# Forecast_1

Diomides_Mavroyiannis

25/11/2020

#Loading the data

```
data2 <- read.csv("C:/Users/Dio and Nono/Desktop/Forecasting/Rstuff/Assignment2/Second bi-weekly assignm
data3 <- read.csv("C:/Users/Dio and Nono/Desktop/Forecasting/Rstuff/Assignment2/Second bi-weekly assignm

#Preparing the library
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(Mcomp)
library(fpp)
```

```
## Loading required package: fma
```

```
## Loading required package: expsmooth
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

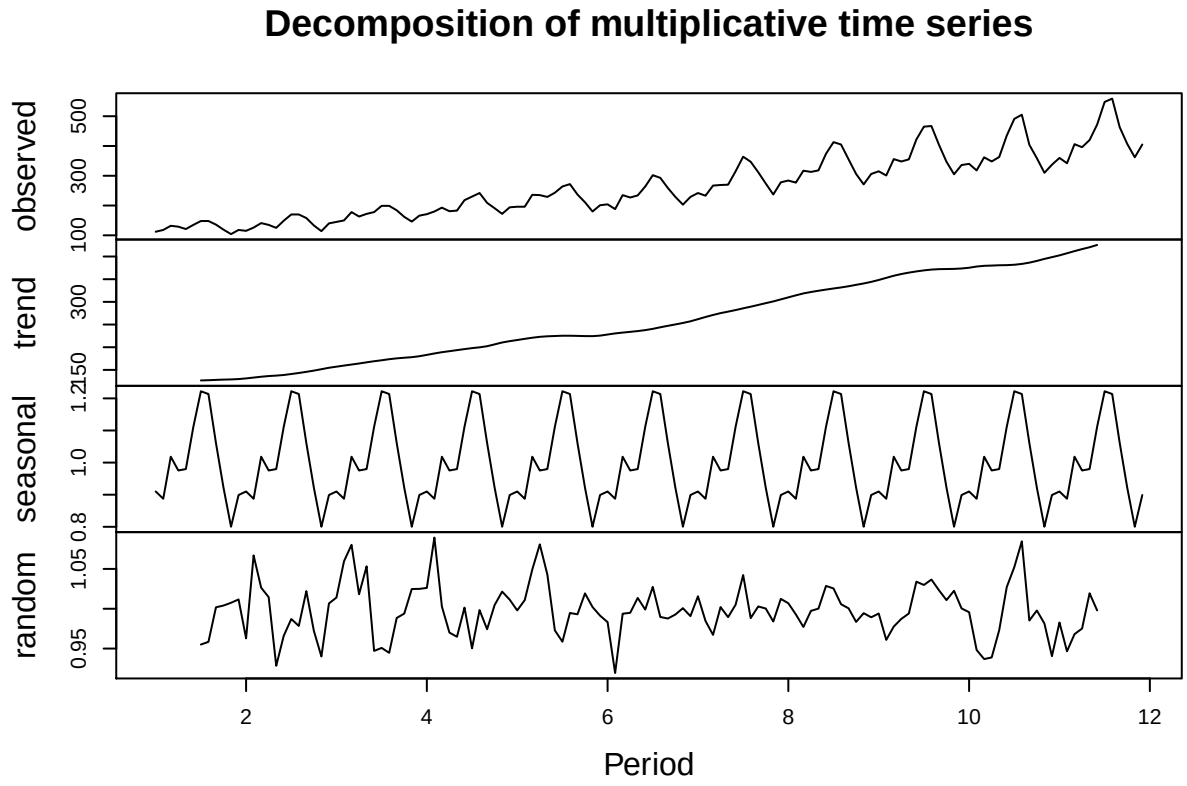
```
## Loading required package: tseries
```

