Final Model

Eric Breton

April 15, 2019

```
install.packages("XML")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
install.packages("RCurl")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
install.packages("RVest")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
install.packages("reshape2")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
install.packages("tidyr")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
install.packages("data.table")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```r
install.packages("stringi")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)

## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```r
install.packages("sqldf")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)

## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```r
install.packages("dplyr")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)

## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```r
install.packages('corrplot')
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)

## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```r
install.packages("ggplot2")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)

## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```r
install.packages("factoextra")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)

## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```r
install.packages("cowplot")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)

## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
install.packages("cluster")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
install.packages("NbClust")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
install.packages("FNN")
```

```
## Installing package into 'C:/Users/ebret/OneDrive/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
library(NbClust)
```

```
## Warning: package 'NbClust' was built under R version 3.5.2
```

```
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 3.5.3
```

```
library(cowplot)
```

```
## Warning: package 'cowplot' was built under R version 3.5.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.3

##
## Attaching package: 'cowplot'

## The following object is masked from 'package:ggplot2':
##
##     ggsave
```

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 3.5.3
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.5.3
```

```
## corrplot 0.84 loaded
```

```
library(ggplot2)
library(XML)
```

```
## Warning: package 'XML' was built under R version 3.5.3
```

```
library(RCurl)
```

```
## Warning: package 'RCurl' was built under R version 3.5.3
```

```
## Loading required package: bitops
```

```
library(rvest)
```

```
## Loading required package: xml2
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:XML':
##
##     xml
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
##
## 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(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 3.5.3
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.5.3
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:reshape2':
##
##     smiths
```

```
## The following object is masked from 'package:RCurl':
##
##     complete
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.5.3
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:reshape2':
##
##     dcast, melt
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(stringi)
```

```
## Warning: package 'stringi' was built under R version 3.5.3
```

```
library(sqldf)
```

```
## Warning: package 'sqldf' was built under R version 3.5.3
```

```
## Loading required package: gsubfn
```

```
## Warning: package 'gsubfn' was built under R version 3.5.3
```

```
## Loading required package: proto
```

```
## Warning: package 'proto' was built under R version 3.5.3

## Loading required package: RSQLite

## Warning: package 'RSQLite' was built under R version 3.5.3
```

```
library (lattice)
```

```
## Warning: package 'lattice' was built under R version 3.5.2
```

```
library(ggplot2)
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages ----------------------------------------------------------------------

## v tibble  2.1.2     v purrr   0.2.5
## v readr   1.3.1     v stringr 1.3.1
## v tibble  2.1.2     v forcats 0.3.0

## Warning: package 'readr' was built under R version 3.5.2

## -- Conflicts ------------------------------------------------------------------------------- t
## x data.table::between()   masks dplyr::between()
## x tidyr::complete()       masks RCurl::complete()
## x dplyr::filter()         masks stats::filter()
## x data.table::first()     masks dplyr::first()
## x cowplot::ggsave()       masks ggplot2::ggsave()
## x readr::guess_encoding() masks rvest::guess_encoding()
## x dplyr::lag()            masks stats::lag()
## x data.table::last()      masks dplyr::last()
## x purrr::pluck()          masks rvest::pluck()
## x purrr::transpose()      masks data.table::transpose()
## x rvest::xml()            masks XML::xml()
```

```
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 3.5.3

## Loading required package: magrittr

##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract

##
## Attaching package: 'ggpubr'

## The following object is masked from 'package:cowplot':
##
##     get_legend
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.2

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(FNN)
```

```
## Warning: package 'FNN' was built under R version 3.5.3
```

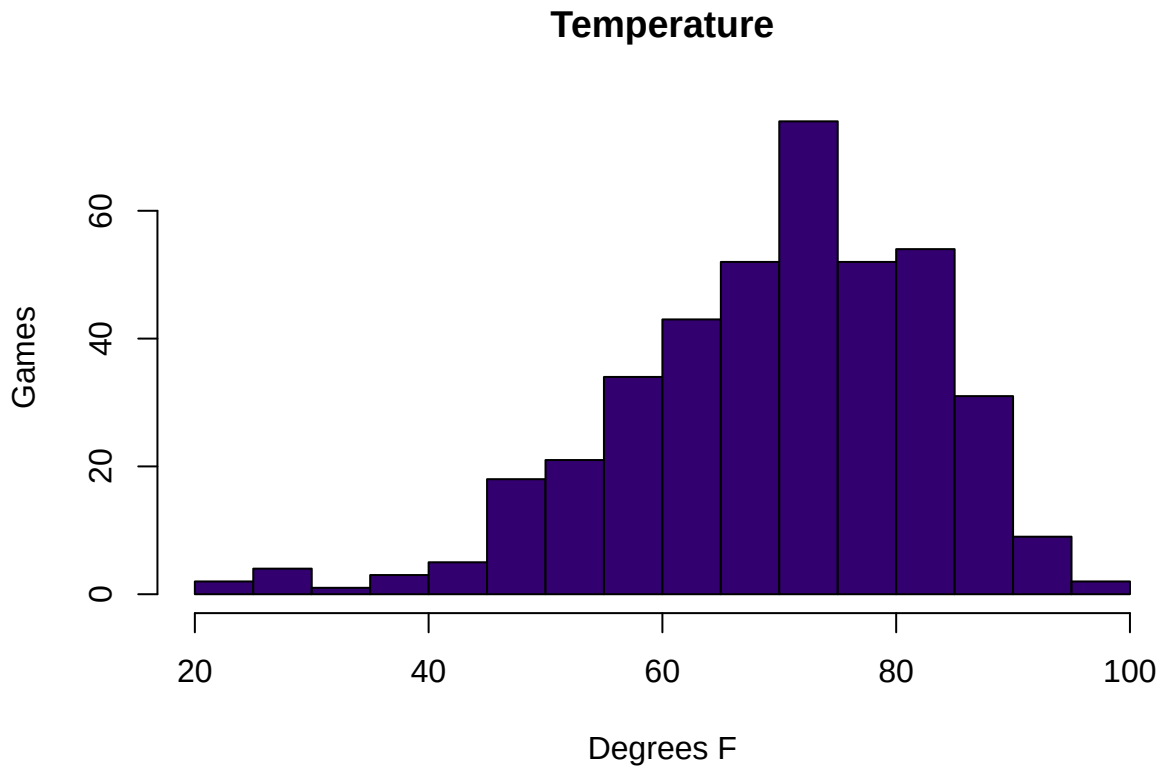
```
setwd("C:/Users/ebret/Desktop/Practicum")
wd<-fread("Clustered_WD_Both.csv",stringsAsFactors = FALSE)
bd<-fread("baseball_Data.csv",stringsAsFactors = FALSE)


