HW_10_1

```
> MoE =
c(29.8,33.2,33.7,35.3,35.5,36.1,36.2,36.3,37.5,37.7,38.7,38.8,39.6,41,42.8,42.8,43.5,45.6,46,46
.9,48,49.3,51.7,62.6,69.8,79.5,80)
> Strength =
c(5.9,7.2,7.3,6.3,8.1,6.8,7,7.6,6.8,6.5,7,6.3,7.9,9,8.2,8.7,7.8,9.7,7.4,7.7,9.7,7.8,7.7,11.6,11
.3,11.8,10.7)
> plot(Strength,MoE)
> boxplot(Strength)
> boxplot(MoE)
> qqnorm(MoE)
> qqnorm(Strength)
> meanMoE = mean(MoE)
> meanStrength = mean(Strength)
> sdMoE = sd(MoE)
> sdStrength = sd(Strength)
> zOfMoe = (MoE -meanMoE)/sdMoE
> zOfStrength = (Strength - meanStrength) / sdStrength
> sumZ = sum(zOfMoe * zOfStrength)
> sumZ / 26
[1] 0.8592721
> cor(MoE,Strength)
[1] 0.8592721
> Sxy = sum((MoE-meanMoE)*(Strength - meanStrength))
> Sxx = sum((MoE - meanMoE)^2)
> Sxy / Sxx
[1] 0.1074821
> meanStrength - 0.107 * meanMoE
[1] 3.314248
> slope = Sxy / Sxx
> intercept = meanStrength - slope*meanMoE
> predictedY = slope*MoE + intercept
> SSE = sum((Strength-predictedY)^2)
> SSE
[1] 18.7356
```

