# Load Data

Loading the data

```
df = read.csv('C:\\Users\\Dell\\OneDrive\\College_2nd\\R Lab\\forestfires.csv')
```

#Statistical Summary

Looking at the data

```
head(df)
```

```
##   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
```

# Statistical summary of the data

```
summary(df)
```

```
##        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  
```

# Structure of the dataframe

```
str(df)
```

```
## 'data.frame':	517 obs. of  13 variables:
##  $ X    : int  7 7 7 8 8 8 8 8 8 7 ...
##  $ Y    : int  5 4 4 6 6 6 6 6 6 5 ...
##  $ month: chr  "mar" "oct" "oct" "mar" ...
##  $ day  : chr  "fri" "tue" "sat" "fri" ...
##  $ FFMC : num  86.2 90.6 90.6 91.7 89.3 92.3 92.3 91.5 91 92.5 ...
##  $ DMC  : num  26.2 35.4 43.7 33.3 51.3 ...
##  $ DC   : num  94.3 669.1 686.9 77.5 102.2 ...
##  $ ISI  : num  5.1 6.7 6.7 9 9.6 14.7 8.5 10.7 7 7.1 ...
##  $ temp : num  8.2 18 14.6 8.3 11.4 22.2 24.1 8 13.1 22.8 ...
##  $ RH   : int  51 33 33 97 99 29 27 86 63 40 ...
##  $ wind : num  6.7 0.9 1.3 4 1.8 5.4 3.1 2.2 5.4 4 ...
##  $ rain : num  0 0 0 0.2 0 0 0 0 0 0 ...
##  $ area : num  0 0 0 0 0 0 0 0 0 0 ...
```

We can see that some values are factors and are termed as chr or num, we will change them after further investigation

# Missing Values

```
any(is.na(df))
```

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

We dont have any missing values

#Plotting

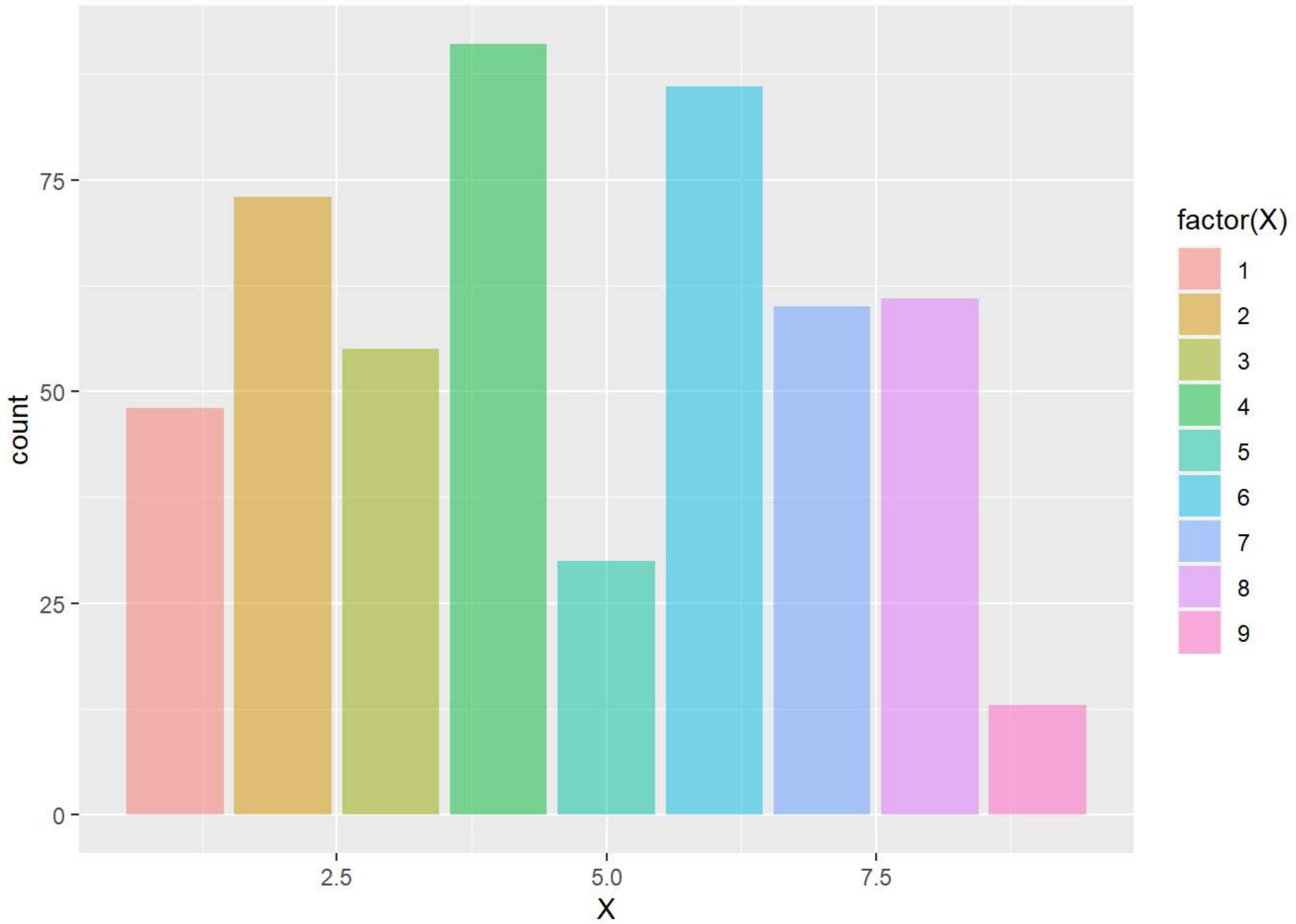
Library for plotting

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.4
```

##Plotting X

```
ggplot(df,aes(X))+
        geom_bar(aes(fill=factor(X)),alpha=0.5)
```

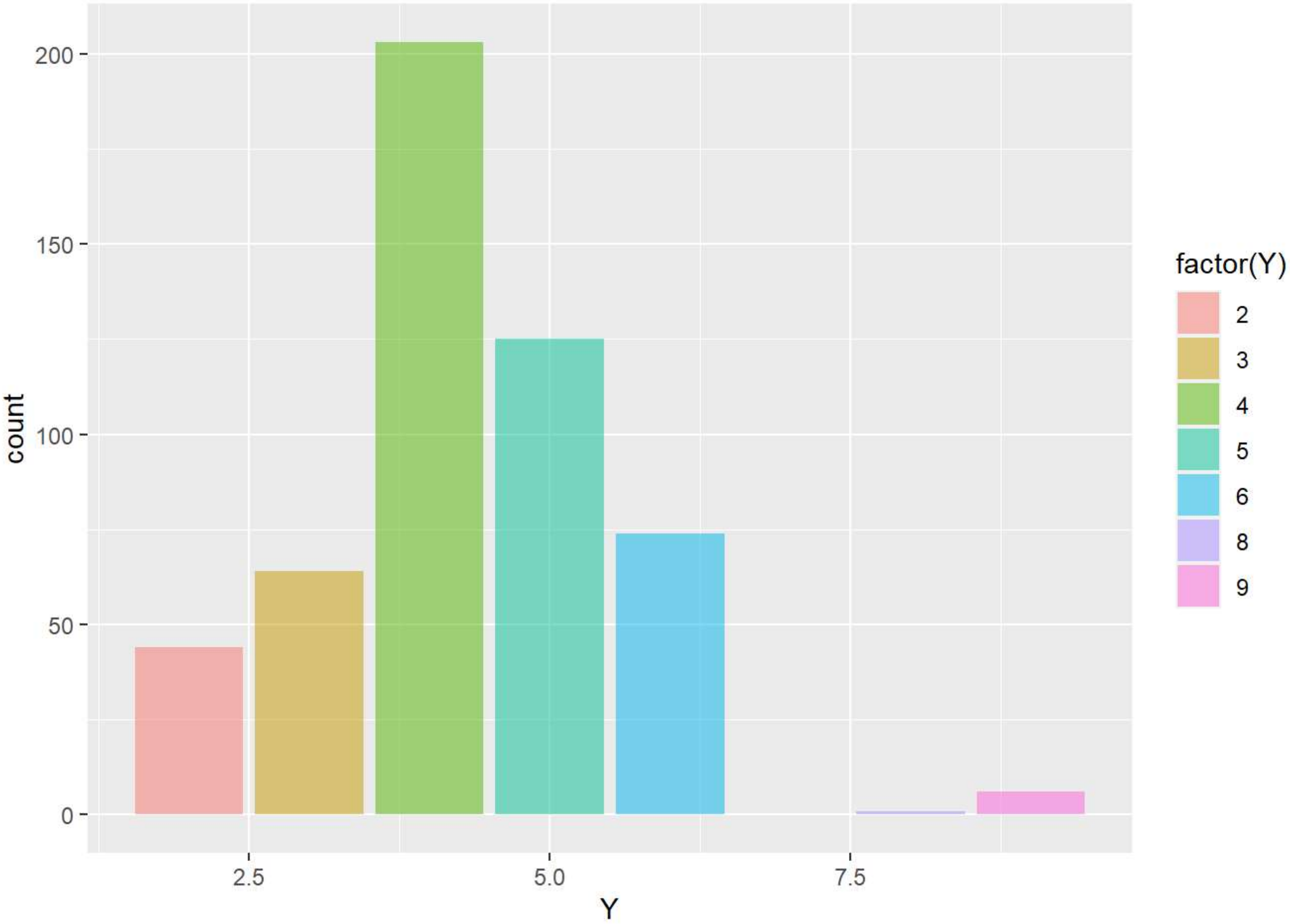



Since X is a factor variable, we will change X to factor