```
#Part A
```

```
#set to time series monthly
tspassengers <- ts(data2$AirPassengers , frequency = 12)
tspassengers_test <- ts(data3$AirPassengers , frequency = 12)
#drop missing
dtspassengers  <-  na.omit(tspassengers)
#decompose monthly
adec <- decompose(dtspassengers , type = "additive")
mdec <- decompose(dtspassengers , type = "multiplicative")
```

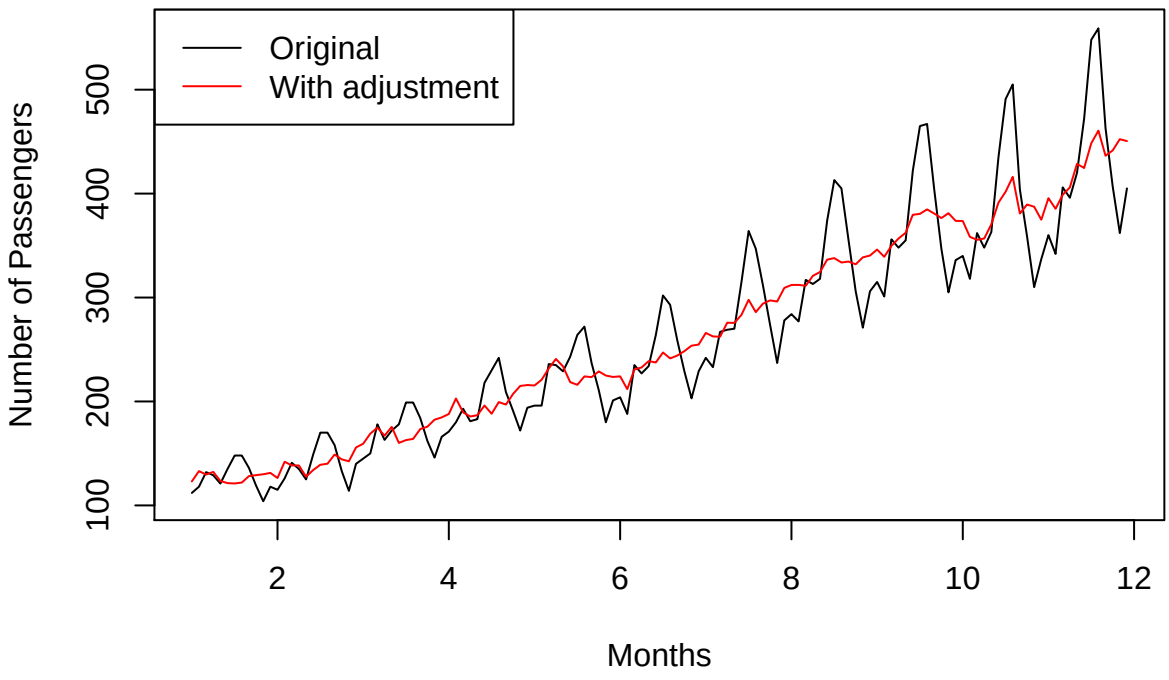
Because the sales are changing over time, the additive adjustment risks over adjusting in years where the sales are below average and under adjusting in years where sales are above average, as such the multiplicative adjustment is more appropriate

```
plot(mdec, type = "l", ylab = "Index", xlab = "Period")
```




```
#Create an adjusted time series
adtspassengers <- dtspassengers / mdec$seasonal
#Create a seasonally adjusted and non adjusted dataset without missing values
series1 <- na.omit(adtspassengers)
series2 <- na.omit(dtspassengers)
```

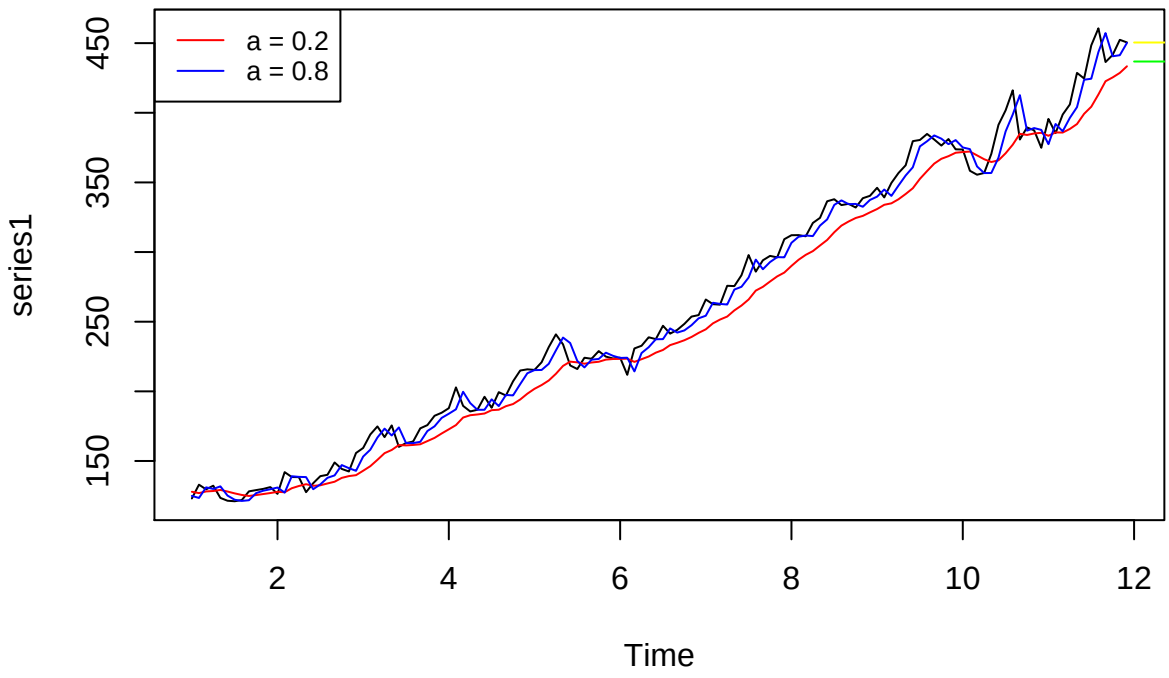
```
plot(series2, type = "l", ylab = "Number of Passengers", xlab = "Months" )
lines(series1, col = 'red')
legend('topleft', legend = c('Original', 'With adjustment'),
       col = c('black', 'red'), lty = 1 )
```