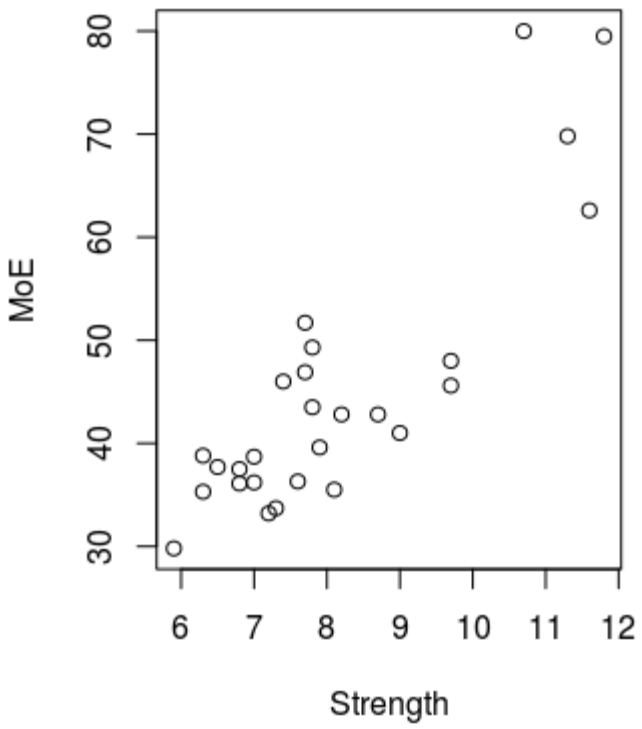


Illustration 1: Scatter Plot of Strength vs MoE

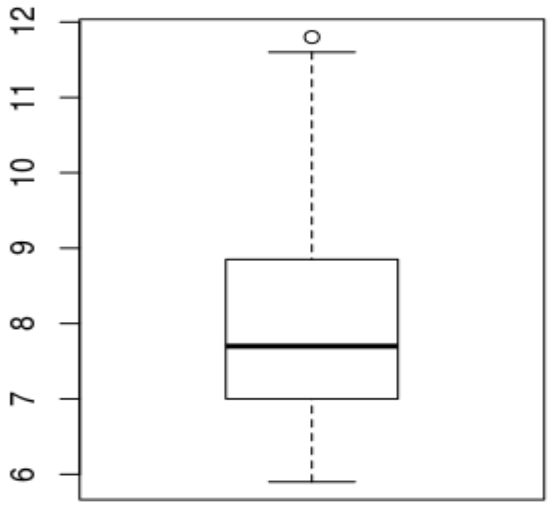


Illustration 2: Box plot of Strength

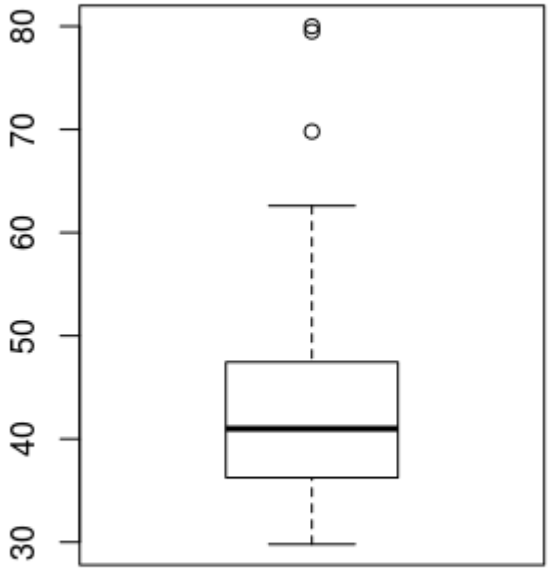


Illustration 3: Box plot of MoE

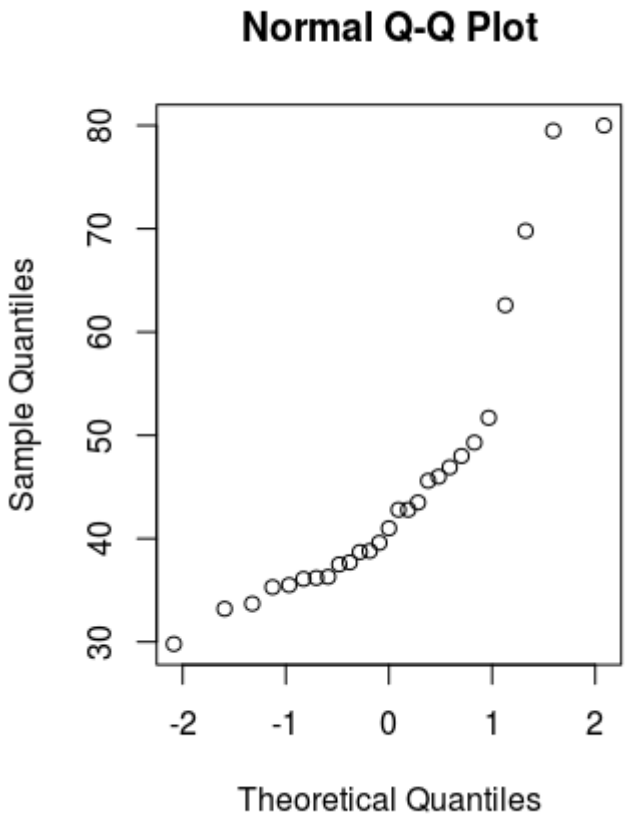


Illustration 4: QQ Plot of MoE

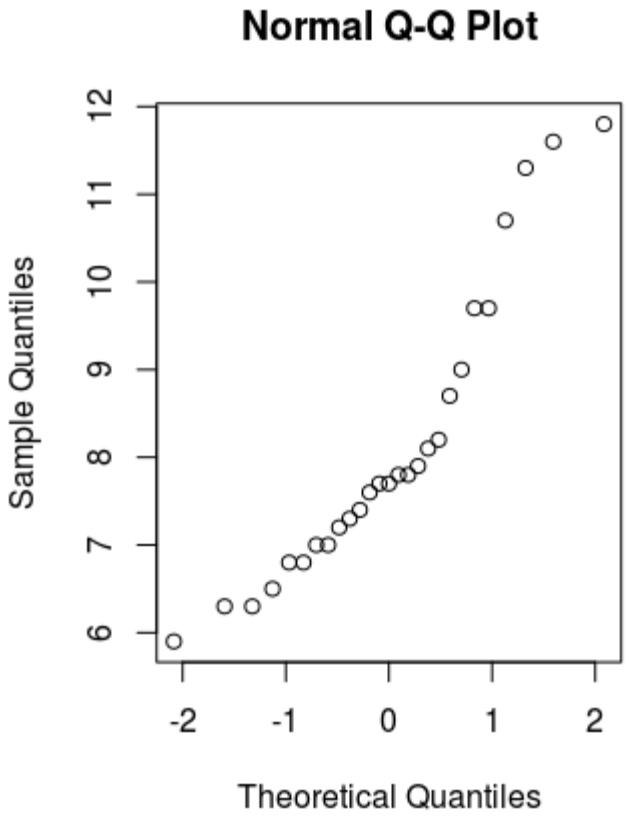


Illustration 5: QQ Plot of Strength

HW_11_1

```
> x = c(45,58,71,7,85,98,108)
> y = c(3.2,3.4,3.47,3.55,3.6,3.7,3.8)
> meanX = mean(x)
> meanY = mean(y)
> Sxy = sum((x-meanX)*(y-meanY))
> Sxx = sum((x-meanX)^2)
> slope = Sxy / Sxx
> intercept = meanY - slope* meanX
> Sxy
[1] 24.58571
> Sxx
[1] 7145.714
> meanX
[1] 67.42857
> x-meanX
[1] -22.428571  -9.428571   3.571429 -60.428571
[5]  17.571429  30.571429  40.571429
> slope
[1] 0.003440624
> meanY
[1] 3.531429
> meanY
[1] 3.531429
> meanX
[1] 67.42857
> intercept
[1] 3.299432