cd<-sqldf('select wd.*,Total from wd join bd where  bd.Date=wd.Date and bd.Day_night=wd.Day_night and H
#View(cd)
# Convert Direction into cardinal
cd<-sqldf('select Date,Day_Night,Total, Cluster, temp,pressure,humidity,wind_speed,wind_deg,clouds_all,
case when wind_deg between "348.75" and "360" then "N"
when wind_deg between "11.25" and "33.75" then "NNE"
when wind_deg between "33.75" and "56.25" then "NE"
when wind_deg between "56.25" and "78.75" then "ENE"
when wind_deg between "78.75" and "101.25" then "E"
when wind_deg between "101.25" and "123.75" then "ESE"
when wind_deg between "123.75" and "146.25" then "SE"
when wind_deg between "146.25" and "168.75" then "SSE"
when wind_deg between "168.75" and "191.25" then "S"
when wind_deg between "191.25" and "213.75" then "SSW"
when wind_deg between "213.75" and "236.25" then "SW"
when wind_deg between "236.25" and "258.75" then "WSW"
when wind_deg between "258.75" and "281.25" then "W"
when wind_deg between "281.25" and "303.75" then "WNW"
when wind_deg between "303.75" and "326.25" then "NW"
when wind_deg between"326.25" and "348.75" then "NNW"
```

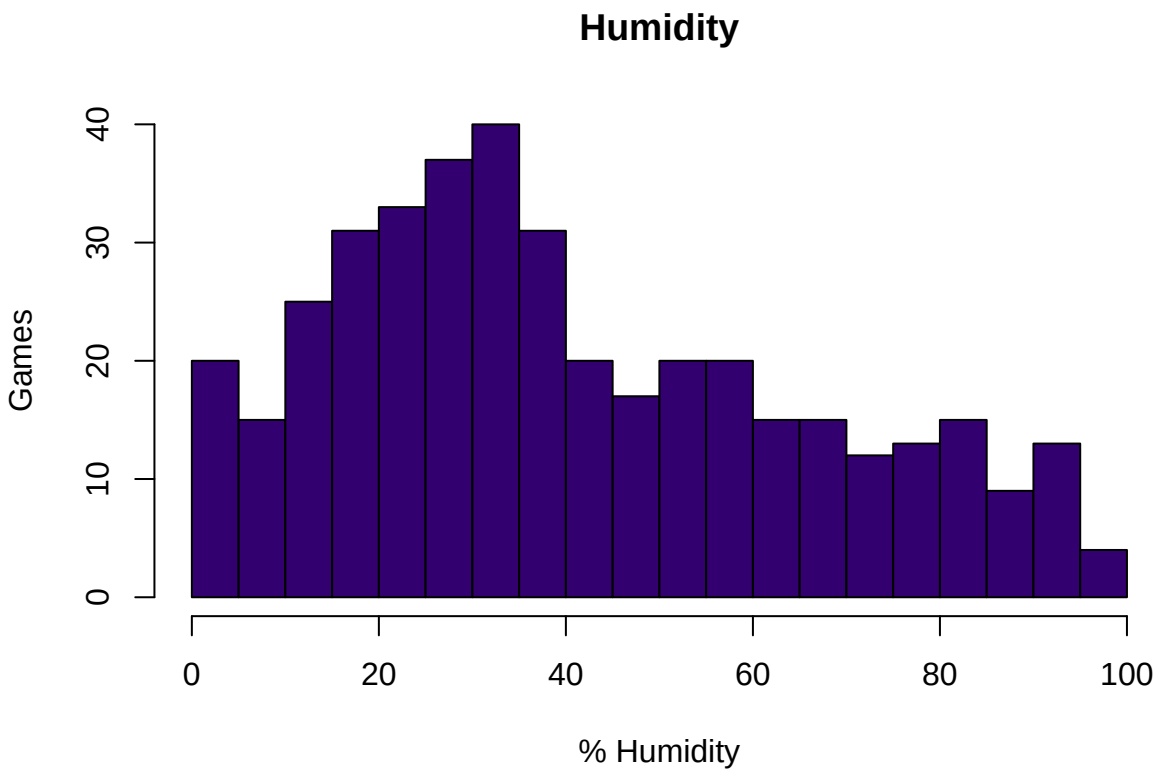
```
else "N" end as card_dir from cd')
View(cd)
##Vew(Combined)
#write.csv(cd, "C:/Users/ebret/Desktop/Practicum/BD_WD_CLustered.csv")
```

Testing Normality

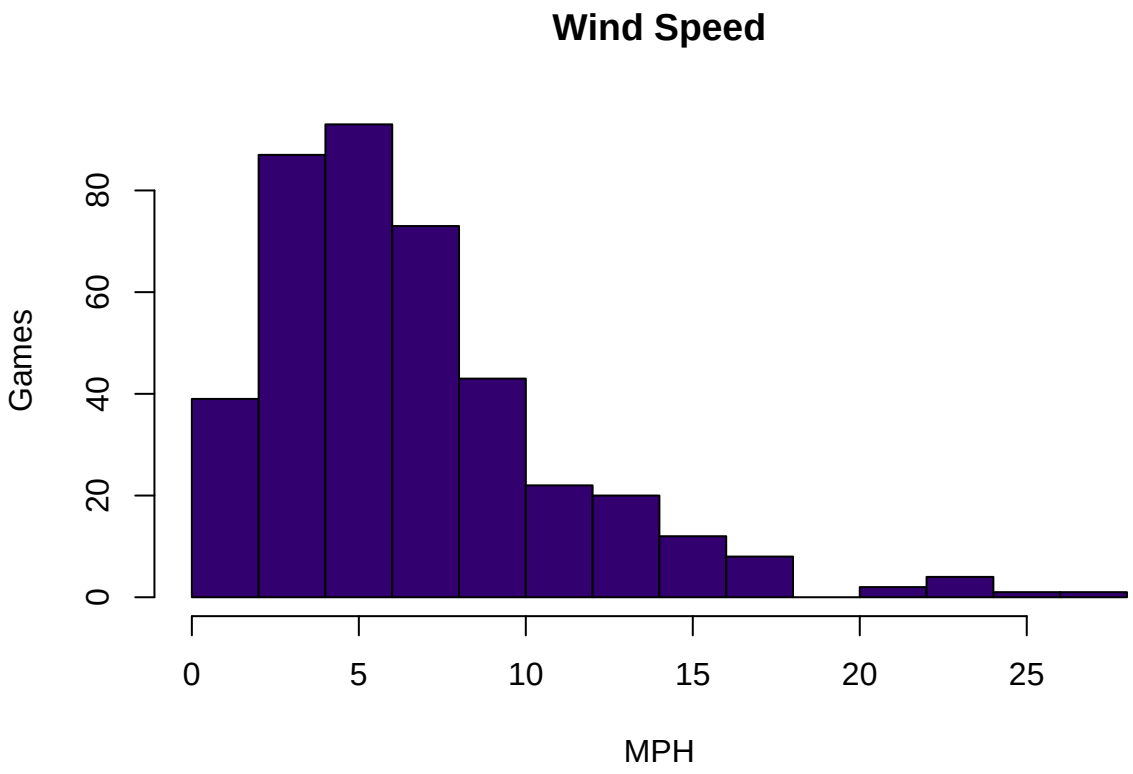
```
hist(cd$temp,main="Temperature",xlab="Degrees F",ylab="Games",breaks=15,col="#33006F")
```



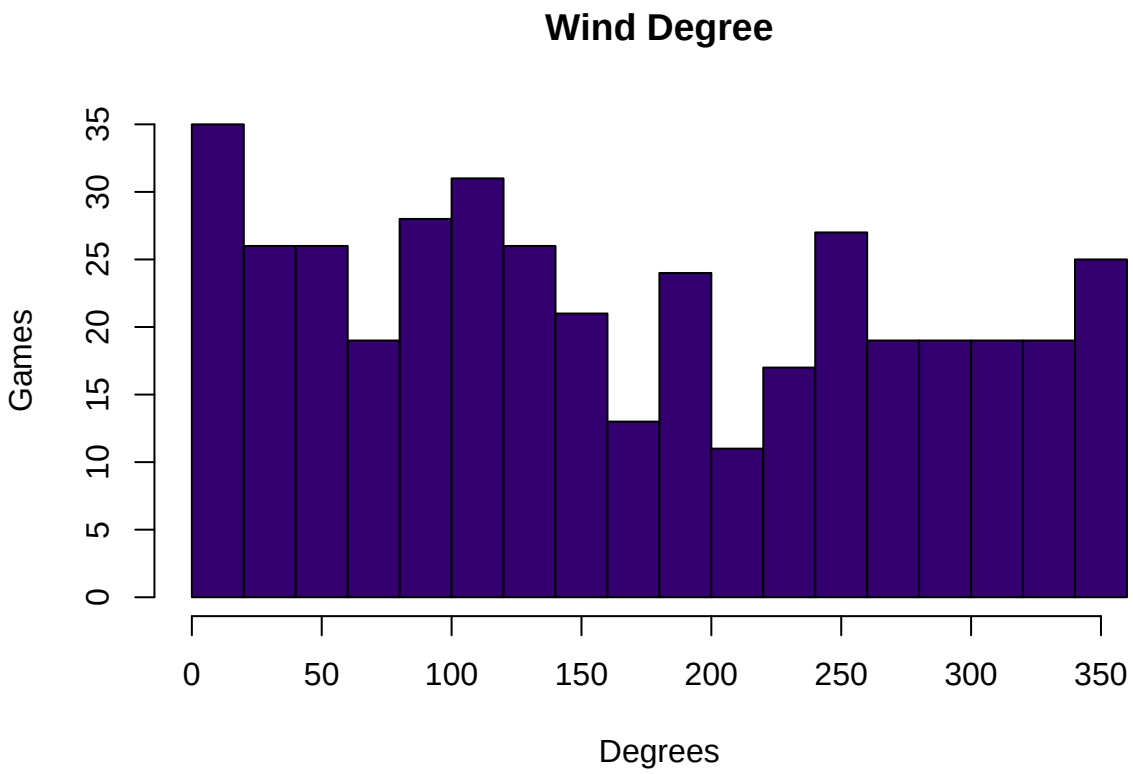

```
hist(cd$humidity,main="Humidity",xlab="% Humidity",ylab="Games",breaks=15,col="#33006F")
```



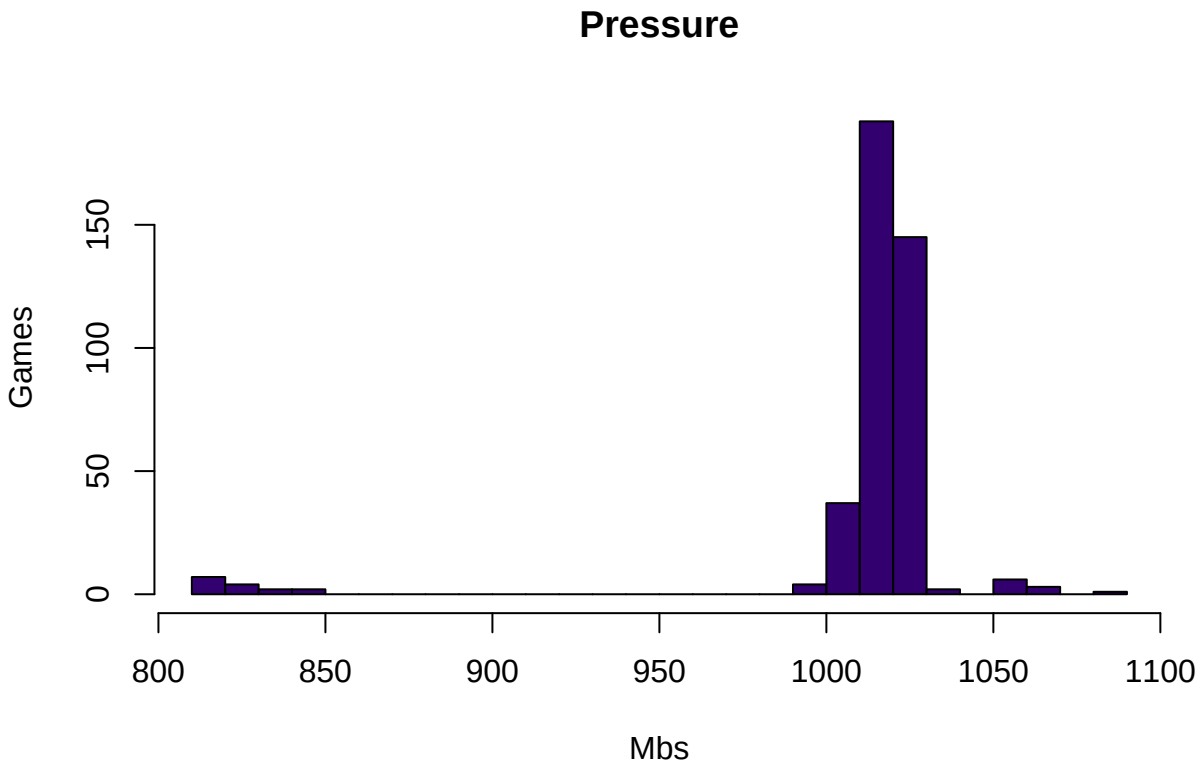