```
#forecast
fit1  <- ses(series1, alpha = 0.2, h=6)
fit2  <- ses(series1, alpha = 0.8, h=6)
```

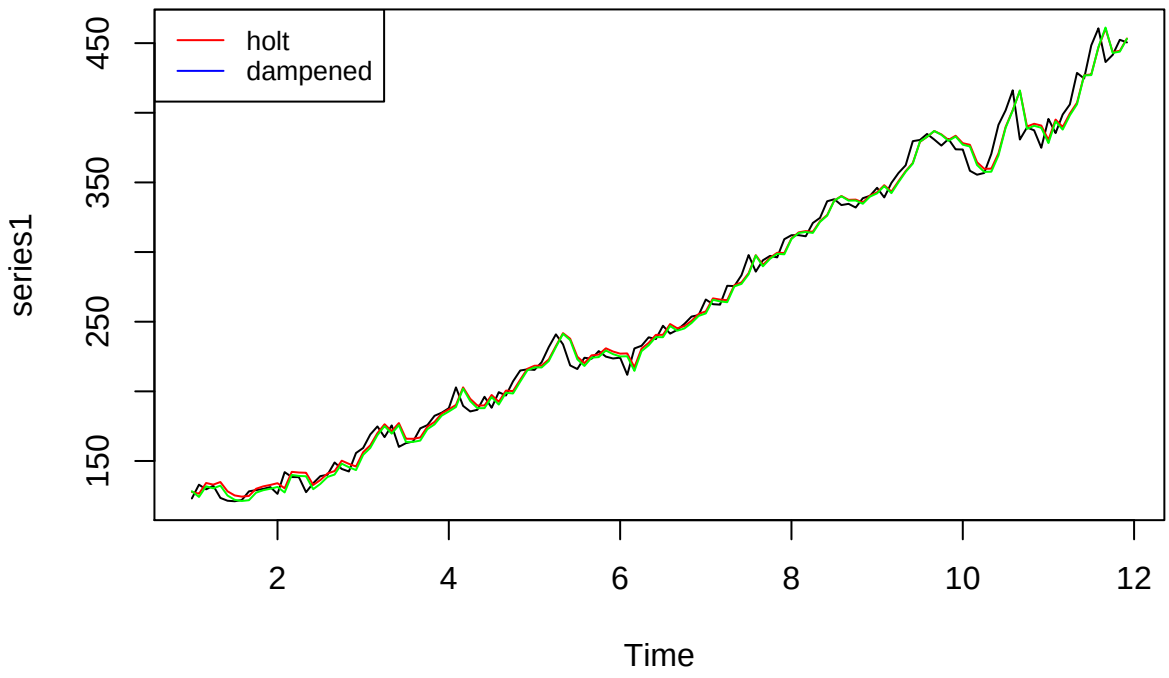
Fit 2 looks better so we will use that

```
plot(series1, type = 'l')
lines(fit1$fitted, col= 'red')
lines(fit1$mean, col= 'green')
lines(fit2$fitted, col= 'blue')
lines(fit2$mean, col= 'yellow')
legend('topleft', legend = c('a = 0.2','a = 0.8'),
       col = c('red', 'blue'), lty = 1, cex = 0.8)
```