> sum((y-meanY)^2)
[1] 0.2364857
> predictedY = slope*x + intercept
> SSExplain = sum((predictedY-meanY)^2)
> SSError = sum((y-predictedY)^2)
> SSExplain + SSError
[1] 0.2364857
> SSExplain
[1] 0.08459019
> SSError
[1] 0.1518955
> 1-(SSError/0.236)
[1] 0.3563749
> sd(y)
[1] 0.1985303
```

HW_11_2

```
> x = runif(100,-1,1)
> error = rnorm(100,0,0.5)
>  y = 2+3*x+error
>
>  meanX = mean(x)
> meanY = mean(y)
> Sxy = sum((x-meanX)*(y-meanY))
> Sxx = sum((x-meanX)^2)
>  slope = Sxy / Sxx
>
>  intercept = meanY - slope*meanX
>  yHat = slope*x + intercept
>  sum((yHat - meanY)*(y-yHat))
[1] 6.327447e-14
```

HW_12_1

```
data <- read.table('http://www.stat.washington.edu/marzban/390/summer17/transform_data.txt')
> xAxis = data[2:101,1]
> yAxis = data[2:101,2]
> xAxis = as.numeric(levels(xAxis))[xAxis]
> yAxis = as.numeric(levels(yAxis))[yAxis]
> plot(xAxis,yAxis)
> plot(sqrt(xAxis),sqrt(yAxis))
```