```
hist(cd$wind_speed,main="Wind Speed",xlab="MPH",ylab="Games",breaks=15,col="#33006F")
```



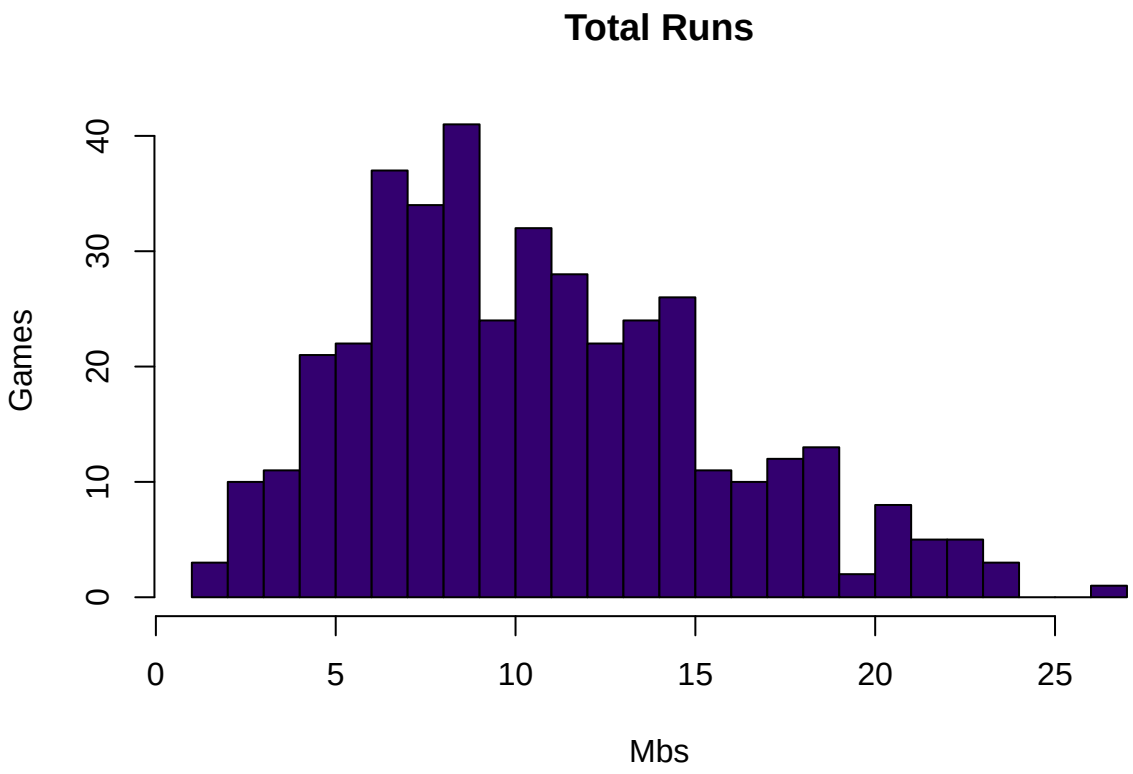

```
hist(cd$wind_deg,main="Wind Degree",xlab="Degrees",ylab="Games",breaks=15,col="#33006F")
```



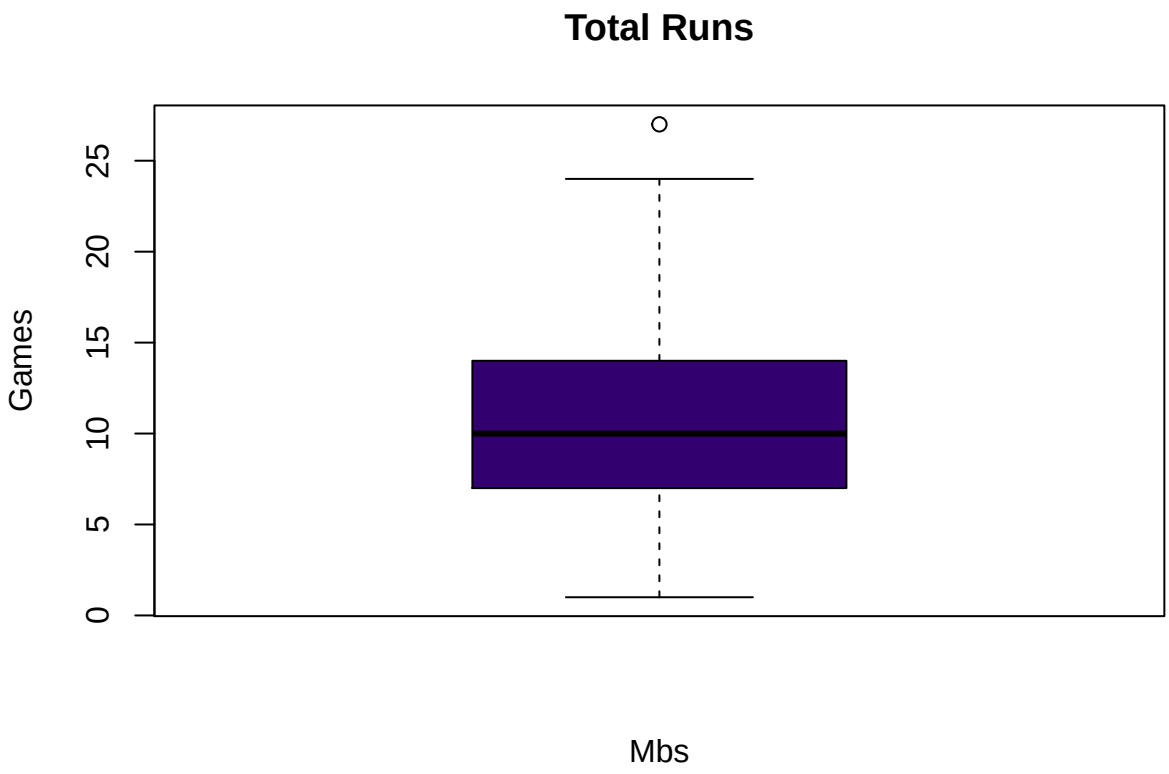

```
hist(cd$pressure,main="Pressure",xlab="Mbs",ylab="Games",breaks=20,col="#33006F")
```



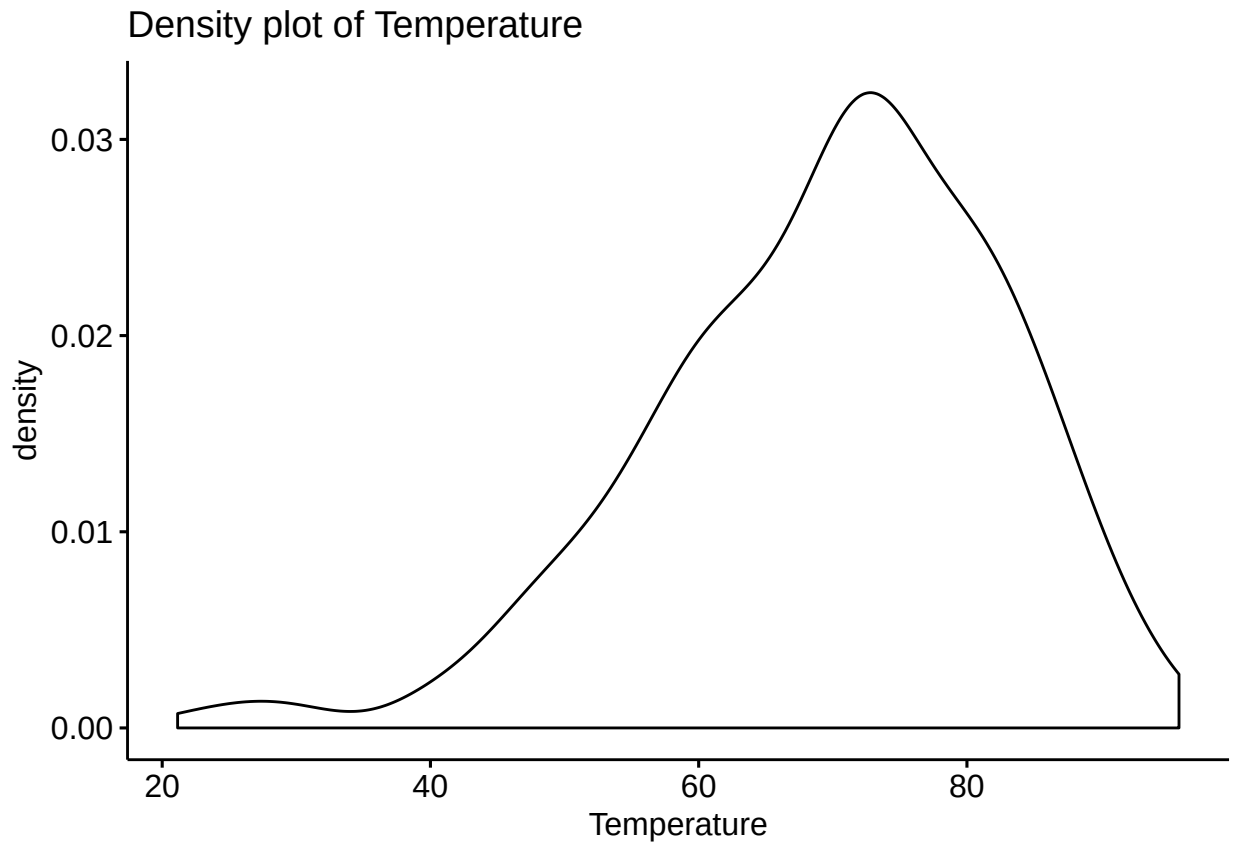

```
hist(cd$Total,main="Total Runs",xlab="Mbs",ylab="Games",breaks=20,col="#33006F")
```



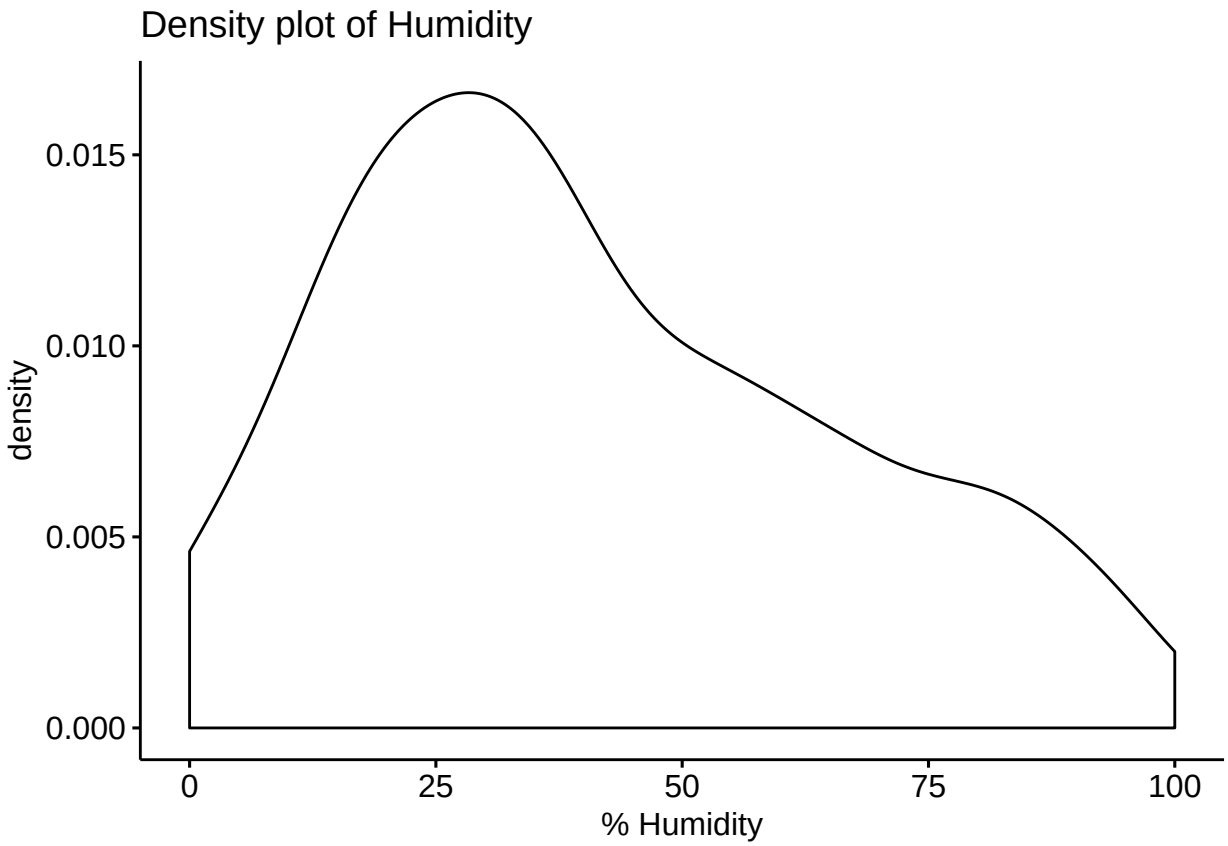

