capstone

Brendan Pham

2023-10-30

R Markdown

STFWI, spatial, temporal, and four fwi indices

STM uses spatial, temporal, and weather variables

FWI uses the fwi indices

M uses the three weather variables

```
# Summary of the dataset
summary(forestfires)
```

```
##        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
```

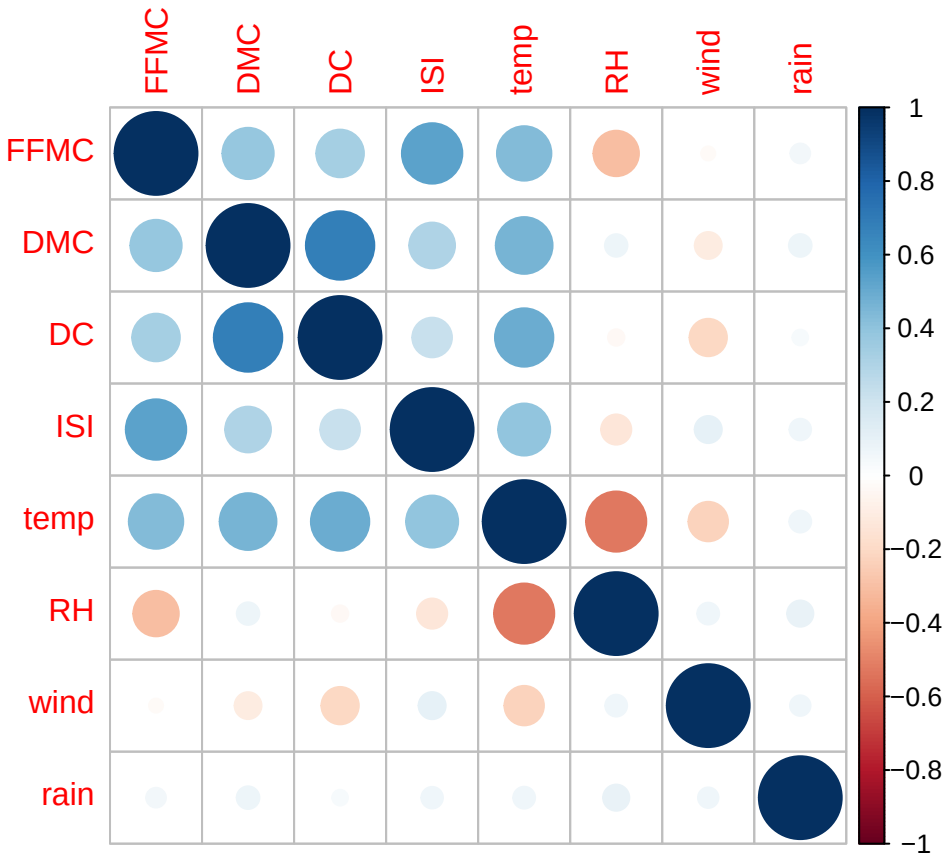
```r
str(forestfires)
```

```
## spc_tbl_ [517 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ X    : num [1:517] 7 7 7 8 8 8 8 8 8 7 ...
##  $ Y    : num [1:517] 5 4 4 6 6 6 6 6 6 5 ...
##  $ month: chr [1:517] "mar" "oct" "oct" "mar" ...
##  $ day  : chr [1:517] "fri" "tue" "sat" "fri" ...
##  $ FFMC : num [1:517] 86.2 90.6 90.6 91.7 89.3 92.3 92.3 91.5 91 92.5 ...
##  $ DMC  : num [1:517] 26.2 35.4 43.7 33.3 51.3 ...
##  $ DC   : num [1:517] 94.3 669.1 686.9 77.5 102.2 ...
##  $ ISI  : num [1:517] 5.1 6.7 6.7 9 9.6 14.7 8.5 10.7 7 7.1 ...
##  $ temp : num [1:517] 8.2 18 14.6 8.3 11.4 22.2 24.1 8 13.1 22.8 ...
##  $ RH   : num [1:517] 51 33 33 97 99 29 27 86 63 40 ...
##  $ wind : num [1:517] 6.7 0.9 1.3 4 1.8 5.4 3.1 2.2 5.4 4 ...
##  $ rain : num [1:517] 0 0 0 0.2 0 0 0 0 0 0 ...
##  $ area : num [1:517] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   X = col_double(),
##   ..   Y = col_double(),
##   ..   month = col_character(),
##   ..   day = col_character(),
##   ..   FFMC = col_double(),
##   ..   DMC = col_double(),
##   ..   DC = col_double(),
##   ..   ISI = col_double(),
##   ..   temp = col_double(),
##   ..   RH = col_double(),
##   ..   wind = col_double(),
##   ..   rain = col_double(),
##   ..   area = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

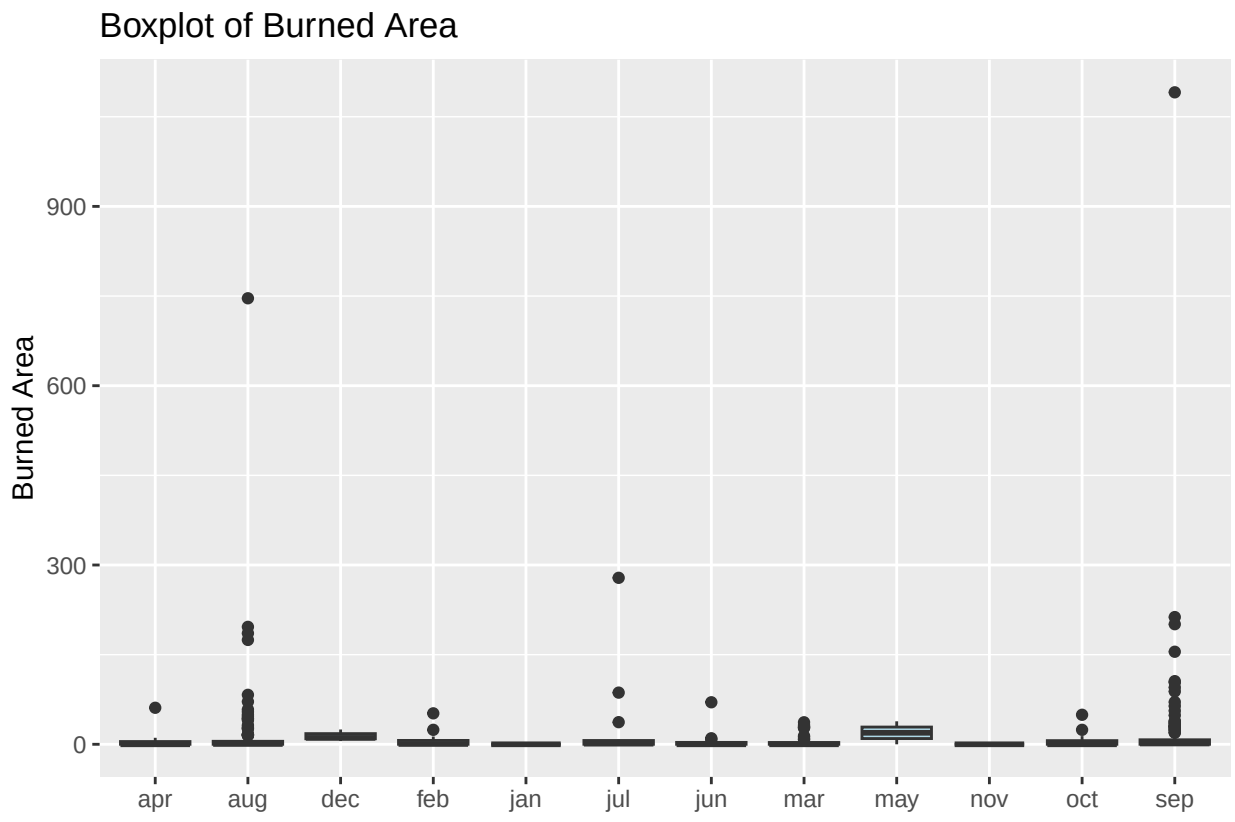
```r
head(forestfires)
```

```
## # A tibble: 6 x 13
##       X     Y month day    FFMC   DMC    DC   ISI  temp    RH  wind  rain  area
##   <dbl> <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     7     5 mar   fri    86.2  26.2  94.3   5.1   8.2    51   6.7   0       0
## 2     7     4 oct   tue    90.6  35.4 669.    6.7  18      33   0.9   0       0
## 3     7     4 oct   sat    90.6  43.7 687.    6.7  14.6    33   1.3   0       0
## 4     8     6 mar   fri    91.7  33.3  77.5   9     8.3    97   4     0.2     0
## 5     8     6 mar   sun    89.3  51.3 102.    9.6  11.4    99   1.8   0       0
## 6     8     6 aug   sun    92.3  85.3 488    14.7  22.2    29   5.4   0       0
```

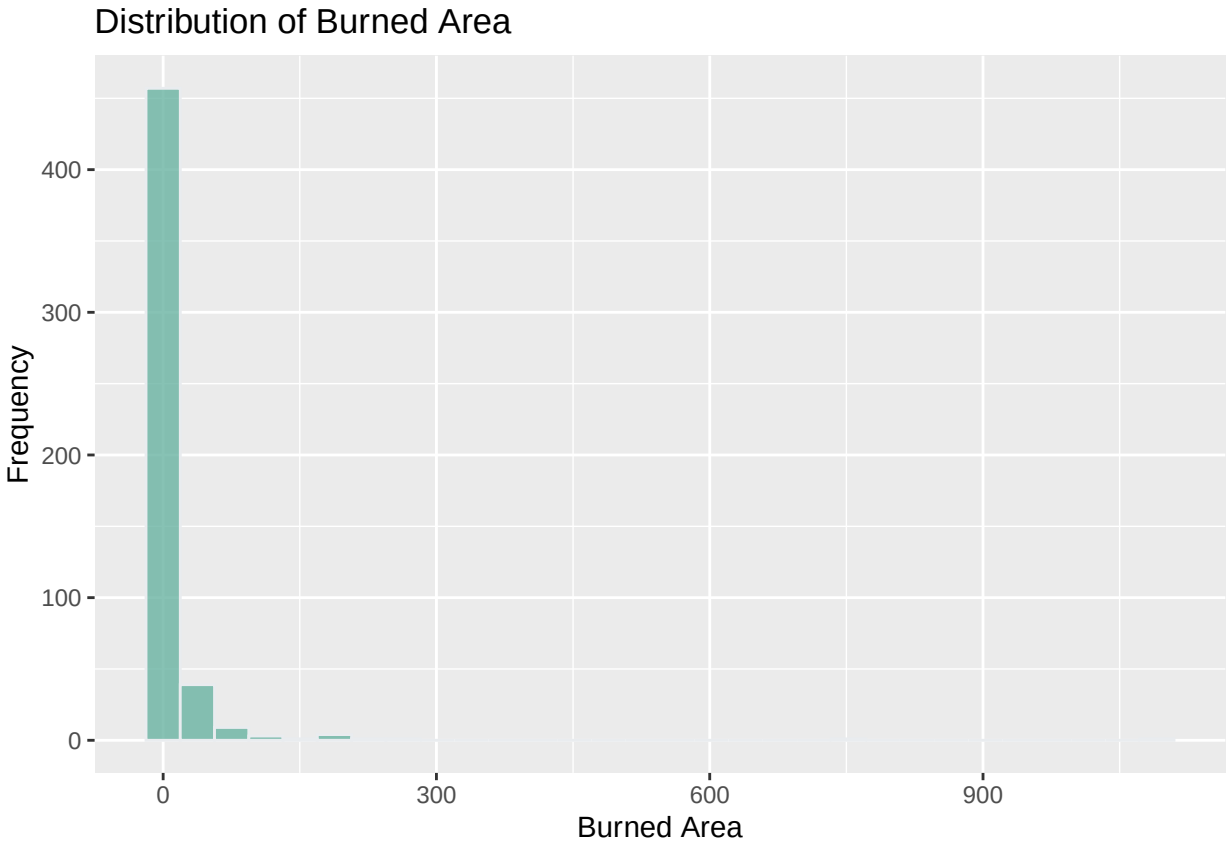
```r
# Visualize the d->istribution of the target variable
# Log transformation (adding 1 to handle zero values)


# Check feature correlations
correlation_matrix <- cor(forestfires[, c("FFMC", "DMC", "DC", "ISI", "temp", "RH", "wind", "rain")])
corrplot(correlation_matrix)
```



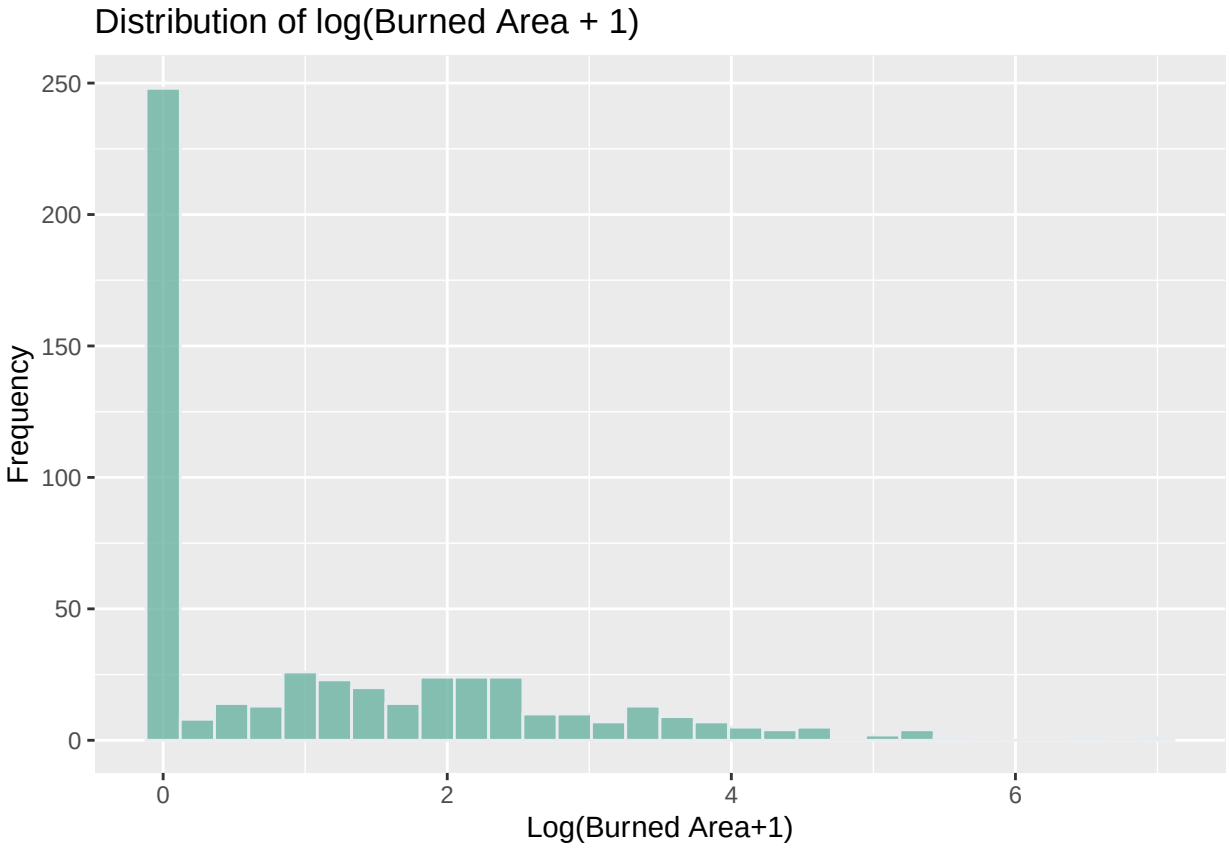

```
ggplot(forestfires, aes(x = month, y = area)) +
  geom_boxplot(fill = "lightblue") +
  labs(title = "Boxplot of Burned Area", x = "", y = "Burned Area")
```



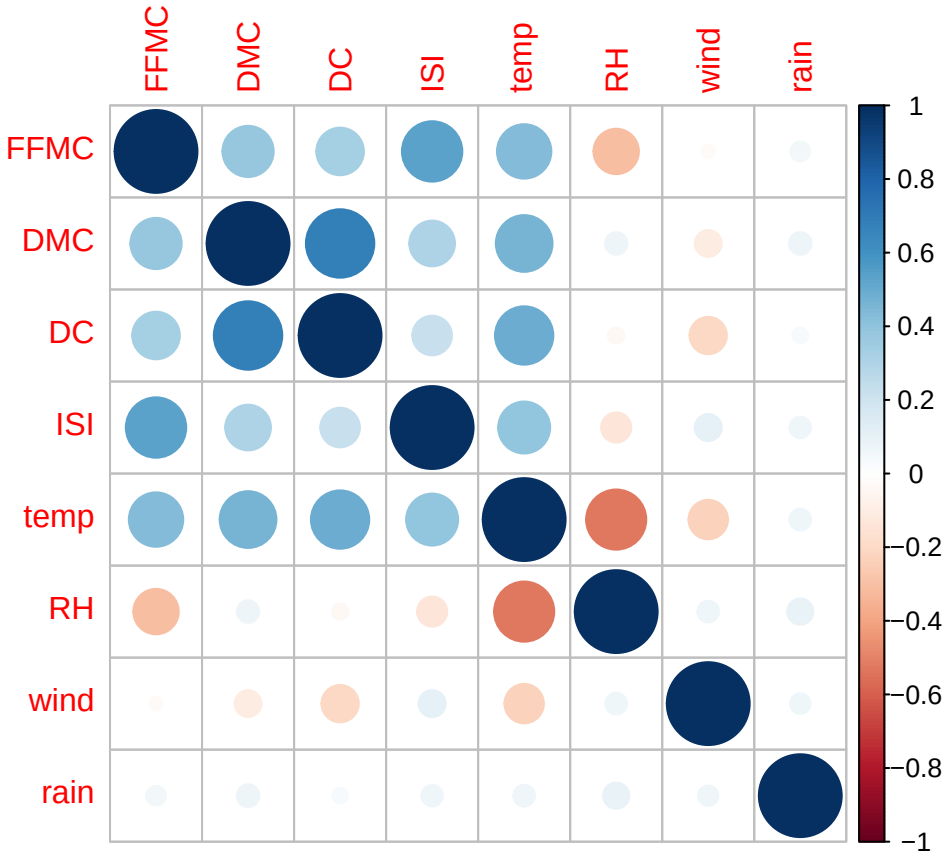

