Final Project EDA

Andrea Cui

04 August, 2023

```
firedata <- read.csv("forestfires.csv")
firedata <- as.data.frame(firedata)

# Load the necessary packages
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library(ggplot2)

# Load the dataset
data <- read.csv("forestfires.csv")

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

```
##   X Y month day FFMC  DMC    DC  ISI temp RH wind rain area
## 1 7 5   mar fri 86.2 26.2  94.3  5.1  8.2 51  6.7  0.0    0
## 2 7 4   oct tue 90.6 35.4 669.1  6.7 18.0 33  0.9  0.0    0
## 3 7 4   oct sat 90.6 43.7 686.9  6.7 14.6 33  1.3  0.0    0
## 4 8 6   mar fri 91.7 33.3  77.5  9.0  8.3 97  4.0  0.2    0
## 5 8 6   mar sun 89.3 51.3 102.2  9.6 11.4 99  1.8  0.0    0
## 6 8 6   aug sun 92.3 85.3 488.0 14.7 22.2 29  5.4  0.0    0
```

```
# Get a summary of the dataset
summary(data)
```

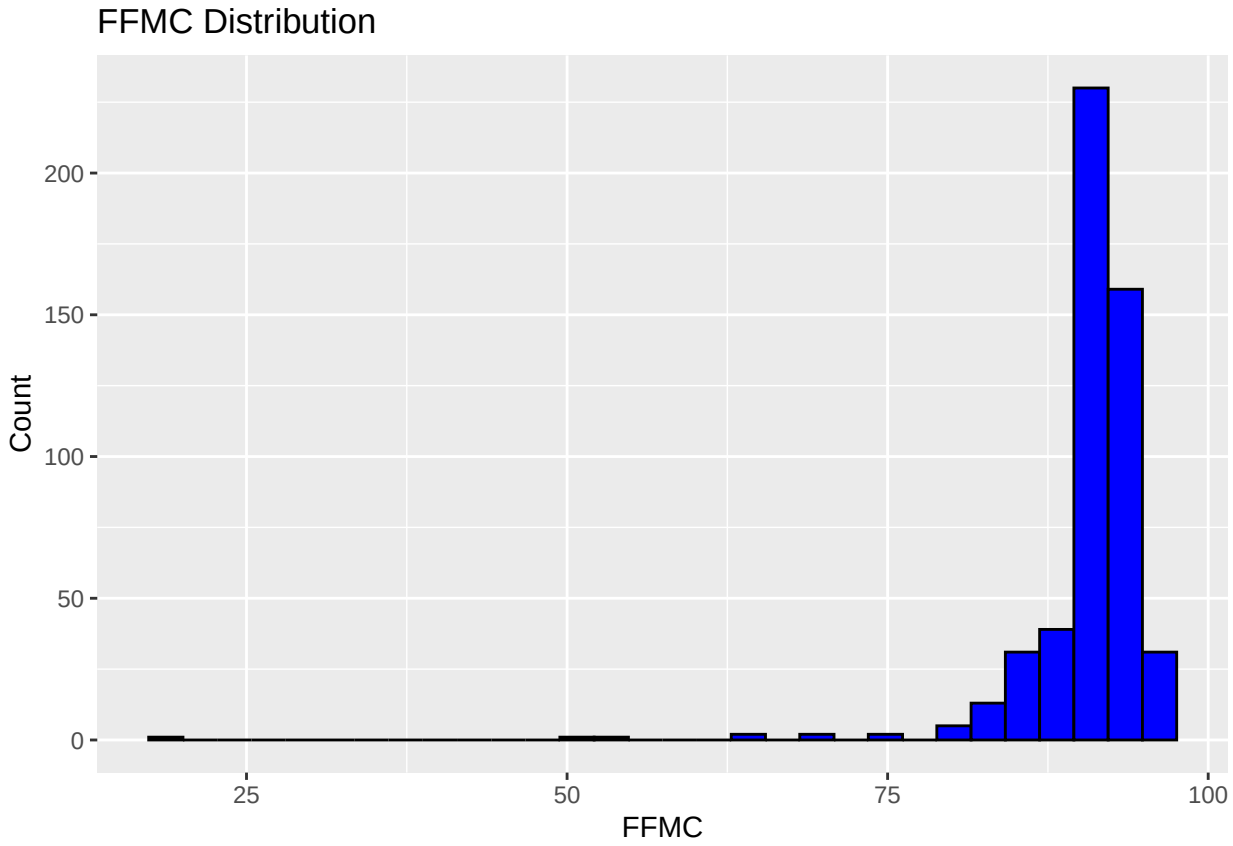
```
##        X               Y          month               day
##  Min.   :1.000   Min.   :2.0   Length:517         Length:517
##  1st Qu.:3.000   1st Qu.:4.0   Class :character   Class :character
```

```
##  Median :4.000   Median :4.0   Mode  :character   Mode  :character
##  Mean   :4.669   Mean   :4.3
##  3rd Qu.:7.000   3rd Qu.:5.0
##  Max.   :9.000   Max.   :9.0
##       FFMC            DMC              DC             ISI
##  Min.   :18.70   Min.   :  1.1   Min.   :  7.9   Min.   : 0.000
##  1st Qu.:90.20   1st Qu.: 68.6   1st Qu.:437.7   1st Qu.: 6.500
##  Median :91.60   Median :108.3   Median :664.2   Median : 8.400
##  Mean   :90.64   Mean   :110.9   Mean   :547.9   Mean   : 9.022
##  3rd Qu.:92.90   3rd Qu.:142.4   3rd Qu.:713.9   3rd Qu.:10.800
##  Max.   :96.20   Max.   :291.3   Max.   :860.6   Max.   :56.100
##       temp             RH              wind            rain
##  Min.   : 2.20   Min.   : 15.00   Min.   :0.400   Min.   :0.00000
##  1st Qu.:15.50   1st Qu.: 33.00   1st Qu.:2.700   1st Qu.:0.00000
##  Median :19.30   Median : 42.00   Median :4.000   Median :0.00000
##  Mean   :18.89   Mean   : 44.29   Mean   :4.018   Mean   :0.02166
##  3rd Qu.:22.80   3rd Qu.: 53.00   3rd Qu.:4.900   3rd Qu.:0.00000
##  Max.   :33.30   Max.   :100.00   Max.   :9.400   Max.   :6.40000
##       area
##  Min.   :   0.00
##  1st Qu.:   0.00
##  Median :   0.52
##  Mean   :  12.85
##  3rd Qu.:   6.57
##  Max.   :1090.84
```

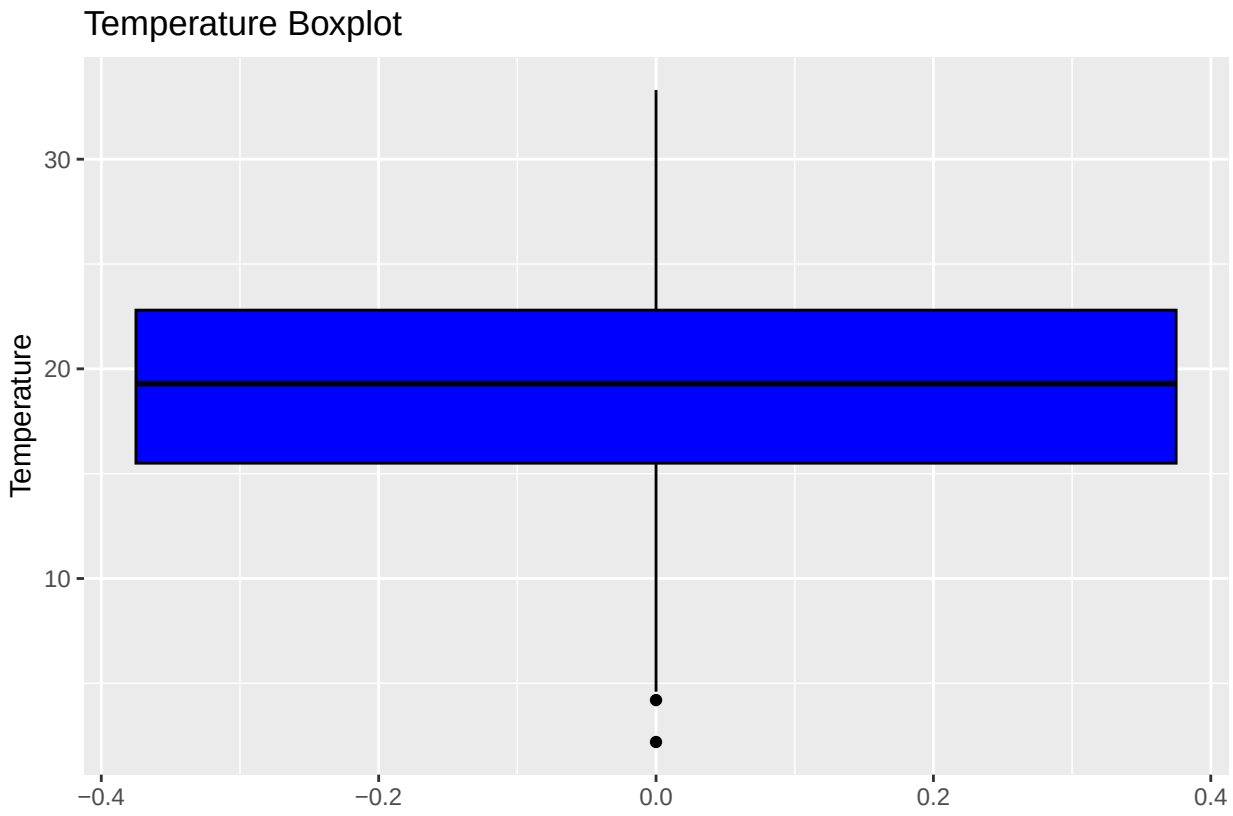
Data Visualizations:

