to be analyzed for outliers
attributes <- c("temp", "atemp", "hum", "windspeed", "casual", "registered", "cnt")

# Loop through attributes and plot boxplots for each
for(attribute in attributes) {
  print(plot_outliers(bike_data, attribute))
}
```



3. Split the dataset into train and test dataset

```
# Load necessary libraries
install.packages("readr")
library(readr) # For reading CSV files

install.packages("caret")
library(caret) # For data splitting and modeling


# Split the dataset into training and test sets
# First, set a seed for reproducibility
set.seed(123)

# Splitting the data - 70% for training, 30% for testing
splitIndex <- createDataPartition(bike_data$cnt, p = .7, list = FALSE, times = 1)
trainData <- bike_data[ splitIndex,]
testData  <- bike_data[-splitIndex,]

# Just to confirm the size of the split
cat("Training set rows:", nrow(trainData), "\n")
cat("Test set rows:", nrow(testData), "\n")

# Fitting the model on the training data
model <- lm(cnt ~ temp, data = trainData)

# Summary of the model to check its performance metrics
summary(model)

# Predicting on the test data
predictions <- predict(model, newdata = testData)

# Comparing actual vs predicted values (for evaluation, consider using metrics like RMSE,
MAE)
comparison <- data.frame(Actual = testData$cnt, Predicted = predictions)
head(comparison)
```

Console | Terminal | Background Jobs

R 4.3.2 · ~/

```
> library(readr) # For reading CSV files
> library(caret) # For data splitting and modeling
> # Split the dataset into training and test sets
> # First, set a seed for reproducibility
> set.seed(123)
> # Splitting the data - 70% for training, 30% for testing
> splitIndex <- createDataPartition(bike_data$cnt, p = .7, list = FALSE, times = 1)
> trainData <- bike_data[ splitIndex,]
> testData  <- bike_data[-splitIndex,]
> # Just to confirm the size of the split
> cat("Training set rows:", nrow(trainData), "\n")
Training set rows: 515
> cat("Test set rows:", nrow(testData), "\n")
Test set rows: 216
> # Fitting the model on the training data
> model <- lm(cnt ~ temp, data = trainData)
> # Summary of the model to check its performance metrics
> summary(model)

Call:
lm(formula = cnt ~ temp, data = trainData)

Residuals:
    Min      1Q  Median      3Q     Max
-4114.6 -1111.9  -164.7  1096.9  3735.8

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1206.8      189.6   6.364 4.37e-10 ***
temp          6658.8      357.8  18.609  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1499 on 513 degrees of freedom
Multiple R-squared:  0.403,     Adjusted R-squared:  0.4018
F-statistic: 346.3 on 1 and 513 DF,  p-value: < 2.2e-16

> # Predicting on the test data
> predictions <- predict(model, newdata = testData)
> # Comparing actual vs predicted values (for evaluation, consider using metrics like RMSE, MA
E)
> comparison <- data.frame(Actual = testData$cnt, Predicted = predictions)
> head(comparison)
   Actual Predicted
1     985  3498.506
2     801  3627.094
3    1349  2514.317
9     822  2127.901
10   1321  2211.136
15   1248  2760.486
>
```

Files | Plots | Packages | Help | Viewer | Presentation

4. Create a model using the random forest algorithm

```
# Load necessary libraries

install.packages("randomForest")
library(randomForest)

# Create a Random Forest model
# We will predict the 'cnt' variable using all other variables except 'instant', 'dteday', 'casual', and 'registered'
model <- randomForest(cnt ~ . -instant -dteday -casual -registered, data=trainData, ntree=500, importance=TRUE)

# Step 4: Evaluate the model
# Predict on test data
predictions <- predict(model, testData)

# Calculate RMSE (Root Mean Squared Error)
rmse <- sqrt(mean((predictions - testData$cnt)^2))
cat("RMSE on test data: ", rmse, "\n")

# Calculate R-squared value
r2 <- cor(predictions, testData$cnt)^2
cat("R-squared: ", r2, "\n")

# Print the importance of variables
importance(model)
```

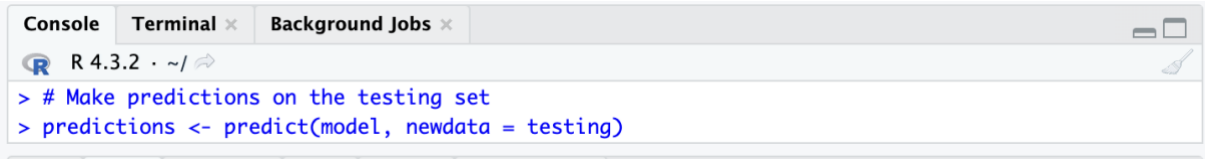
```
Console  Terminal  Background Jobs
R 4.3.2 · ~/
> library(randomForest)
> # Create a Random Forest model
> # We will predict the 'cnt' variable using all other variables except 'instant', 'dteday', 'casual', and 'registered'
> model <- randomForest(cnt ~ . -instant -dteday -casual -registered, data=trainData, ntree=500, importance=TRUE)
> # Step 4: Evaluate the model
> # Predict on test data
> predictions <- predict(model, testData)
> # Calculate RMSE (Root Mean Squared Error)
> rmse <- sqrt(mean((predictions - testData$cnt)^2))
> cat("RMSE on test data: ", rmse, "\n")
RMSE on test data:  700.2809
> # Calculate R-squared value
> r2 <- cor(predictions, testData$cnt)^2
> cat("R-squared: ", r2, "\n")
R-squared:  0.8798572
> # Print the importance of variables
> importance(model)
             %IncMSE IncNodePurity
season     24.556588     133044430
yr        108.468422     465642430
mnth       18.183348     181835159
holiday     2.228197       5117330
weekday     4.505050      48573108
workingday  5.446017       6704082
weathersit 19.146647      50907377
temp       26.172281     425907306
atemp      25.066236     374906104
hum        26.448283     109479233
windspeed  10.470000      59865082
year        3.776658      33419315
>
```

5. Predict the performance of the model on the test dataset

```
# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(bike_data$cnt, p = 0.7, list = FALSE)
training <- bike_data[trainIndex,]
testing <- bike_data[-trainIndex,]

# Train a machine learning model
model <- train(cnt ~ season + yr + mnth + holiday + weekday + workingday + weathersit +
temp + atemp + hum + windspeed, data = training, method = "lm")
```

```
Console  Terminal  Background Jobs
R 4.3.2 · ~/
> # Split the data into training and testing sets
> set.seed(123)
> trainIndex <- createDataPartition(bike_data$cnt, p = 0.7, list = FALSE)
> training <- bike_data[trainIndex,]
> testing <- bike_data[-trainIndex,]
> # Train a machine learning model
> model <- train(cnt ~ season + yr + mnth + holiday + weekday + workingday + weathersit +
+                  temp + atemp + hum + windspeed, data = training, method = "lm")
```

```
# Make predictions on the testing set
predictions <- predict(model, newdata = testing)
```




```
# Evaluate the model performance
performance <- RMSE(predictions, testing$cnt)
print(paste("Root Mean Squared Error:", performance))
```