```
df$X = as.factor(df$X)
```

##Plotting Y

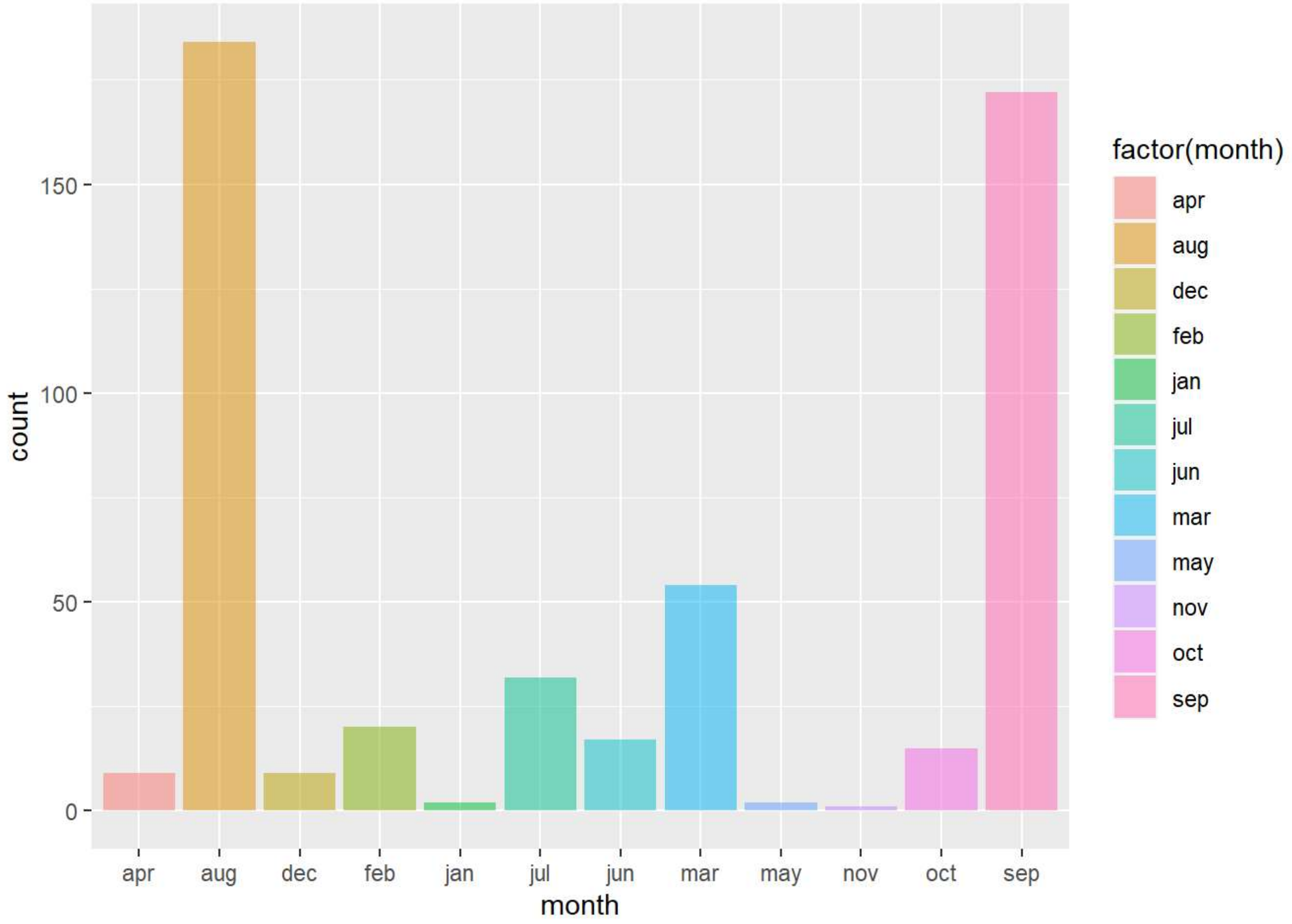
```
ggplot(df,aes(Y))+
        geom_bar(aes(fill=factor(Y)),alpha=0.5)
```




```
df$Y = as.factor(df$Y)
```

# Plotting Month

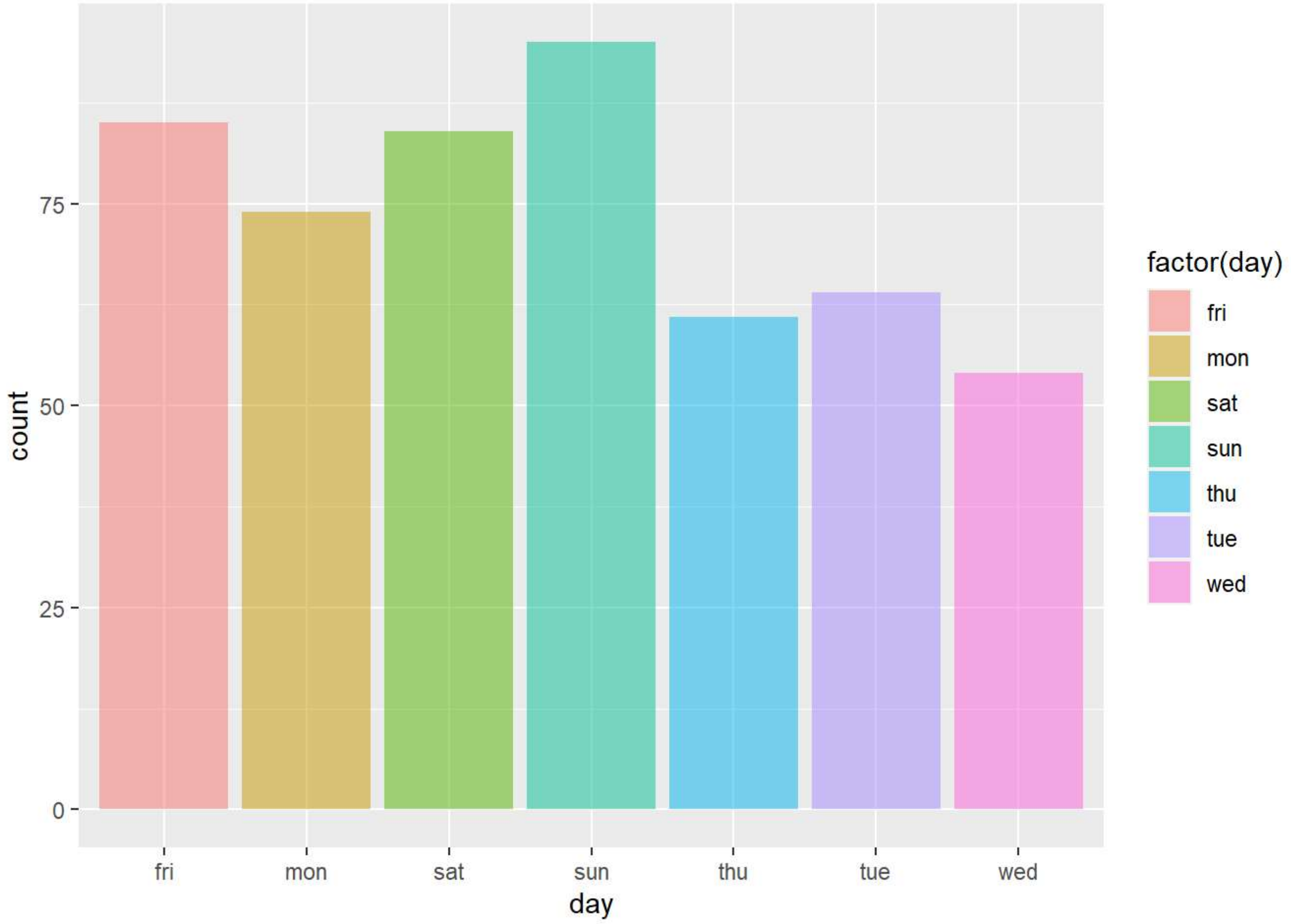
```
ggplot(df,aes(month))+
        geom_bar(aes(fill=factor(month)),alpha=0.5)
```