```
ggplot(forestfires,aes(x = area)) +geom_histogram(fill="#69b3a2", color="#e9ecef", alpha=0.8, bins = 30)
  labs(title = "Distribution of Burned Area", x = "Burned Area", y = "Frequency")
```



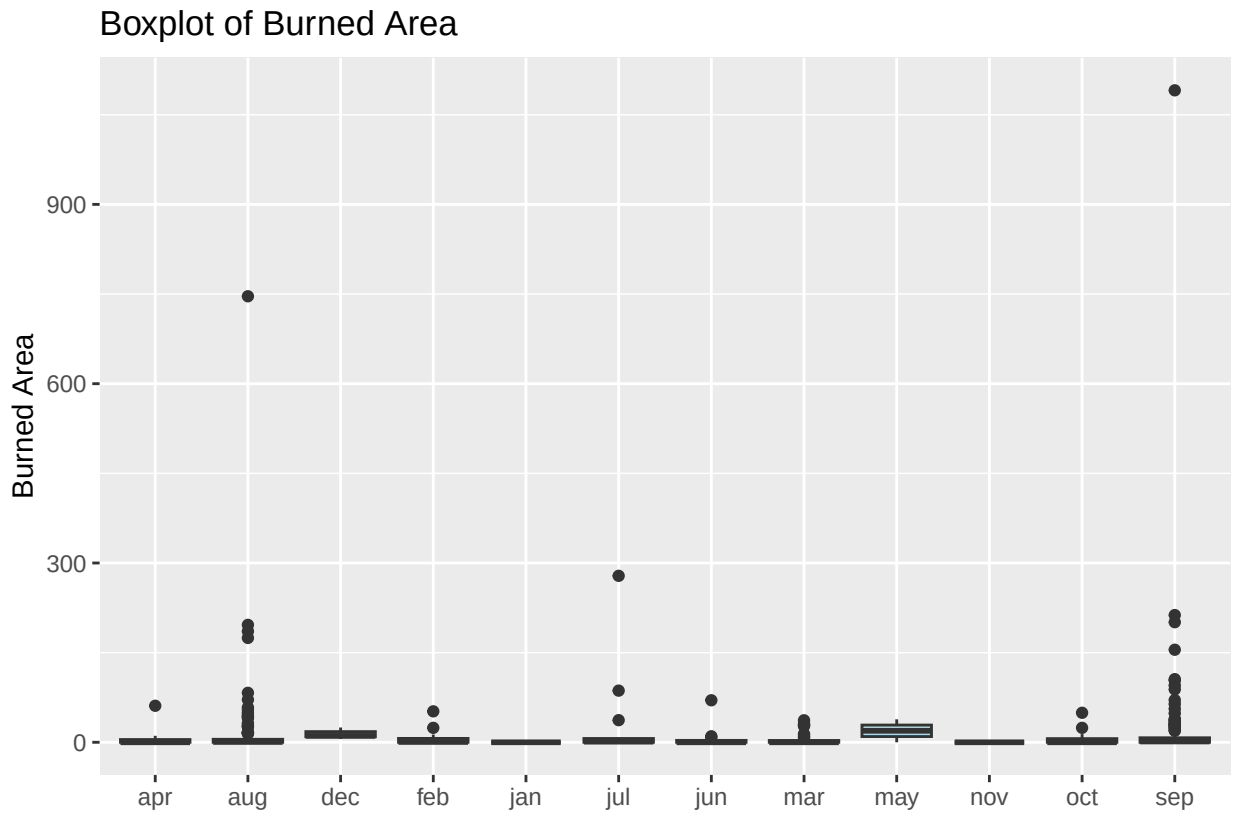

```
ggplot(forestfires,aes(x = log(area+1))) + geom_histogram(fill="#69b3a2", color="#e9ecef", alpha=0.8, bi
    labs(title = "Distribution of log(Burned Area + 1)", x = "Log(Burned Area+1)", y = "Frequency")
```




```r
# Check feature correlations
correlation_matrix <- cor(forestfires[, c("FFMC", "DMC", "DC", "ISI", "temp", "RH", "wind", "rain")])
corrplot(correlation_matrix)
```

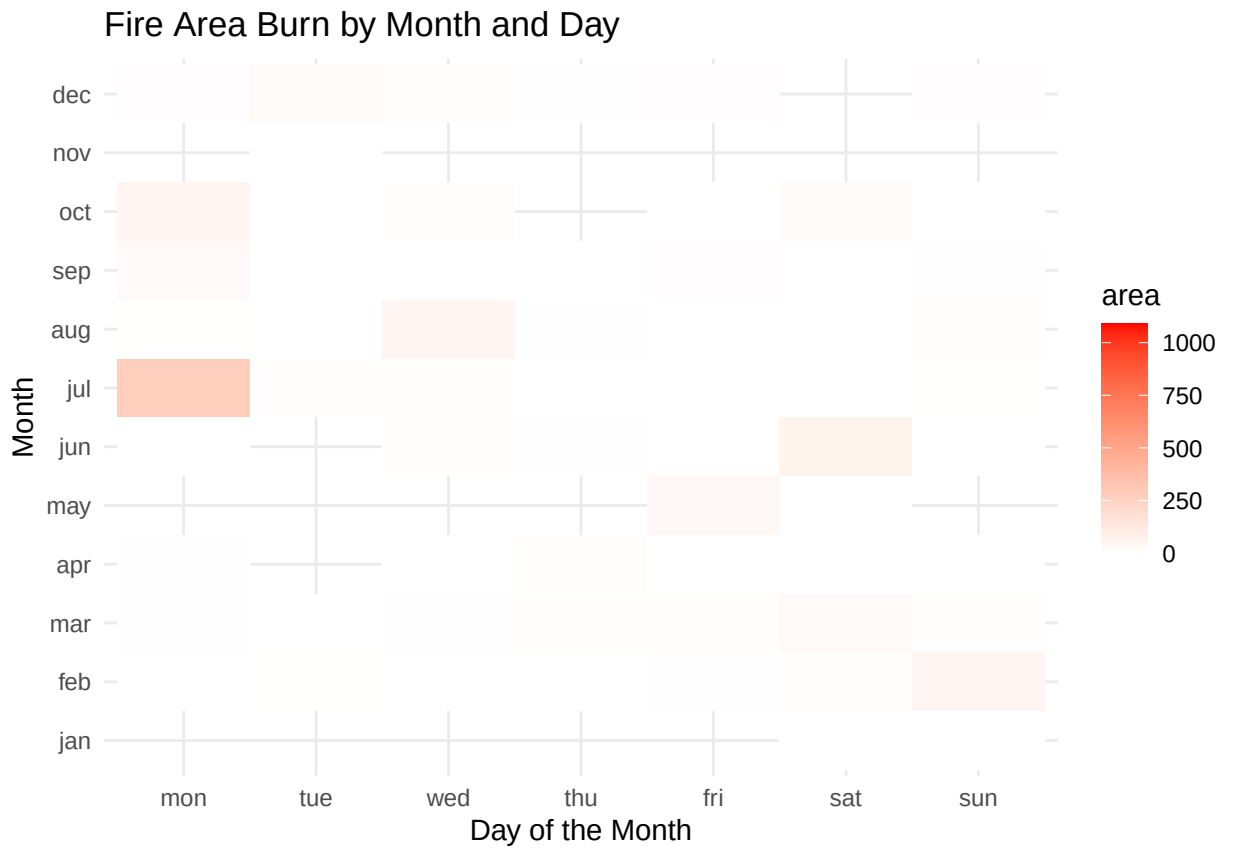



```
ggplot(forestfires, aes(x = month, y = area)) +
  geom_boxplot(fill = "lightblue") +
  labs(title = "Boxplot of Burned Area", x = "", y = "Burned Area")
```




```
forestfires$month <- factor(forestfires$month, levels =
  c("jan","feb","mar",
  "apr", "may", "jun", "jul",  "aug",
  "sep", "oct", "nov", "dec"))
forestfires$day <- factor(forestfires$day, levels =
  c("mon","tue","wed","thu","fri","sat","sun"))

forestfires$season = 0
for (i in 1:length(forestfires$month)) {
  if (forestfires$month[i] %in% c("dec", "jan", "feb")) {
     forestfires$season[i] = "winter"
 } else if (forestfires$month[i] %in% c("mar", "apr", "may")) {
     forestfires$season[i] = "spring"
  }else if (forestfires$month[i] %in% c("jun", "jul", "aug")) {
     forestfires$season[i] = "summer"
  }else  forestfires$season[i] = "autumn"
}

forestfires$season = as.factor(forestfires$season)



# Create a heatmap using ggplot2
ggplot(forestfires, aes(x = day, y = month, fill = area)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
```

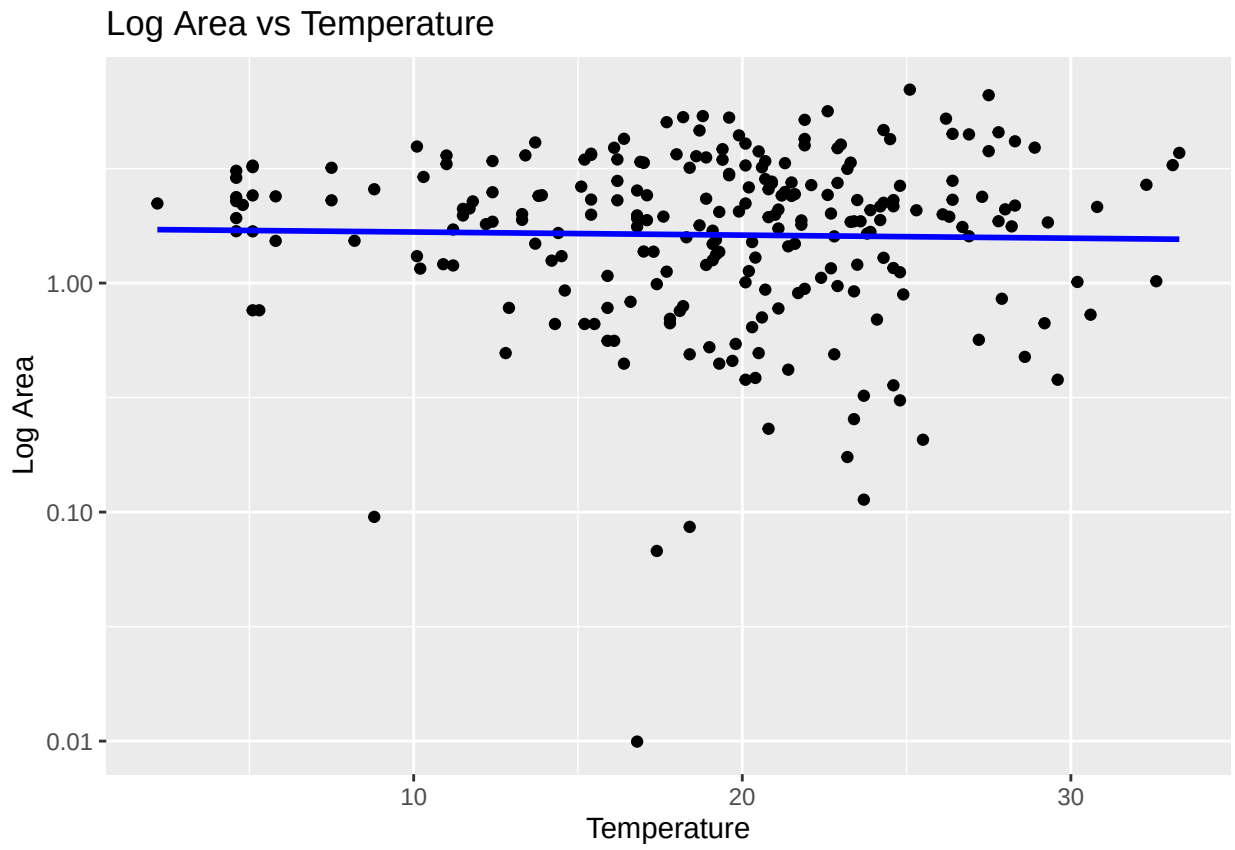
```
  labs(title = "Fire Area Burn by Month and Day", x = "Day of the Month", y = "Month") +
  theme_minimal()
```




```
ggplot(forestfires, aes(x = temp, y = log(area))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear regression line
  labs(title = "Log Area vs Temperature",
       x = "Temperature",
       y = "Log Area") +
  scale_y_log10()
```

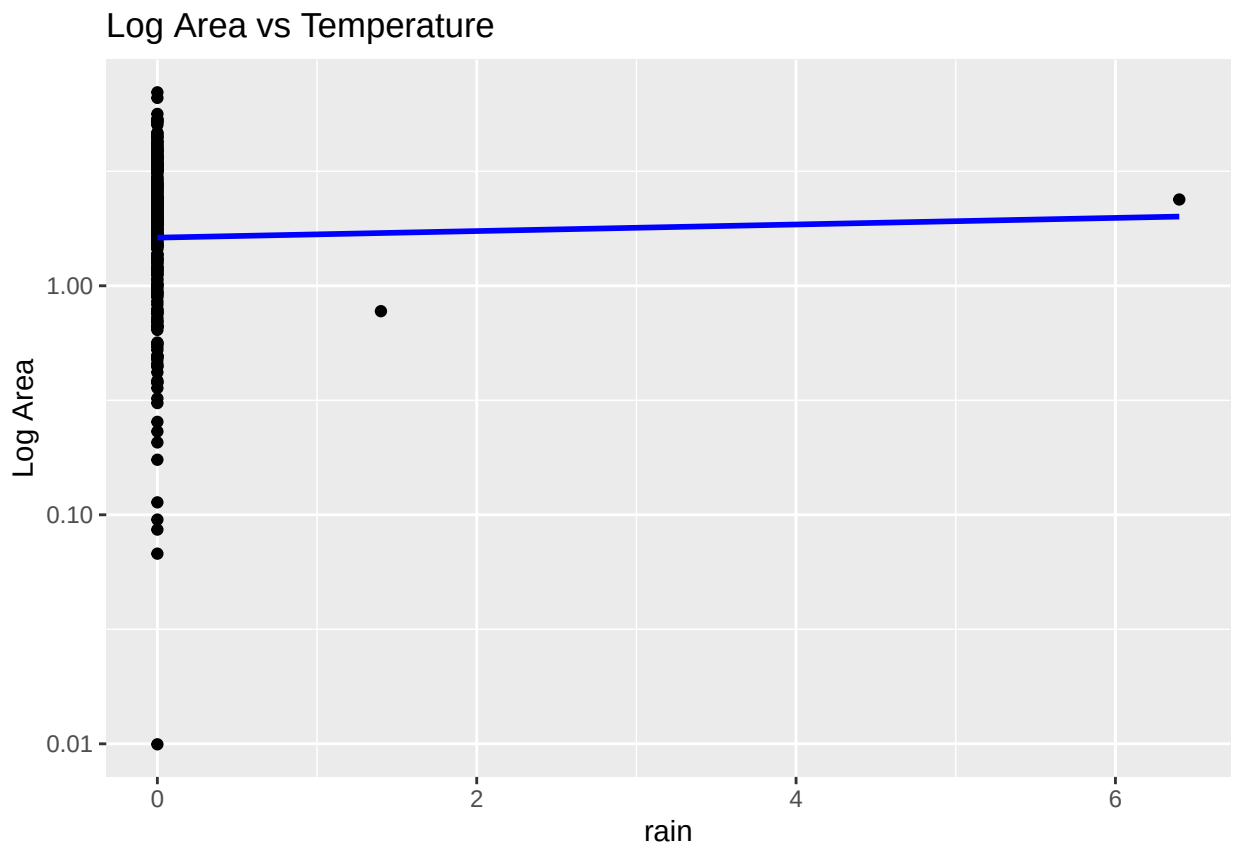
```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 274 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 274 rows containing missing values (‘geom_point()‘).
```



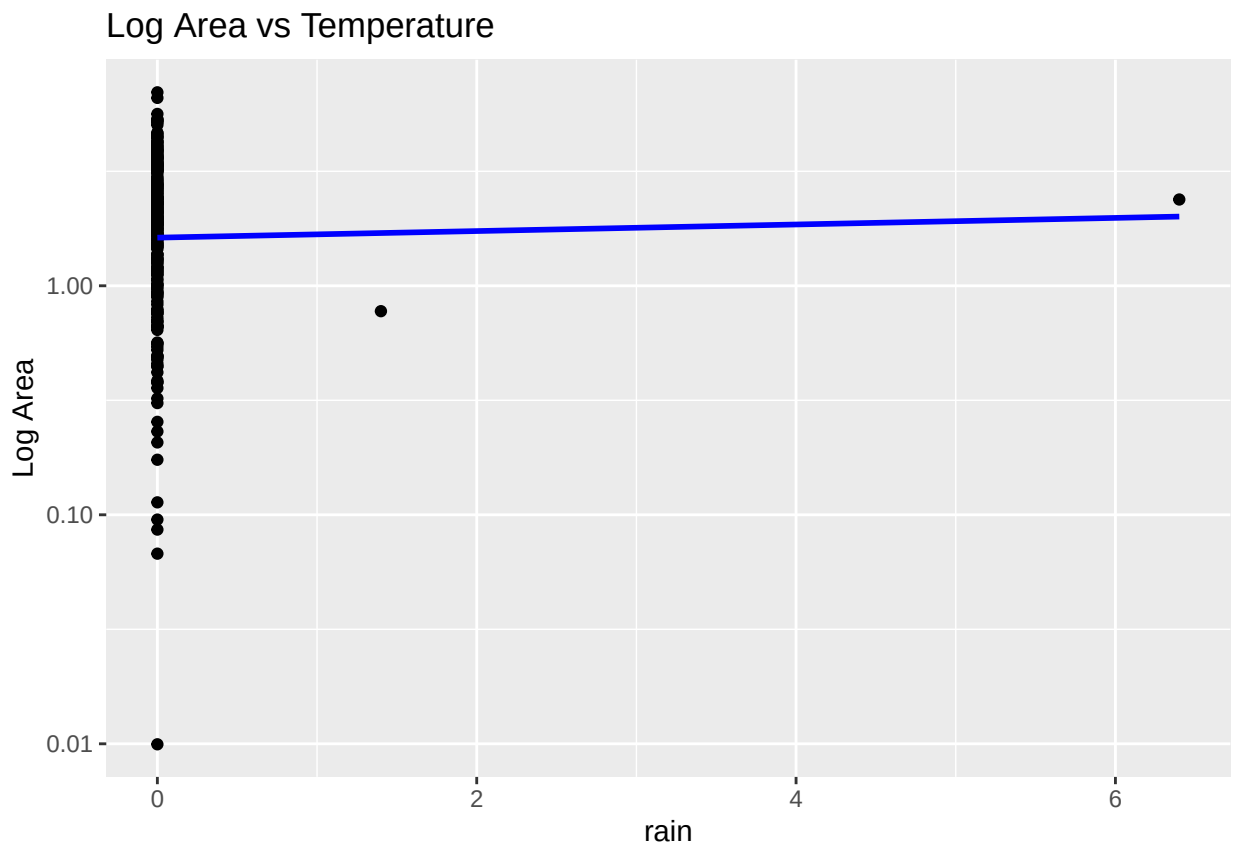

```
ggplot(forestfires, aes(x = rain, y = log(area))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear regression line
  labs(title = "Log Area vs Temperature",
       x = "rain",
       y = "Log Area") +
  scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## ‘geom_smooth()‘ using formula = ’y ~ x’