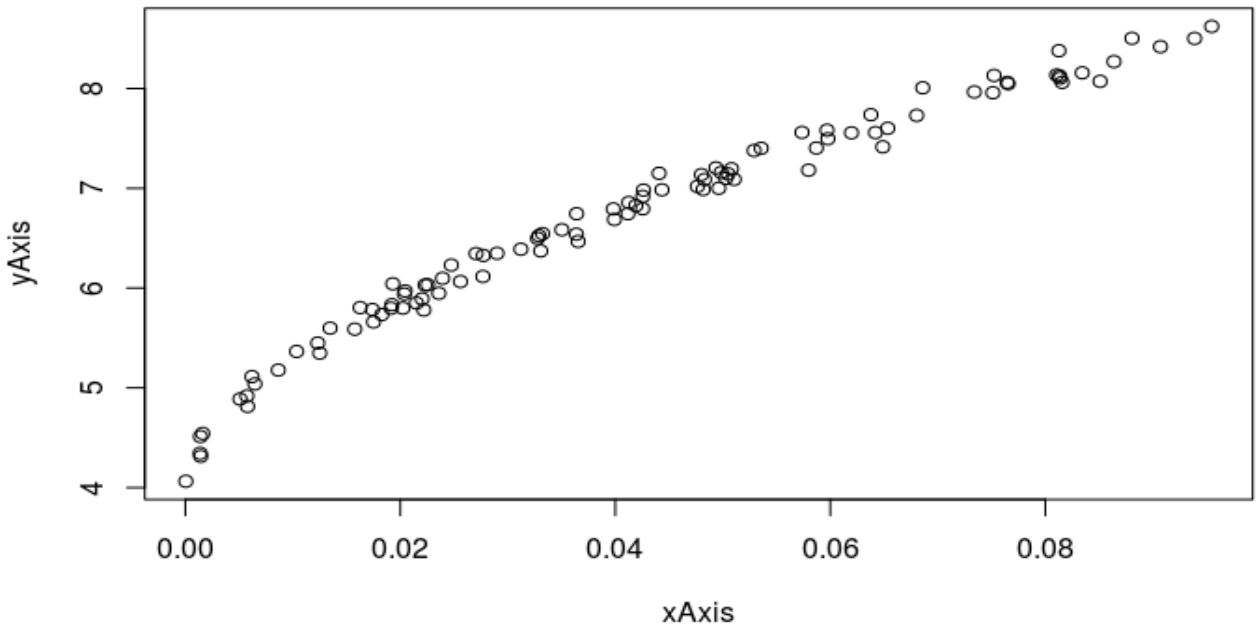


Illustration 3: Original Data

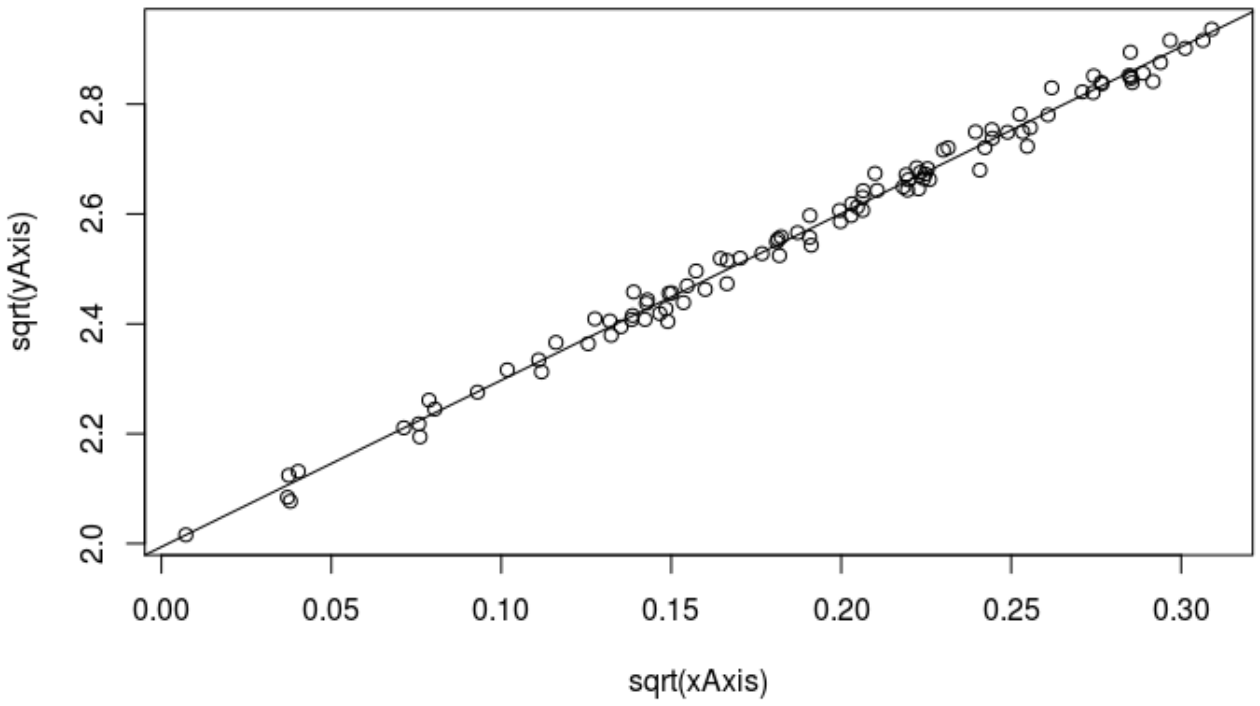
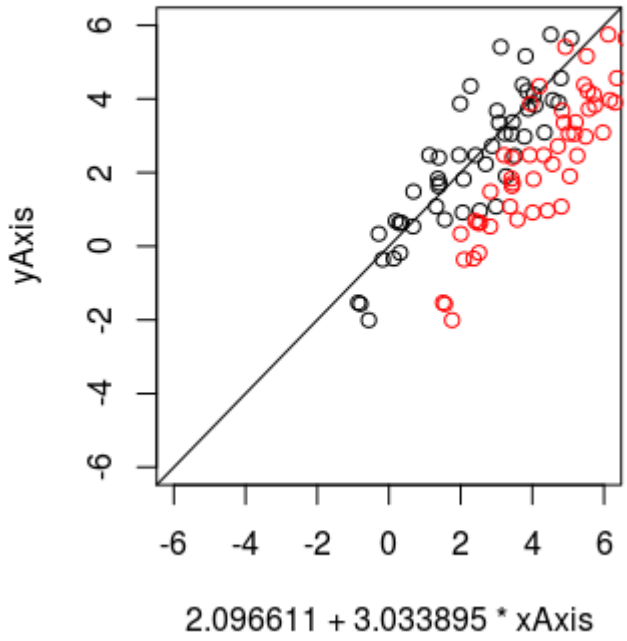