```
# Histogram for the FFMC variable
ggplot(data, aes(x=FFMC)) +
  geom_histogram(fill='blue', color='black') +
  labs(title="FFMC Distribution", x="FFMC", y="Count")
```

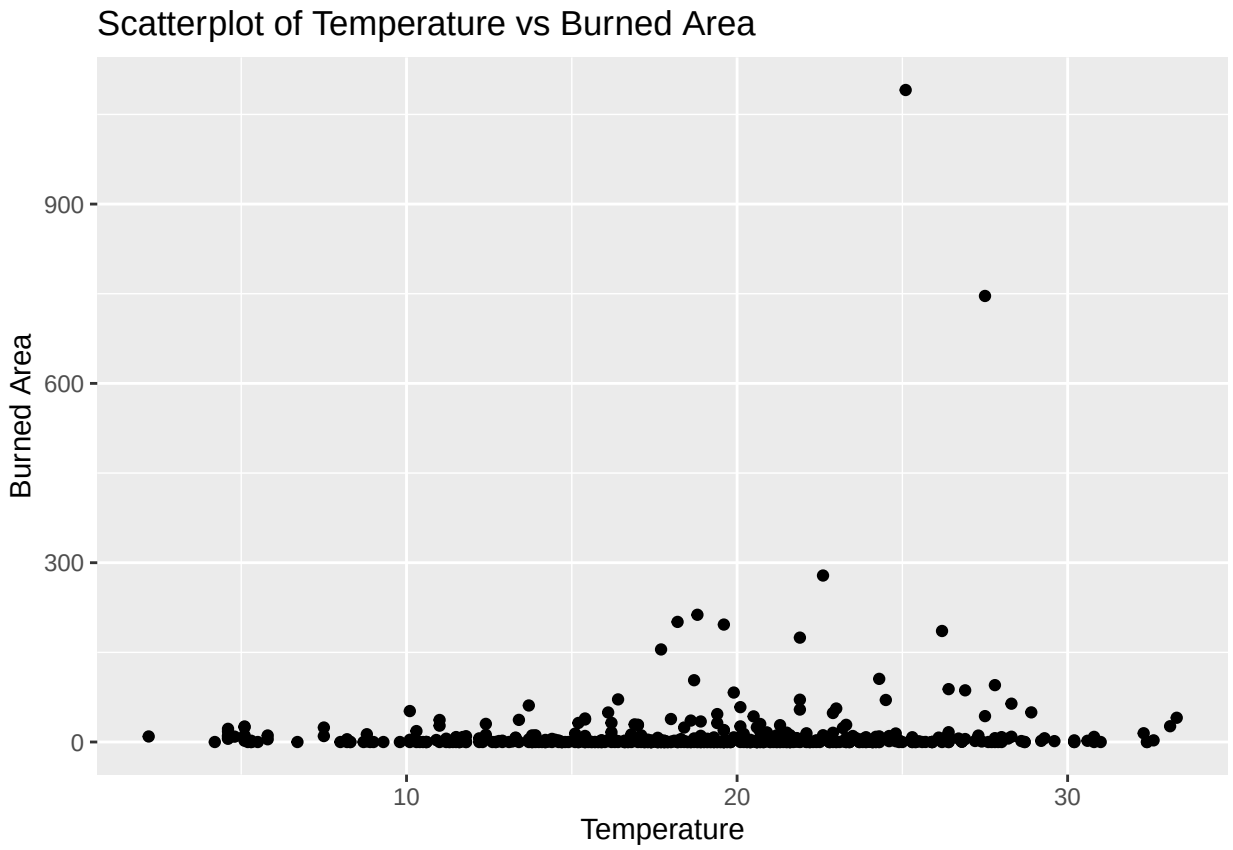
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Boxplot for temperature
ggplot(data, aes(y=temp)) +
  geom_boxplot(fill='blue', color='black') +
  labs(title="Temperature Boxplot", x="", y="Temperature")
```