```
df$month = as.factor(df$month)
```

# Plotting Day

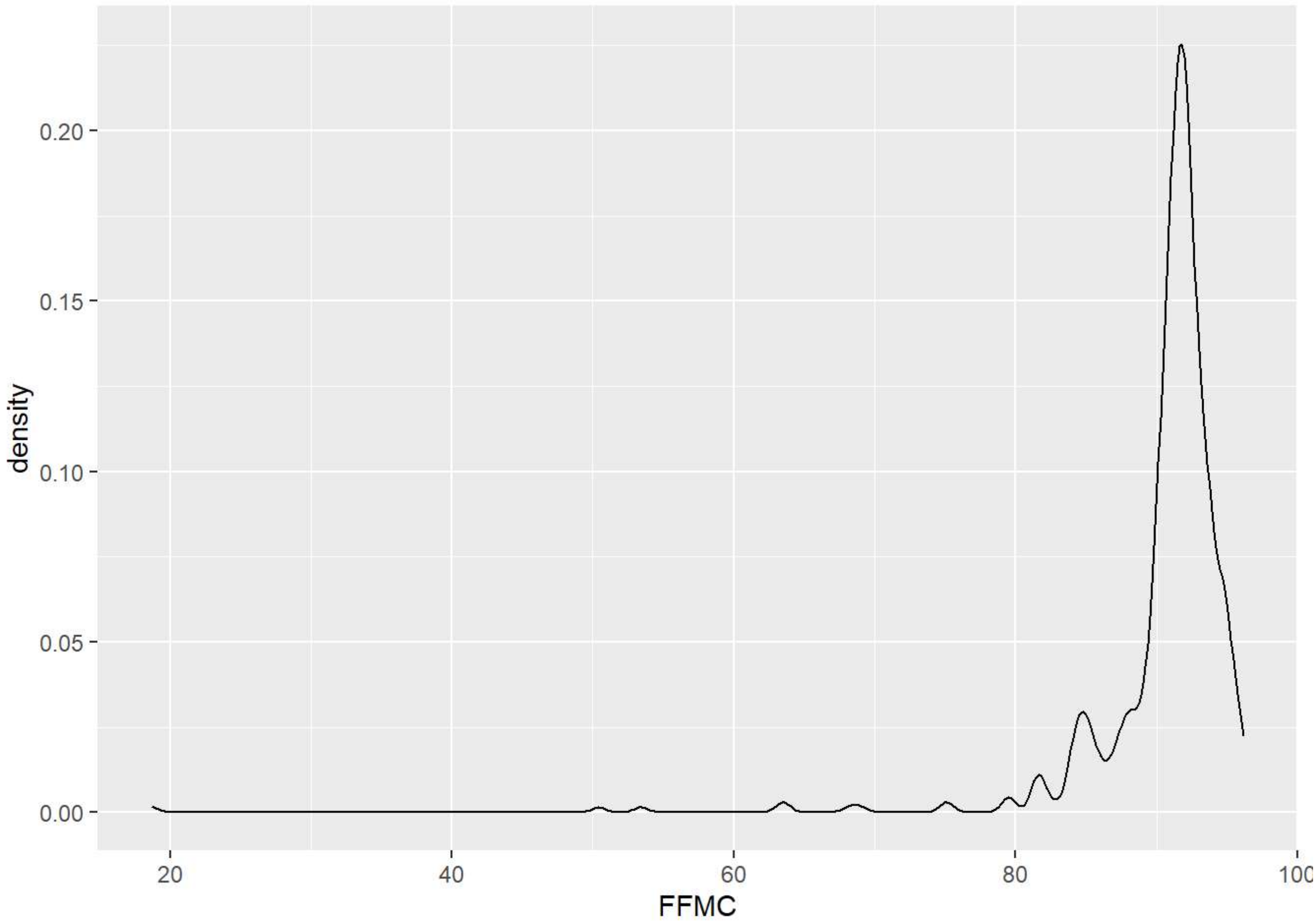
```
ggplot(df,aes(day))+
        geom_bar(aes(fill=factor(day)),alpha=0.5)
```




```
df$day = as.factor(df$day)
```

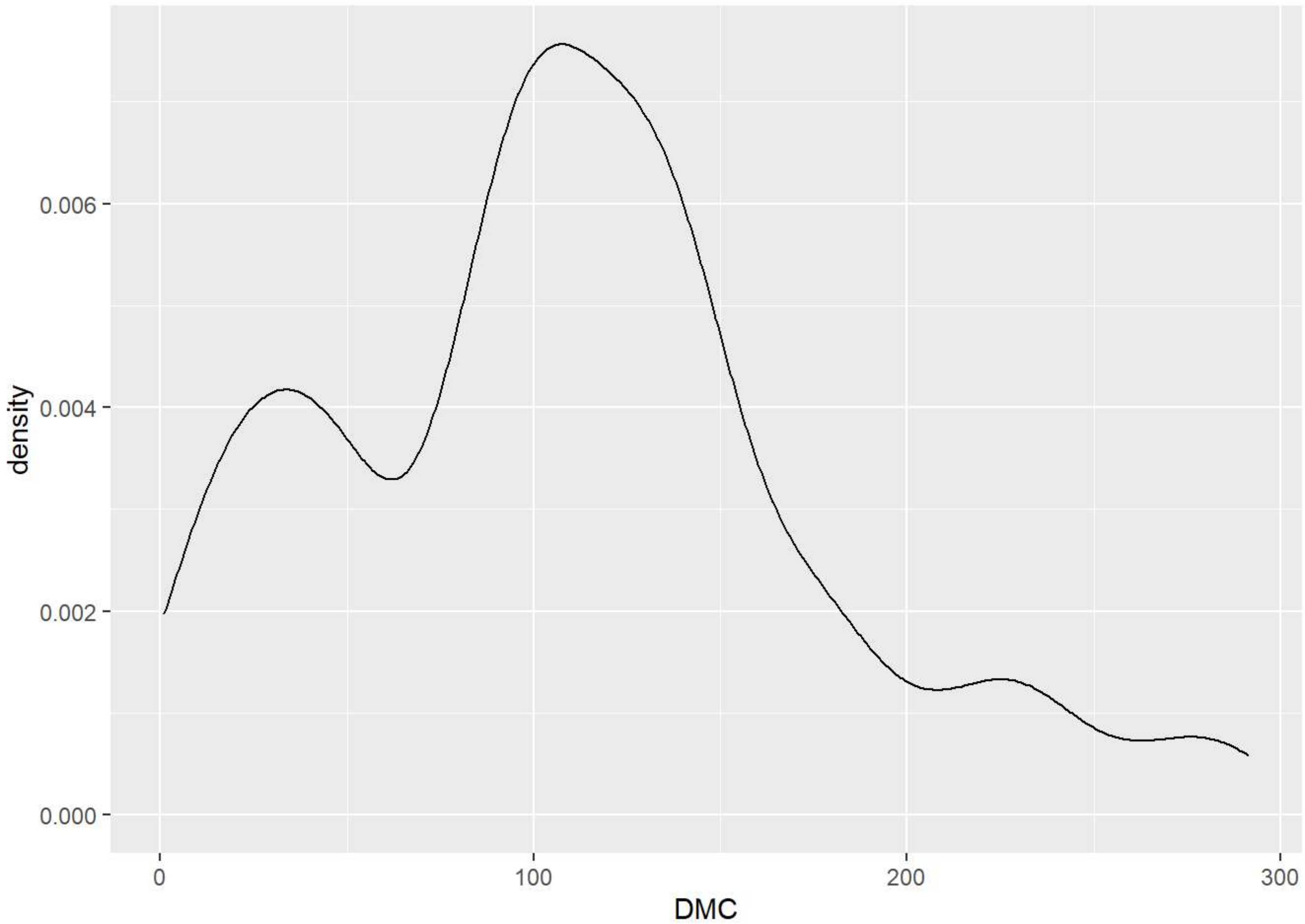
# Plotting FFMC