```
boxplot(cd$Total,main="Total Runs",xlab="Mbs",ylab="Games",breaks=20,col="#33006F")
```



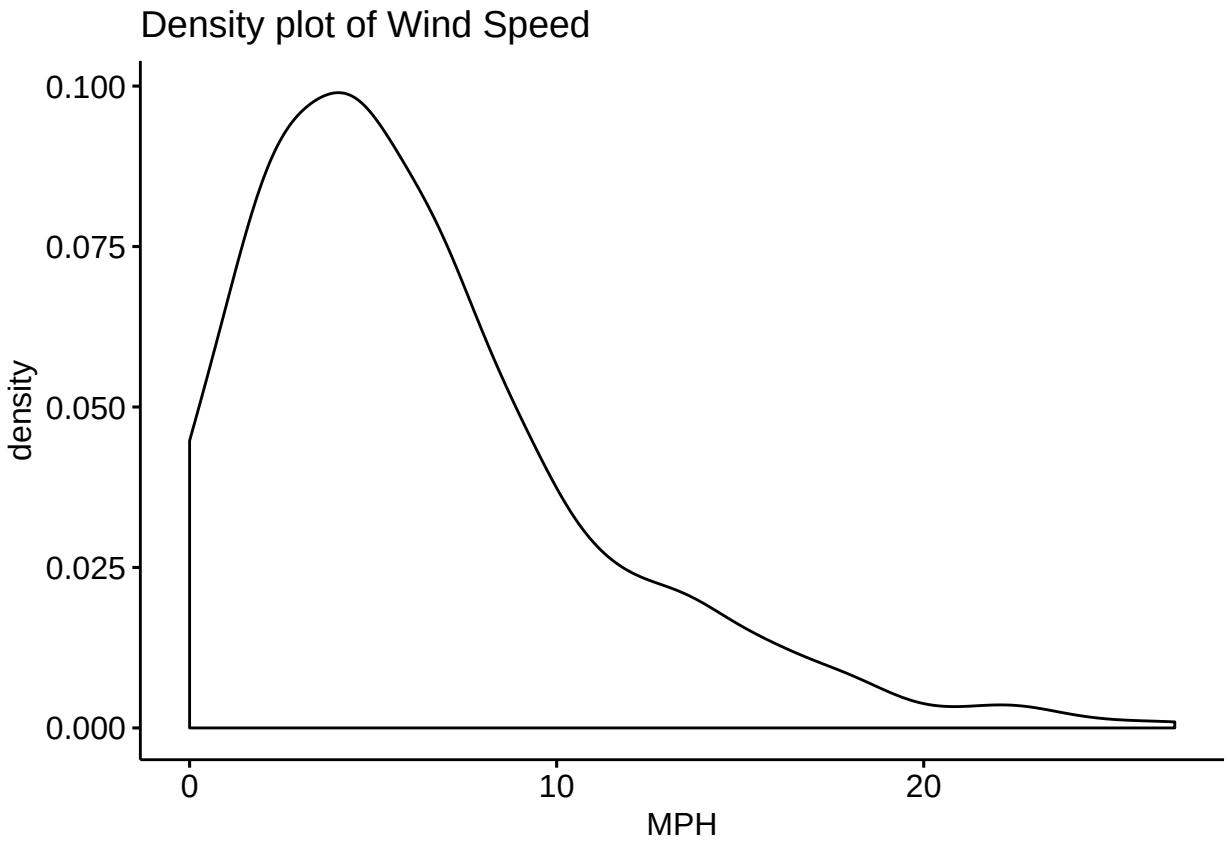

```
ggdensity(cd$temp,  main = "Density plot of Temperature",xlab = "Temperature")
```



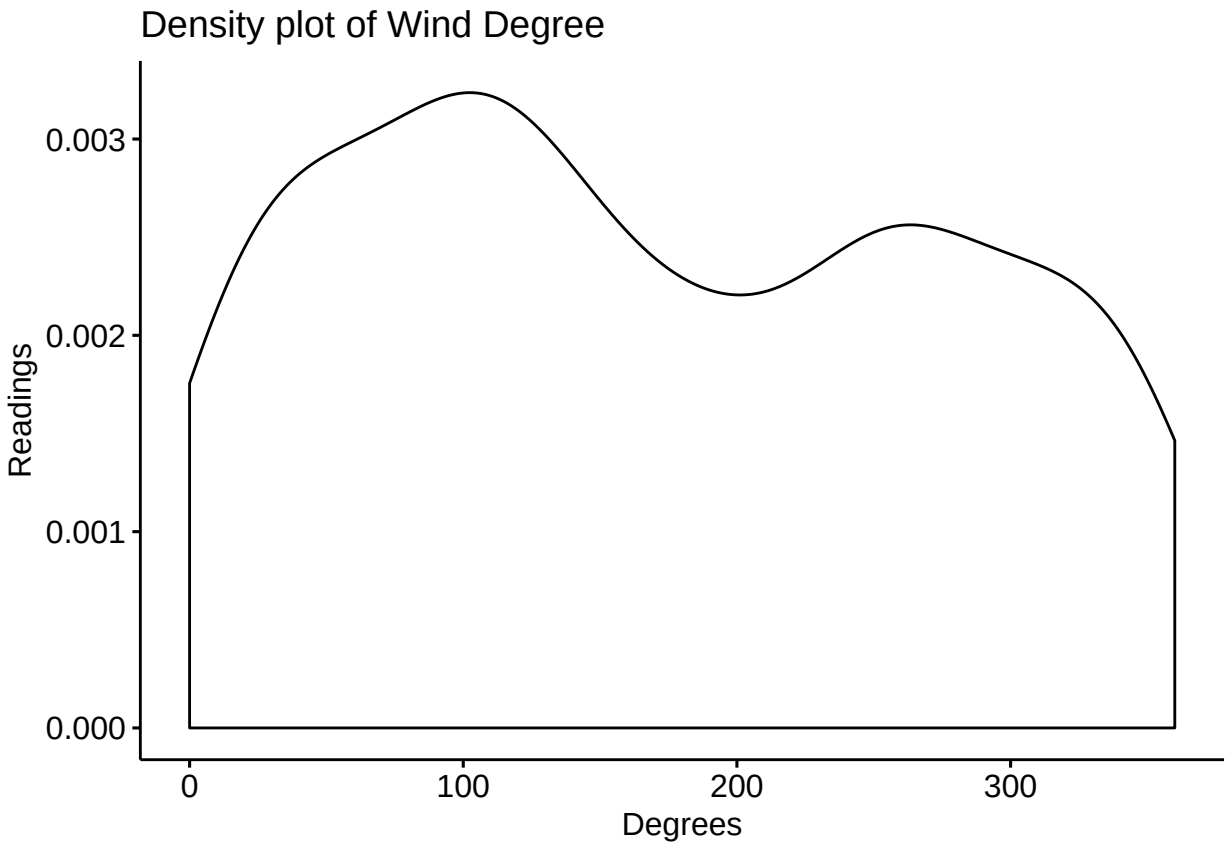

```
ggdensity(cd$humidity,main="Density plot of Humidity",xlab="% Humidity")
```



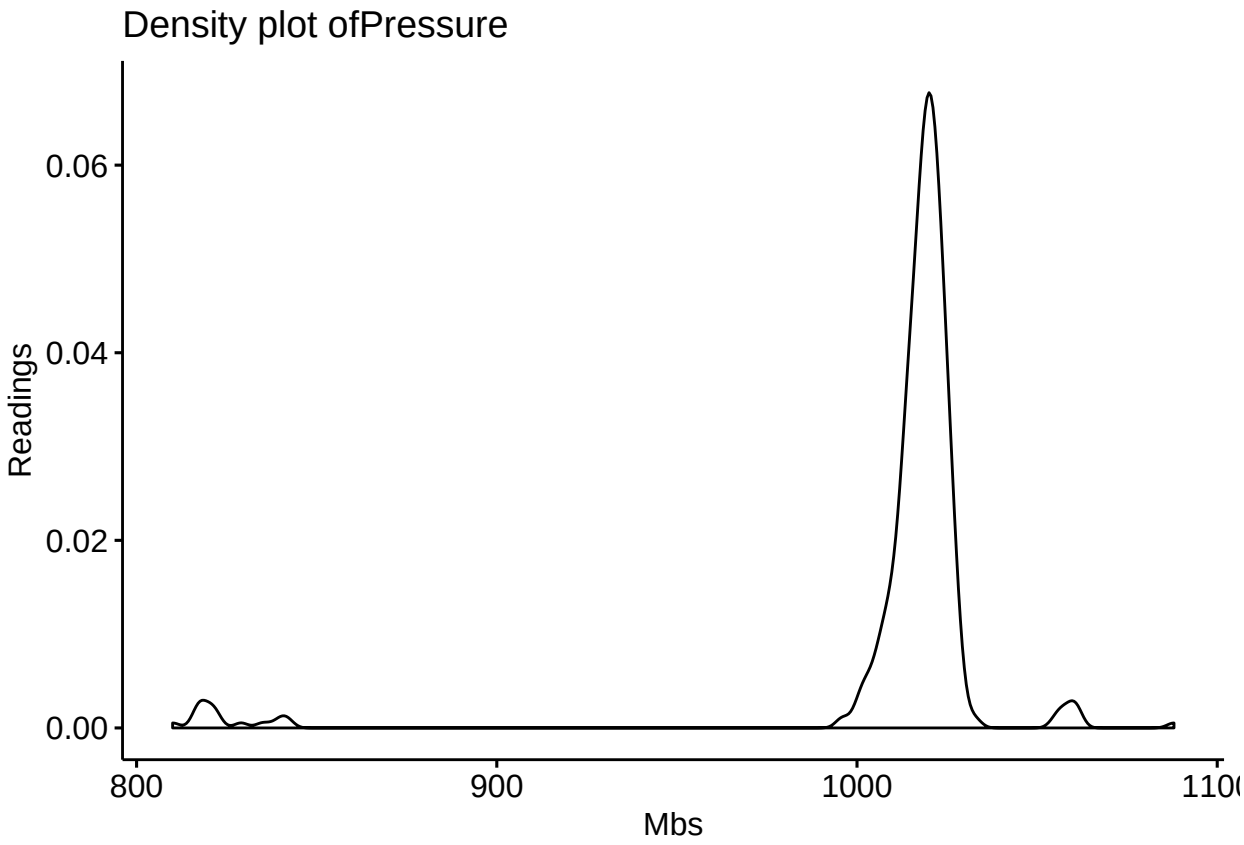