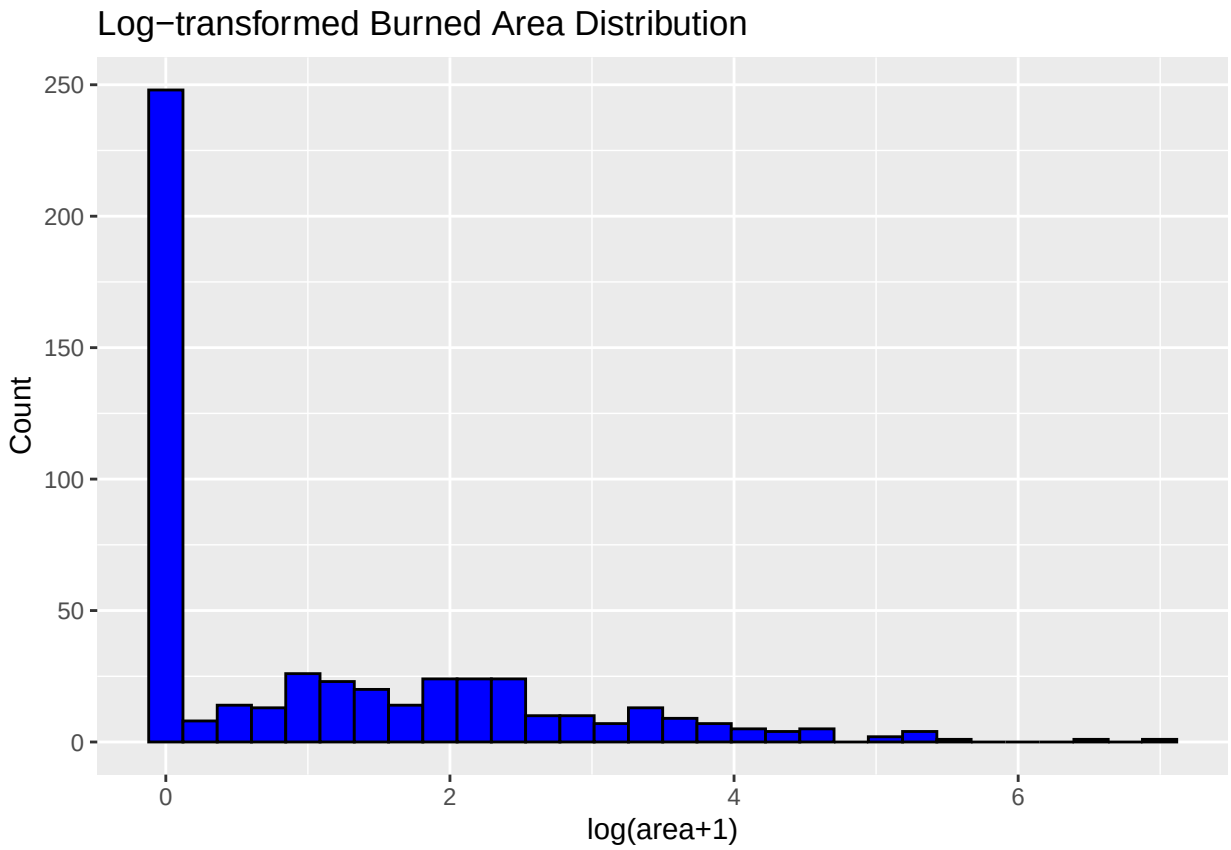



```
ggplot(data, aes(x=temp, y=area)) +
  geom_point() +
  labs(title="Scatterplot of Temperature vs Burned Area", x="Temperature", y="Burned Area")
```




```
# Histogram for the log-transformed area
ggplot(data, aes(x=log(area + 1))) +
  geom_histogram(fill='blue', color='black') +
  labs(title="Log-transformed Burned Area Distribution", x="log(area+1)", y="Count")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



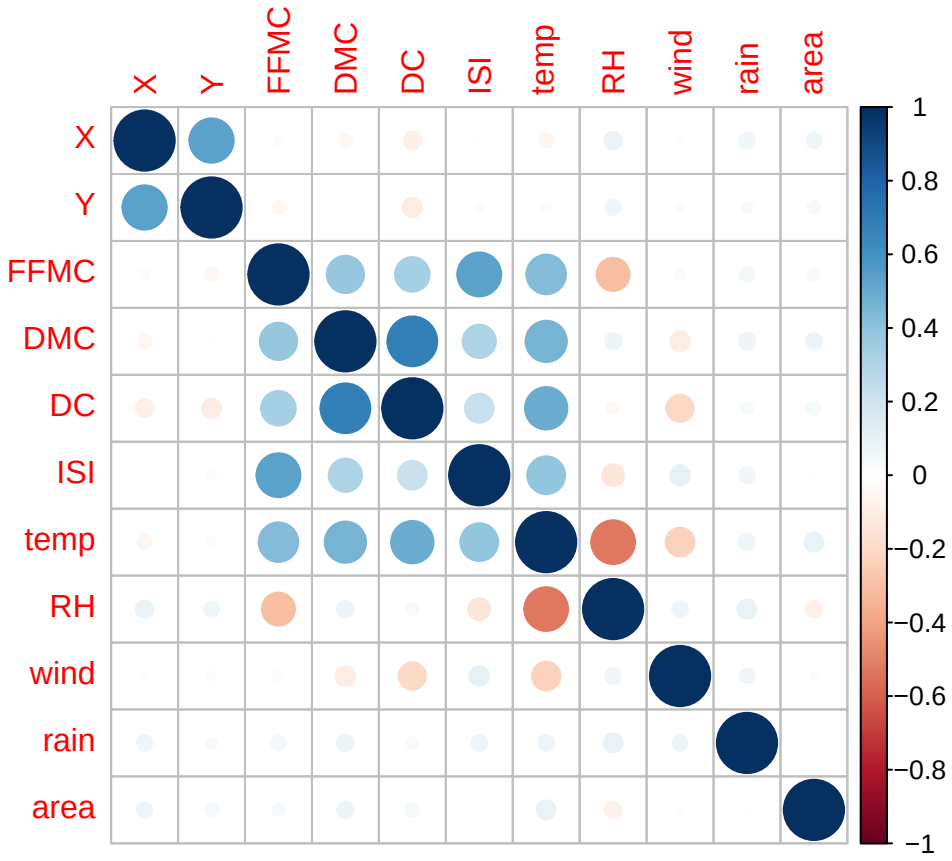

```
# In the code above, I added 1 to the area before taking the logarithm to avoid
# undefined values since log(0) is not defined.
```

```
# Load required package
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
# Compute the correlation matrix
correlationMatrix <- cor(data[,sapply(data, is.numeric)])

# Generate the correlation plot
corrplot(correlationMatrix, method = "circle")
```

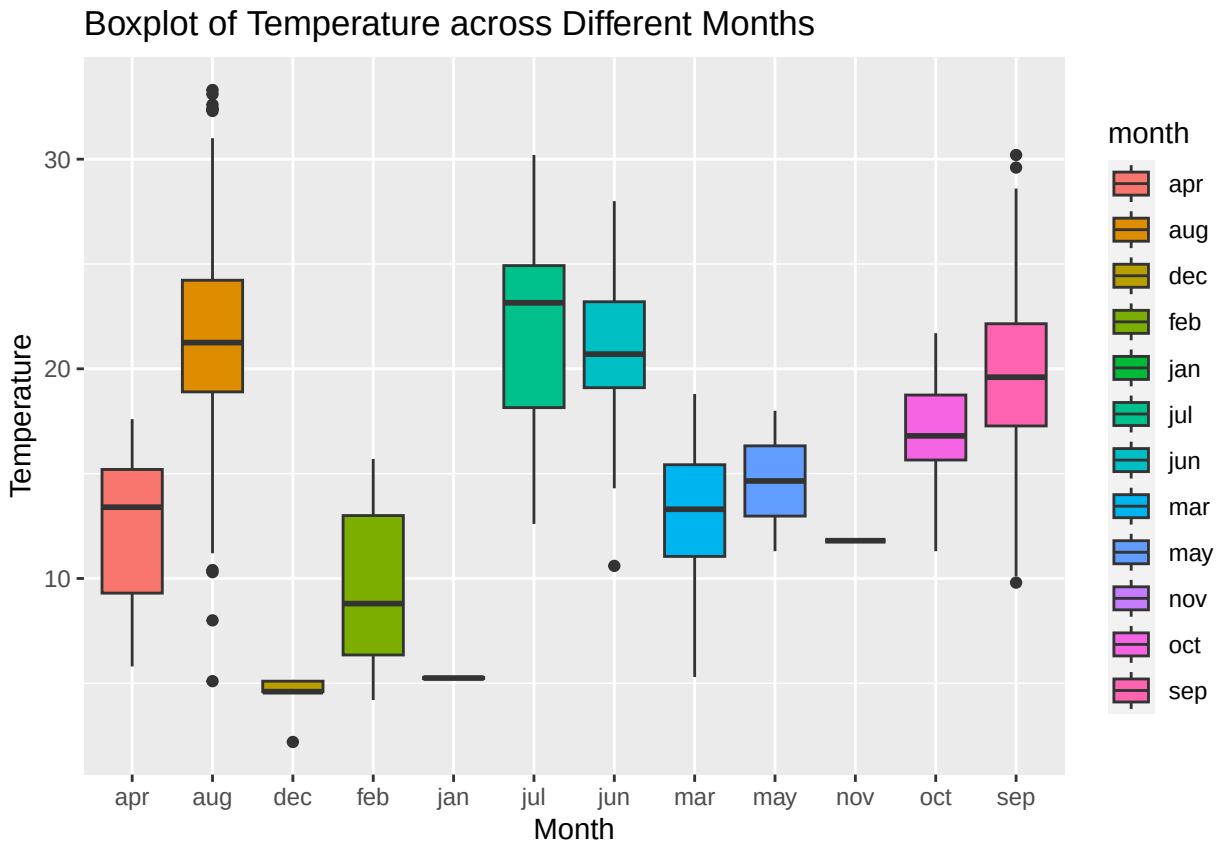