Illustration 4: Transformed and linear data

HW_12_3

```
> data = read.table('http://www.stat.washington.edu/marzban/390/summer17/bias_0_data.txt')
> xAxis = data[2:51,1]
> yAxis = data[2:51,2]
> xAxis = as.numeric(levels(xAxis))[xAxis]

> yAxis = as.numeric(levels(yAxis))[yAxis]

> xVal = lm(yAxis~xAxis)
> xVal[1][1]
$coefficients
(Intercept)       xAxis
   2.096611    3.033895
> plot(2.096611+3.033895*xAxis,yAxis,xlim = range(-6,6),ylim = range(-6,6))

> abline(0,1)
newData = read.table('http://www.stat.washington.edu/marzban/390/summer17/bias_1_data.txt')
> xNewData = newData[2:52,1]
> yNewData = newData[2:52,2]
> xNewData = as.numeric(levels(xNewData))[xNewData]
> yNewData = as.numeric(levels(yNewData))[yNewData]
> lm(yNewData~xNewData)

Call:
```

```
lm(formula = yNewData ~ xNewData)

Coefficients:
(Intercept)     xNewData
      4.044        2.603

> points(2.603*xNewData + 4.044,yNewData,col = 2)
```

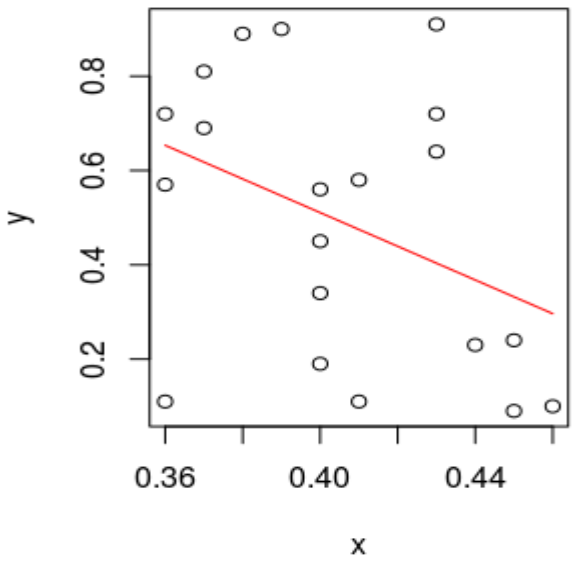


c) We can see that the red line is shifted to the right of the plot. Suggesting that the predicted value is higher than the observed value. This is because of the appearance of an outlier
To be exact, the last data on the bias_1_data has the value (0,100). The y value in this case is odd, in which the regular y value is around -2 or 6 . This in turn affect the coefficient of the linear model because of the outlier. Hence making the prediction shift to the right.