```
ggplot(df, aes(FFMC)) +
  geom_density(alpha=0.5)
```



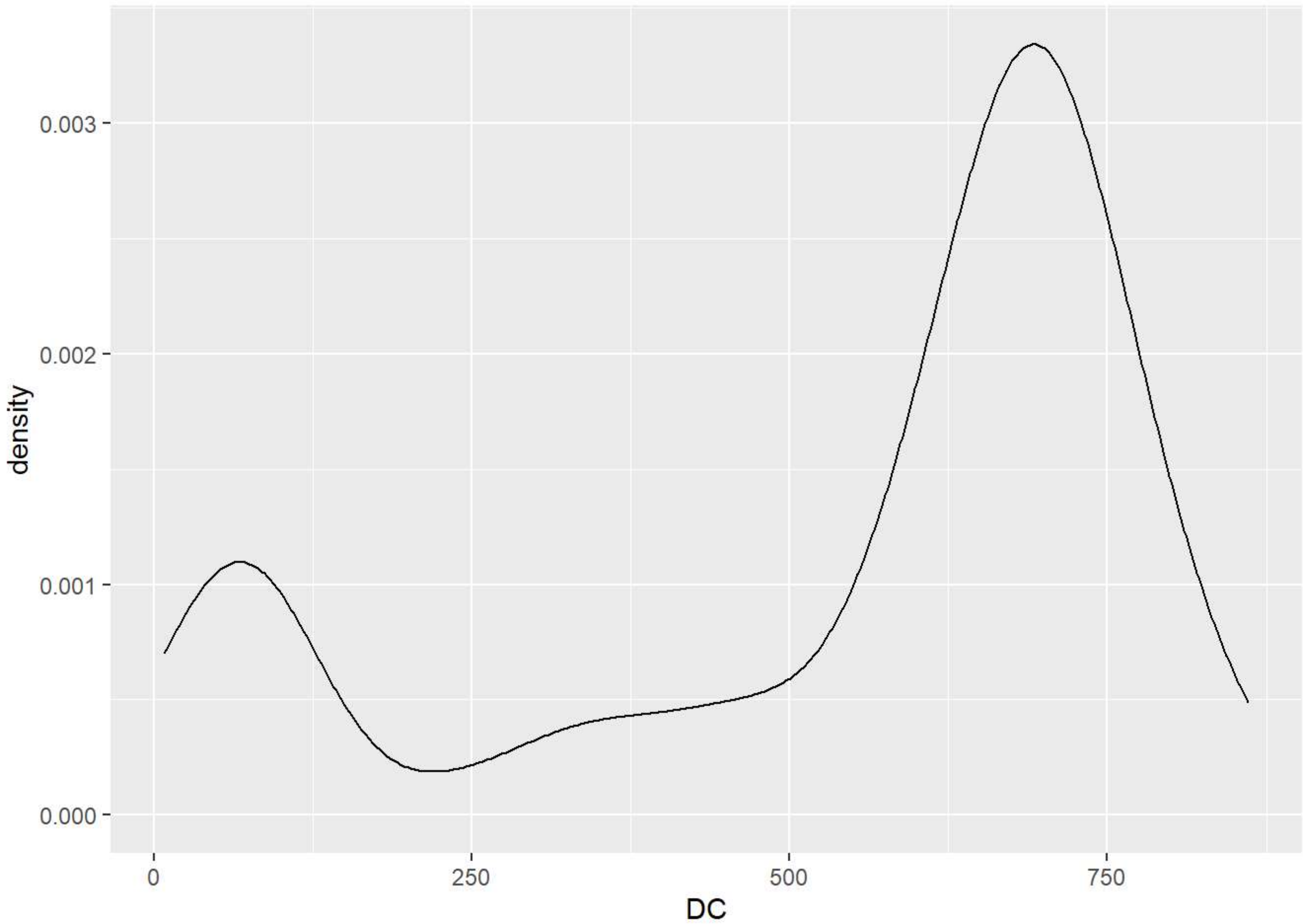

# Plotting DMC

```
ggplot(df, aes(DMC)) +
  geom_density(alpha=0.5)
```



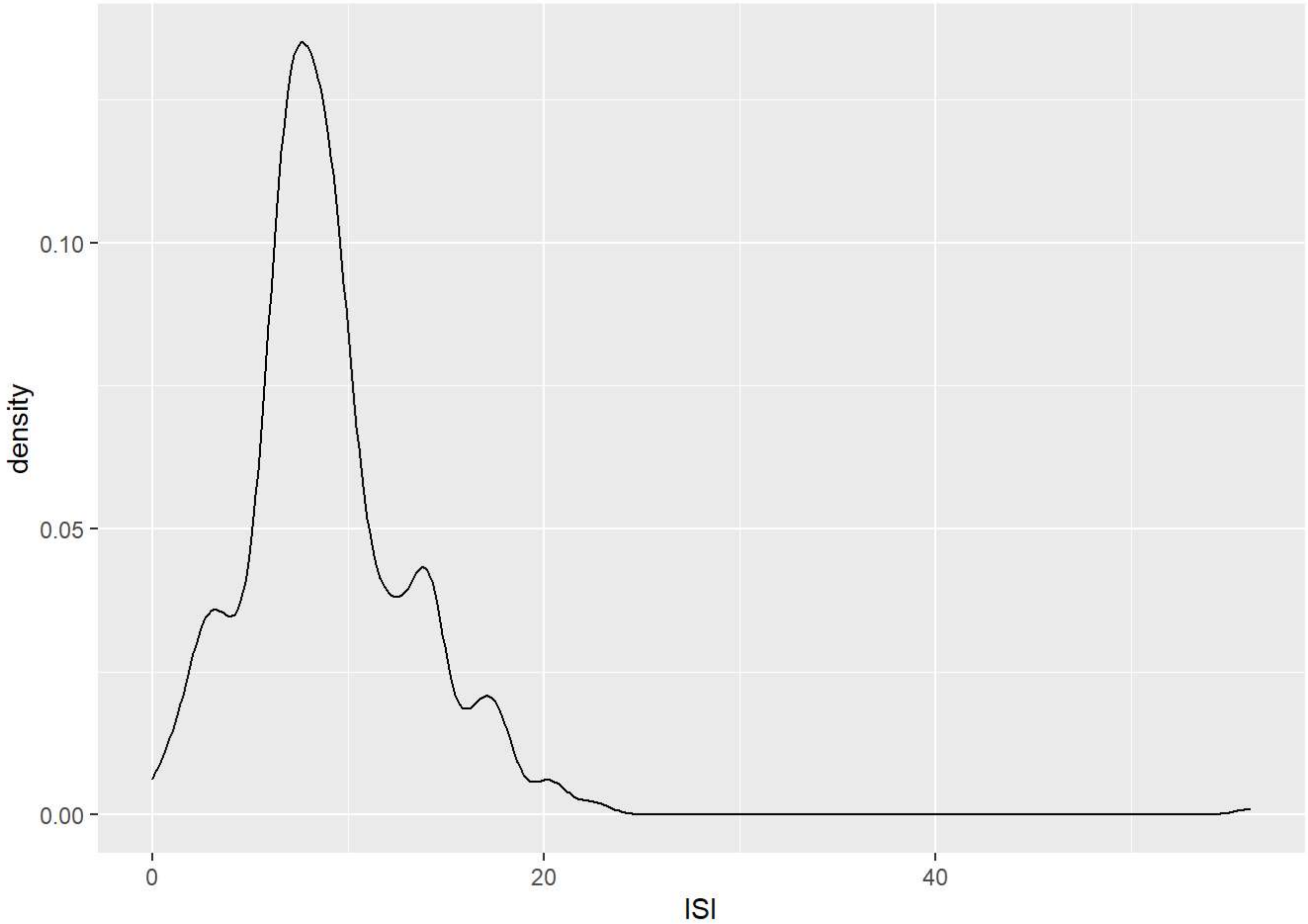

# Plotting DC

```
ggplot(df, aes(DC)) +
  geom_density(alpha=0.5)
```



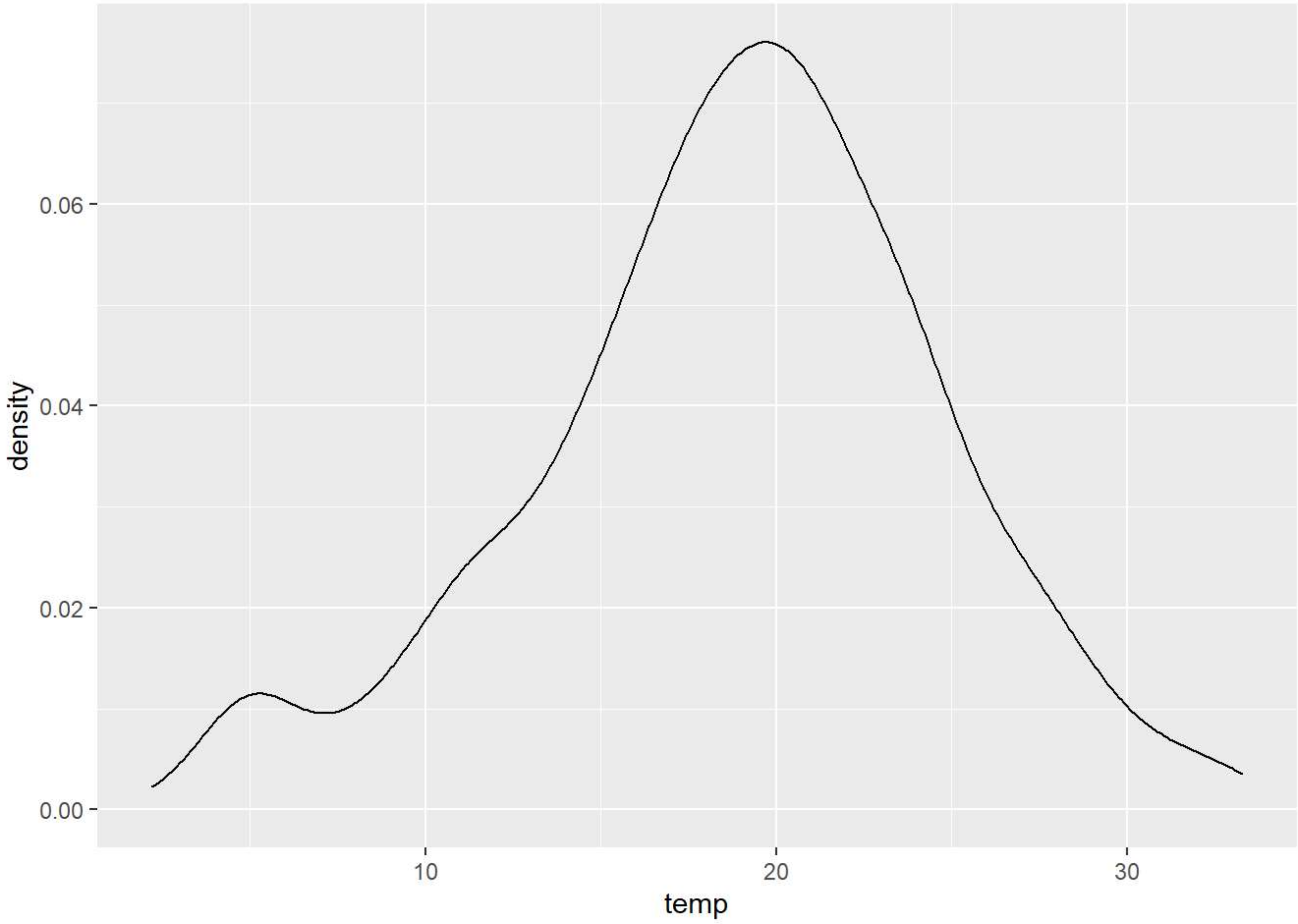

# Plotting ISI