```
ggdensity(cd$wind_speed,main="Density plot of Wind Speed",xlab="MPH")
```



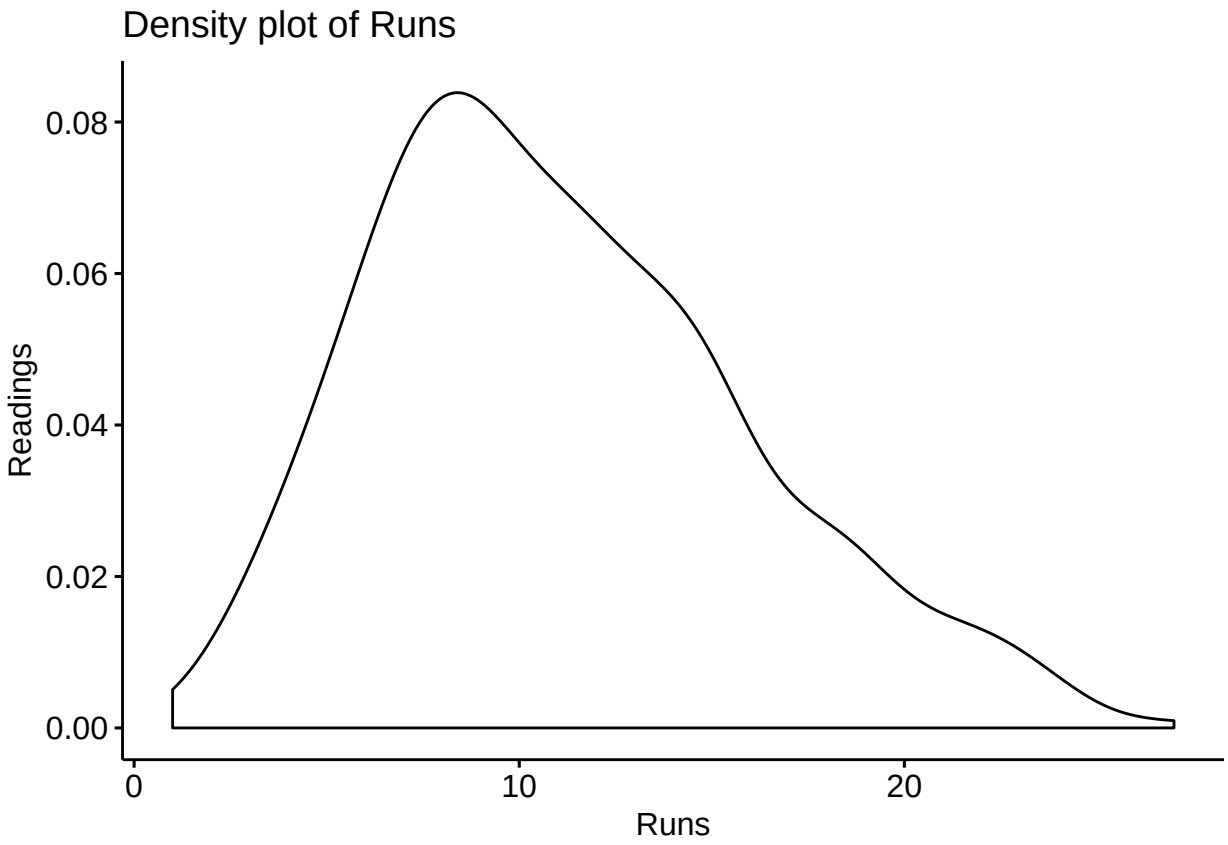

```
ggdensity(cd$wind_deg,main="Density plot of Wind Degree",xlab="Degrees",ylab="Readings")
```



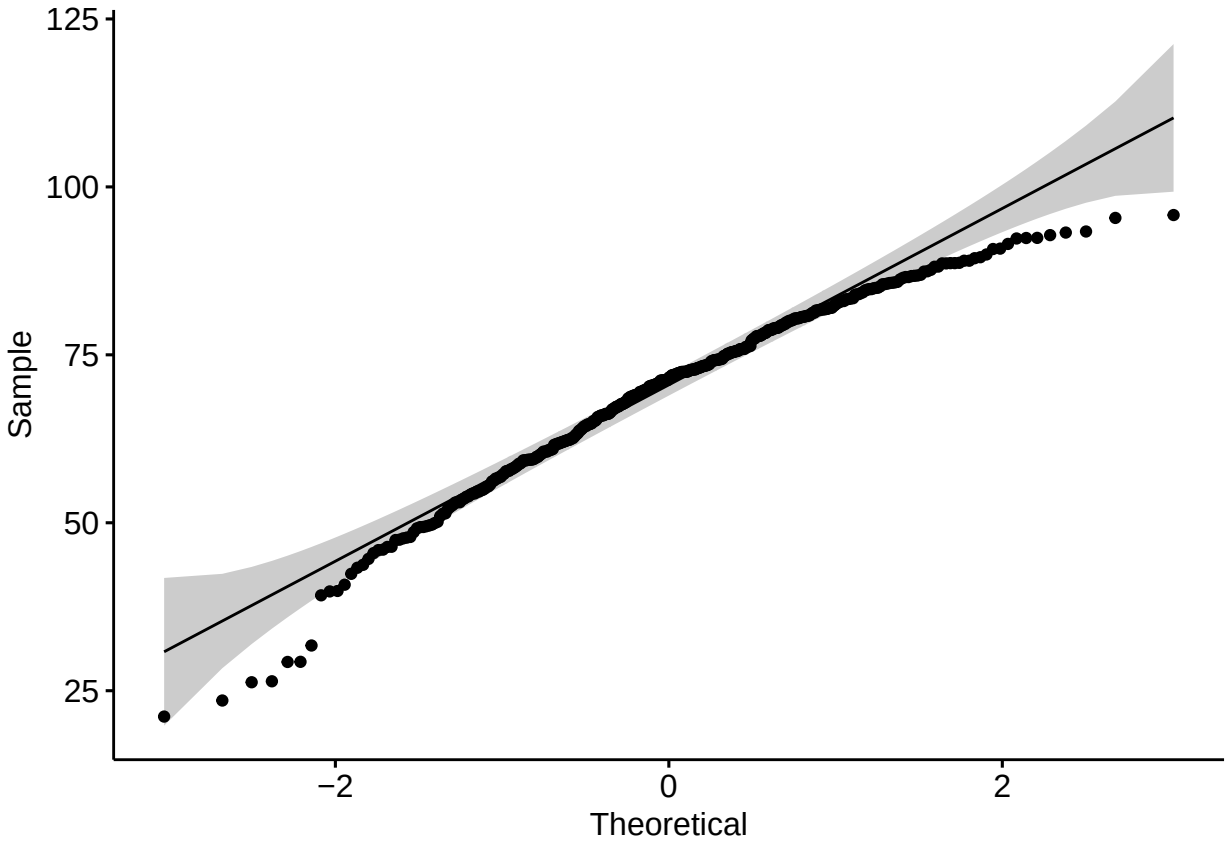

```
ggdensity(cd$pressure,main="Density plot ofPressure",xlab="Mbs",ylab="Readings")
```



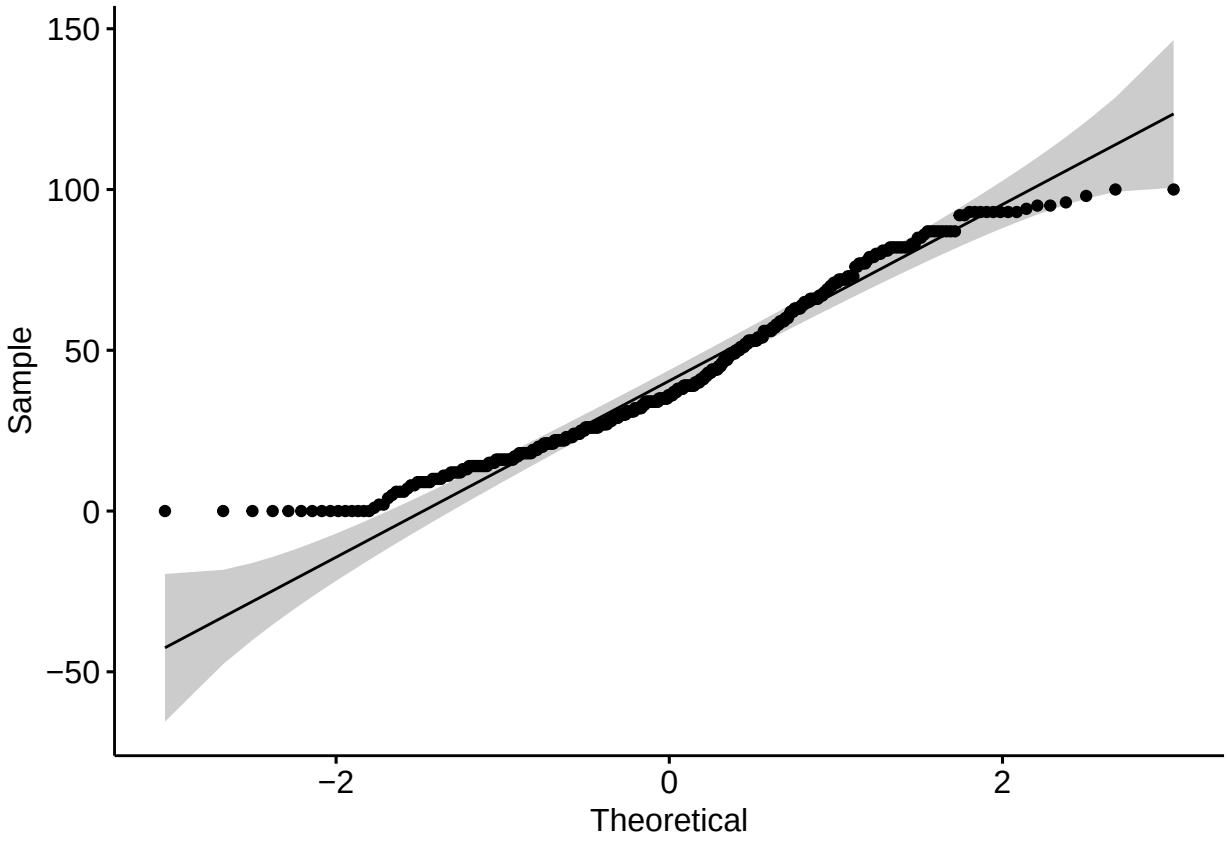