## Warning: Removed 274 rows containing non-finite values (‘stat_smooth()‘).

## Warning: Removed 274 rows containing missing values (‘geom_point()‘).
```



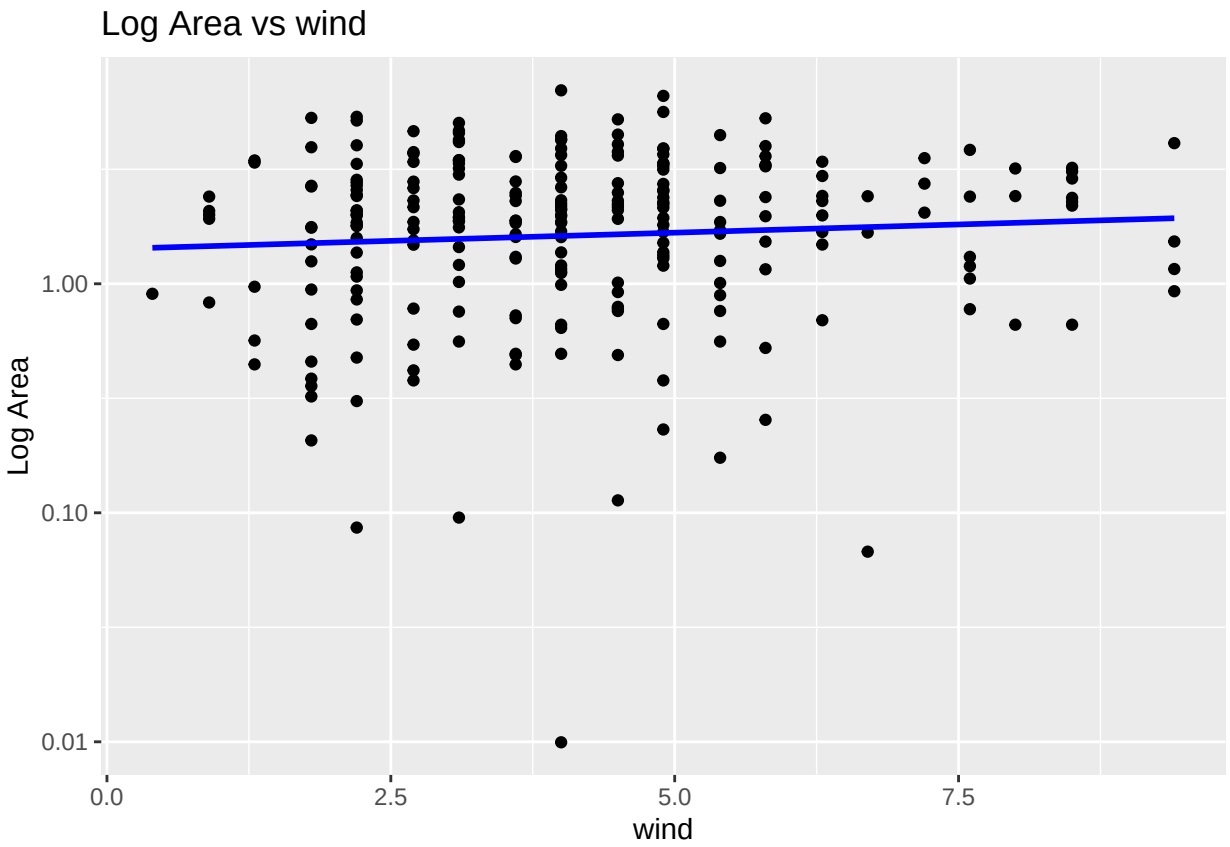

```
ggplot(forestfires, aes(x = rain, y = log(area))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear regression line
  labs(title = "Log Area vs Temperature",
       x = "rain",
       y = "Log Area") +
  scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 274 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 274 rows containing missing values (`geom_point()`).
```



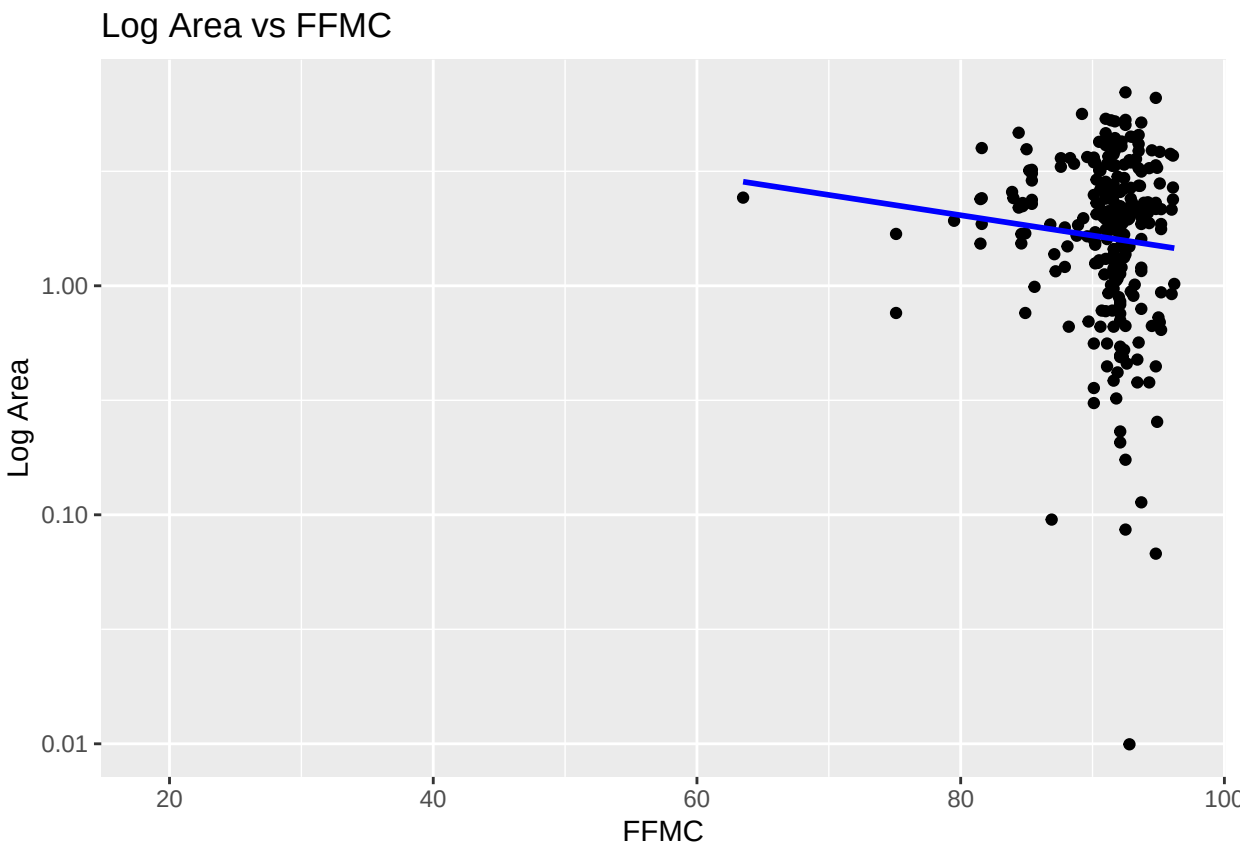

```
ggplot(forestfires, aes(x = wind, y = log(area))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear regression line
  labs(title = "Log Area vs wind",
       x = "wind",
       y = "Log Area") +
  scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 274 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 274 rows containing missing values (`geom_point()`).
```



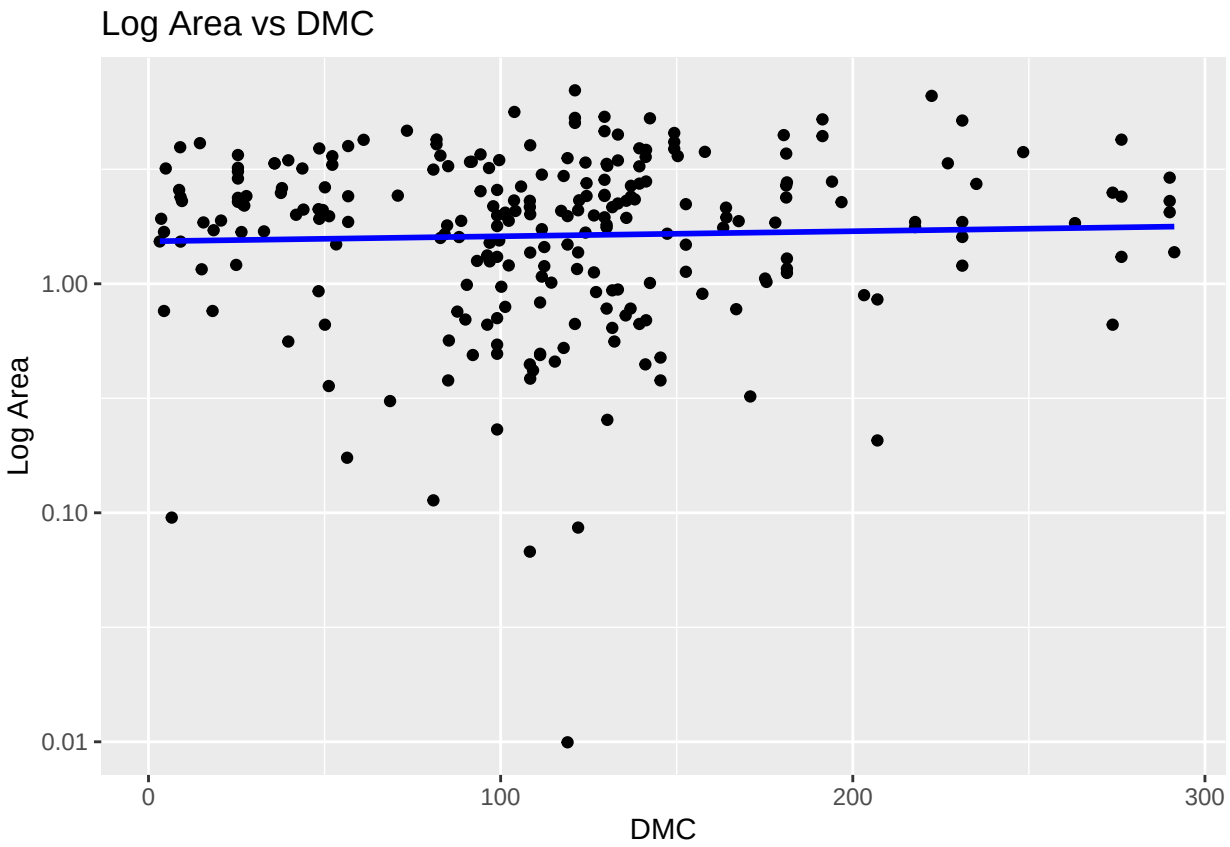

```
ggplot(forestfires, aes(x = FFMC, y = log(area))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear regression line
  labs(title = "Log Area vs FFMC",
       x = "FFMC",
       y = "Log Area") +
  scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 274 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 274 rows containing missing values (`geom_point()`).
```



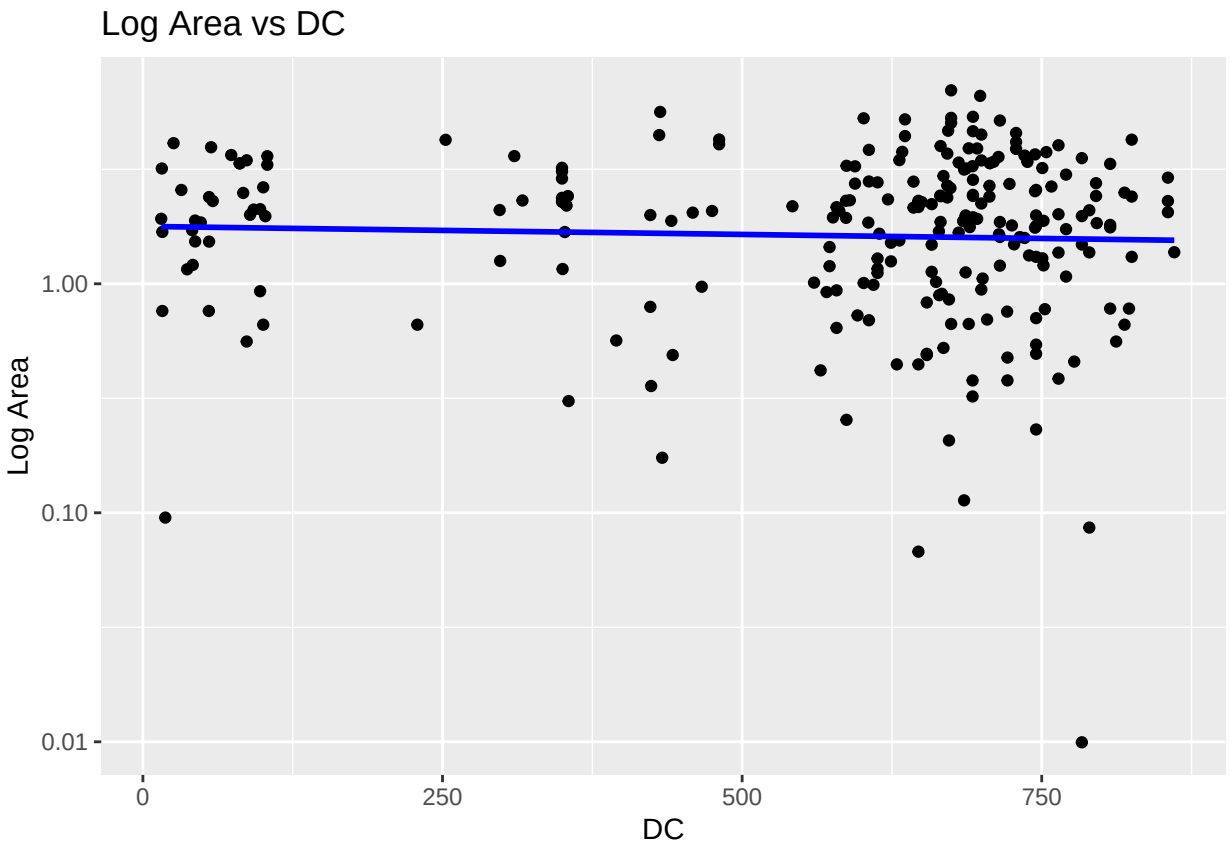

```
ggplot(forestfires, aes(x = DMC, y = log(area))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear regression line
  labs(title = "Log Area vs DMC",
       x = "DMC",
       y = "Log Area") +
  scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 274 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 274 rows containing missing values (`geom_point()`).
```



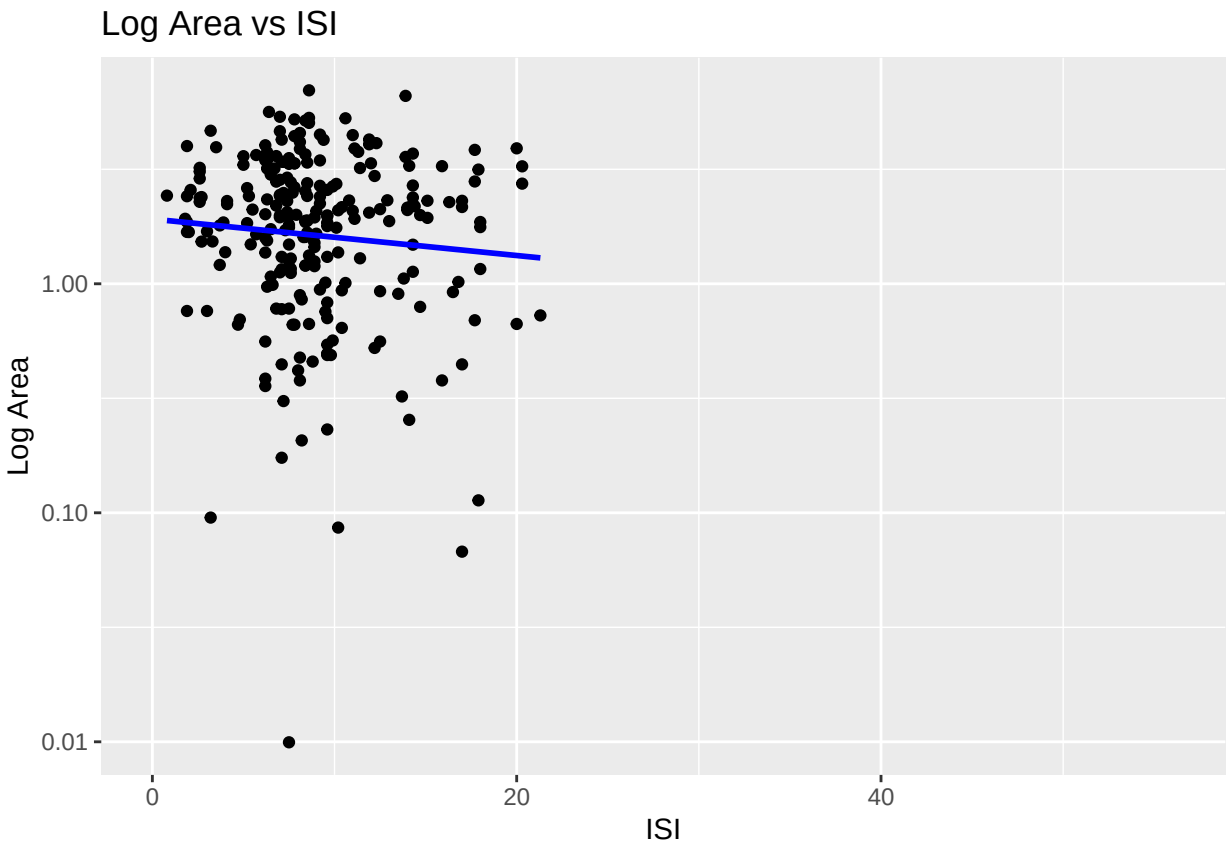

```
ggplot(forestfires, aes(x = DC, y = log(area))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear regression line
  labs(title = "Log Area vs DC",
       x = "DC",
       y = "Log Area") +
  scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 274 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 274 rows containing missing values (`geom_point()`).
```



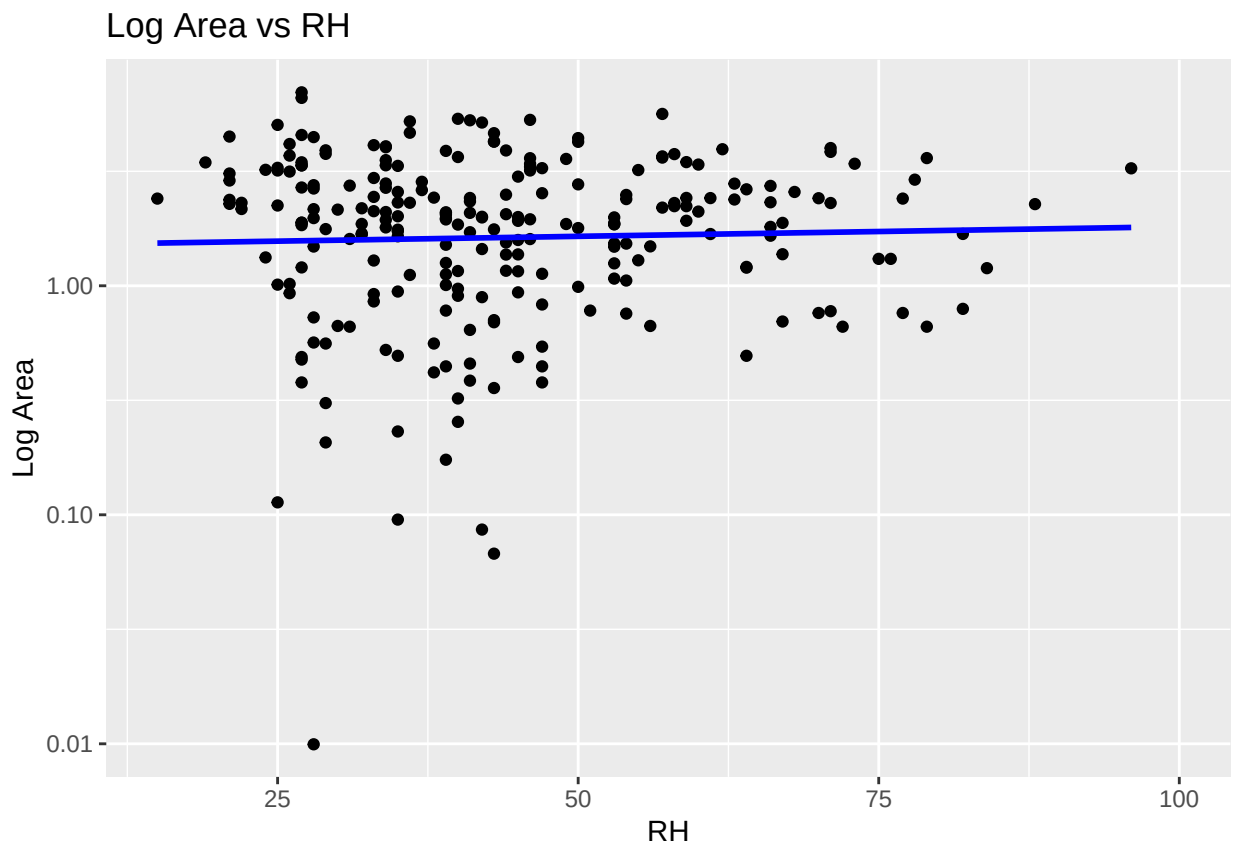