```
ggplot(df, aes(ISI)) +
  geom_density(alpha=0.5)
```



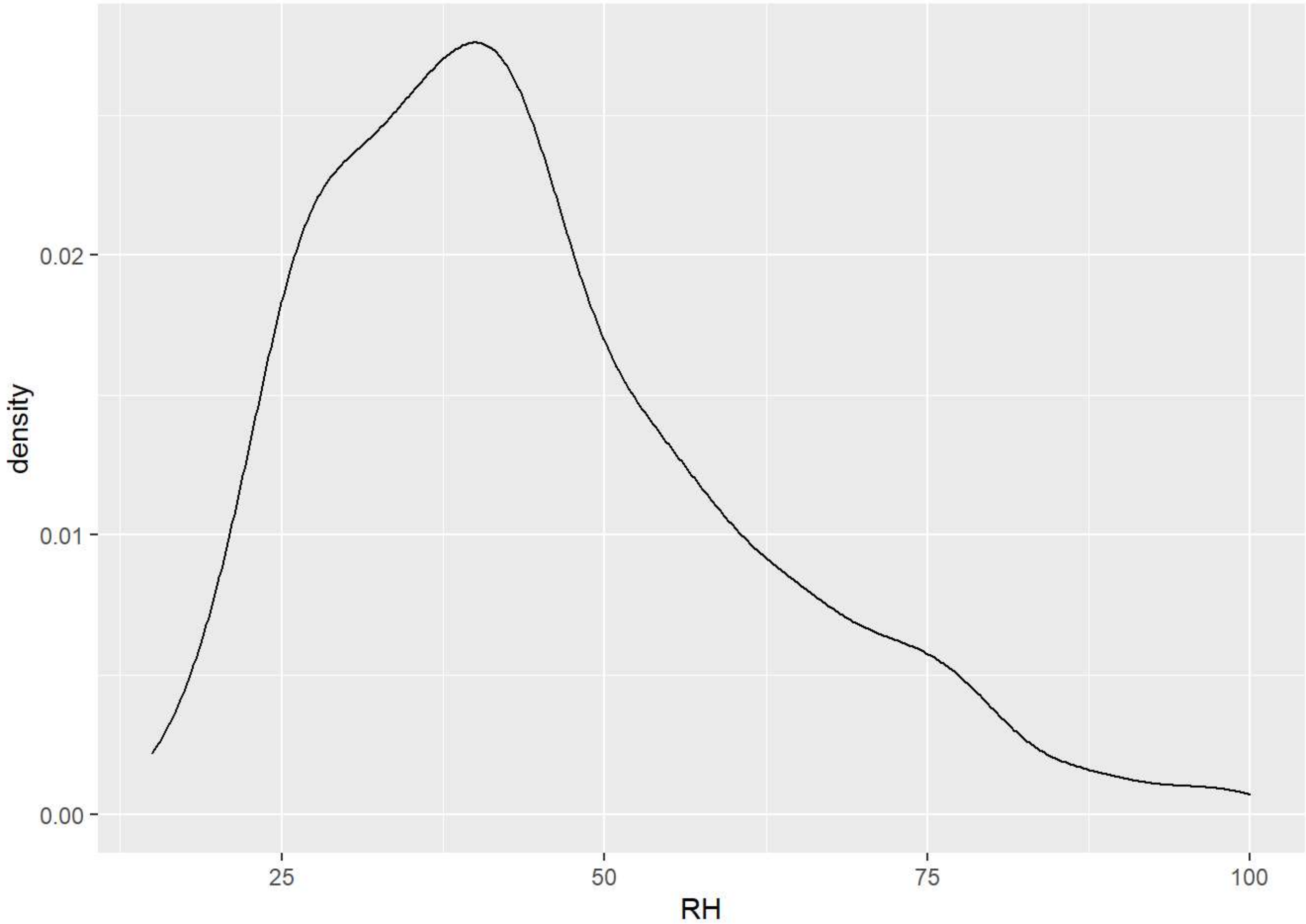

# Plotting temp

```
ggplot(df, aes(temp)) +
  geom_density(alpha=0.5)
```



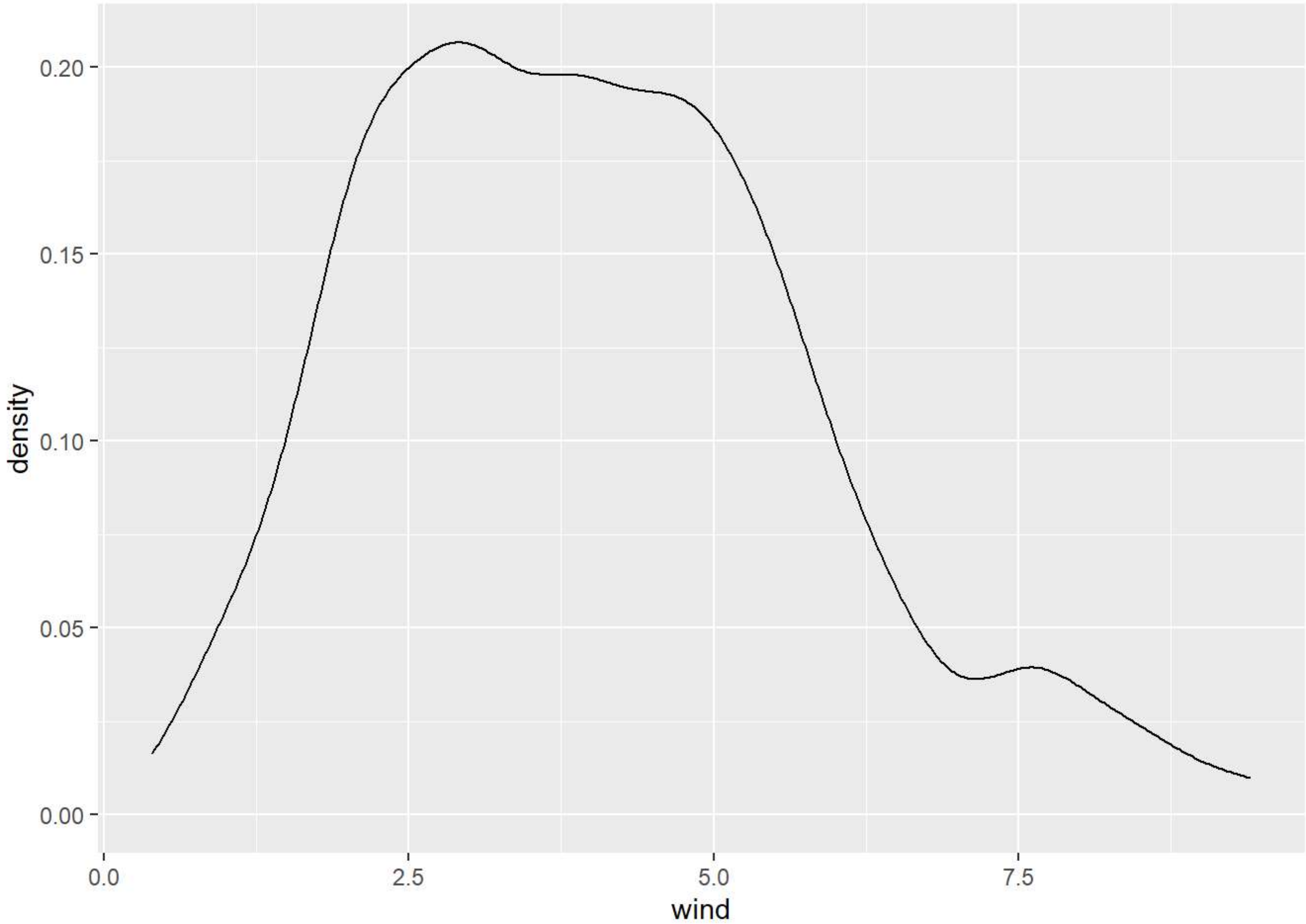

# Plotting RH