```
model1 <- holt(series1, h=3)
model2 <- holt(series1, damped = TRUE,  h=3)
```

```
plot(series1, type = 'l')
lines(model1$fitted , col= 'red')
lines(model2 $fitted , col= 'green')
legend('topleft', legend = c('holt','dampened'),
       col = c('red', 'blue'), lty = 1, cex = 0.8)
```




To compare we split our series1 into train and test

```
train <- series1
test <- na.omit(tspassengers_test)

trainsimple <- ses(train, alpha = 0.8, h=12)
trainholt <- holt(train, h=12)
traindamped <- holt(train, damped = TRUE,  h=12)

#Compute the in sample accuracy
mse_in_simple  <- mean( (train - trainsimple$fitted )^2 )
mse_in_holt  <- mean( (train - trainholt$fitted )^2 )
mse_in_damped  <- mean( (train - traindamped$fitted )^2 )

#Compute the out sample accuracy
mse_out_simple  <- mean( (test - trainsimple$fitted )^2 )
mse_out_holt  <- mean( (test - trainholt$fitted )^2 )
mse_out_damped  <- mean( (test - traindamped$fitted )^2 )

c(mse_in_simple, mse_out_simple)
```

```
## [1]     86.01671 128188.90171
```

```
c(mse_in_holt,mse_out_holt )
```

```
## [1]     76.50818 126044.72363
```

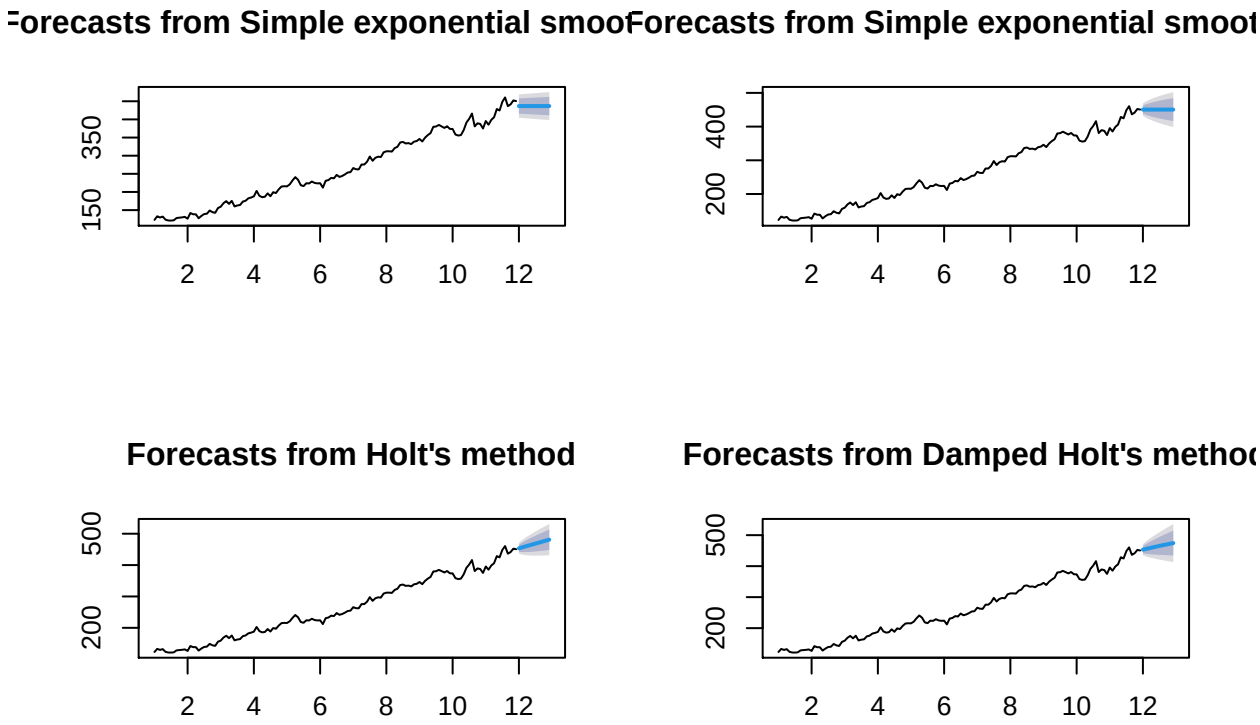
```r
c(mse_in_damped, mse_out_damped)
```