```
ggplot(forestfires, aes(x = ISI, y = log(area))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear regression line
  labs(title = "Log Area vs ISI",
       x = "ISI",
       y = "Log Area") +
  scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 274 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 274 rows containing missing values (`geom_point()`).
```



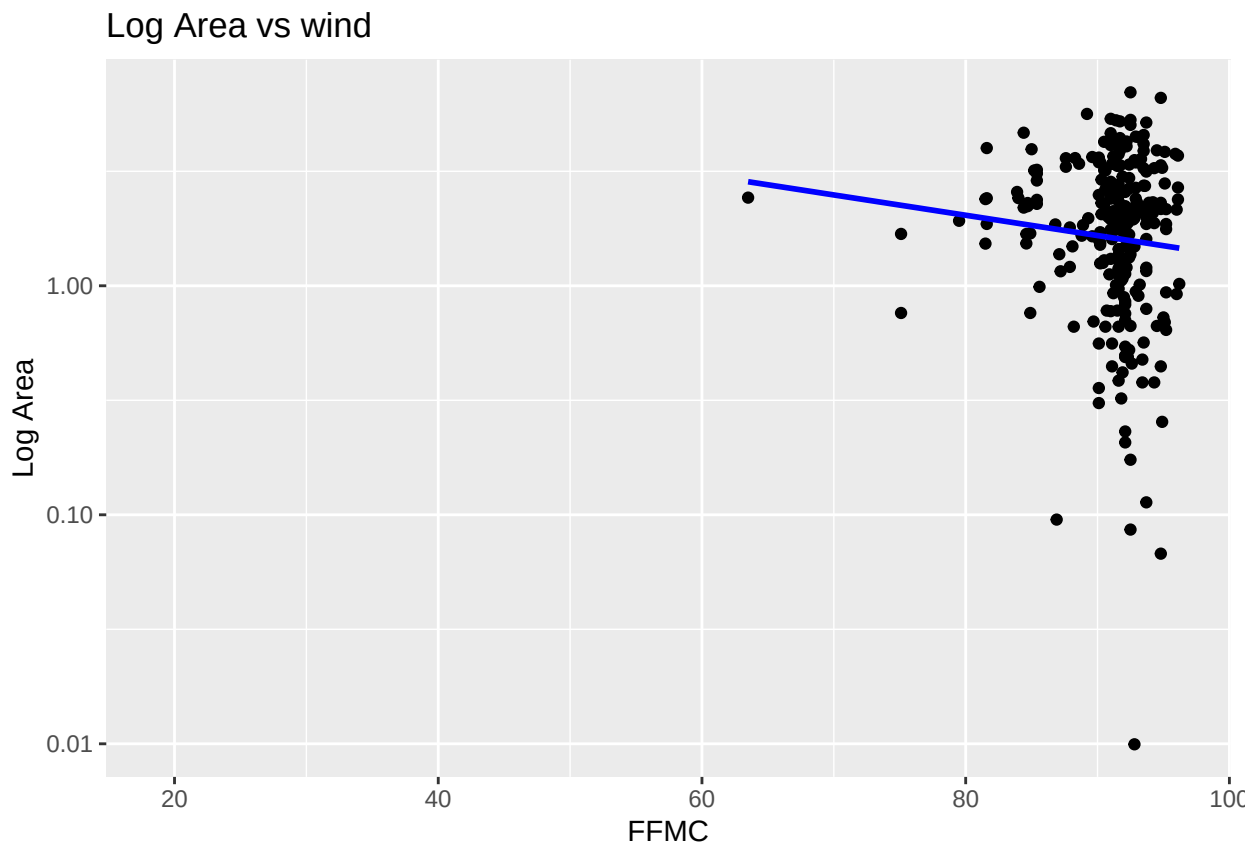

```
ggplot(forestfires, aes(x = RH, y = log(area))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear regression line
  labs(title = "Log Area vs RH",
       x = "RH",
       y = "Log Area") +
  scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 274 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 274 rows containing missing values (`geom_point()`).
```



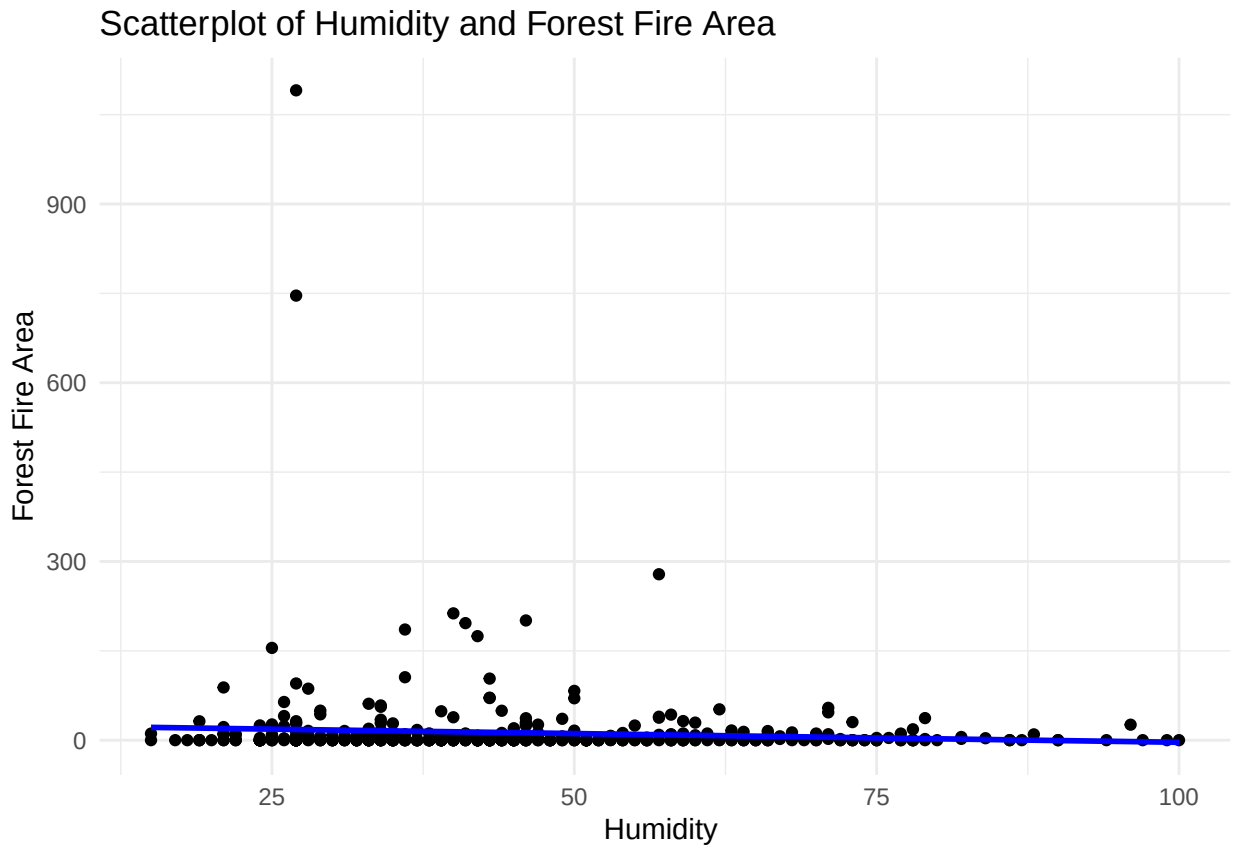

```
ggplot(forestfires, aes(x = FFMC, y = log(area))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear regression line
  labs(title = "Log Area vs wind",
       x = "FFMC",
       y = "Log Area") +
  scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 274 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 274 rows containing missing values (`geom_point()`).
```




```
ggplot(data = forestfires, aes(x = RH, y = area)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(
    title = "Scatterplot of Humidity and Forest Fire Area",
    x = "Humidity ",
    y = "Forest Fire Area"
  ) +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



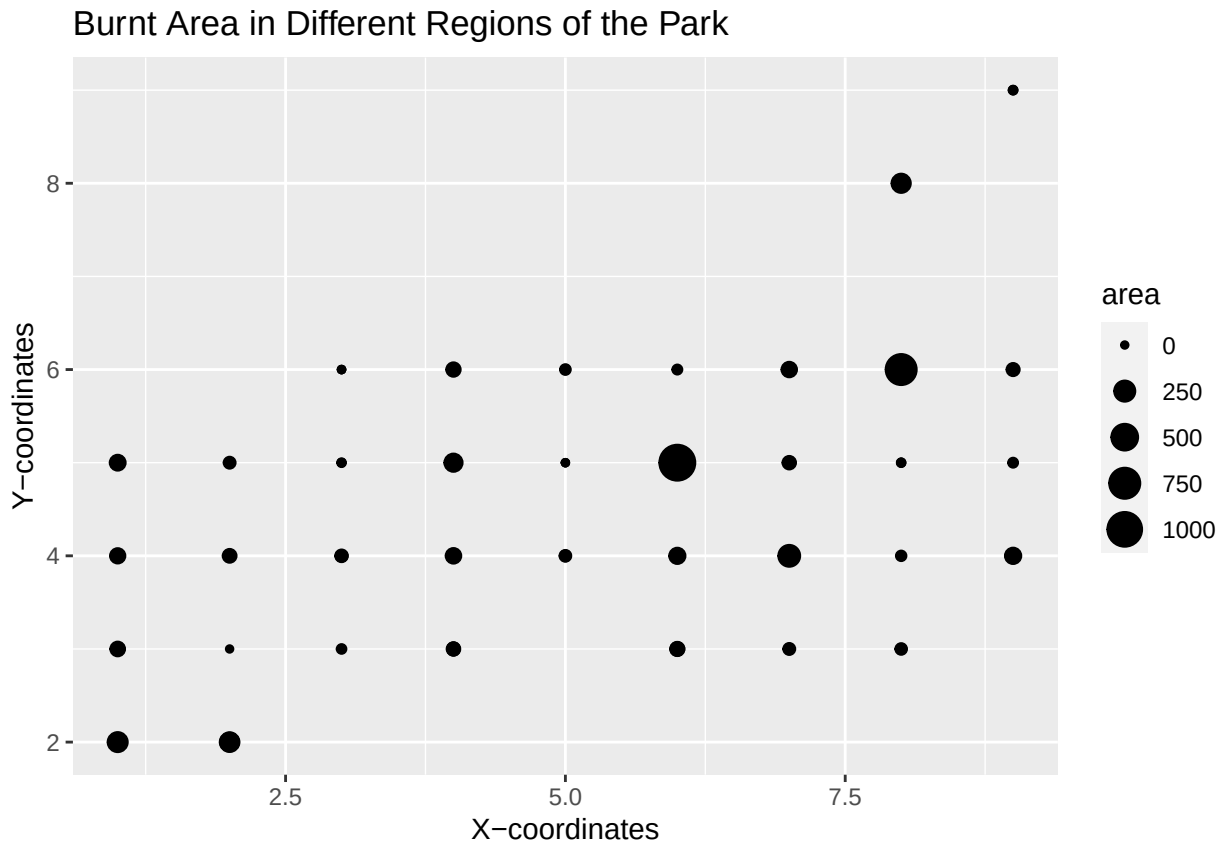

```
library(caret)
forestfires$day <-as.numeric(forestfires$day)
forestfires$month <-as.numeric(forestfires$month)
forestfires$season <-as.numeric(forestfires$season)
summary(forestfires)
```

```
##        X               Y           month             day             FFMC
##  Min.   :1.000   Min.   :2.0   Min.   : 1.000   Min.   :1.000   Min.   :18.70
##  1st Qu.:3.000   1st Qu.:4.0   1st Qu.: 7.000   1st Qu.:2.000   1st Qu.:90.20
##  Median :4.000   Median :4.0   Median : 8.000   Median :5.000   Median :91.60
##  Mean   :4.669   Mean   :4.3   Mean   : 7.476   Mean   :4.259   Mean   :90.64
##  3rd Qu.:7.000   3rd Qu.:5.0   3rd Qu.: 9.000   3rd Qu.:6.000   3rd Qu.:92.90
##  Max.   :9.000   Max.   :9.0   Max.   :12.000   Max.   :7.000   Max.   :96.20
##       DMC              DC             ISI              temp
##  Min.   :  1.1   Min.   :  7.9   Min.   : 0.000   Min.   : 2.20
##  1st Qu.: 68.6   1st Qu.:437.7   1st Qu.: 6.500   1st Qu.:15.50
##  Median :108.3   Median :664.2   Median : 8.400   Median :19.30
##  Mean   :110.9   Mean   :547.9   Mean   : 9.022   Mean   :18.89
##  3rd Qu.:142.4   3rd Qu.:713.9   3rd Qu.:10.800   3rd Qu.:22.80
##  Max.   :291.3   Max.   :860.6   Max.   :56.100   Max.   :33.30
##        RH              wind            rain              area
##  Min.   : 15.00   Min.   :0.400   Min.   :0.00000   Min.   :   0.00
##  1st Qu.: 33.00   1st Qu.:2.700   1st Qu.:0.00000   1st Qu.:   0.00
##  Median : 42.00   Median :4.000   Median :0.00000   Median :   0.52
##  Mean   : 44.29   Mean   :4.018   Mean   :0.02166   Mean   :  12.85
```

```
##  3rd Qu.: 53.00   3rd Qu.:4.900   3rd Qu.:0.00000   3rd Qu.:   6.57
##  Max.   :100.00   Max.   :9.400   Max.   :6.40000   Max.   :1090.84
##      season
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :3.000
##  Mean   :2.207
##  3rd Qu.:3.000
##  Max.   :4.000
```

```
idx <- createDataPartition(forestfires$area, p = 0.80, list = FALSE)

forest_train <- forestfires[idx,]
forest_test <- forestfires[-idx,]
```

```
ggplot(data = forestfires, aes(x=X,y=Y, size = area), color = "blue") +
  geom_point() + labs(title = "Burnt Area in Different Regions of the Park", x = "X-coordinates", y = "
```



$$N_h = \frac{N_s}{\alpha * (N_i + N_o)}$$

Rule One: As the complexity in the relationship between the input data and the desired output increases, the number of the processing elements in the hidden layer should also increase.

Rule Two: If the process being modeled is separable into multiple stages, then additional hidden layer(s) may be required. If the process is not separable into stages, then additional layers may simply enable memorization of the training set, and not a true general solution effective with other data.

Commenting out prediction accuracy because seeds are not reproducable in the pdf knit, tried solutions

```
start.time <- Sys.time()

nn_model <- neuralnet(
  formula = log(area+1)~.-month-X-Y,
  data = forest_train,
  hidden = c(5,2),
  linear.output =  TRUE,
  stepmax=2e05,
  learningrate = 0.01
)
end.time <- Sys.time()
```

```
y_pred <- predict(nn_model, newdata = forest_test)

# Calculate Mean Squared Error (MSE)
rmse <- sqrt(mean((log(forest_test$area +1) - y_pred)^2))

# Calculate R-squared (variance explained)
y_mean <- mean(log(forest_test$area +1))
tss <- sum((log(forest_test$area +1) - y_mean)^2)
rss <- sum((log(forest_test$area +1) - y_pred)^2)
r_squared <- 1 - (rss / tss)
mae_value <- MAE(y_pred, forest_test$area)


# Print metrics
print(paste("Mean Absolute Error (MAE):", mae_value))
```

```
## [1] "Mean Absolute Error (MAE): 19.0098455804758"
```

```
print(paste("Root Mean Squared Error (MSE):", rmse))
```

```
## [1] "Root Mean Squared Error (MSE): 1.50005522173778"
```

```
print(paste("R-squared (variance explained):", r_squared))
```

```
## [1] "R-squared (variance explained): 0.00304142271102859"
```

```
nn_metrics <- data.frame(
  MAE = round(mae_value,4),
  RMSE = round(rmse,4),
  PercentVarianceExplained = round(r_squared,3)
)
 kable(table(nn_metrics), caption = "Neural Network")
```

Table 1: Neural Network

MAE	RMSE	PercentVarianceExplained	Freq
19.0098	1.5001	0.003	1

```
# set.seed(3645121)
#
# start.time <- Sys.time()
#
# nn_model1 <- neuralnet(
#   formula = log(area+1)~.-month,
#   data = forest_train,
#   hidden = c(5,2),
#   linear.output =  TRUE,
#   stepmax=2e05,
#   learningrate = 0.01
# )
# end.time <- Sys.time()
```

```
# y_pred <- predict(nn_model1, newdata = forest_test)
#
# # Calculate Mean Squared Error (MSE)
# rmse <- sqrt(mean((log(forest_test$area +1) - y_pred)^2))
#
# # Calculate R-squared (variance explained)
# y_mean <- mean(log(forest_test$area +1))
# tss <- sum((log(forest_test$area +1) - y_mean)^2)
# rss <- sum((log(forest_test$area +1) - y_pred)^2)
# r_squared <- 1 - (rss / tss)
# mae_value <- MAE(y_pred, forest_test$area)
#
#
# # Print metrics
# print(paste("Mean Absolute Error (MAE):", mae_value))
# print(paste("Root Mean Squared Error (MSE):", rmse))
# print(paste("R-squared (variance explained):", r_squared))
#
# nn_metrics <- data.frame(
#   MAE = round(mae_value,4),
#   RMSE = round(rmse,4),
#   PercentVarianceExplained = round(r_squared,3)
# )
#  kable(table(nn_metrics), caption = "Neural Network")
```

```
# set.seed(42831674)
#
# start.time <- Sys.time()
#
# nn_model2 <- neuralnet(
#   formula = log(area+1)~.-month-X-Y,
#   data = forest_train,
#   hidden = c(5,3),
```

```
#   linear.output =  TRUE,
#   stepmax=2e05,
#   learningrate = 0.01
# )
# end.time <- Sys.time()
```

```
#comment because its not reporducable

# plot(nn_model2)
# y_pred <- predict(nn_model2, newdata = forest_test)
#
# # Calculate Mean Squared Error (MSE)
# rmse <- sqrt(mean((log(forest_test$area +1) - y_pred)^2))
#
# # Calculate R-squared (variance explained)
# y_mean <- mean(log(forest_test$area +1))
# tss <- sum((log(forest_test$area +1) - y_mean)^2)
# rss <- sum((log(forest_test$area +1) - y_pred)^2)
# r_squared <- 1 - (rss / tss)
# mae_value <- MAE(y_pred, forest_test$area)
#
#
# # Print metrics
# print(paste("Mean Absolute Error (MAE):", mae_value))
# print(paste("Root Mean Squared Error (MSE):", rmse))
# print(paste("R-squared (variance explained):", r_squared))
#
# nn_metrics <- data.frame(
#   MAE = round(mae_value,4),
#   RMSE = round(rmse,4),
#   PercentVarianceExplained = round(r_squared,3)
# )
#  kable(table(nn_metrics), caption = "Neural Network")
```

```
start.time <- Sys.time()

nn_model3 <- neuralnet(
  formula = log(area+1)~FFMC + DMC + DC + ISI + season,
  data = forest_train,
  hidden = c(5,2),
  linear.output =  TRUE,
  stepmax=2e05,
  learningrate = 0.01
)
end.time <- Sys.time()
```

```
plot(nn_model3)
y_pred <- predict(nn_model3, newdata = forest_test)

# Calculate Mean Squared Error (MSE)
rmse <- sqrt(mean((log(forest_test$area +1) - y_pred)^2))

# Calculate R-squared (variance explained)
```

```r
y_mean <- mean(log(forest_test$area +1))
tss <- sum((log(forest_test$area +1) - y_mean)^2)
rss <- sum((log(forest_test$area +1) - y_pred)^2)
r_squared <- 1 - (rss / tss)
mae_value <- MAE(y_pred, forest_test$area)


# Print metrics
print(paste("Mean Absolute Error (MAE):", mae_value))
```

```
## [1] "Mean Absolute Error (MAE): 19.026515034918"
```

```r
print(paste("Root Mean Squared Error (MSE):", rmse))
```

```
## [1] "Root Mean Squared Error (MSE): 1.52332594872127"
```

```r
print(paste("R-squared (variance explained):", r_squared))
```

```
## [1] "R-squared (variance explained): -0.028130635236967"
```

```r
nn_metrics <- data.frame(
  MAE = round(mae_value,4),
  RMSE = round(rmse,4),
  PercentVarianceExplained = round(r_squared,3)
)
 kable(table(nn_metrics), caption = "Neural Network")
```