```
ggplot(df, aes(RH)) +
  geom_density(alpha=0.5)
```



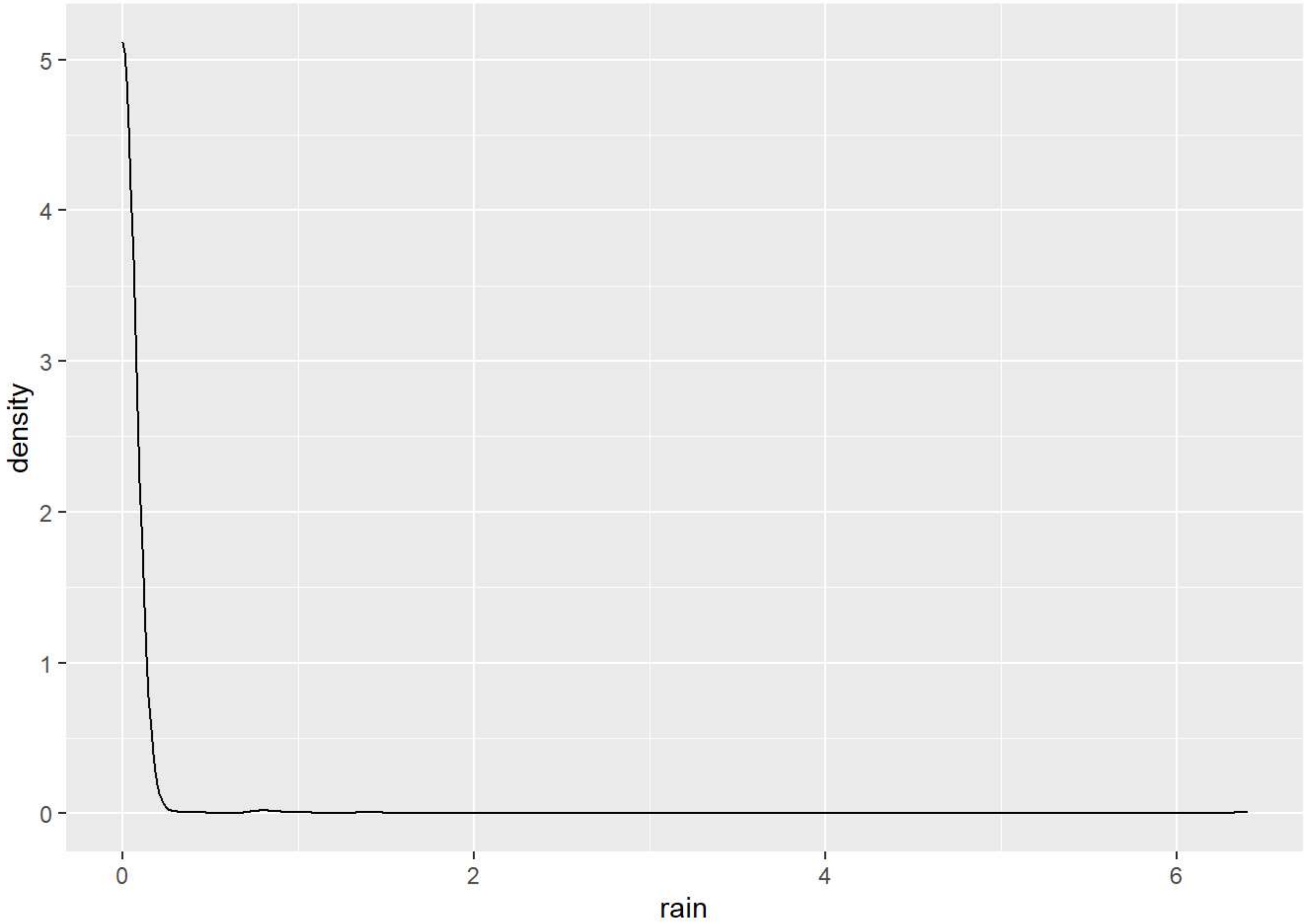

# Plotting wind

```
ggplot(df, aes(wind)) +
  geom_density(alpha=0.5)
```



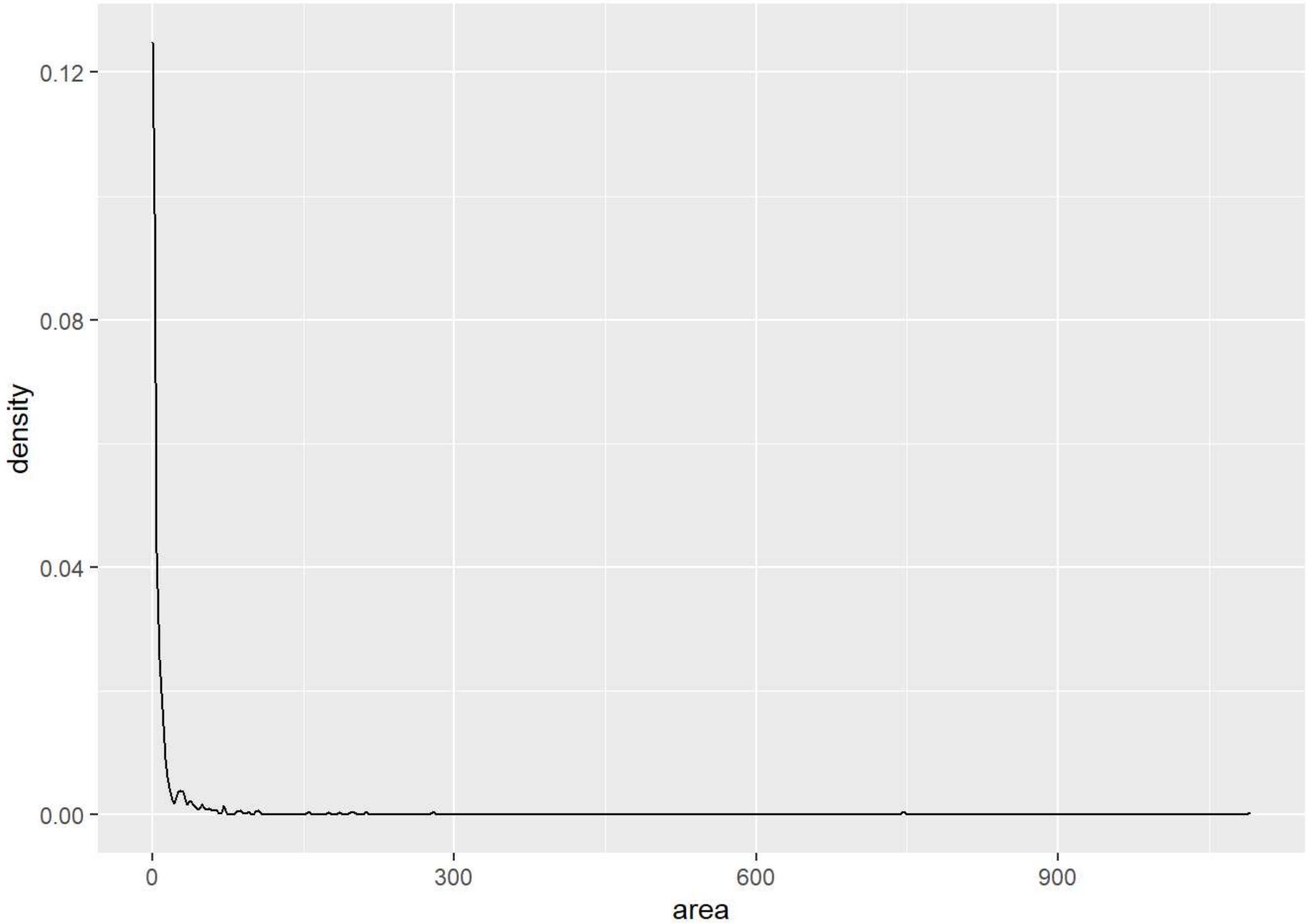

# Plotting rain

```
ggplot(df, aes(rain)) +
  geom_density(alpha=0.5)
```




# Plotting area

```
ggplot(df, aes(area)) +
  geom_density(alpha=0.5)
```




# Modelling

Libraries needed