```
# Check for missing values
sapply(data, function(x) sum(is.na(x)))
```

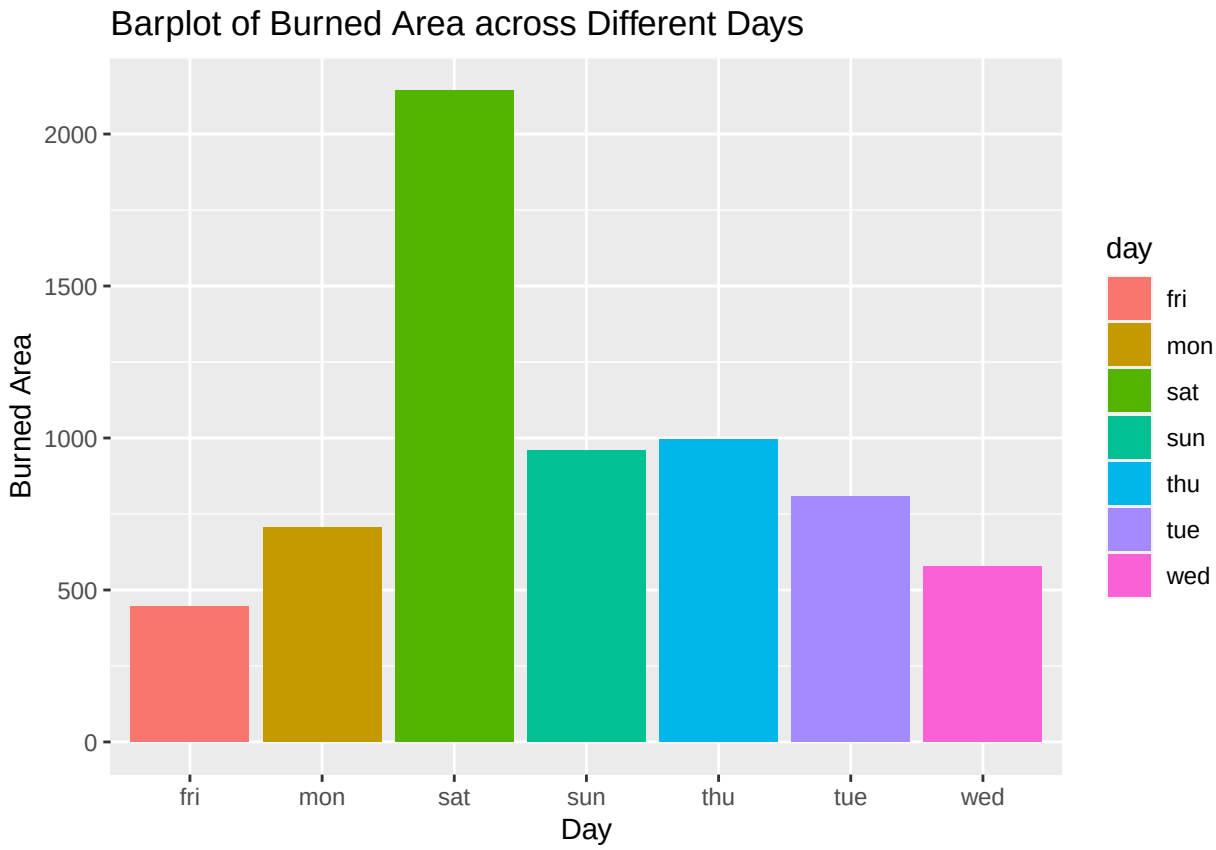
```
##     X     Y month   day  FFMC   DMC    DC   ISI  temp    RH  wind  rain  area
##     0     0     0     0     0     0     0     0     0     0     0     0     0
```

Comparisons Between Variables:

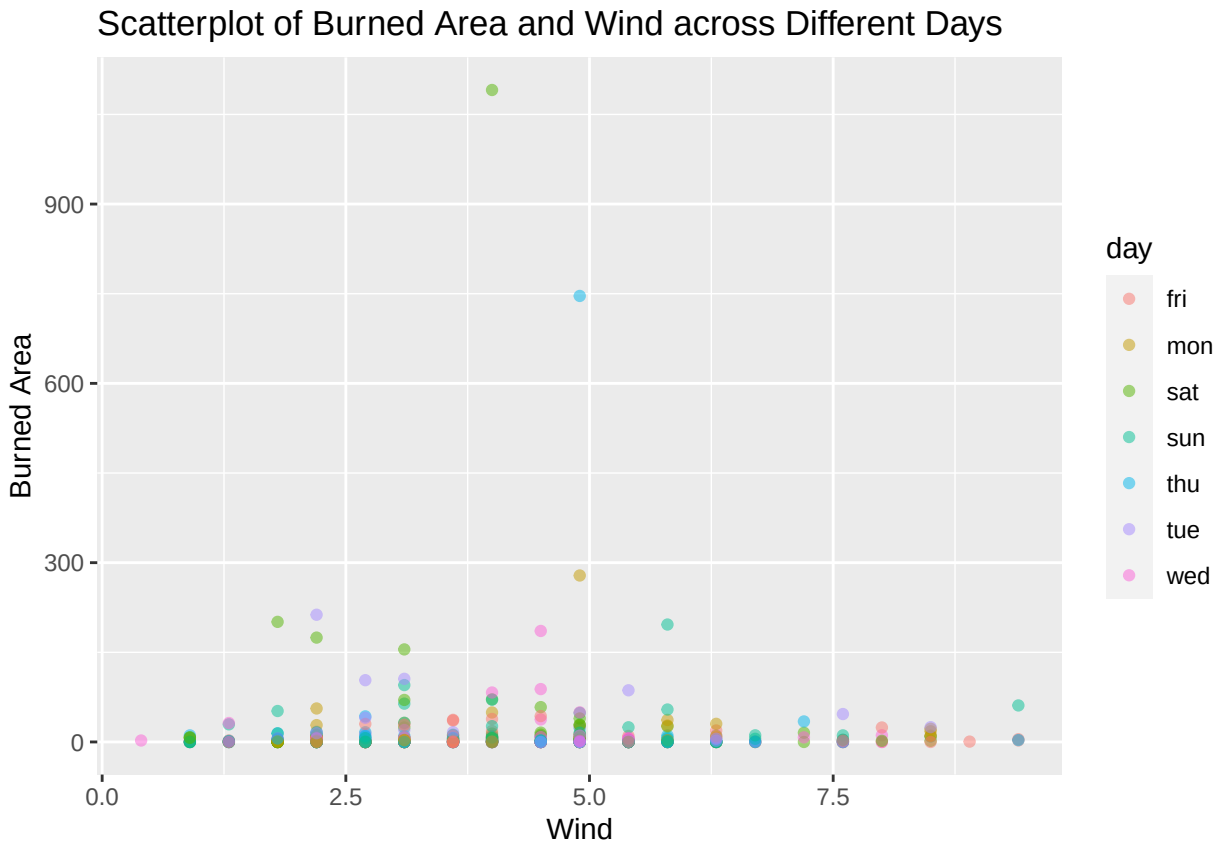
```
# Boxplot of Temperature across Different Months
ggplot(data, aes(x=month, y=temp, fill=month)) +
  geom_boxplot() +
  labs(title="Boxplot of Temperature across Different Months", x="Month", y="Temperature")
```




```
# Barplot of Burned Area across Different Days
ggplot(data, aes(x=day, y=area, fill=day)) +
  geom_bar(stat="identity") +
  labs(title="Barplot of Burned Area across Different Days", x="Day", y="Burned Area")
```