Table 2: Neural Network

MAE	RMSE	PercentVarianceExplained	Freq
19.0265	1.5233	-0.028	1

```r
# set.seed(42831674)
#
# start.time <- Sys.time()
#
# nn_model4 <- neuralnet(
#   formula = log(area+1)~season + temp + RH + wind,
#   data = forest_train,
#   hidden = c(5,3),
#   linear.output =  TRUE,
#   stepmax=2e05,
#   learningrate = 0.01
# )
# end.time <- Sys.time()
```

```r
# y_pred <- predict(nn_model4, newdata = forest_test)
#
# # Calculate Mean Squared Error (MSE)
```

```
# rmse <- sqrt(mean((log(forest_test$area +1) - y_pred)^2))
#
# # Calculate R-squared (variance explained)
# y_mean <- mean(log(forest_test$area +1))
# tss <- sum((log(forest_test$area +1) - y_mean)^2)
# rss <- sum((log(forest_test$area +1) - y_pred)^2)
# r_squared <- 1 - (rss / tss)
# mae_value <- MAE(y_pred, forest_test$area)
#
#
# # Print metrics
# print(paste("Mean Absolute Error (MAE):", mae_value))
# print(paste("Root Mean Squared Error (MSE):", rmse))
# print(paste("R-squared (variance explained):", r_squared))
#
# nn_metrics <- data.frame(
#   MAE = round(mae_value,4),
#   RMSE = round(rmse,4),
#   PercentVarianceExplained = round(r_squared,3)
# )
#  kable(table(nn_metrics), caption = "Neural Network")
```

```
time.taken <- round(end.time - start.time,2)
time.taken
```

```
## Time difference of 6.54 secs
```

```
set.seed(4281001)

features <- c("X","Y","FFMC", "DMC", "DC", "ISI", "temp", "RH", "wind", "rain")
target <- "area"


# Train the Random Forest model
rf_model <- randomForest(
  formula = as.formula(paste(target, "~", paste(features, collapse = "+"))),
  data = forest_train
)

ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 5)

grid_mtry = data.frame(mtry = seq(1,12))


rf_cv <- train(area~.,
               data = forest_train,
               method = "rf",
               trControl = ctrl,
               tunegrid = grid_mtry,
               importance = TRUE)
rf_cv
```

```
## Random Forest
```

```
##
## 416 samples
##  13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 332, 333, 332, 334, 333, 333, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared     MAE
##    2    46.50969  0.024120198  17.39258
##    7    48.78178  0.011156801  18.52065
##   13    49.53863  0.004861585  18.35826
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
# Make predictions on the test set
predictions <- predict(rf_model, newdata = forest_test)
predictions
```

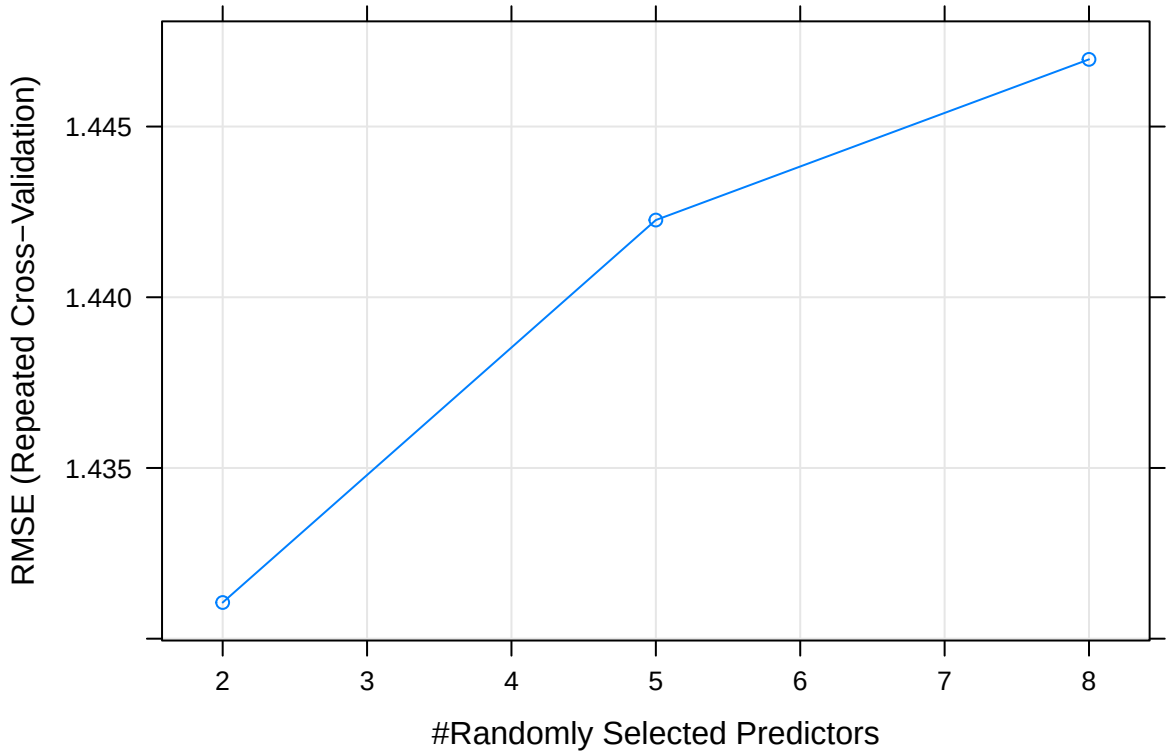
```
##          1          2          3          4          5          6          7
##   7.643423  16.050834   9.490454   8.345971  15.514963   9.495105  18.063252
##          8          9         10         11         12         13         14
##  16.521931   1.896844   1.896844   8.984386   9.450860   7.540405  23.854201
##         15         16         17         18         19         20         21
##  91.054453  12.816732  35.139598  33.185619   5.425086   4.942934  13.444628
##         22         23         24         25         26         27         28
##   4.010753  10.542062   2.406645  12.905589   7.767906  24.107319   6.948272
##         29         30         31         32         33         34         35
##  33.115840   5.116178  18.706532  15.792555  30.770304   8.345971  10.709795
##         36         37         38         39         40         41         42
##  20.040710   5.363919   6.430765   2.216551  21.085371  10.297803  23.264913
##         43         44         45         46         47         48         49
##  29.808649  10.797921   2.382491   5.414268 141.066623   8.828379   4.153461
##         50         51         52         53         54         55         56
##   2.010024   2.866403   7.734552  10.332123  12.257782   7.402454   2.797120
##         57         58         59         60         61         62         63
##  44.986123  11.202927  11.079720   5.477185   5.477185   7.890059  16.014715
##         64         65         66         67         68         69         70
##  17.970355  11.990919  10.596758   2.317945  17.743696  16.138025  34.743464
##         71         72         73         74         75         76         77
##   1.916857  10.404226  24.256556  15.161988  23.346274  13.136468  14.368614
##         78         79         80         81         82         83         84
##   9.155027  25.054291  22.598758  30.342678 100.364912  30.385633  13.790211
##         85         86         87         88         89         90         91
##  19.924277   6.298167  10.353038   9.901679   8.977445   8.114963   8.707463
##         92         93         94         95         96         97         98
##   3.897973  13.354102  14.133074   2.010407   6.977519   6.944654  11.889681
##         99        100        101
##   7.118686  23.530611   8.389144
```

Random Forrest

```r
set.seed(4281001)

ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 5)

grid_mtry = data.frame(mtry = seq(1,length(colnames(forest_train))-1))

rf_cv <- train(form = log(area+1)~.-season-X-Y-month-rain,
               data = forest_train,
               method = "rf",
               trControl = ctrl,
               tunegrid = grid_mtry,
               importance = TRUE,
               ntree = 500)
rf_cv
```

```
## Random Forest
##
## 416 samples
##  13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 333, 334, 333, 332, 332, 332, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared    MAE
##   2     1.431060  0.01284360  1.173116
##   5     1.442261  0.01266224  1.181112
##   8     1.446967  0.01345541  1.185686
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```r
plot(rf_cv)
```




```
set.seed(51564561)
start.time <- Sys.time()


mod_rf <- randomForest(log(area+1)~.-X-Y-month-rain,data = forest_train, mtry = 2, ntree =500)
mod_rf
```

```
##
## Call:
##  randomForest(formula = log(area + 1) ~ . - X - Y - month - rain,      data = forest_train, mtry = 2
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 2.047437
##                     % Var explained: -9.1
```

```
end.time <- Sys.time()
time.taken <- round(end.time - start.time,2)
time.taken
```

```
## Time difference of 0.34 secs
```

```r
y_pred <- predict(mod_rf, newdata = forest_test)

# Calculate Mean Squared Error (MSE)
rmse <- sqrt(mean((log(forest_test$area +1) - y_pred)^2))

# Calculate R-squared (variance explained)
y_mean <- mean(log(forest_test$area +1))
tss <- sum((log(forest_test$area +1) - y_mean)^2)
rss <- sum((log(forest_test$area +1) - y_pred)^2)
r_squared <- 1 - (rss / tss)
mae_value <- MAE(y_pred, forest_test$area)


print(paste("Mean Absolute Error (MAE):", mae_value))
```

```
## [1] "Mean Absolute Error (MAE): 19.036156550682"
```

```r
print(paste("Root Mean Squared Error (MSE):", rmse))
```

```
## [1] "Root Mean Squared Error (MSE): 1.59423575507464"
```

```r
print(paste("R-squared (variance explained):", r_squared))
```

```
## [1] "R-squared (variance explained): -0.126076020142954"
```

```r
nn_metrics <- data.frame(
  MAE = round(mae_value,4),
  RMSE = round(rmse,4),
  PercentVarianceExplained = round(r_squared,3)
)
 kable(table(nn_metrics), caption = "Random Forest")
```

Table 3: Random Forest

MAE	RMSE	PercentVarianceExplained	Freq
19.0362	1.5942	-0.126	1

```r
set.seed(4821)
mod_rf <- randomForest(log(area+1)~FFMC + DMC + DC + ISI + season,data = forest_train, mtry = 2, ntree =

mod_rf
```

```
##
## Call:
##  randomForest(formula = log(area + 1) ~ FFMC + DMC + DC + ISI +      season, data = forest_train, mt
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 2.200302
##                     % Var explained: -17.25
```

```r
y_pred <- predict(mod_rf, newdata = forest_test)

# Calculate Mean Squared Error (MSE)
rmse <- sqrt(mean((log(forest_test$area +1) - y_pred)^2))

# Calculate R-squared (variance explained)
y_mean <- mean(log(forest_test$area +1))
tss <- sum((log(forest_test$area +1) - y_mean)^2)
rss <- sum((log(forest_test$area +1) - y_pred)^2)
r_squared <- 1 - (rss / tss)
mae_value <- MAE(y_pred, forest_test$area)


# Print metrics
print(paste("Mean Absolute Error (MAE):", mae_value))
```

```
## [1] "Mean Absolute Error (MAE): 19.0359042591728"
```

```r
print(paste("Root Mean Squared Error (MSE):", rmse))
```

```
## [1] "Root Mean Squared Error (MSE): 1.66646336429638"
```

```r
print(paste("R-squared (variance explained):", r_squared))
```

```
## [1] "R-squared (variance explained): -0.23042220763644"
```

```r
nn_metrics <- data.frame(
  MAE = round(mae_value,4),
  RMSE = round(rmse,4),
  PercentVarianceExplained = round(r_squared,3)
)
 kable(table(nn_metrics), caption = "Neural Network")
```

Table 4: Neural Network

MAE	RMSE	PercentVarianceExplained	Freq
19.0359	1.6665	-0.23	1

```r
set.seed(4821)
mod_rf <- randomForest(log(area+1)~season + temp +RH+wind,data = forest_train, mtry = 2, ntree =500)

mod_rf
```

```
##
## Call:
##  randomForest(formula = log(area + 1) ~ season + temp + RH + wind,      data = forest_train, mtry = 2
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
```

```
##
##           Mean of squared residuals: 2.064479
##                     % Var explained: -10.01
```

```
y_pred <- predict(mod_rf, newdata = forest_test)

# Calculate Mean Squared Error (MSE)
rmse <- sqrt(mean((log(forest_test$area +1) - y_pred)^2))

# Calculate R-squared (variance explained)
y_mean <- mean(log(forest_test$area +1))
tss <- sum((log(forest_test$area +1) - y_mean)^2)
rss <- sum((log(forest_test$area +1) - y_pred)^2)
r_squared <- 1 - (rss / tss)
mae_value <- MAE(y_pred, forest_test$area)


# Print metrics
print(paste("Mean Absolute Error (MAE):", mae_value))
```

```
## [1] "Mean Absolute Error (MAE): 19.0608351238337"
```

```
print(paste("Root Mean Squared Error (MSE):", rmse))
```

```
## [1] "Root Mean Squared Error (MSE): 1.58197781261606"
```

```
print(paste("R-squared (variance explained):", r_squared))
```

```
## [1] "R-squared (variance explained): -0.108825988546794"
```

```
nn_metrics <- data.frame(
  MAE = round(mae_value,4),
  RMSE = round(rmse,4),
  PercentVarianceExplained = round(r_squared,3)
)
 kable(table(nn_metrics), caption = "RF: Temporal")
```

Table 5: RF: Temporal

MAE	RMSE	PercentVarianceExplained	Freq
19.0608	1.582	-0.109	1

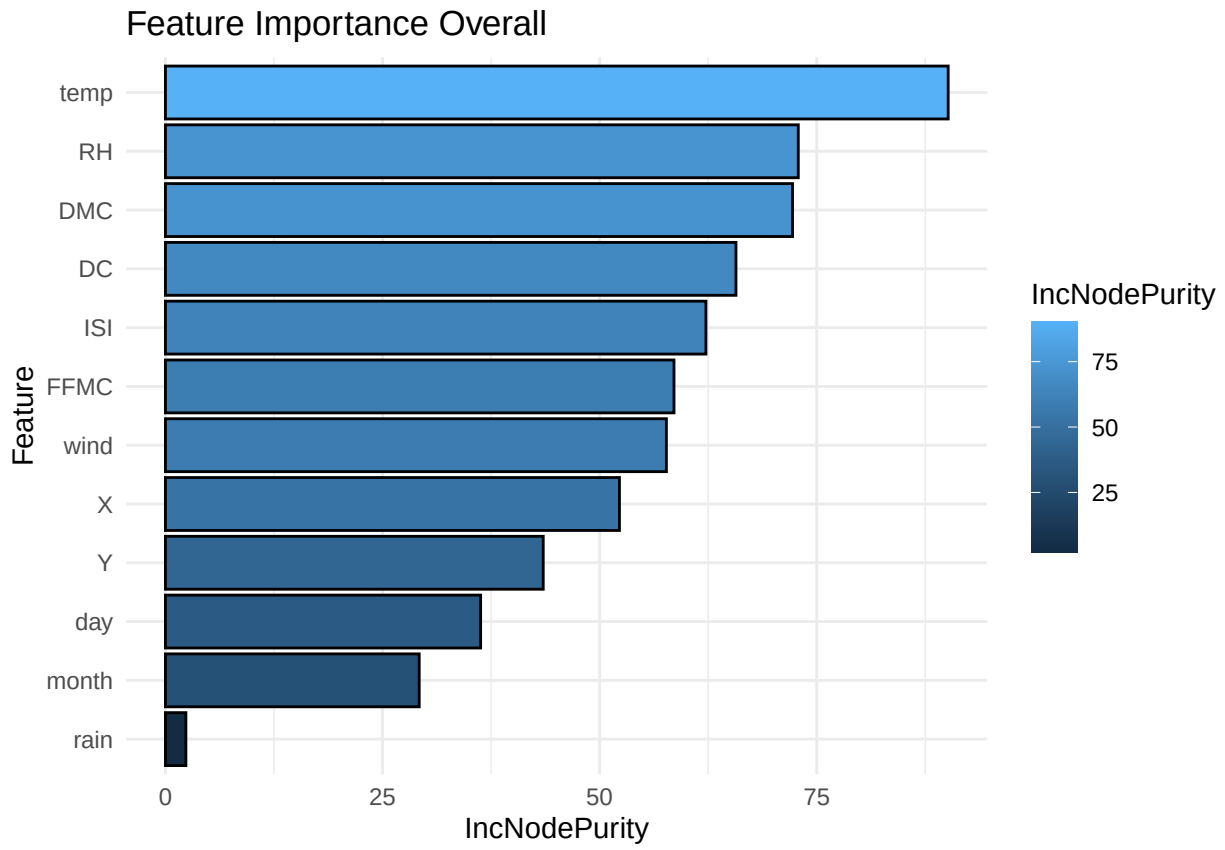
```
features <- c("X","Y","FFMC", "DMC", "DC", "ISI", "temp", "RH", "wind", "rain")
features_t = c("temp", "RH", "wind", "rain")
features_m = c("FFMC", "DMC", "DC", "ISI")
features_misc = c("day","month","X","Y")

rf_model <- randomForest(log(area+1) ~.-season, data = forest_train, mtry = 2)
```

```r
importance_overall <- data.frame(
  Feature = rownames(rf_model$importance),
  Importance = rf_model$importance
)

importance_overall
```

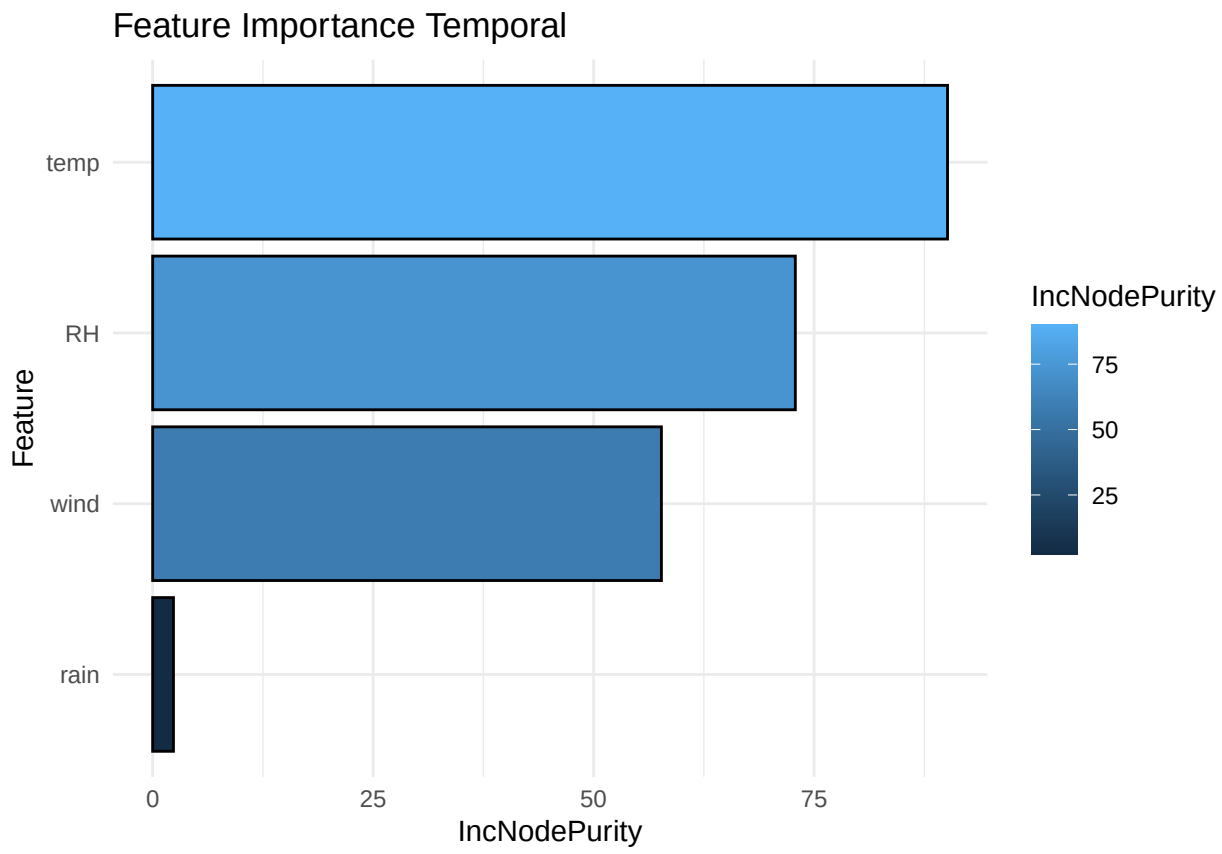
```
##       Feature IncNodePurity
## X           X      52.27632
## Y           Y      43.49984
## month   month      29.22066
## day       day      36.29565
## FFMC     FFMC      58.55884
## DMC       DMC      72.20639
## DC         DC      65.69388
## ISI       ISI      62.23757
## temp     temp      90.12911
## RH         RH      72.86969
## wind     wind      57.68405
## rain     rain       2.35508
```

```r
# Get all feature names
all_features <- colnames(forest_train)

# Subset feature names for the specified group
features_t <- intersect(features_t, all_features)
features_m <- intersect(features_m, all_features)
features_misc <- intersect(features_misc,all_features)

# Extract feature importance for the specified group of features
importance <- rf_model$importance[which(rownames(rf_model$importance) %in% features_t), , drop = FALSE]

# Create a data frame for plotting
importance1_df <- data.frame(
  Features = features_t,
  Importance = importance
)

# Extract feature importance for the specified group of features
importance <- rf_model$importance[which(rownames(rf_model$importance) %in% features_m), , drop = FALSE]

importance2_df <- data.frame(
  Features = features_m,
  Importance = importance
)

# Extract feature importance for the specified group of features
importance <- rf_model$importance[which(rownames(rf_model$importance) %in% features_misc), , drop = FALS

importance3_df <- data.frame(
  Features = features_misc,
  Importance = importance
)
```