HW_12_4

```
> x= c( 0.4, 0.46, 0.41, 0.45, 0.44, 0.36, 0.45, 0.43, 0.38, 0.39, 0.4, 0.4, 0.41, 0.37,
+                0.43, 0.37, 0.4, 0.36, 0.43, 0.36)
>
>
> y = c(0.19, 0.1, 0.11, 0.09, 0.23, 0.11, 0.24, 0.91, 0.89,
+                 0.9,0.34,0.45,0.58,0.69,0.72,0.81,0.56,0.57,0.64,0.72)
> lm(y~x)

Call:
lm(formula = y ~ x)

Coefficients:
```

```
(Intercept)           x
      1.939      -3.571

> plot(x,y)
lines(x[order(x)], -3.571*x[order(x)] + 1.939 , col = 2)
plot(-3.571*x+1.939, y - (-3.571*x+1.939))
summary(interpret)$r.squared
[1] 0.1532254
> se = sqrt(( 1 / (20 - 2)) * sum((y - (-3.571*x+1.939))^2))
> se
[1] 0.2777241
plot(x,y)
lm(y ~ poly(x, 2,raw = TRUE))
> lines(orderdX, -11.56+ 63.23*orderdX -82.16*orderdX^2,col = 2)
> plot(1/x, y)
```



a) The coeffiecient of the regression line has the slope -3.571 and intercept at 1.939. Meaning that an increase in 1 of the x value will decrease the y value at -3.571. Since this is a valuation ranging from 0 to 1 , we can also say that an increase in 0.01 will decrease the y value at 0.03571

c) The R square value come out at 0.1532. Meaning that only 15% of the variability in y can be explained in X. Meaning that 85% of the change in Y do not depend on X but instead on different variable that also affect Y in addition to X

d) The standard deviation of error is 0.27. Meaning that on average, the error between the predicted value of y is differe +- 0.27 from the actual value of y. This is a huge error since the data is only within a range of 0 – 1 and the error is about 20% already

b) The residual plot of the error between the difference between y and the predicted value of y vs predicted value of y appear in a random order. This conclude that the linear model is not good because there are too many error in the plot.

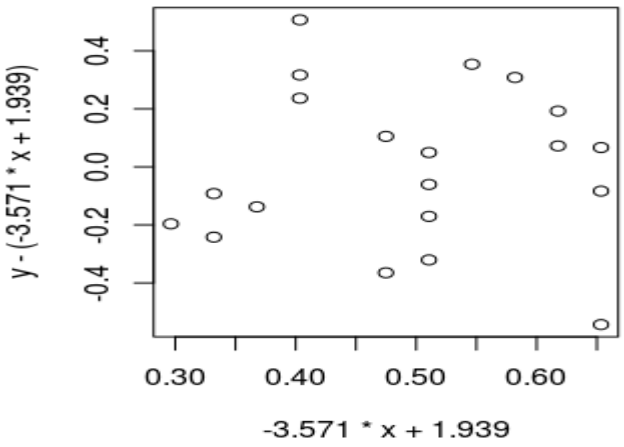


Illustration 5: Residual Plot

e) yes, the data need to be transformed into another degree polynomial or alter the data. However, since the R^2 is still small, it is best to find another variable that also relate to Y so that the data can be more accurate