```
## [1]     80.18093 127851.95272
```

```r
#So the lowest MSE error is the Holt

model1 <- ses(series1, alpha = 0.2, h=12)
model2 <- ses(series1, alpha = 0.8, h=12)
model3 <- holt(series1, h=12)
model4 <- holt(series1, damped = TRUE, h=12)
```

```r
par(mfrow = c(2,2))
plot(model1);plot(model2)
plot(model3);plot(model4)
```

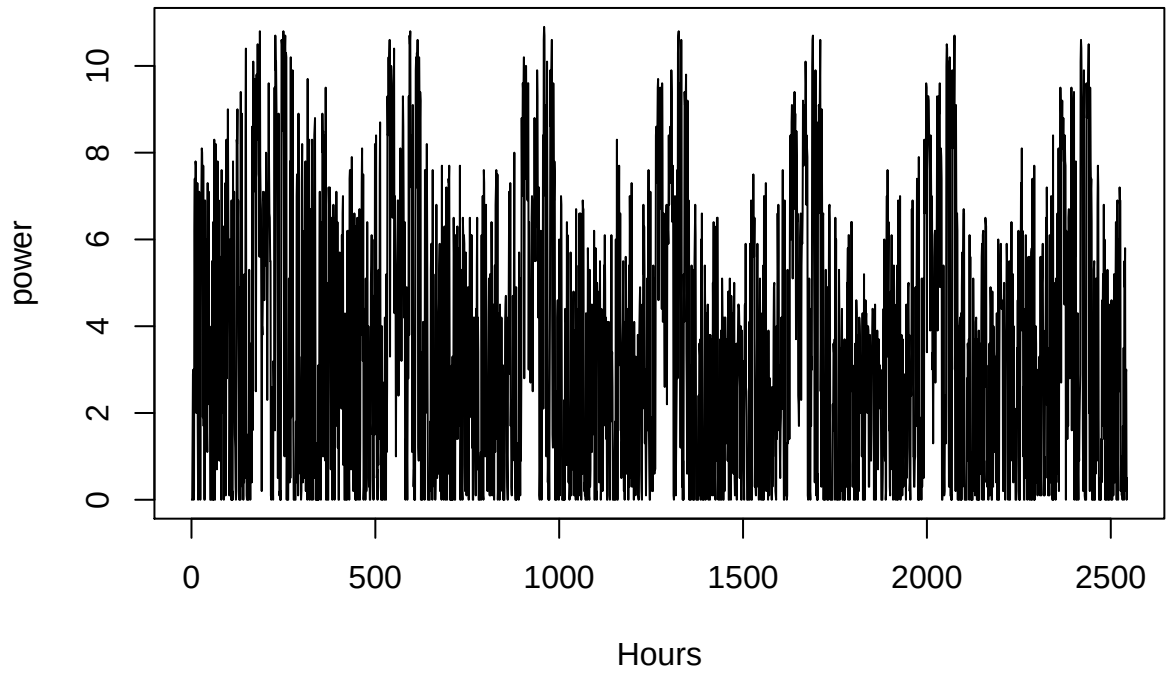



Exercise B

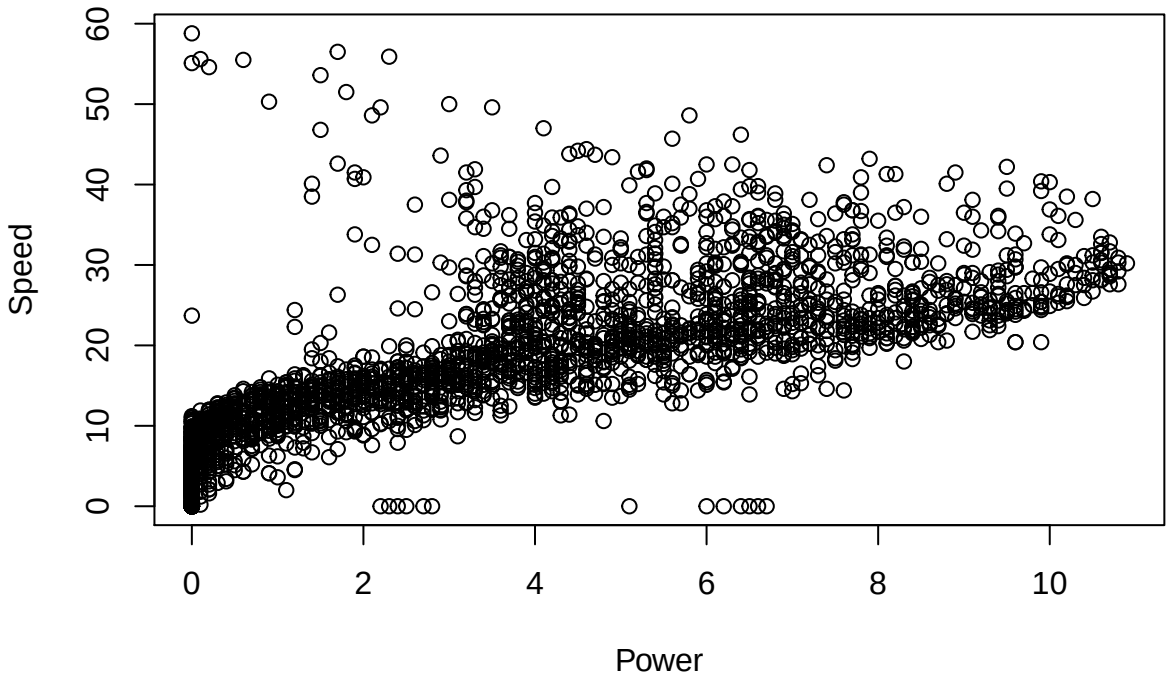
```r
#Make dataset without the first column the data
trainset <- data2[-1]
testset <- data3[-1]