```
# Scatterplot of Burned Area and Wind with respect to different days of the week
ggplot(data, aes(x=wind, y=area, color=day)) +
  geom_point(alpha=0.5) +
  labs(title="Scatterplot of Burned Area and Wind across Different Days", x="Wind", y="Burned Area")
```



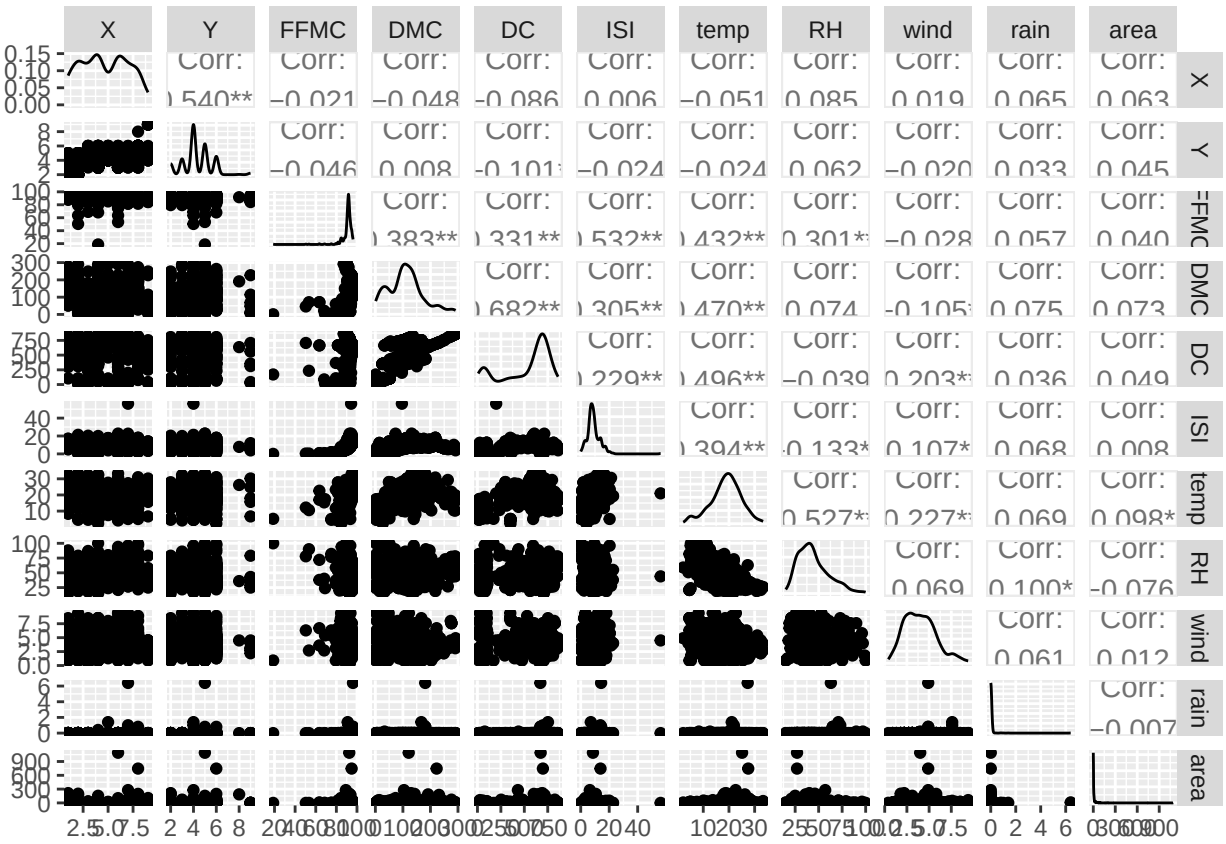
pair plots:

```
# Load required package
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

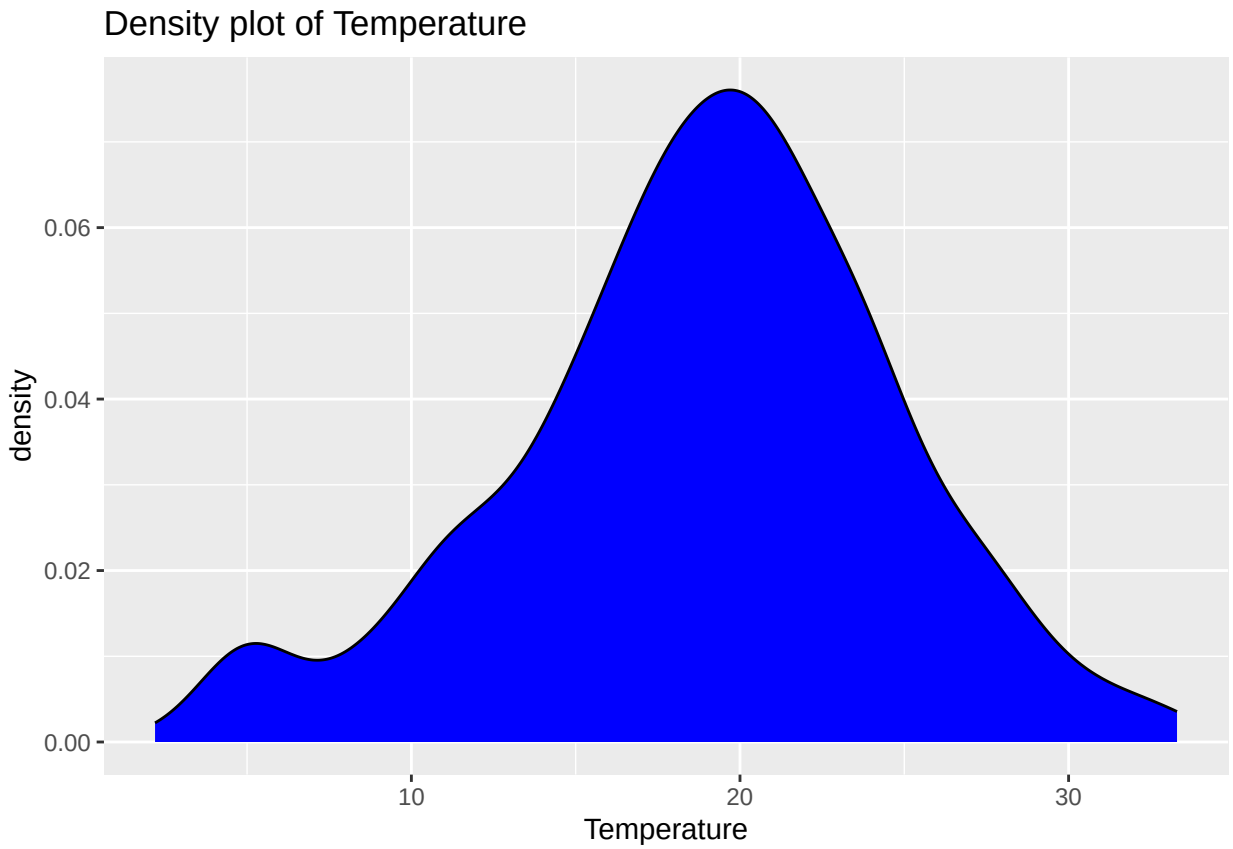
```
# Select numerical variables to avoid clutter
data_num <- data[, sapply(data, is.numeric)]

# Generate pairs plot
ggpairs(data_num)
```