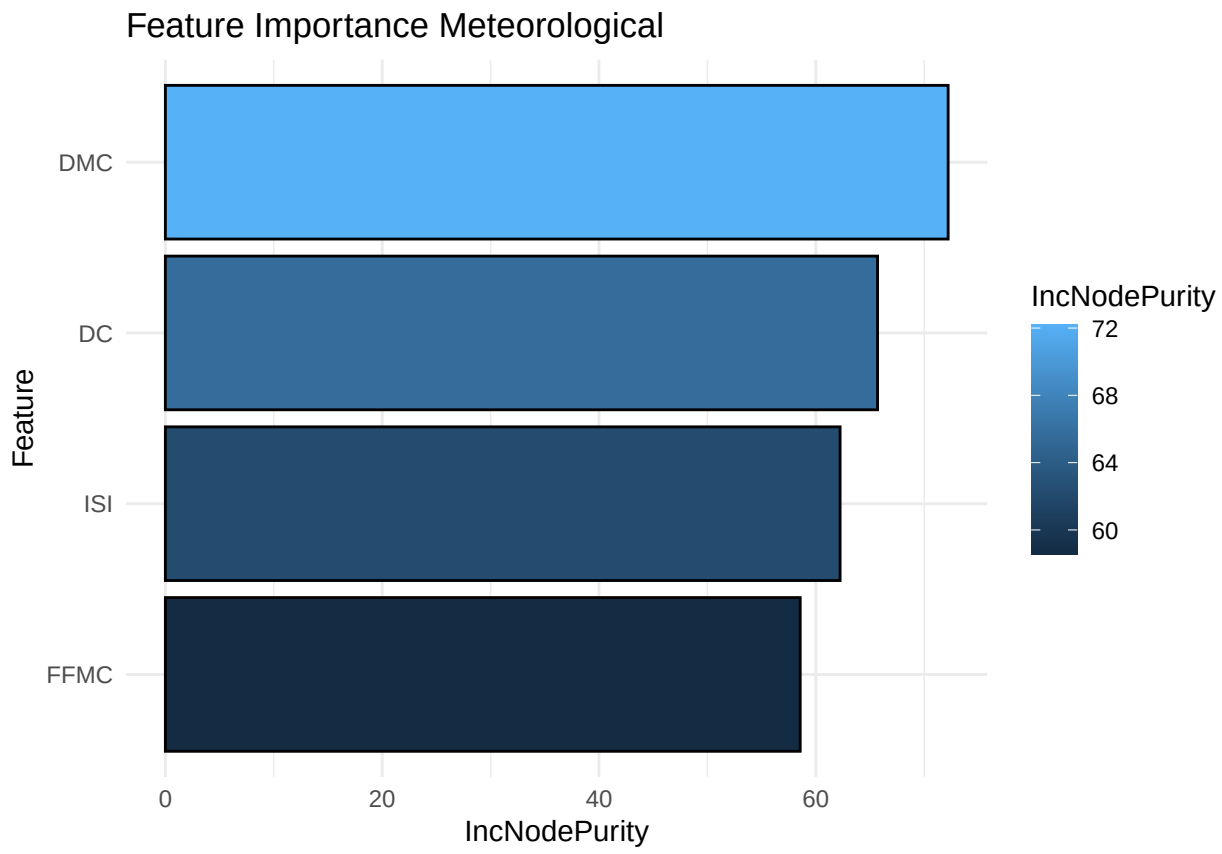
```
ggplot(importance_overall, aes(y = reorder(Feature, IncNodePurity),x = IncNodePurity, fill =IncNodePurit
  geom_bar(stat = "identity", color = "black") +
  labs(title = "Feature Importance Overall",
       x = "IncNodePurity",
       y = "Feature") +
  theme_minimal()
```



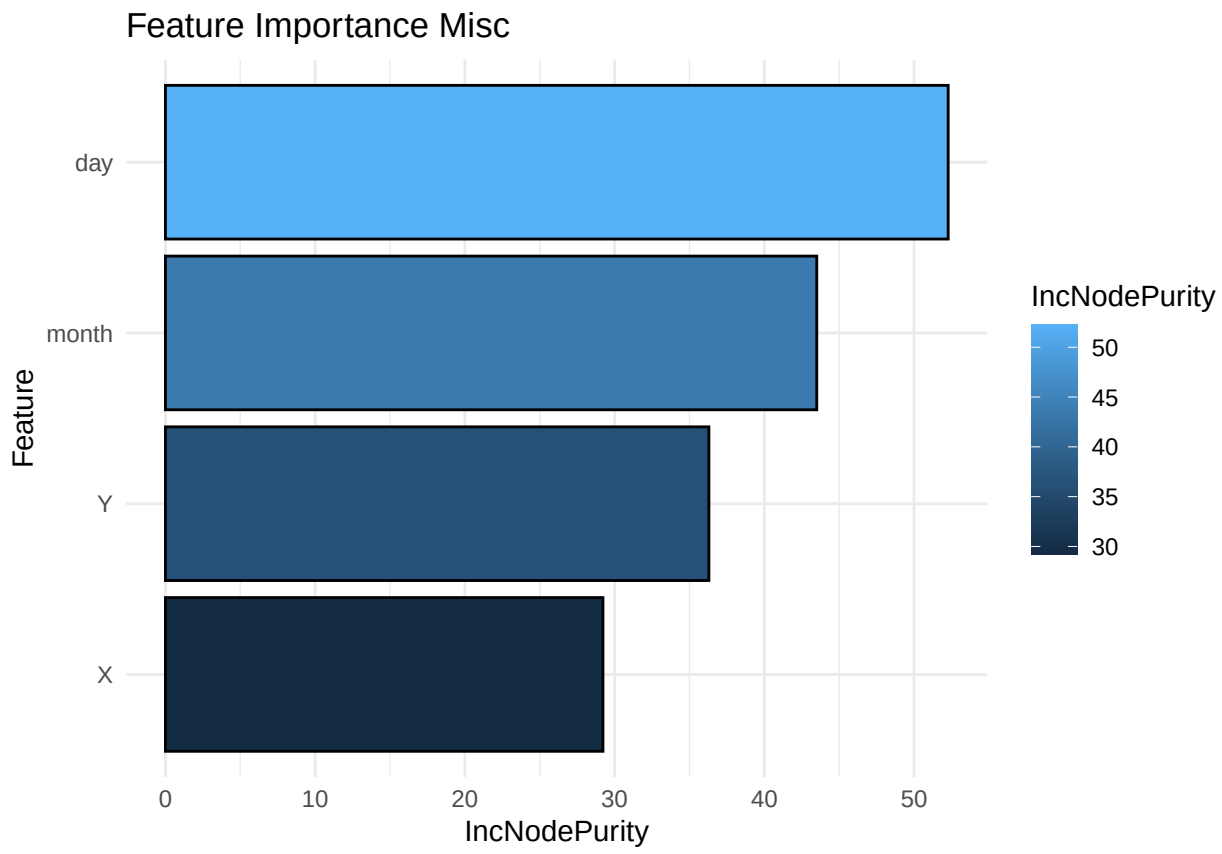

```
ggplot(importance1_df, aes(y = reorder(Features, IncNodePurity),x = IncNodePurity, fill =IncNodePurity)
  geom_bar(stat = "identity", color = "black") +
  labs(title = "Feature Importance Temporal",
       x = "IncNodePurity",
       y = "Feature") +
  theme_minimal()
```




```
ggplot(importance2_df, aes(y = reorder(Features, IncNodePurity),x = IncNodePurity, fill =IncNodePurity))
  geom_bar(stat = "identity", color = "black") +
  labs(title = "Feature Importance Meteorological",
       x = "IncNodePurity",
       y = "Feature") +
  theme_minimal()
```



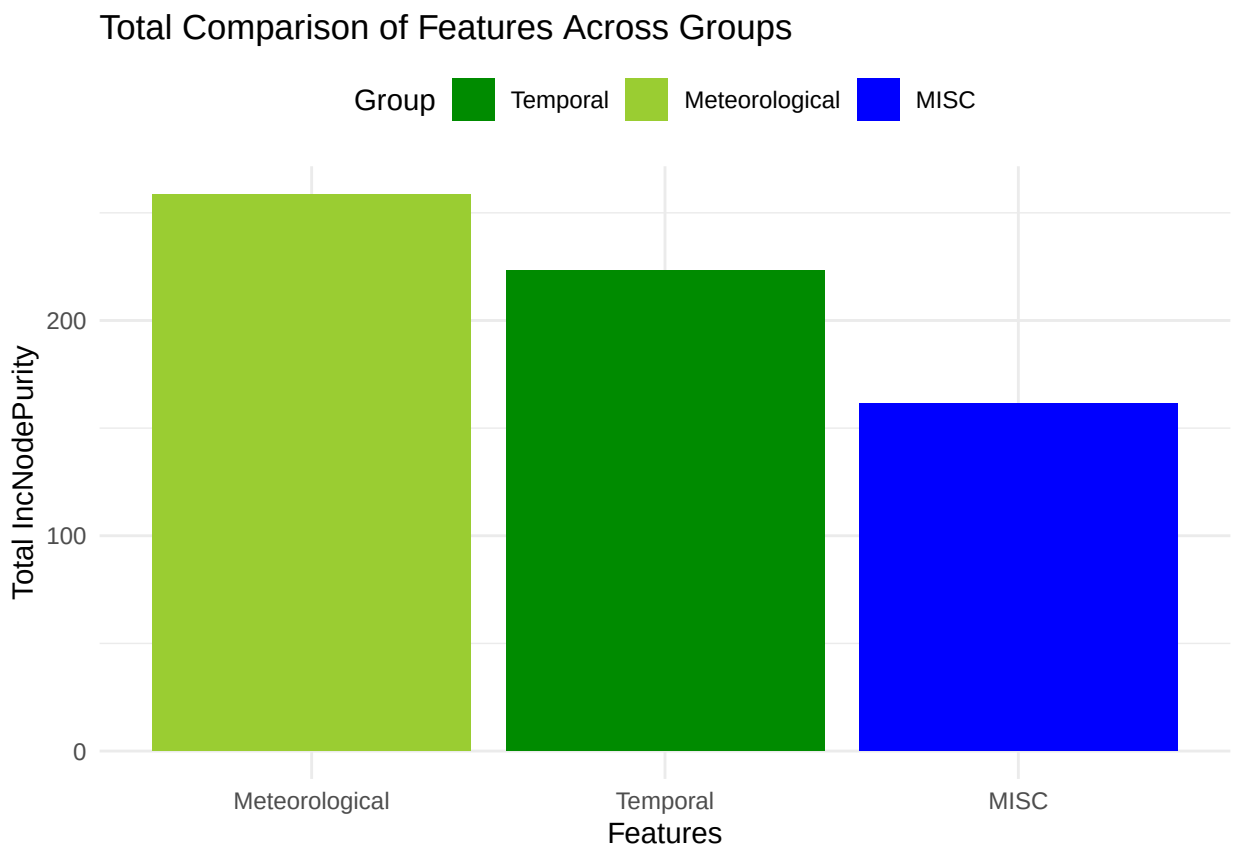

```
ggplot(importance3_df, aes(y = reorder(Features, IncNodePurity),x = IncNodePurity, fill =IncNodePurity))
  geom_bar(stat = "identity", color = "black") +
  labs(title = "Feature Importance Misc",
       x = "IncNodePurity",
       y = "Feature") +
  theme_minimal()
```



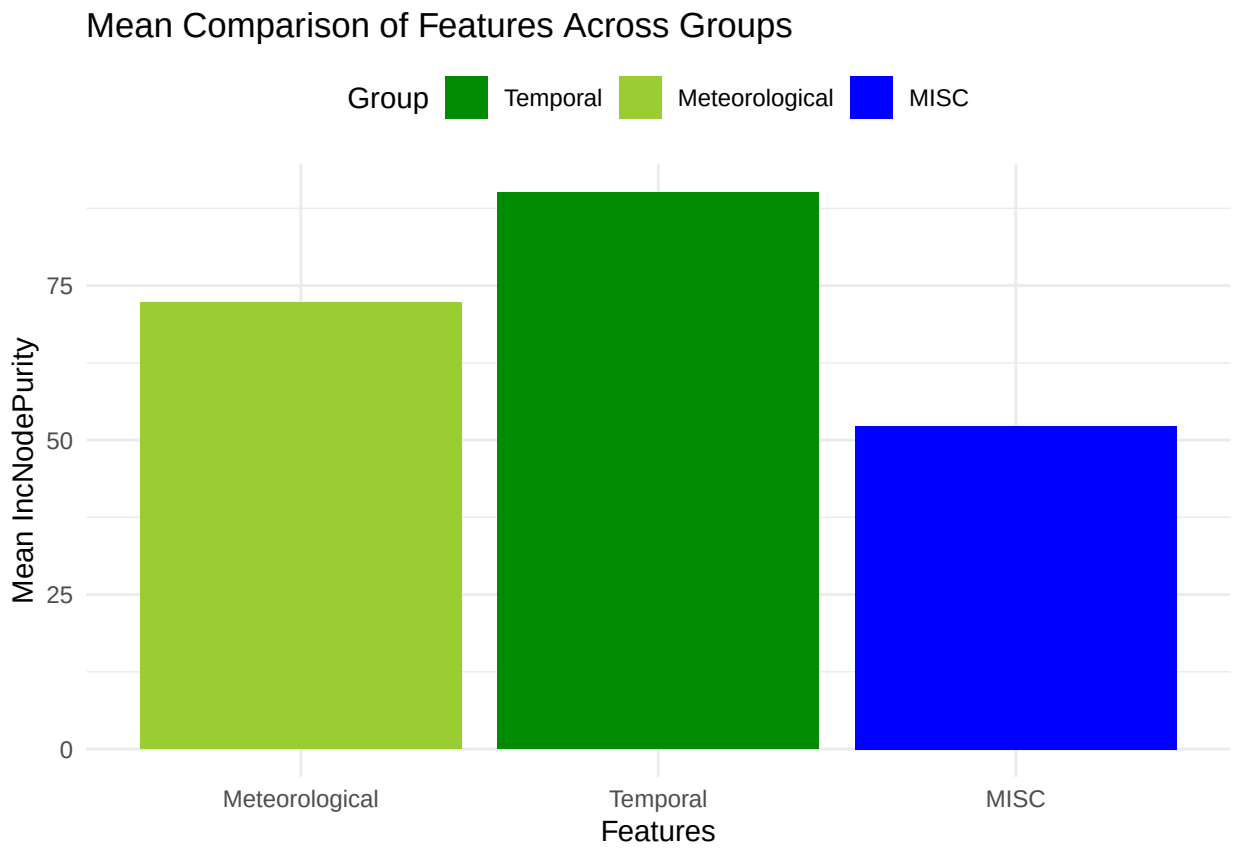

```
all_data <- rbind(
  data.frame(Feature = "Temporal", Value = importance1_df$IncNodePurity, Group = "Temporal"),
  data.frame(Feature = "Meteorological", Value = importance2_df$IncNodePurity, Group = "Meteorological"),
  data.frame(Feature = "MISC", Value = importance3_df$IncNodePurity, Group = "MISC")
)
all_data$Group <- factor(all_data$Group, levels = c("Temporal", "Meteorological", "MISC"))

total_data <- all_data %>%
  group_by(Group, Feature) %>%
  summarise(Total = sum(Value))
```

```
## `summarise()` has grouped output by 'Group'. You can override using the
## `.groups` argument.
```

```
ggplot(total_data, aes(x = reorder(Feature,-Total), y = Total, fill = Group)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Total Comparison of Features Across Groups", x = "Features", y = "Total IncNodePurity") +
  scale_fill_manual(values = c("Temporal" = "green4", "Meteorological" = "yellowgreen", "MISC" = "blue")) +
  theme_minimal() +
  theme(legend.position = "top")
```



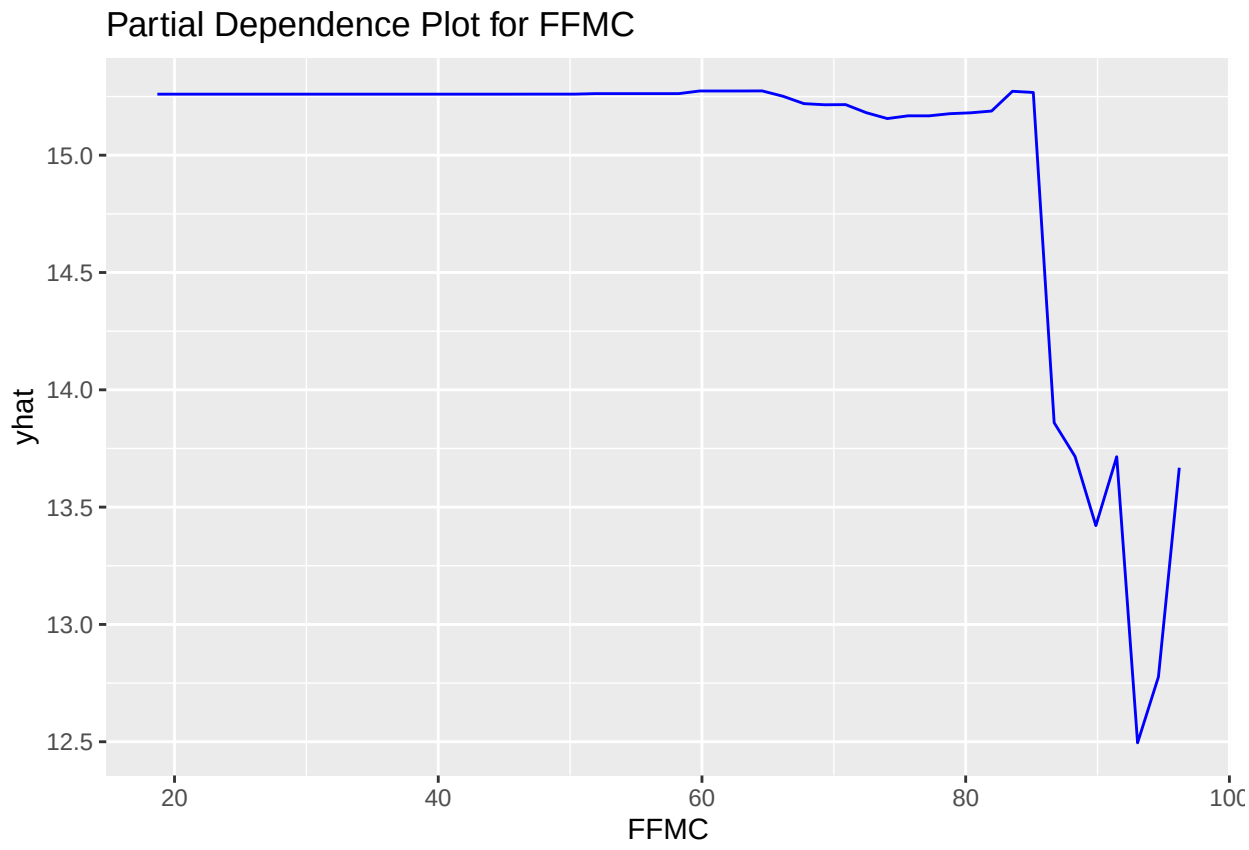

```
ggplot(all_data, aes(x = reorder(Feature,-Value), y = Value, fill = Group)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Mean Comparison of Features Across Groups", x = "Features", y = "Mean IncNodePurity") +
  scale_fill_manual(values = c("Temporal" = "green4", "Meteorological" = "yellowgreen", "MISC" = "blue"
  theme_minimal() +
  theme(legend.position = "top")
```