```
ggdensity(cd$Total,main="Density plot of Runs",xlab="Runs",ylab="Readings")
```



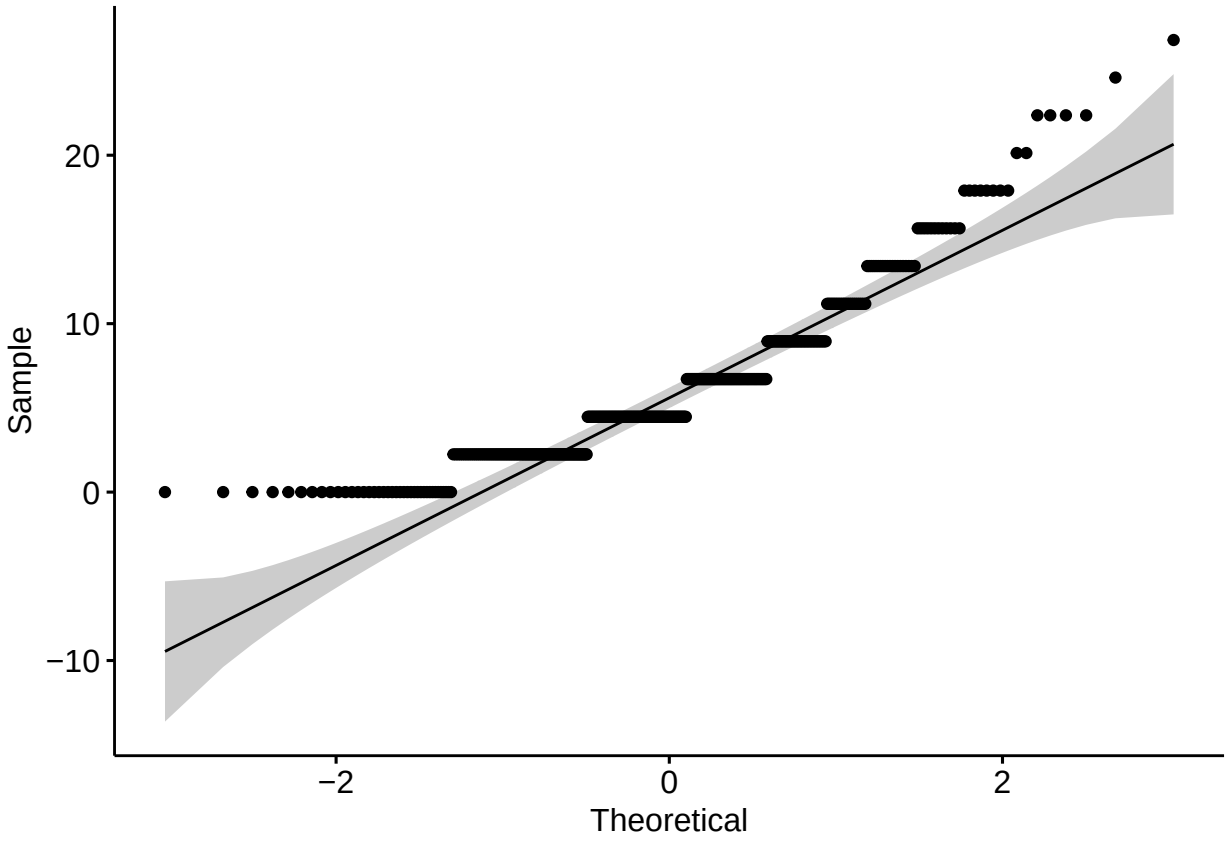