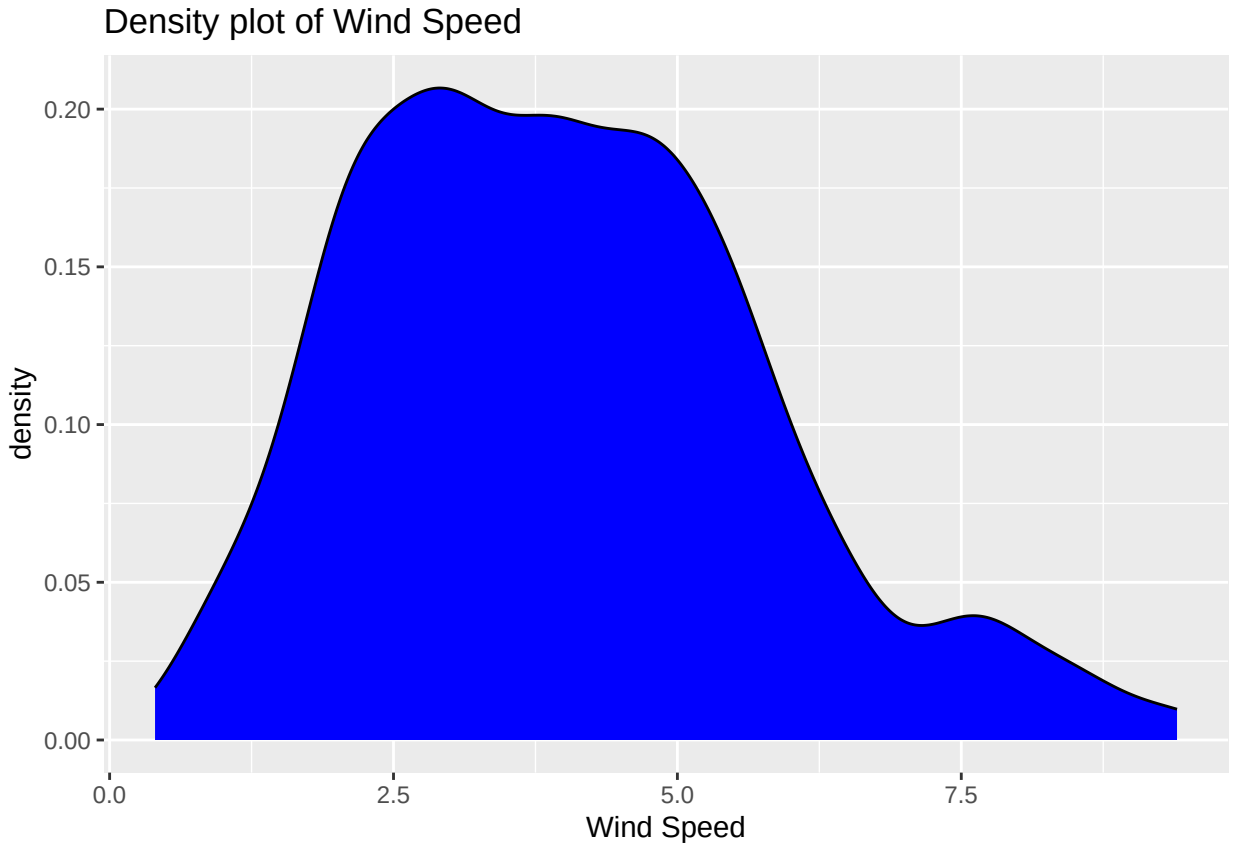


Density Plots:

```
# Density plot of Temperature
ggplot(data, aes(x=temp)) +
  geom_density(fill='blue') +
  labs(title="Density plot of Temperature", x="Temperature")
```

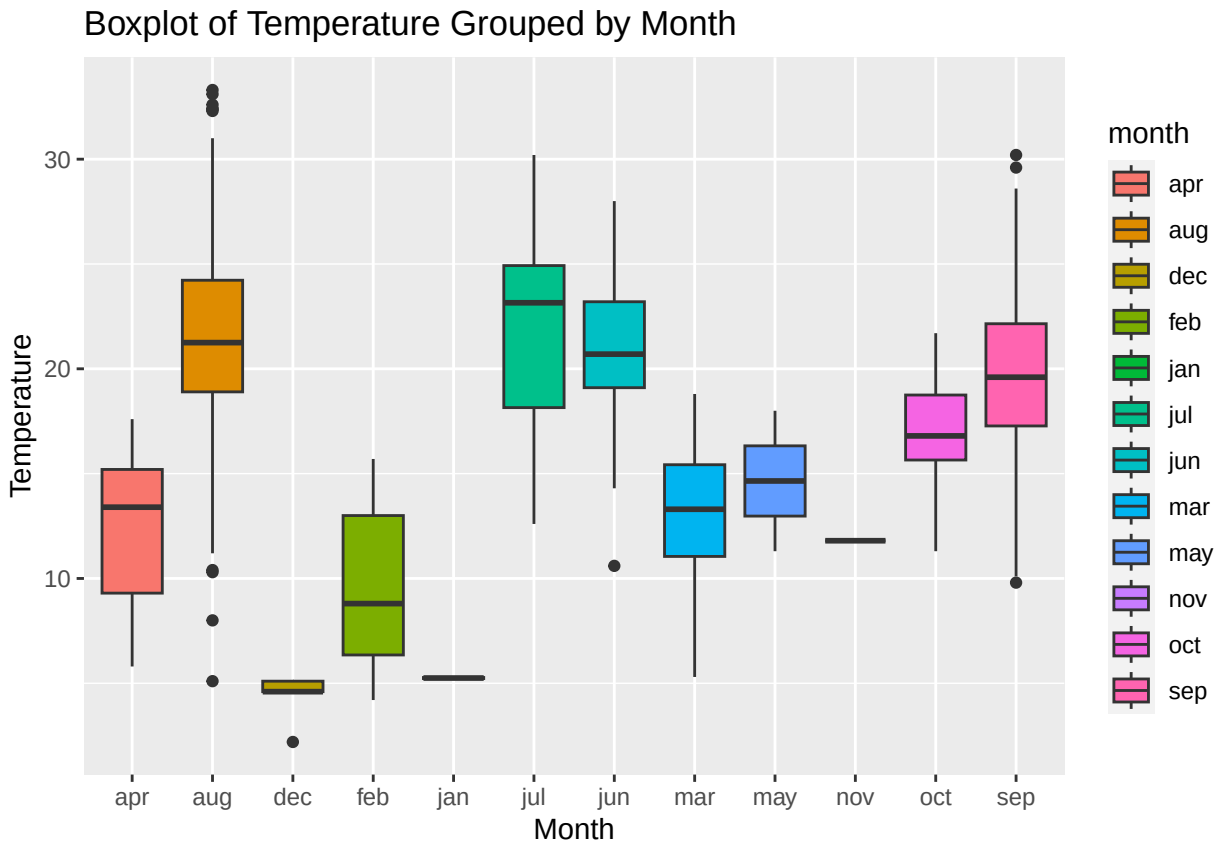



```
# Density plot of Wind Speed
ggplot(data, aes(x=wind)) +
  geom_density(fill='blue') +
  labs(title="Density plot of Wind Speed", x="Wind Speed")
```

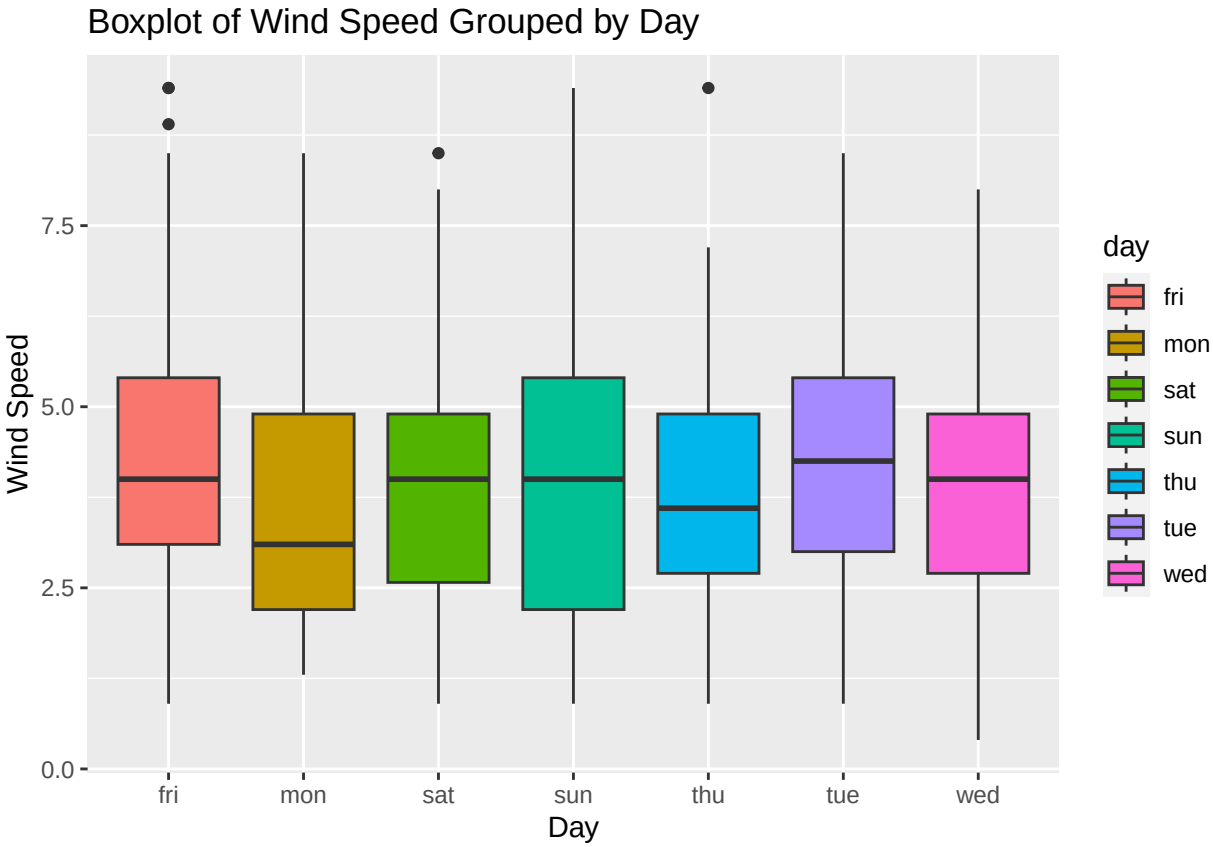


Interactions Between Categorical and Continuous Variables:

```
# Boxplot of temperature grouped by months
ggplot(data, aes(x=month, y=temp, fill=month)) +
  geom_boxplot() +
  labs(title="Boxplot of Temperature Grouped by Month", x="Month", y="Temperature")
```