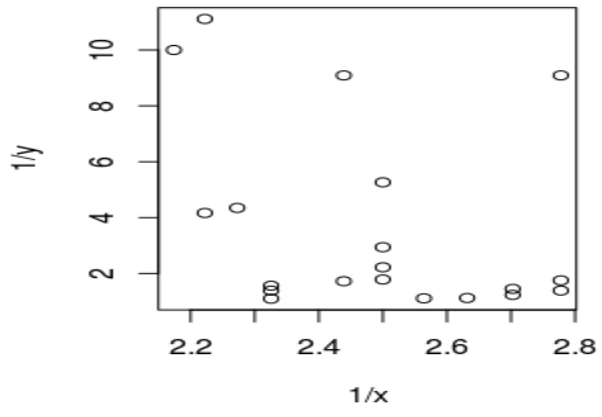


Illustration 6: 1/x and 1/y transformation

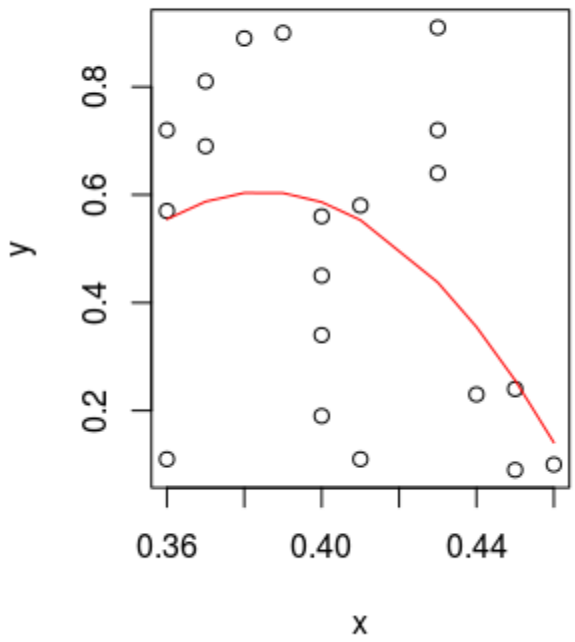


Illustration 7: Second Degree Polynomial

3.33

```
> Thickness =
c(220,220,220,220,370,370,370,370,440,440,440,440,680,680,680,680,860,860,860,860)
> Strength =
c(24,22,19.1,15.5,26.3,24.6,23.1,21.2,25.2,24,21.7,19.2,17,14.9,13,11.8,12.2,11.2,6.6,2.8)

> fit2 = lm(Strength ~ poly(Thickness, 2,raw = TRUE))
> fit2

Call:
lm(formula = Strength ~ poly(Thickness, 2, raw = TRUE))

Coefficients:
                      (Intercept)
                        1.452e+01
poly(Thickness, 2, raw = TRUE)1
                        4.323e-02
poly(Thickness, 2, raw = TRUE)2
```

```
                        -6.001e-05

> orderdX = Thickness[order(Thickness)]
> plot(Thickness, Strength)
> lines(orderdX, 1.452e+01  + 4.323e-02*orderdX -6.001e-05*orderdX^2 ,col = 2)
> se = sqrt(( 1 / (20 - 2)) * sum((Strength - (1.452e+01  + 4.323e-02*Thickness -6.001e-
05*Thickness^2))^2))
> se
[1] 3.177259
> summary(fit2)$r.squared
[1] 0.779747
1.452e+01  + 4.323e-02*500 -6.001e-05*500^2
[1] 21.1325
```

a) No it is not possible to transform because the data Thickness is considered a qualitative variable. We can't treat it like a continuous variable and transform it (making assumption that the Strength will be correlated with it)

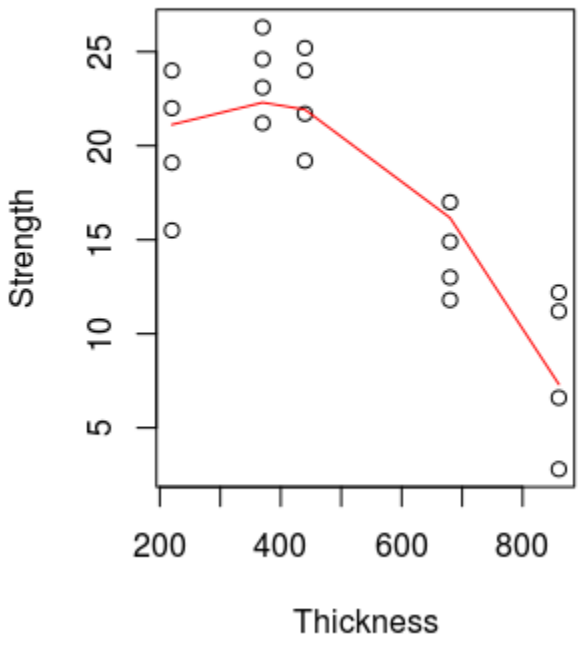


Illustration 8: Quadratic Fitted Line

b) Looking at the R^2 of the quadratic model, we can say that 77% of the Strength can be predicted by using Thickness. Meaning that the thickness variable is responsible for 77% of strength, Another 23% of Strength belong to another variables.
The se also sugesst that on average, the predicted value of strength using Thickness is +- 3.17 only from the observed value