```
ggqqplot(cd$temp)
```



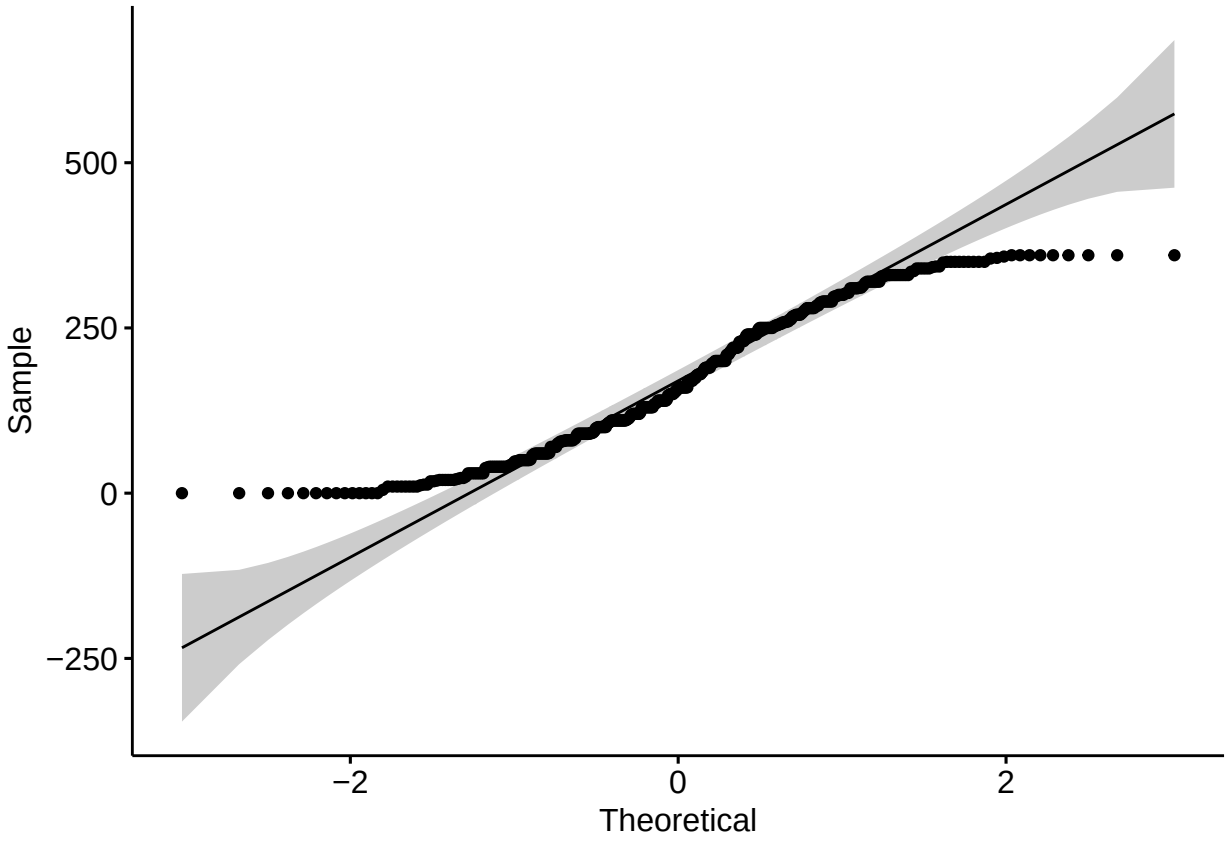

```
ggqqplot(cd$humidity)
```



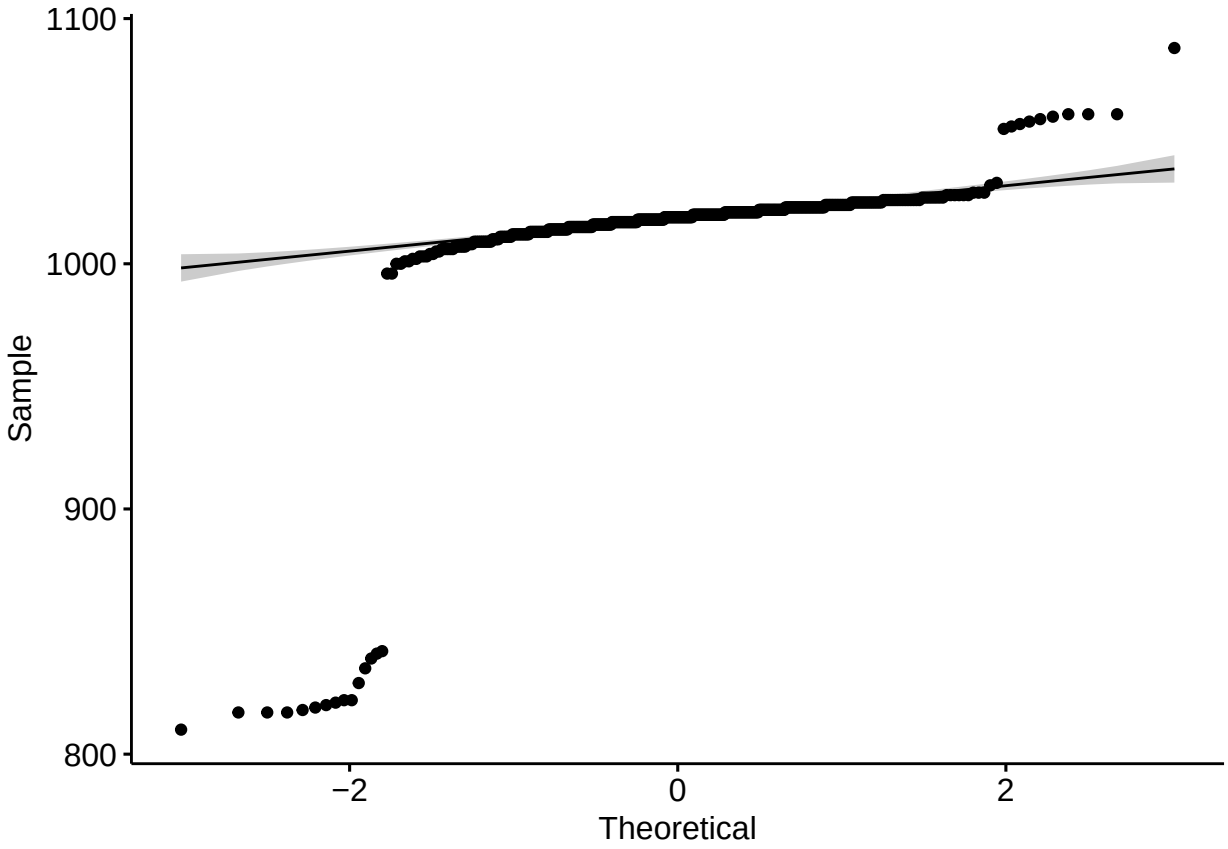

```
ggqqplot(cd$wind_speed)
```



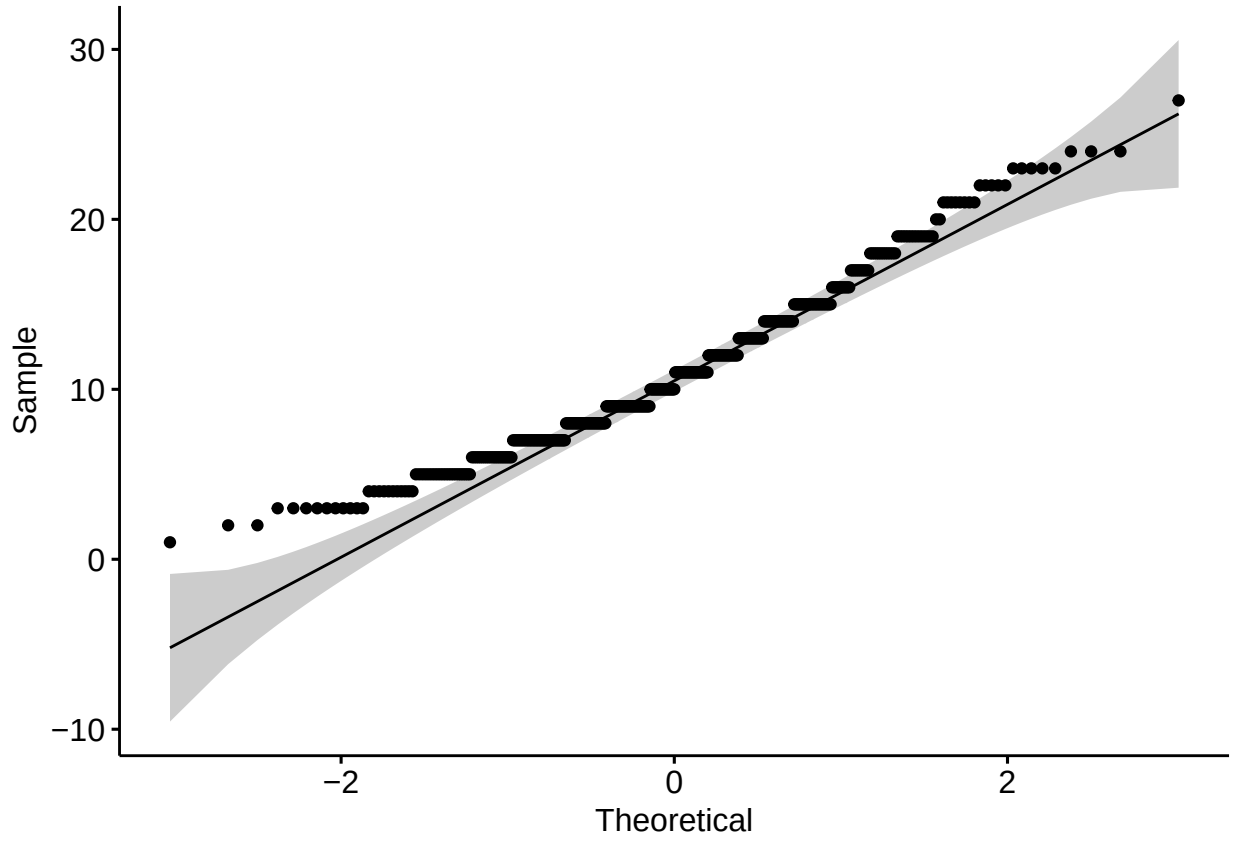

```
ggqqplot(cd$wind_deg)
```




```
ggqqplot(cd$pressure)
```




```
ggqqplot(cd$Total)
```




```
shapiro.test(cd$temp)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  cd$temp
## W = 0.96889, p-value = 1.379e-07
```

```
shapiro.test(cd$humidity)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  cd$humidity
## W = 0.95899, p-value = 3.364e-09
```

```
shapiro.test(cd$wind_speed)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  cd$wind_speed
## W = 0.89032, p-value < 2.2e-16
```

```r
shapiro.test(cd$wind_deg)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  cd$wind_deg
## W = 0.94602, p-value = 5.526e-11
```

```r
shapiro.test(cd$pressure)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  cd$pressure
## W = 0.36986, p-value < 2.2e-16
```

```r
shapiro.test(cd$Total)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  cd$Total
## W = 0.96858, p-value = 1.214e-07
```

From my various tests, it is clear that my data is not normal and thus I will use a Non Parametric statistical method for my analyis. I have chosen to use KNN, because it is simple and powerful without the need to tune complex paramters. Training exampes cab be added easily.

```r
# Create a Data Set with just my Total Runs
#due to smaller data set avg total runs per game is about 1.5 runs less than avg so take vegas line and
line<-10
base_data<-cd

#head(base_data)


#Store my Date to add back as row names
rn<-cd %>% select(Date)

# remove total runs, and dates from my data set
base_data <- cd %>% select(-Date,-card_dir,-Cluster)
# Change my character data to integers

base_data$Day_night<-ifelse(base_data$Day_night=="D",1,0)
data_to_scale<-base_data %>% select(-Day_night,-Total)
un_scaled_data<-base_data %>% select(Day_night,Total)
#head(un_scaled_data)
#Veriify my data is all integers or numeric so that I can scale it
#str(base_data)
#scaling my data so that distance is not skewed by the different times of measurements that I am using.
#str(base_data)
```

```
scaled_bd<-scale(data_to_scale)
scaled_combined<-cbind(un_scaled_data,scaled_bd)
#head(scaled_combined)
```

```
#Set seed so that my process is repatable
set.seed(42)



dt<-data.table(scaled_combined)
dt$Total<-ifelse(dt$Total>line,1,0)
dt$Total=as.factor(dt$Total)

##Vew(dt)
# targs<-dt %>% select(Total)
# feats<-dt %>% select(-Total)
# Seperate my classifier from my variables
targs <- dt[, Total]
feats <- dt[, -'Total']


#80% of my sample size
train.idxs <- createDataPartition(targs, p = 0.8)$Resample1
##Vew(train.idxs)

# create features and targets
tr.feats <- feats[train.idxs]
#Vew(tr.feats)
tr.targs <- targs[train.idxs]
##Vew(tr.targs)
te.feats <- feats[-train.idxs]
te.targs <- targs[-train.idxs]
```

This function will tell me the average number of times my prediction of Y is accurate.

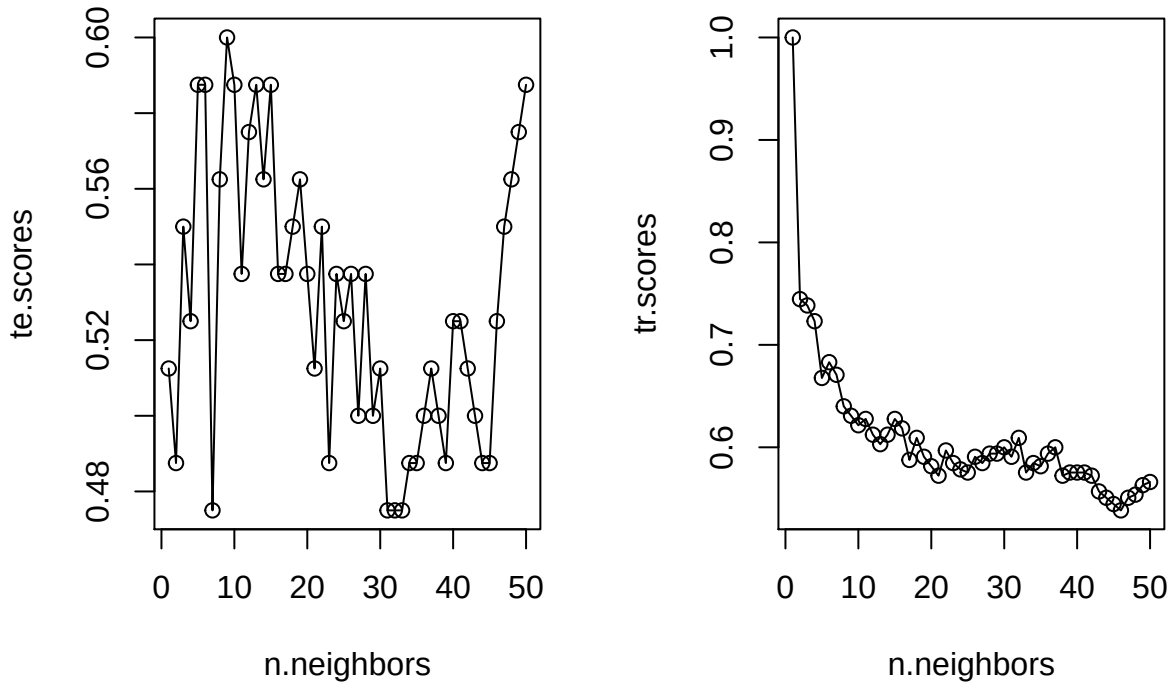
```
# calculate accuracy
accuracy <- function(y.true, y.pred) {
  return(mean(y.true == y.pred))
}
```

```
#Creating two vectors for my testing and training scores to
te.scores <- c()
tr.scores <- c()

#Generating a list of whole numbers that I can loop through for analyzing the best number of nearest nu
n.neighbors <- seq(1, 50)

#Running a loop for all k values between 1 through 50 which will give me the data necessary to determin
for (k in n.neighbors) {
#Training the data on my training features (atmospheric conditions) classifying the O/U on my test set
te.preds <- knn(train = tr.feats, cl = tr.targs, test = te.feats, k = k)
#Training the data on my training features (atmospheric conditions) classifying the O/U on my training
  tr.preds <- knn(train = tr.feats, cl = tr.targs, test = tr.feats, k = k)
```

```
#Returns the cores for both knn analysis from my accuracy function which is taking in the predicted val
  te.scores <- c(te.scores, accuracy(te.preds, te.targs))
  tr.scores <- c(tr.scores, accuracy(tr.preds, tr.targs))
}
par(mfrow=c(1, 2))
plot(n.neighbors, te.scores)
lines(n.neighbors, te.scores)
plot(n.neighbors, tr.scores)
lines(n.neighbors, tr.scores)
```