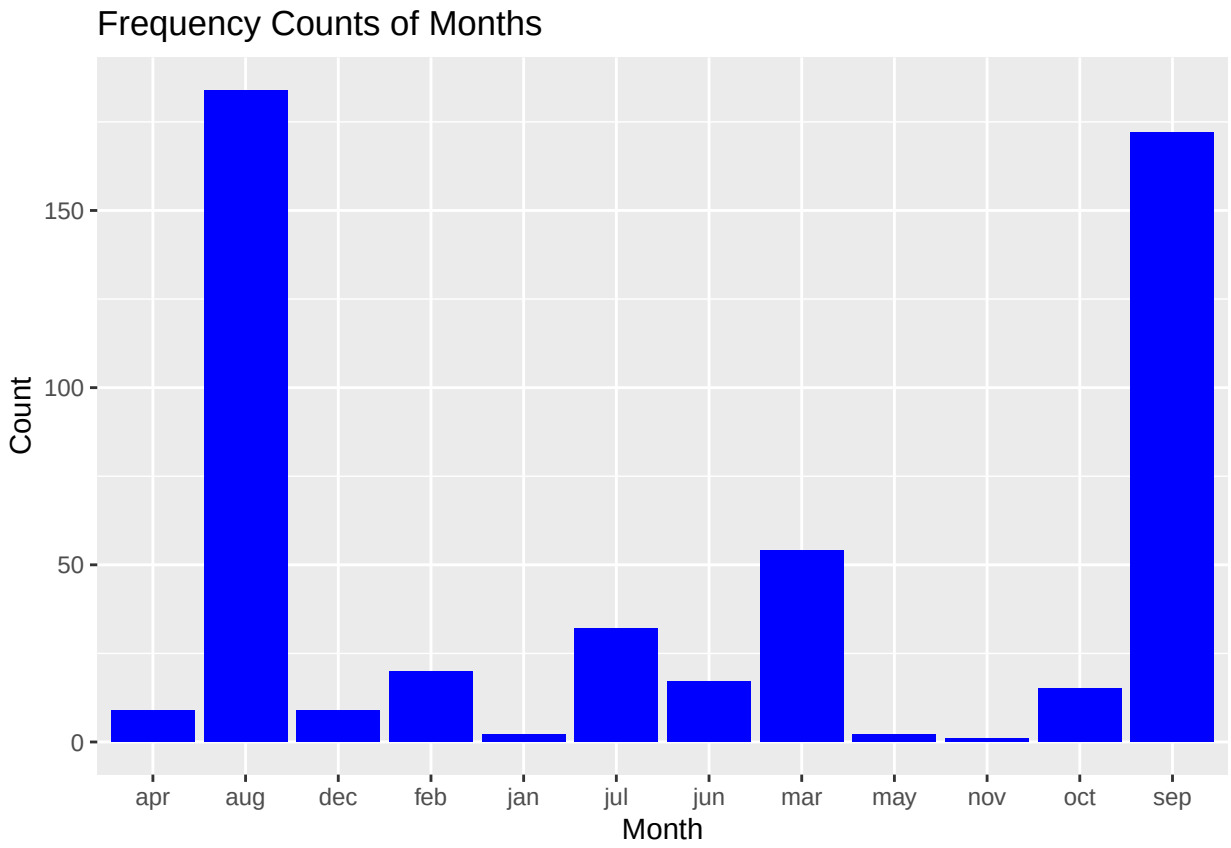



```r
# Boxplot of wind speed grouped by day
ggplot(data, aes(x=day, y=wind, fill=day)) +
  geom_boxplot() +
  labs(title="Boxplot of Wind Speed Grouped by Day", x="Day", y="Wind Speed")
```

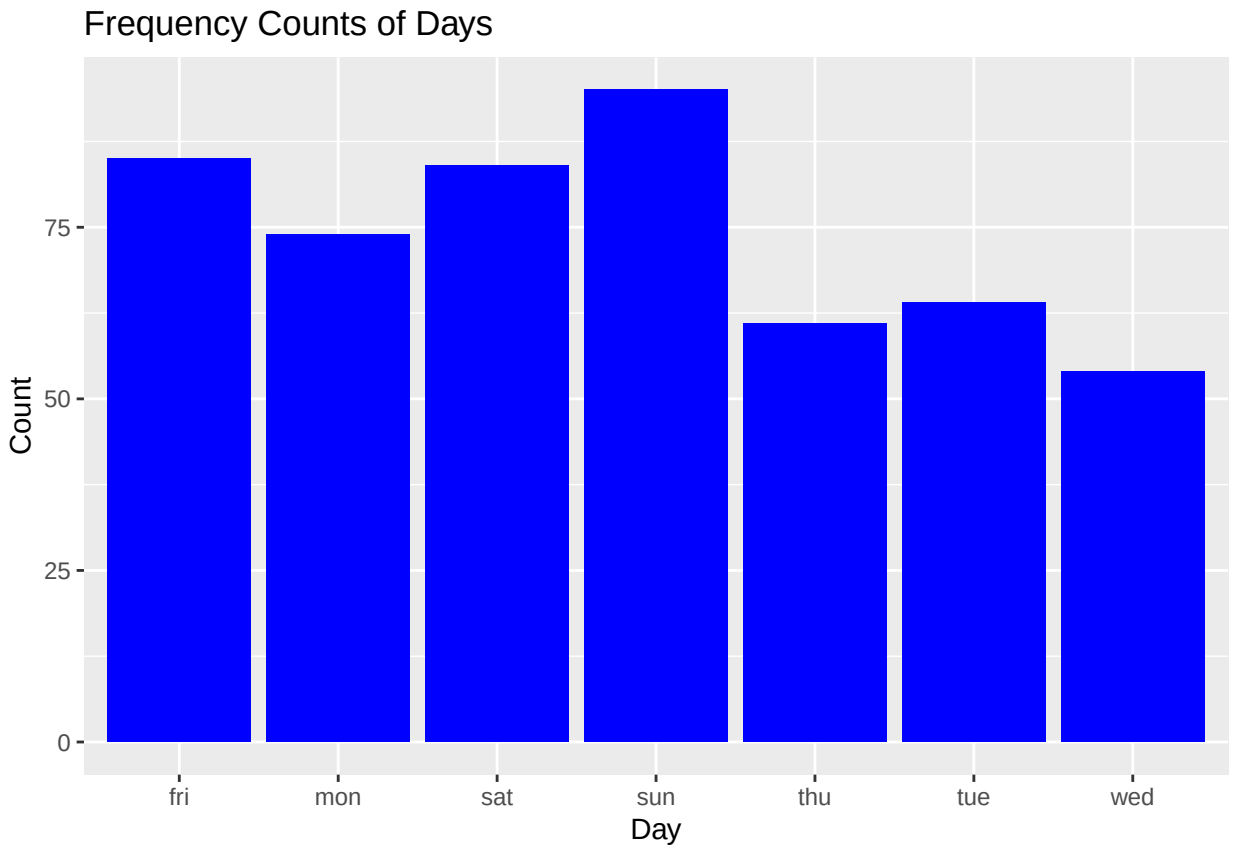


Frequency Counts of Categorical Variables:

```
# Frequency counts of months
ggplot(data, aes(x=month)) +
  geom_bar(fill='blue') +
  labs(title="Frequency Counts of Months", x="Month", y="Count")
```




```
# Frequency counts of days
ggplot(data, aes(x=day)) +
  geom_bar(fill='blue') +
  labs(title="Frequency Counts of Days", x="Day", y="Count")
```



ML

```
# Load necessary packages

library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

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

```r
library(dplyr)
library(caret)
```

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

```r
# Load the data
forest_fire_data <- read.csv("forestfires.csv")
df <- forest_fire_data
# Preprocess the data