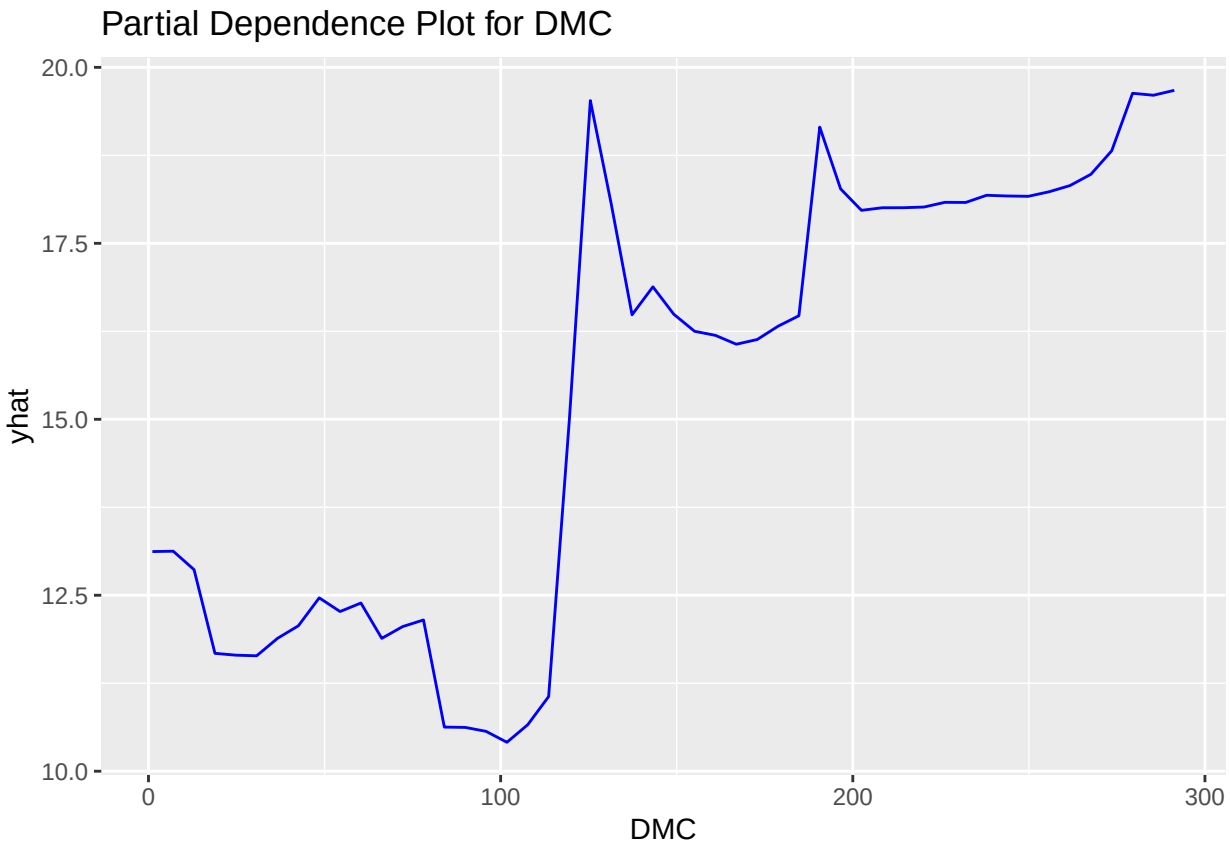

```
rf_model <- randomForest(area ~ FFMC + DMC + DC + ISI + temp + RH + wind + rain, data = forest_train)

# Create PDPs for meteorological features
pdp_FFMC <- partial(rf_model, pred.var = "FFMC", grid.resolution = 50)
pdp_DMC <- partial(rf_model, pred.var = "DMC", grid.resolution = 50)
pdp_DC <- partial(rf_model, pred.var = "DC", grid.resolution = 50)
pdp_ISI <- partial(rf_model, pred.var = "ISI", grid.resolution = 50)

# Create PDPs for temporal features
pdp_temp <- partial(rf_model, pred.var = "temp", grid.resolution = 50)
pdp_RH <- partial(rf_model, pred.var = "RH", grid.resolution = 50)
pdp_WIND <- partial(rf_model, pred.var = "wind", grid.resolution = 50)
pdp_rain <- partial(rf_model, pred.var = "rain", grid.resolution = 50)


# Plot PDPs for meteorological features


pdp_to_ggplot <- function(pdp, feature_name) {
  ggplot(pdp, aes_string(x = feature_name, y = "yhat")) +
    geom_line(color = "blue") +
    labs(title = paste("Partial Dependence Plot for", feature_name))
}
```

```
# Plot PDPs for meteorological features
(ggplot_FFMC <- pdp_to_ggplot(pdp_FFMC, "FFMC"))
```

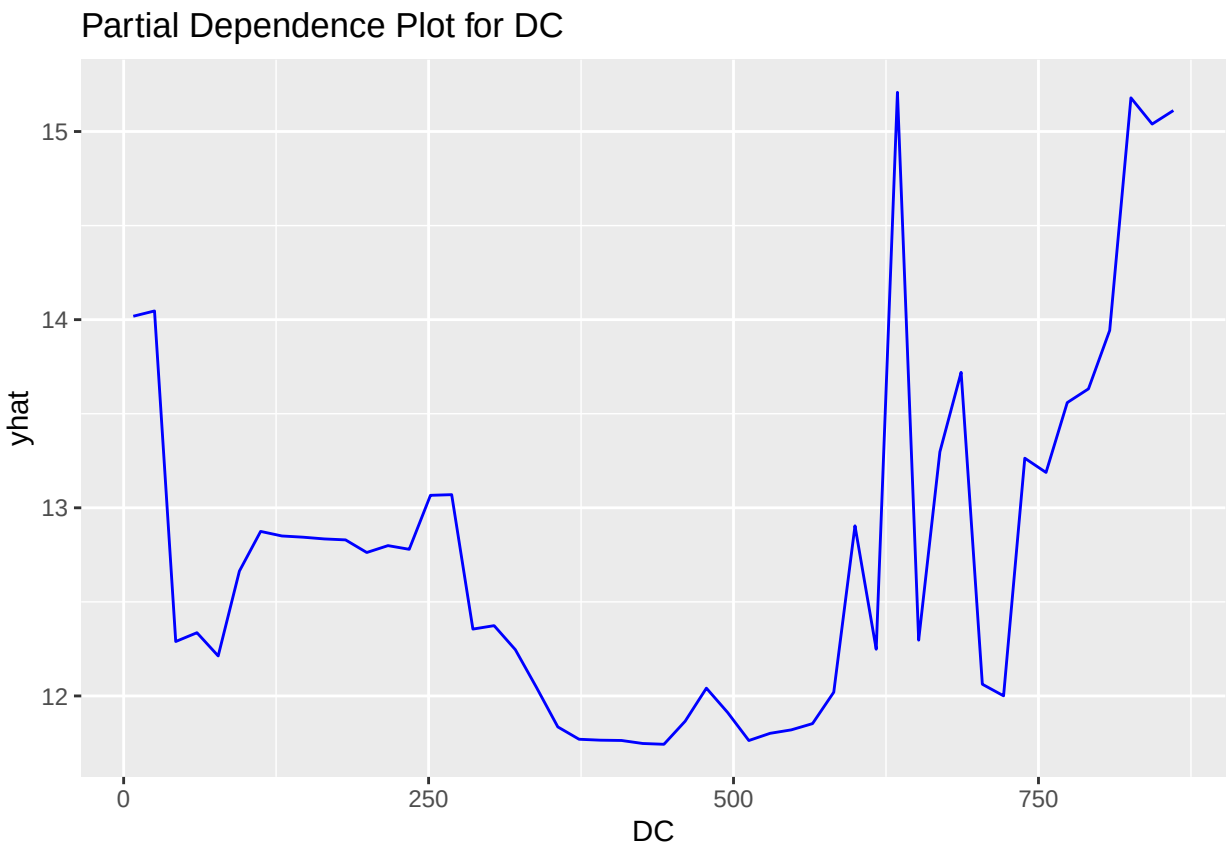
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation ideoms with `aes()`
```



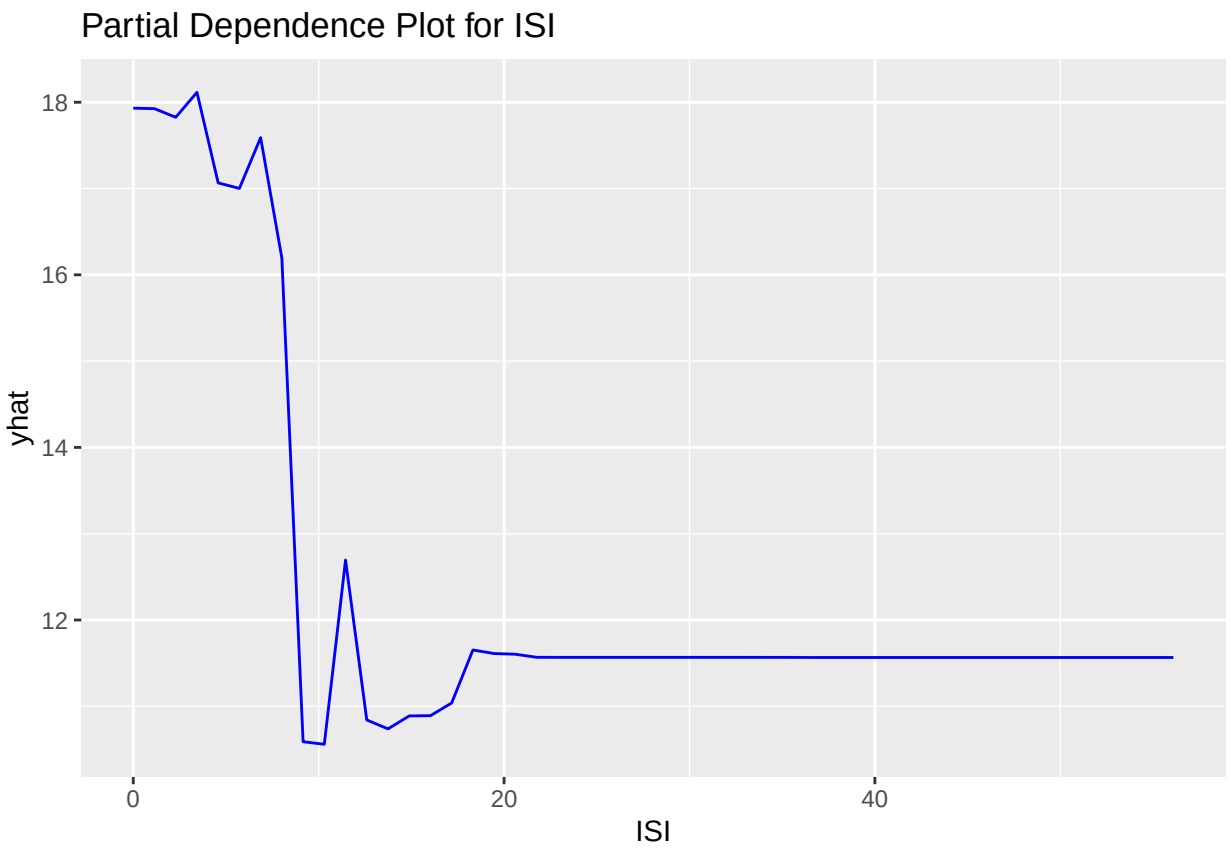

```
(ggplot_DMC <- pdp_to_ggplot(pdp_DMC, "DMC"))
```



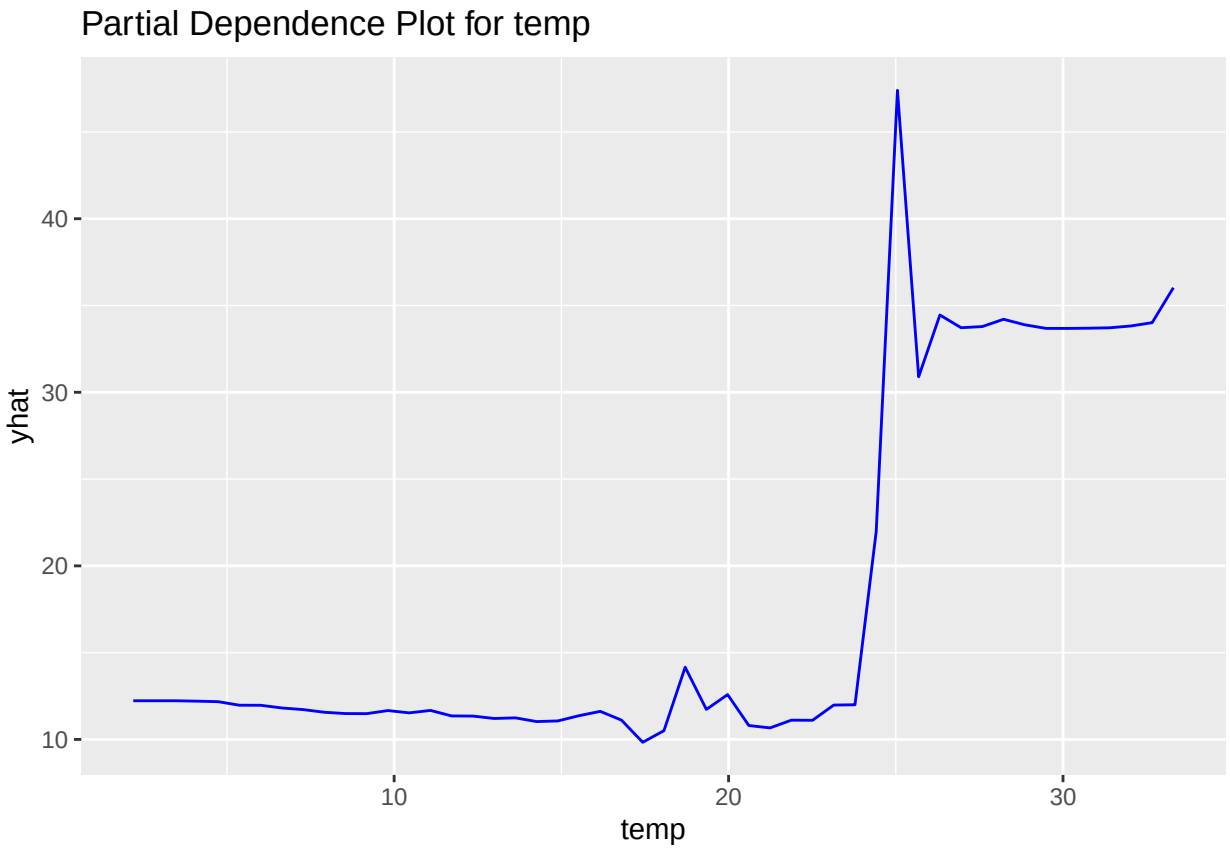

```
(ggplot_DC <- pdp_to_ggplot(pdp_DC , "DC"))
```



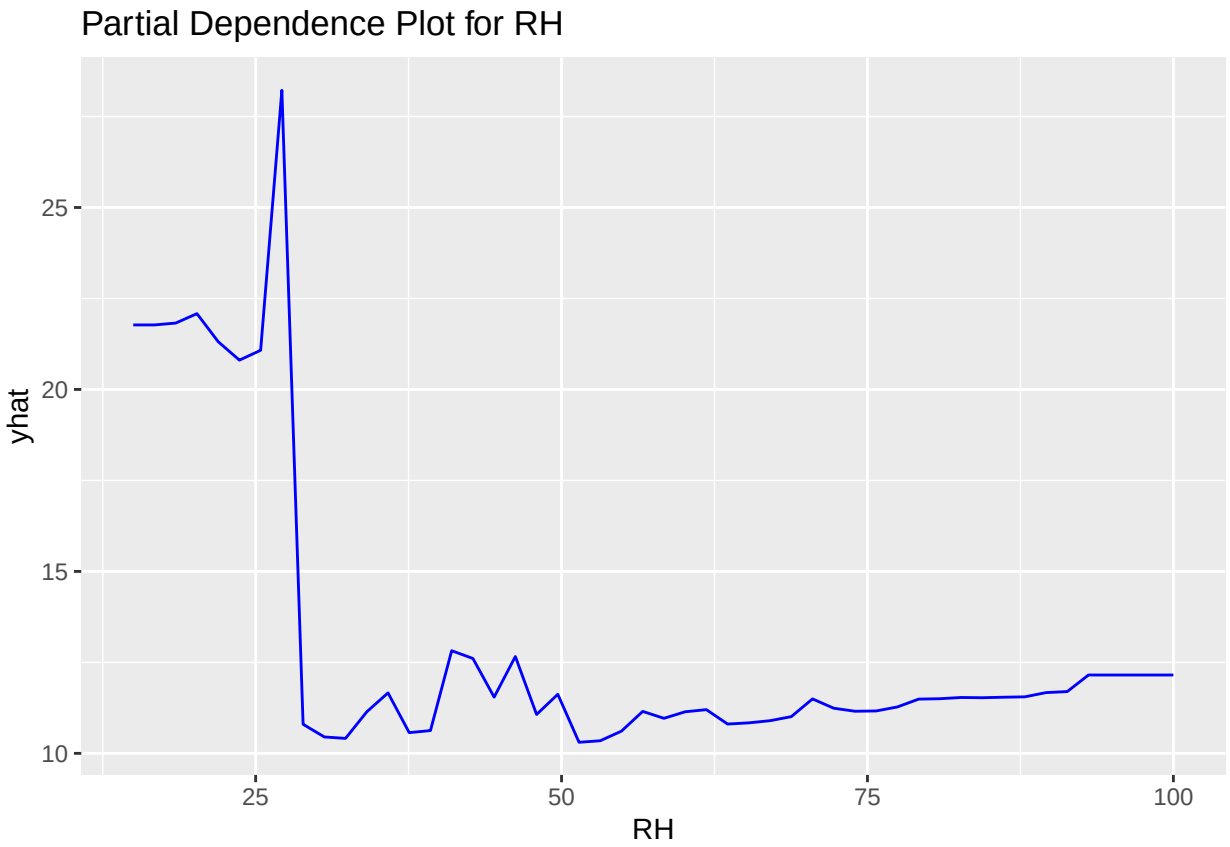

```
(ggplot_ISI <- pdp_to_ggplot(pdp_ISI, "ISI"))
```



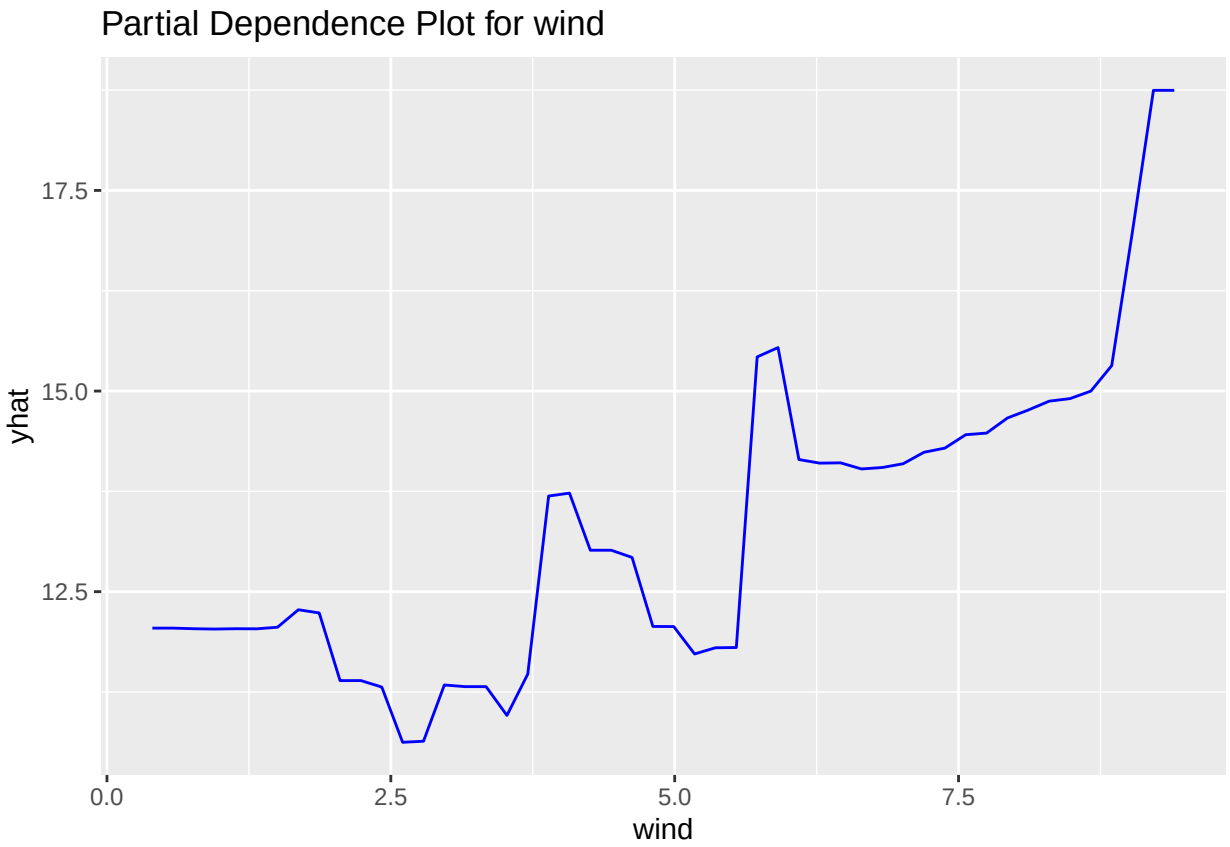