```
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.0.5
```

Create test and train The following function creates a general train test dataset, catTools can also be used.

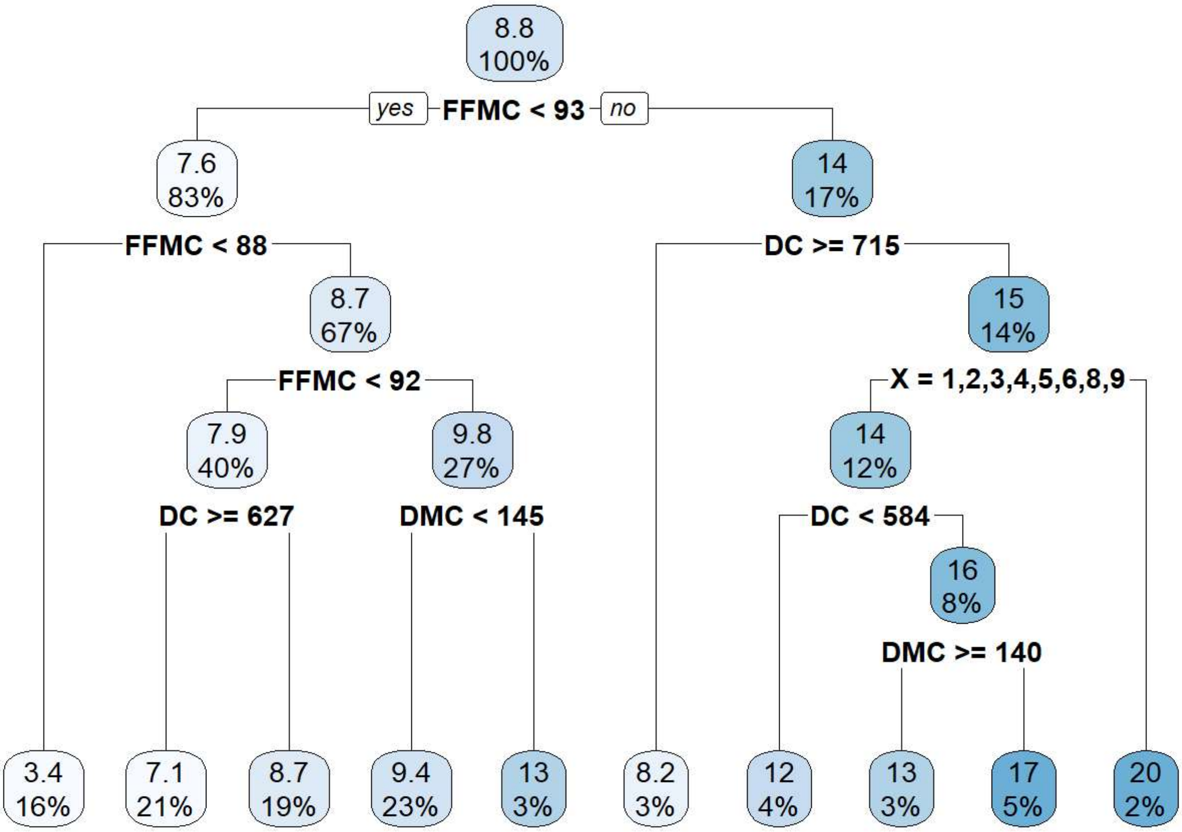
```
create_train_test <- function(data, size = 0.8, train = TRUE)
{
  #' create_train_test(df, size = 0.8, train = TRUE)
  #' arguments:
  #' @param df: Dataset used to train the model.
  #' @param size: Size of the split. By default, 0.8. Numerical value
  #' @param train: If set to `TRUE`, the function creates the train set, otherwise the test s
et. Default value sets to `TRUE`. Boolean value.You      need to add a Boolean parameter beca
use R does not allow to return two data frames simultaneously.

  #' @return test/train data


    n_row = nrow(data)
    total_row = size * n_row
    train_sample <- 1: total_row
    if (train == TRUE) {
        return (data[train_sample, ])
    } else {
        return (data[-train_sample, ])
    }
}
```

Getting data

```
data_train <- create_train_test(df, 0.8, train = TRUE)
data_test <- create_train_test(df, 0.8, train = FALSE)
```

fITTING the data

```
fit <- rpart(ISI~., data = data_train, method = 'anova')
rpart.plot(fit)
```




From the tree, we can understand that, FFMC, DC, DMC and X are important for the prediction of ISI. Based on the tree, we can say which data can occur with percentage value given. For FFMC less than 93, DC is checked and if FFMC is greater than 93, it is again checked against 88. This happens **because of the greedy approach taken by decision trees, which does not take into account what other parts can do, it goes with first best**.

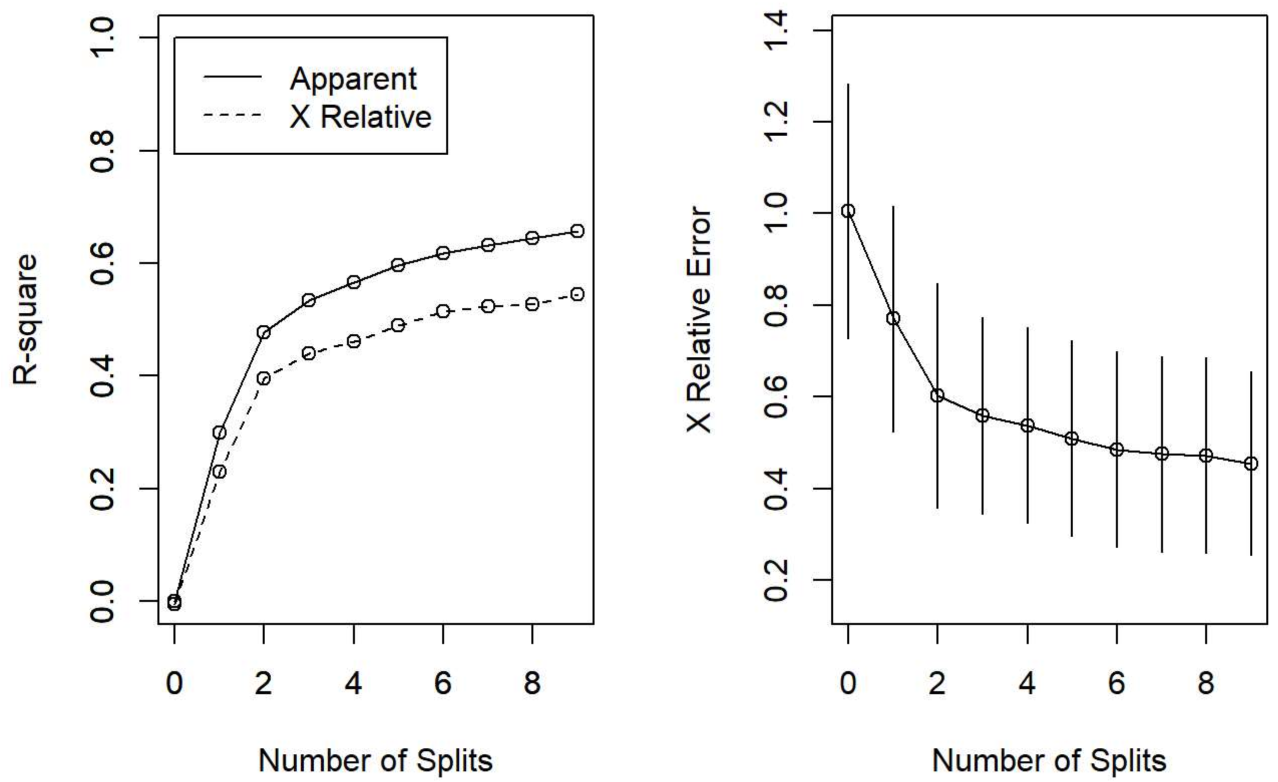
```
fit
```