```
max.acc.idx <- which.max(te.scores)
print(paste('best accuracy of',
            round(te.scores[max.acc.idx], 2),
            'at k=',
            n.neighbors[max.acc.idx],
            sep = ' '))
```

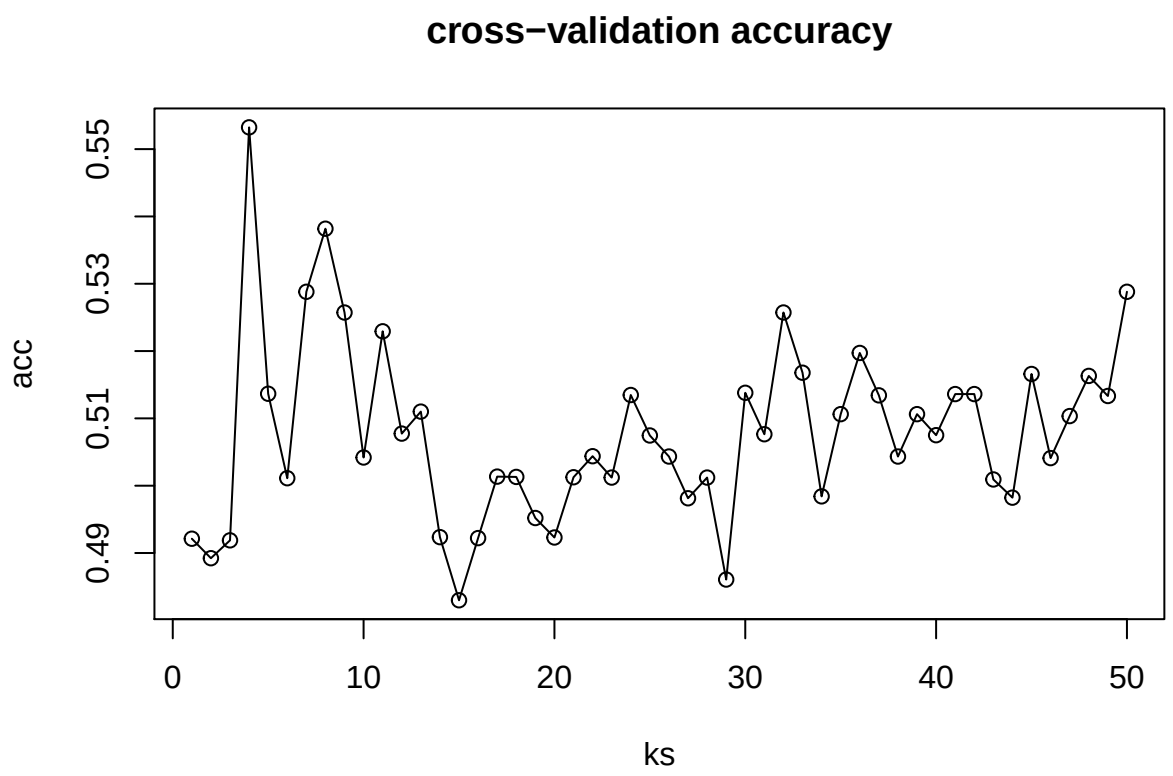
```
## [1] "best accuracy of 0.6 at k= 9"
```

I will do some cross validation on the training set. The results of this will change multiple times due to the randomness of how the data will get split. My prior results showed a value of 9 for K at 60% accuracy.

The first portion of this I will use 5 folds in my cross validation which will divide my dat into 5 observation groups. One of these folds will be held out and use as the validating fold. From this the mean squared error is calculated. The process is than repeated 5 times for each of my observation groups with each group being held out once, this will give me an estimate of the test error.

```
trControl <- trainControl(method  = "cv",
                          number  = 5)
fit <- train(Total ~ .,
             method     = 'knn',
             tuneGrid   = expand.grid(k = 1:50),
             trControl  = trControl,
             preProcess = c("center", "scale"),
             metric     = "Accuracy",
             data       = dt[train.idxs])

ks <- fit$results$k
acc <- fit$results$Accuracy
par(mfrow = c(1, 1))
plot(ks, acc, main = 'cross-validation accuracy')
lines(ks, acc)
```




```
max.acc.idx <- which.max(fit$results$Accuracy)
print(paste('best accuracy of',
            round(acc[max.acc.idx], 2),
            'at k=', ks[max.acc.idx],
            sep = ' '))
```

```
## [1] "best accuracy of 0.55 at k= 4"
```

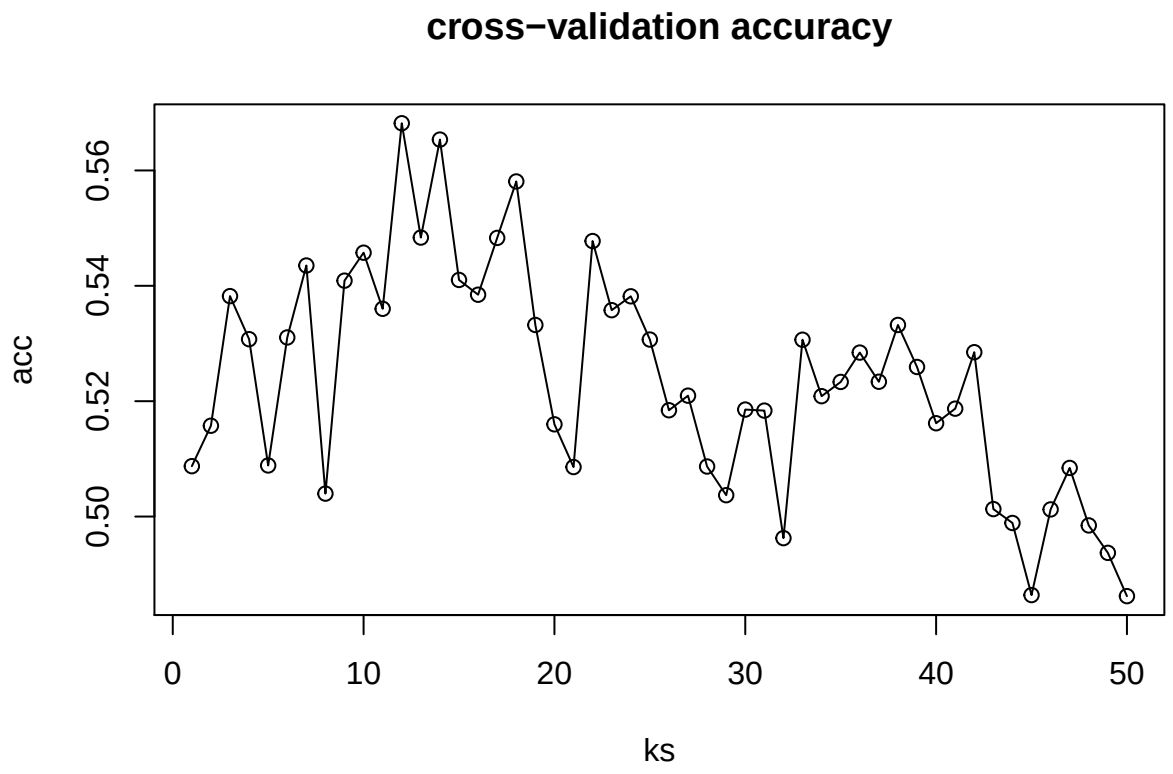
Following the cross validation (ran 3 times) on the training dataset I got an average k value of 7 with an

accuracy of 55%. Running cross validation on the whole dataset.

```
trControl <- trainControl(method  = "cv",
                          number  = 5)

fit <- train(Total ~ .,
             method     = 'knn',
             tuneGrid   = expand.grid(k = 1:50),
             trControl  = trControl,
             preProcess = c("center", "scale"),
             metric     = "Accuracy",
             data       = dt)

ks <- fit$results$k
acc <- fit$results$Accuracy
par(mfrow = c(1, 1))
plot(ks, acc, main = 'cross-validation accuracy')
lines(ks, acc)
```




```
max.acc.idx <- which.max(fit$results$Accuracy)

print(paste('best accuracy of',
            round(acc[max.acc.idx], 2),
            'at k=', ks[max.acc.idx],
            sep = ' '))
```

```
## [1] "best accuracy of 0.57 at k= 12"
```

It may even be best in this case to use a moving average of the accuracy – in that case, around 18 would be best.

```
preds <- predict(fit, dt[, -'Total'])
confusionMatrix(preds, dt[, Total])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 142  87
##          1  61 115
##
##                Accuracy : 0.6346
##                  95% CI : (0.5856, 0.6816)
##     No Information Rate : 0.5012
##     P-Value [Acc > NIR] : 4.43e-08
##
##                   Kappa : 0.2689
##  Mcnemar's Test P-Value : 0.03988
##
##             Sensitivity : 0.6995
##             Specificity : 0.5693
##          Pos Pred Value : 0.6201
##          Neg Pred Value : 0.6534
##              Prevalence : 0.5012
##          Detection Rate : 0.3506
##    Detection Prevalence : 0.5654
##       Balanced Accuracy : 0.6344
##
##        'Positive' Class : 0
##
```

Overall, the best k seems to be in the range of 7-9 with an accuracy average around 57 between all three runs which I did.

The following will be how I actually run the data, ideally I will find a way to automatically load the weather data I need to my existing weather data set, then select the date in the where clause and input into the model

```
test<-fread("test.csv",stringsAsFactors = FALSE)
testing<-rbind(wd,test)
##Vew(testing)
test<-sqldf('select temp,   pressure,   humidity,   wind_speed, wind_deg,   clouds_all,Date,case when Da
to_Scale<-test %>% select( temp,    pressure,   humidity,   wind_speed, wind_deg,   clouds_all)
re_up<-test %>% select(Day_night,Date)
re_up$Day_night=as.numeric(re_up$Day_night)
test_scaled<-scale(to_Scale)
test<-cbind(re_up,test_scaled)
##Vew(test)
##Vew(tr.targs)
```

```
test<-sqldf('select temp,   pressure,   humidity,   wind_speed, wind_deg,   clouds_all,Day_night from t
            where Date="2017-04-05" and Day_night="1"')
##Vew(test)
str(test)
```

```
## 'data.frame':    1 obs. of  7 variables:
##  $ temp      : num -1.36
##  $ pressure  : num 0.207
##  $ humidity  : num -0.424
##  $ wind_speed: num -0.389
##  $ wind_deg  : num -1.41
##  $ clouds_all: num -0.597
##  $ Day_night : num 1
```

```
new <- knn(train = dt[,-'Total'], cl = dt$Total, test = test, k = 8)
View(new)
```

My model predicts an under for a day game based on the weather for April 5th 2017.