```
# Plot PDPs for temporal features
(ggplot_temp <- pdp_to_ggplot(pdp_temp, "temp"))
```



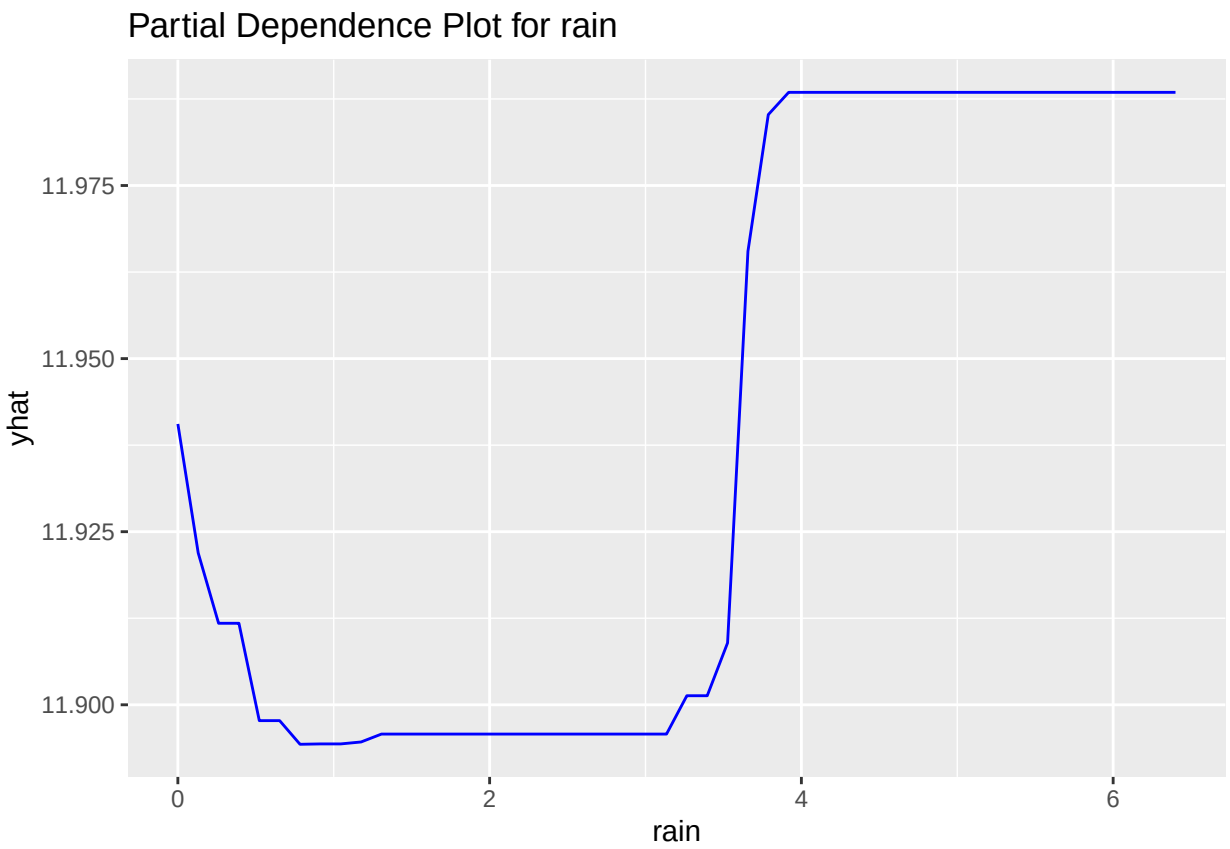

```
(ggplot_RH <- pdp_to_ggplot(pdp_RH, "RH"))
```




```
(ggplot_WIND <- pdp_to_ggplot(pdp_WIND, "wind"))
```




```
(ggplot_rain <- pdp_to_ggplot(pdp_rain, "rain"))
```



Simulation

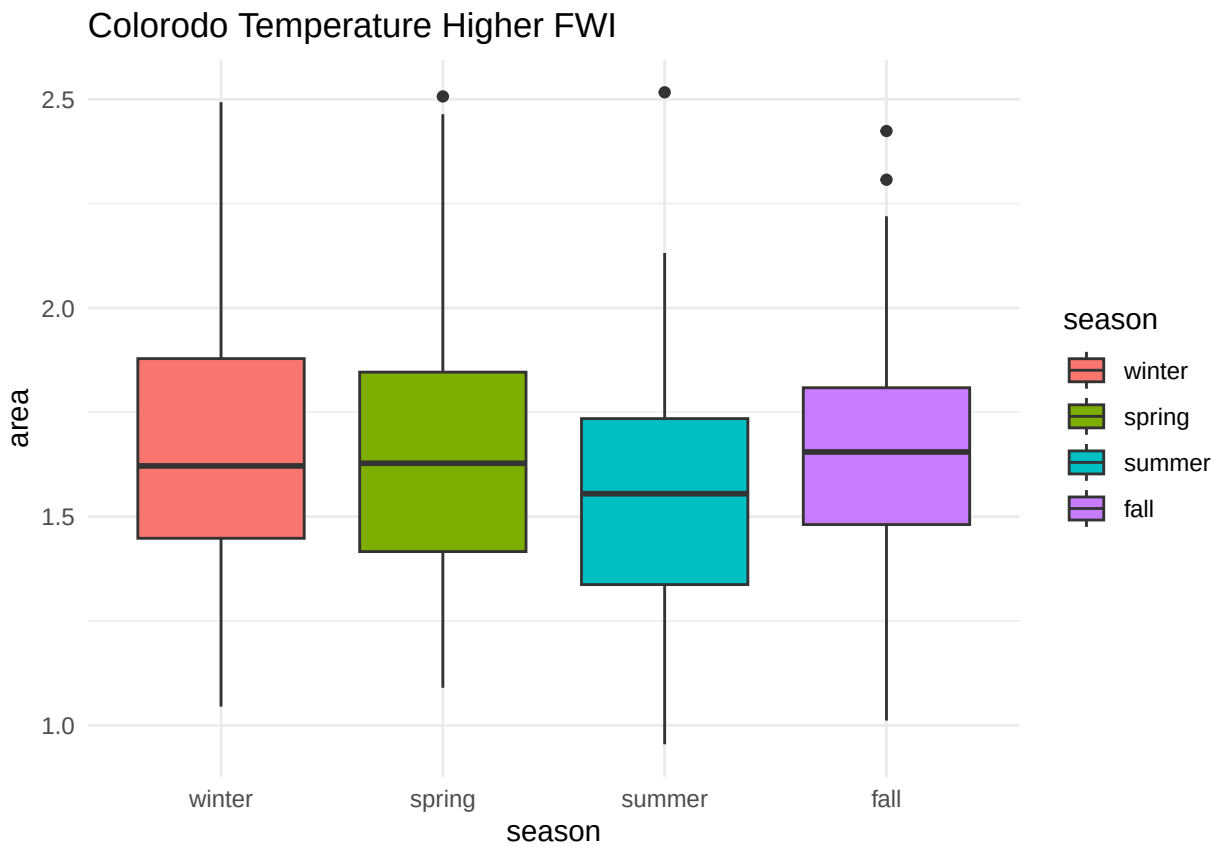
Scenario 1

```
summary(forestfires)
```

```
##        X               Y           month           day            FFMC
##  Min.   :1.000   Min.   :2.0   Min.   : 1.000   Min.   :1.000   Min.   :18.70
##  1st Qu.:3.000   1st Qu.:4.0   1st Qu.: 7.000   1st Qu.:2.000   1st Qu.:90.20
##  Median :4.000   Median :4.0   Median : 8.000   Median :5.000   Median :91.60
##  Mean   :4.669   Mean   :4.3   Mean   : 7.476   Mean   :4.259   Mean   :90.64
##  3rd Qu.:7.000   3rd Qu.:5.0   3rd Qu.: 9.000   3rd Qu.:6.000   3rd Qu.:92.90
##  Max.   :9.000   Max.   :9.0   Max.   :12.000   Max.   :7.000   Max.   :96.20
##       DMC              DC             ISI              temp
##  Min.   :  1.1   Min.   :  7.9   Min.   : 0.000   Min.   : 2.20
##  1st Qu.: 68.6   1st Qu.:437.7   1st Qu.: 6.500   1st Qu.:15.50
##  Median :108.3   Median :664.2   Median : 8.400   Median :19.30
##  Mean   :110.9   Mean   :547.9   Mean   : 9.022   Mean   :18.89
##  3rd Qu.:142.4   3rd Qu.:713.9   3rd Qu.:10.800   3rd Qu.:22.80
##  Max.   :291.3   Max.   :860.6   Max.   :56.100   Max.   :33.30
##        RH              wind            rain              area
##  Min.   : 15.00   Min.   :0.400   Min.   :0.00000   Min.   :   0.00
##  1st Qu.: 33.00   1st Qu.:2.700   1st Qu.:0.00000   1st Qu.:   0.00
##  Median : 42.00   Median :4.000   Median :0.00000   Median :   0.52
```

```
##  Mean   : 44.29   Mean   :4.018   Mean   :0.02166   Mean   :  12.85
##  3rd Qu.: 53.00   3rd Qu.:4.900   3rd Qu.:0.00000   3rd Qu.:   6.57
##  Max.   :100.00   Max.   :9.400   Max.   :6.40000   Max.   :1090.84
##      season
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :3.000
##  Mean   :2.207
##  3rd Qu.:3.000
##  Max.   :4.000
```

```r
# higher DMC DC ISI FFMC, same wind rh temp
n <- 1000
simulated_data <- data.frame(
  season = sample(c("winter", "spring", "summer", "fall"), n, replace = TRUE),
  FFMC = runif(n, min = 70, max = 90),
  DMC = runif(n, min = 100, max = 400),
  DC = runif(n, min = 300, max = 900),
  ISI = runif(n, min = 10, max = 70),
  temp = runif(n, min = 20, max = 35),
  RH = runif(n, min = 15, max = 100),
  wind = runif(n, min = 0, max = 10),
  rain = runif(n, min = 0, max = 5),
  X = sample(1:7, n, replace = TRUE),
  Y = sample(1:7, n, replace = TRUE),
  day = sample(1:7, n, replace = TRUE),
  month = sample(1:7, n, replace = TRUE),
  rain = sample(1:7, n, replace = TRUE),
  area = FALSE
)


simulated_data$temperature_F[simulated_data$season == "winter"] <- runif(sum(simulated_data$season == "
simulated_data$temperature_F[simulated_data$season == "spring"] <- runif(sum(simulated_data$season == "
simulated_data$temperature_F[simulated_data$season == "summer"] <- runif(sum(simulated_data$season == "
simulated_data$temperature_F[simulated_data$season == "fall"] <- runif(sum(simulated_data$season == "fa

# Convert temperatures from Fahrenheit to Celsius
simulated_data$temp <- (simulated_data$temperature_F)

rf_model2 <- randomForest(log(area+1) ~.-X-Y-month-rain, data = forest_train, mtry = 2)

# Predict area burned for the simulated data
simulated_data$area <-exp(predict(rf_model2, newdata = simulated_data)-1)

simulated_data$season <- factor(simulated_data$season, levels = c("winter", "spring", "summer", "fall"))

ggplot(simulated_data, aes(x = season,area,fill = season)) +geom_boxplot() + labs(title = "Colorodo Temp
```




```
summary(forestfires)
```

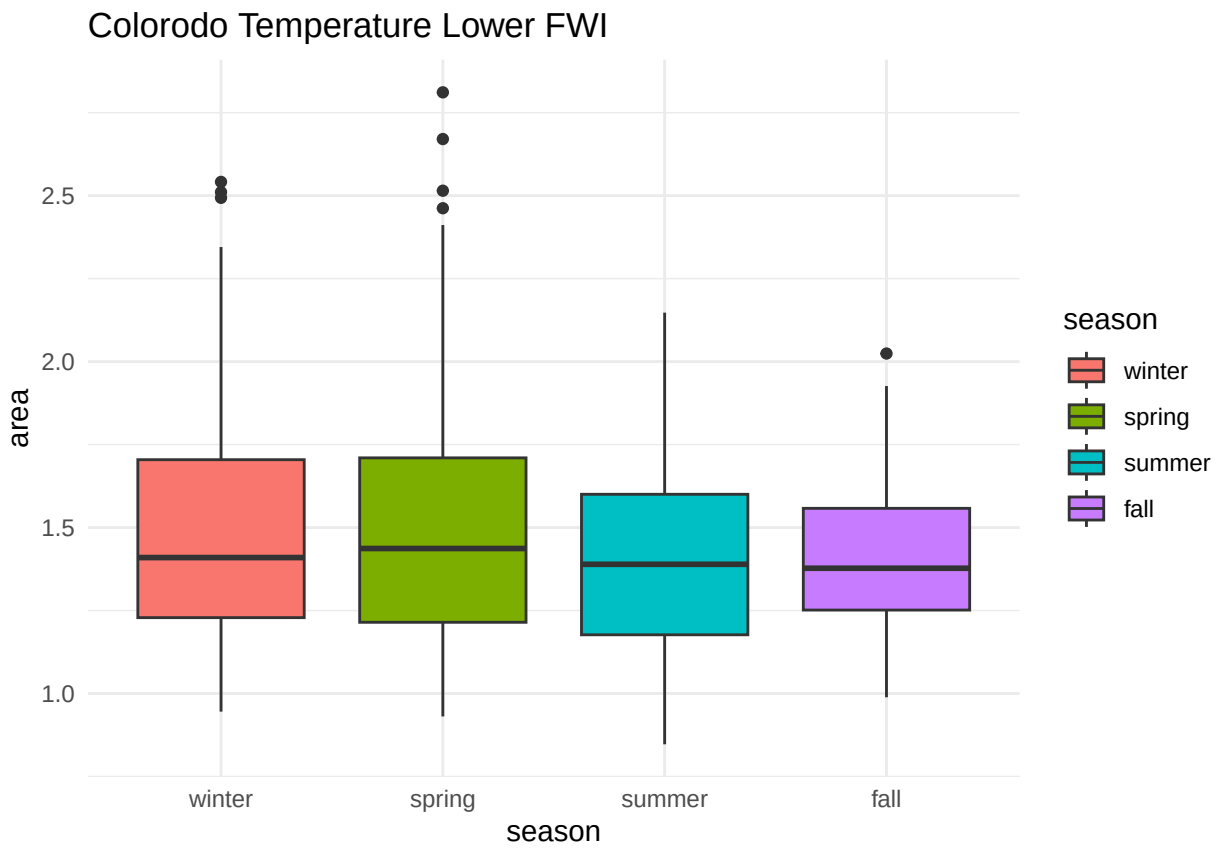
```
##        X               Y              month              day              FFMC
##  Min.   :1.000   Min.   :2.0   Min.   : 1.000   Min.   :1.000   Min.   :18.70
##  1st Qu.:3.000   1st Qu.:4.0   1st Qu.: 7.000   1st Qu.:2.000   1st Qu.:90.20
##  Median :4.000   Median :4.0   Median : 8.000   Median :5.000   Median :91.60
##  Mean   :4.669   Mean   :4.3   Mean   : 7.476   Mean   :4.259   Mean   :90.64
##  3rd Qu.:7.000   3rd Qu.:5.0   3rd Qu.: 9.000   3rd Qu.:6.000   3rd Qu.:92.90
##  Max.   :9.000   Max.   :9.0   Max.   :12.000   Max.   :7.000   Max.   :96.20
##       DMC              DC              ISI              temp
##  Min.   :  1.1   Min.   :  7.9   Min.   : 0.000   Min.   : 2.20
##  1st Qu.: 68.6   1st Qu.:437.7   1st Qu.: 6.500   1st Qu.:15.50
##  Median :108.3   Median :664.2   Median : 8.400   Median :19.30
##  Mean   :110.9   Mean   :547.9   Mean   : 9.022   Mean   :18.89
##  3rd Qu.:142.4   3rd Qu.:713.9   3rd Qu.:10.800   3rd Qu.:22.80
##  Max.   :291.3   Max.   :860.6   Max.   :56.100   Max.   :33.30
##        RH              wind            rain              area
##  Min.   : 15.00   Min.   :0.400   Min.   :0.00000   Min.   :   0.00
##  1st Qu.: 33.00   1st Qu.:2.700   1st Qu.:0.00000   1st Qu.:   0.00
##  Median : 42.00   Median :4.000   Median :0.00000   Median :   0.52
##  Mean   : 44.29   Mean   :4.018   Mean   :0.02166   Mean   :  12.85
##  3rd Qu.: 53.00   3rd Qu.:4.900   3rd Qu.:0.00000   3rd Qu.:   6.57
##  Max.   :100.00   Max.   :9.400   Max.   :6.40000   Max.   :1090.84
##      season
##  Min.   :1.000
```

```
##  1st Qu.:1.000
##  Median :3.000
##  Mean   :2.207
##  3rd Qu.:3.000
##  Max.   :4.000
```

```
# higher DMC DC ISI FFMC, same wind rh temp
n < - 1000
```

```
## [1] FALSE
```

```
simulated_data <- data.frame(
  season = sample(c("winter", "spring", "summer", "fall"), n, replace = TRUE),
  FFMC = runif(n, min = 50, max = 70),
  DMC = runif(n, min = 50, max = 300),
  DC = runif(n, min = 100, max = 600),
  ISI = runif(n, min = 10, max = 50),
  temp = runif(n, min = 20, max = 35),
  RH = runif(n, min = 15, max = 100),
  wind = runif(n, min = 0, max = 10),
  rain = runif(n, min = 0, max = 5),
  X = sample(1:7, n, replace = TRUE),
  Y = sample(1:7, n, replace = TRUE),
  day = sample(1:7, n, replace = TRUE),
  month = sample(1:7, n, replace = TRUE),
  rain = sample(1:7, n, replace = TRUE),
  area = FALSE
)

simulated_data$temperature_F[simulated_data$season == "winter"] <- runif(sum(simulated_data$season == "
simulated_data$temperature_F[simulated_data$season == "spring"] <- runif(sum(simulated_data$season == "
simulated_data$temperature_F[simulated_data$season == "summer"] <- runif(sum(simulated_data$season == "
simulated_data$temperature_F[simulated_data$season == "fall"] <- runif(sum(simulated_data$season == "fa

# Convert temperatures from Fahrenheit to Celsius
simulated_data$temp <- (simulated_data$temperature_F)

rf_model2 <- randomForest(log(area+1) ~.-X-Y-month-rain, data = forest_train, mtry = 2)

# Predict area burned for the simulated data
simulated_data$area <-exp(predict(rf_model2, newdata = simulated_data)-1)

simulated_data$season <- factor(simulated_data$season, levels = c("winter", "spring", "summer", "fall"))

ggplot(simulated_data, aes(x = season,area,fill = season)) +geom_boxplot() + labs(title = "Colorodo Temp
```


Colorodo Temperature Lower FWI
area
2.5
2.0
1.5
1.0
winter
spring
summer
fall
season
season
winter
spring
summer
fall