```
## n= 413
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 413 8287.2320000  8.750363
##    2) FFMC< 93.05 343 3118.3970000  7.642274
##      4) FFMC< 88.15 67  164.2478000  3.432836 *
##      5) FFMC>=88.15 276 1478.7550000  8.664130
##       10) FFMC< 91.75 165  572.0124000  7.869091
##         20) DC>=626.65 88  211.6899000  7.126136 *
##         21) DC< 626.65 77  256.2345000  8.718182 *
##       11) FFMC>=91.75 111  647.4157000  9.845946
##         22) DMC< 144.8 97  407.7464000  9.362887 *
##         23) DMC>=144.8 14   60.2092900 13.192860 *
##    3) FFMC>=93.05 70 2684.0120000 14.180000
##      6) DC>=714.5 11    0.1472727  8.154545 *
##      7) DC< 714.5 59 2210.0390000 15.303390
##       14) X=1,2,3,4,5,6,8,9 51  411.3192000 14.496080
##         28) DC< 583.9 17   55.1611800 12.358820 *
##         29) DC>=583.9 34  239.6776000 15.564710
##           58) DMC>=140.3 14   41.5000000 13.400000 *
##           59) DMC< 140.3 20   86.6520000 17.080000 *
##       15) X=7 8 1553.5800000 20.450000 *
```

# Evaluations

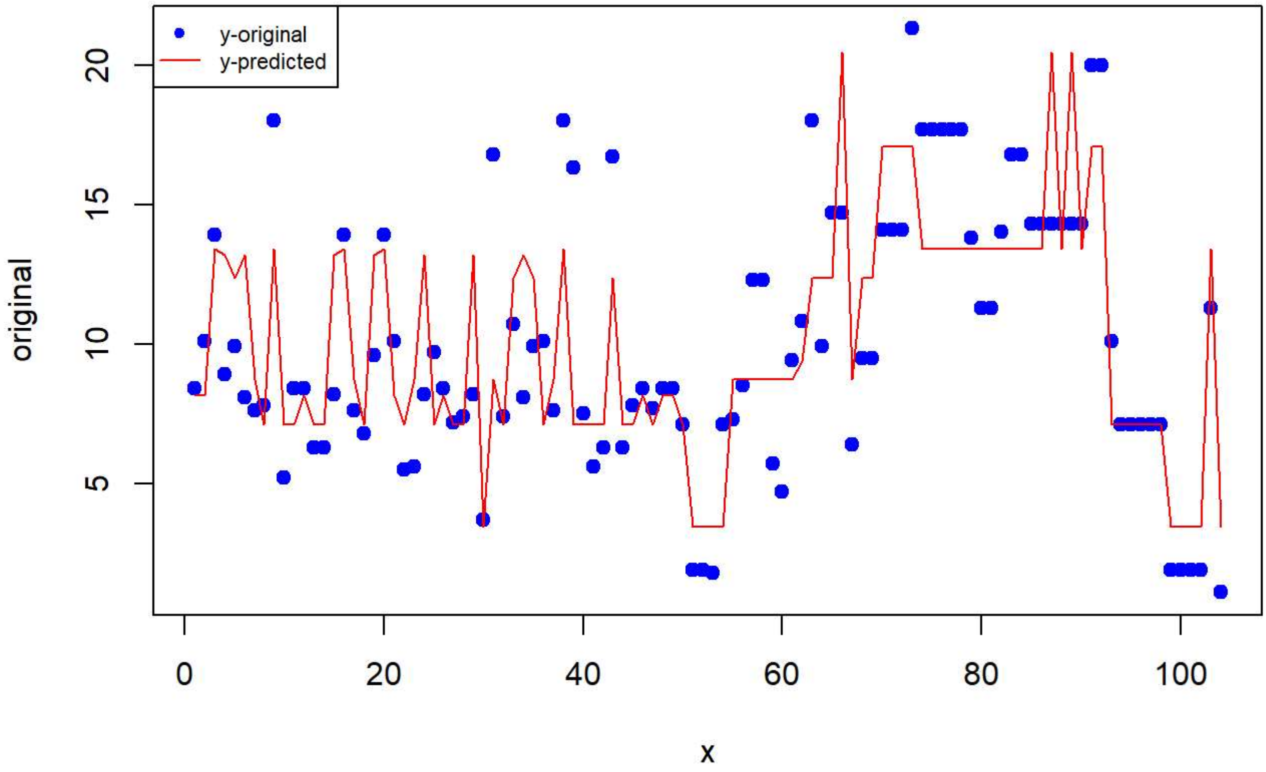
```
par(mfrow=c(1,2))
rsq.rpart(fit)
```

```
##
## Regression tree:
## rpart(formula = ISI ~ ., data = data_train, method = "anova")
##
## Variables actually used in tree construction:
## [1] DC   DMC  FFMC X
##
## Root node error: 8287.2/413 = 20.066
##
## n= 413
##
##          CP nsplit rel error  xerror    xstd
## 1  0.299838      0   1.00000 1.00552 0.27729
## 2  0.178032      1   0.70016 0.77036 0.24517
## 3  0.057175      2   0.52213 0.60377 0.24465
## 4  0.031292      3   0.46495 0.55944 0.21395
## 5  0.029580      4   0.43366 0.53847 0.21371
## 6  0.021655      5   0.40408 0.50995 0.21331
## 7  0.014055      6   0.38243 0.48562 0.21340
## 8  0.013458      7   0.36837 0.47546 0.21322
## 9  0.012560      8   0.35491 0.47212 0.21323
## 10 0.010000      9   0.34235 0.45534 0.20101
```




```
predict_unseen <-predict(fit, data_test, method = 'anova')
```

```
original = data_test$ISI

x=1:length(original)

plot(x, original,pch=19, col="blue")
lines(x, predict_unseen, col="red")
legend("topleft", legend = c("y-original", "y-predicted"),
       col = c("blue", "red"), pch = c(19,NA), lty = c(NA,1),  cex = 0.7)
```




From the above graph we can see the deviation from the truth in some cases is very high, which will effect our $R^2$

```
res <- cbind(predict_unseen,data_test$ISI)
colnames(res) <- c('predicted','ground truth')
res <- as.data.frame(res)
```

```
res
```

```
##     predicted ground truth
## 414  8.154545          8.4
## 415  8.154545         10.1
## 416 13.400000         13.9
## 417 13.192857          8.9
## 418 12.358824          9.9
## 419 13.192857          8.1
## 420  8.718182          7.6
## 421  7.126136          7.8
## 422 13.400000         18.0
## 423  7.126136          5.2
## 424  7.126136          8.4
## 425  8.154545          8.4
## 426  7.126136          6.3
## 427  7.126136          6.3
## 428 13.192857          8.2
## 429 13.400000         13.9
## 430  8.718182          7.6
## 431  7.126136          6.8
## 432 13.192857          9.6
## 433 13.400000         13.9
## 434  8.154545         10.1
## 435  7.126136          5.5
## 436  8.718182          5.6
## 437 13.192857          8.2
## 438  7.126136          9.7
## 439  8.154545          8.4
## 440  7.126136          7.2
## 441  7.126136          7.4
## 442 13.192857          8.2
## 443  3.432836          3.7
## 444  8.718182         16.8
## 445  7.126136          7.4
## 446 12.358824         10.7
## 447 13.192857          8.1
## 448 12.358824          9.9
## 449  7.126136         10.1
## 450  8.718182          7.6
## 451 13.400000         18.0
## 452  7.126136         16.3
## 453  7.126136          7.5
## 454  7.126136          5.6
## 455  7.126136          6.3
## 456 12.358824         16.7
## 457  7.126136          6.3
## 458  7.126136          7.8
## 459  8.154545          8.4
## 460  7.126136          7.7
## 461  8.154545          8.4
## 462  8.154545          8.4
## 463  7.126136          7.1
## 464  3.432836          1.9
## 465  3.432836          1.9
## 466  3.432836          1.8
## 467  3.432836          7.1
## 468  8.718182          7.3
## 469  8.718182          8.5
```

```
## 470  8.718182         12.3
## 471  8.718182         12.3
## 472  8.718182          5.7
## 473  8.718182          4.7
## 474  8.718182          9.4
## 475  9.362887         10.8
## 476 12.358824         18.0
## 477 12.358824          9.9
## 478 12.358824         14.7
## 479 20.450000         14.7
## 480  8.718182          6.4
## 481 12.358824          9.5
## 482 12.358824          9.5
## 483 17.080000         14.1
## 484 17.080000         14.1
## 485 17.080000         14.1
## 486 17.080000         21.3
## 487 13.400000         17.7
## 488 13.400000         17.7
## 489 13.400000         17.7
## 490 13.400000         17.7
## 491 13.400000         17.7
## 492 13.400000         13.8
## 493 13.400000         11.3
## 494 13.400000         11.3
## 495 13.400000         14.0
## 496 13.400000         16.8
## 497 13.400000         16.8
## 498 13.400000         14.3
## 499 13.400000         14.3
## 500 20.450000         14.3
## 501 13.400000         14.3
## 502 20.450000         14.3
## 503 13.400000         14.3
## 504 17.080000         20.0
## 505 17.080000         20.0
## 506  7.126136         10.1
## 507  7.126136          7.1
## 508  7.126136          7.1
## 509  7.126136          7.1
## 510  7.126136          7.1
## 511  7.126136          7.1
## 512  3.432836          1.9
## 513  3.432836          1.9
## 514  3.432836          1.9
## 515  3.432836          1.9
## 516 13.400000         11.3
## 517  3.432836          1.1
```

```
sse<-sum((res$predicted-res$`ground truth`)^2)
sst<-sum((mean(df$ISI)-res$`ground truth`)^2)
r2=1-(sse/sst)
print(paste(sse, "is the SSE"))
```

```
## [1] "903.696894845512 is the SSE"
```

```
print(paste(r2, "is the R square"))
```

```
## [1] "0.624929675377415 is the R square"
```

The overall accuracy can be seen. As seen in the graph the SSE is high, because of the deviations and that in turns effects $R^2$. From the accuracy we can say that **Overall the ISI of a variable can be explained up to** $62.49$ **by the model. That is, given the other variables, we can correctly predict the ISI,** $62.49$ **of the time**