#Fit a regression
mlr <- lm(power ~ speed, data=trainset)
frc_mlr <- as.numeric(predict(mlr, testset))
```

```r
plot(data2$power , type = "l", ylab = "power", xlab = "Hours" )
```




```r
plot(x =data2$power, y = data2$speed, xlab = 'Power', ylab = 'Speed' )
```




```
cor(data2$power,data2$speed)
```

```
## [1] 0.7326155
```

This is a pretty high level of correlation! Now on to the neural network!

```
#Rescale the data
trainset_s <- trainset
trainset_s$speed <- (trainset_s$speed-min(trainset_s$speed))/(max(trainset_s$speed)-min(trainset_s$speed
trainset_s$power <- (trainset_s$power-min(trainset_s$power))/(max(trainset_s$power)-min(trainset_s$power

#Rescale the data
testset_s <- testset
testset_s$speed <- (testset_s$speed-min(testset_s$speed))/(max(testset_s$speed)-min(testset_s$speed))
testset_s$power <- (testset_s$power-min(testset_s$power))/(max(testset_s$power)-min(testset_s$power))

library(neuralnet)
set.seed(512)
mlp1 <- neuralnet(power ~ speed, data=trainset_s,
                  hidden = 1, act.fct = "logistic", linear.output = FALSE)
#This is our forecast but scaled
frc_mlp_s <- as.numeric(predict(mlp1, testset_s))


#Now we unscale it
frc_mlp <- frc_mlp_s*(max(trainset$power)-min(trainset$power))+min(trainset$power)
data.frame(frc_mlp,testset$power)
```

```
##      frc_mlp testset.power
## 1  6.4465076           2.4
## 2  5.9771731           3.0
## 3  4.7681263           2.1
## 4  6.1319886           1.9
## 5  0.2635972           0.0
## 6  0.2669094           0.0
## 7  2.9209639           0.0
## 8  5.1799698           1.5
## 9  6.5992996           7.4
## 10 3.5208850           0.8
## 11 6.4223166           4.6
## 12 3.7174968           1.2
## 13 6.5992499           9.0
## 14 1.3439708           1.2
## 15 0.2964732           0.0
## 16 6.5297536           4.4
## 17 0.4097138           0.0
## 18 6.3110250           3.1
## 19 6.3943474           6.9
## 20 6.5994690           8.6
## 21 5.1169709           3.1
## 22 6.5994711           8.9
## 23 6.0335454           2.2
## 24 6.5661749           7.6
```

```
#So this is our prediction  the data
```

Finally, we compare the actual data with our predictions.

```
#RMSE of regression and neural network
sqrt(mean((testset$power-frc_mlr)^2))
```

```
## [1] 1.387682
```

```
sqrt(mean((testset$power-frc_mlp)^2))
```

```
## [1] 2.419144
```