# Convert categorical variables to factors
forest_fire_data$month <- as.factor(forest_fire_data$month)
forest_fire_data$day <- as.factor(forest_fire_data$day)
```

```r
# Log transform
df$area <- log(1 + df$area)

# Creating bins in the target, to stratify in train/test split
df$area_bin <- cut(df$area, breaks = 5)

# Dropping the 'area' column from the dataframe to create the predictors dataframe 'X'
X <- df[, !(names(df) %in% "area")]

# Creating the target variable 'y'
y <- df$area
```

split the dataset:

```r
# Set the seed for reproducibility
set.seed(99)

# Split the data into training and test sets
index <- createDataPartition(df$area_bin, p = 0.7, list = FALSE)
X_train <- X[index, ]
Y_train <- y[index]
X_test <- X[-index, ]
Y_test <- y[-index]

# Drop the 'area_bin' column from the training and test sets
X_train$area_bin <- NULL
X_test$area_bin <- NULL
```

We will compare a ML model (Random Forest or "RF") to our base model (Linear Regression) using the basic metrics Root Mean Square Error. The description of Random Forest will be disclosed later.

First we take a look at the base model, which is our linear regression:

```r
# Load necessary libraries
library(Metrics)
```

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

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

```r
# Fit the linear regression model
lr <- lm(Y_train ~ ., data = X_train)

# Make predictions and transform them back using exp
y_pred_test <- exp(predict(lr, newdata = X_test)) - 1

# Compute the RMSE for the test set
rmse_test <- sqrt(mean((exp(Y_test) - 1 - y_pred_test)^2))

# Compute the MAE for the test set
mae_test <- mean(abs(exp(Y_test) - 1 - y_pred_test))

# Print RMSE and MAE for the test set
print(paste0("Test set RMSE: ", round(rmse_test, 2)))
```

```
## [1] "Test set RMSE: 23.43"
```

```r
print(paste0("Test set MAE: ", round(mae_test, 2)))
```

```
## [1] "Test set MAE: 8.8"
```

```r
# Predict on the training set
y_pred_train <- exp(predict(lr, newdata = X_train)) - 1

# Compute the RMSE for the training set
rmse_train <- sqrt(mean((exp(Y_train) - 1 - y_pred_train)^2))

# Print RMSE for the training set
print(paste0("Train RMSE: ", round(rmse_train, 2)))
```

```
## [1] "Train RMSE: 75.02"
```

Then we will use RF.

First we use grid search to do the hyperparameter searching:

```r
# Load necessary libraries
library(caret)
library(randomForest)
set.seed(22)
# Set up train control
ctrl <- trainControl(method = "cv", number = 10)

# Define the grid for the tuning parameters
grid <- expand.grid(
  mtry = c(2, 3, 4),  # Number of variables available for splitting at each tree node
  splitrule = "variance",
  min.node.size = c(1, 3, 5)  # Minimum number of observations that must exist in a node in order for a
```

```
)

# Combine X_train and Y_train into a single dataframe
train_data <- cbind(Y_train, X_train)

# Run the model with the combined dataframe
model <- train(
  Y_train ~ .,
  data = train_data,
  method = "ranger",
  trControl = ctrl,
  tuneGrid = grid
)
#rf_model <- randomForest(target ~ ., data = df, ntree = 500)

# Print the best tuning parameters
print(model$bestTune)
```

```
##   mtry splitrule min.node.size
## 2    2  variance             3
```

```
#print(model)
```

In the context of Random Forests, mtry is a tuning parameter that specifies the number of variables randomly sampled as candidates at each split.

The splitrule parameter in a Random Forest algorithm specifies the rule or metric used to decide on the best split at each node in the decision tree.

There are several options for the splitrule parameter:

"gini": Uses the Gini impurity to measure the quality of a split. The Gini impurity measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen. This is the default for classification tasks.

"variance": Used for regression tasks. It measures the quality of a split by computing the reduction in variance of the response variable.

"auc": It measures the quality of a split by computing the Area Under the Receiver Operating Characteristic Curve (AUC-ROC).

The grid search gives us "variance".

The min.node.size parameter in a Random Forest algorithm determines the minimum number of observations that must exist in a node for a split to be attempted.

In simple terms, if you set min.node.size to 5, for example, it means that any subset of your data with less than 5 instances won't be split any further. This is a way to control overfitting because it prevents the model from learning too much detail from the training data.

The we use the tuned hyperparameters to build our Random Forest Model:

```
library(randomForest)
library(Metrics)

# Fit the Random Forest Model
rf <- randomForest(Y_train ~ ., data = train_data, ntree = 500, mtry = 2,
```

```
                    nodesize = 3, maxnodes = 10)
test_data <- rbind(X_test, Y_test)


# Make predictions
pred_train <- exp(predict(rf, train_data)) - 1
pred_test <- exp(predict(rf, test_data)) - 1

# Compute RMSE
rmse_train <- rmse(pred_train, exp(train_data$Y_train) - 1)
rmse_test <- rmse(pred_test, exp(Y_test) - 1)
```

```
## Warning in actual - predicted: longer object length is not a multiple of shorter
## object length
```

```
print(paste("Train RMSE: ", rmse_train))
```

```
## [1] "Train RMSE:  75.084456018409"
```

```
print(paste("Test RMSE: ", rmse_test))
```

```
## [1] "Test RMSE:  22.9753712092193"
```

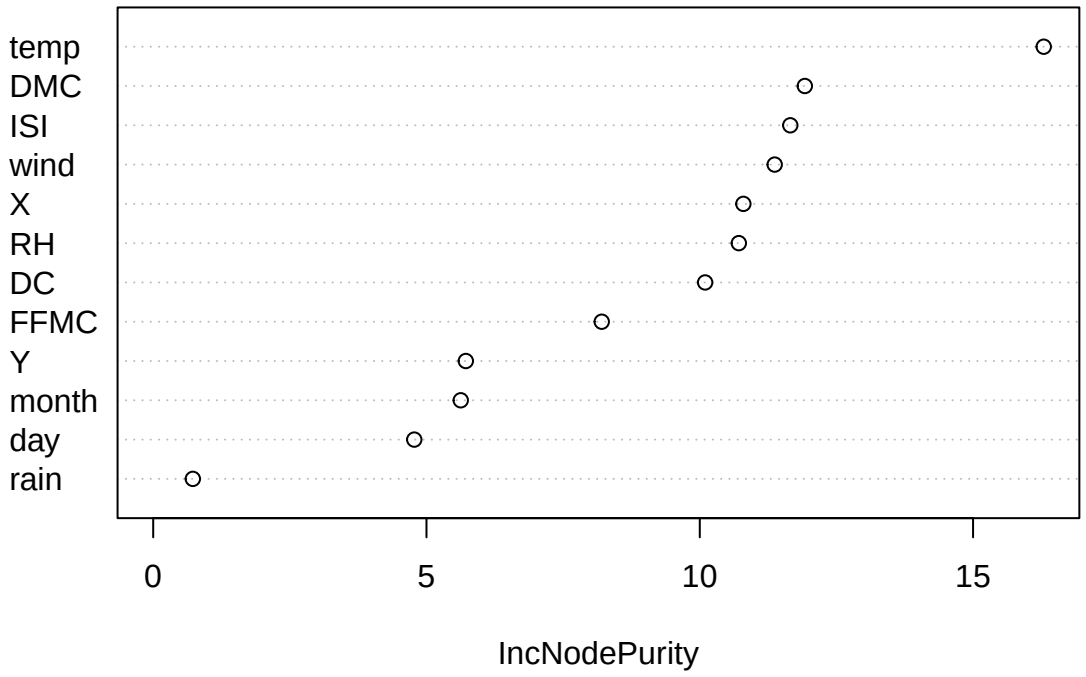
```
# Compute MAE
mae_test <- mae(pred_test, exp(Y_test) - 1)
```

```
## Warning in actual - predicted: longer object length is not a multiple of shorter
## object length
```

```
print(paste("Test MAE: ", mae_test))
```

```
## [1] "Test MAE:  8.43178592005806"
```

```
# Feature importance
importance <- importance(rf)
varImpPlot(rf)
```



The results show that the RMSE from RF is slightly smaller than that of Linear Regression. The important variables for predicting forest fire are shown above.