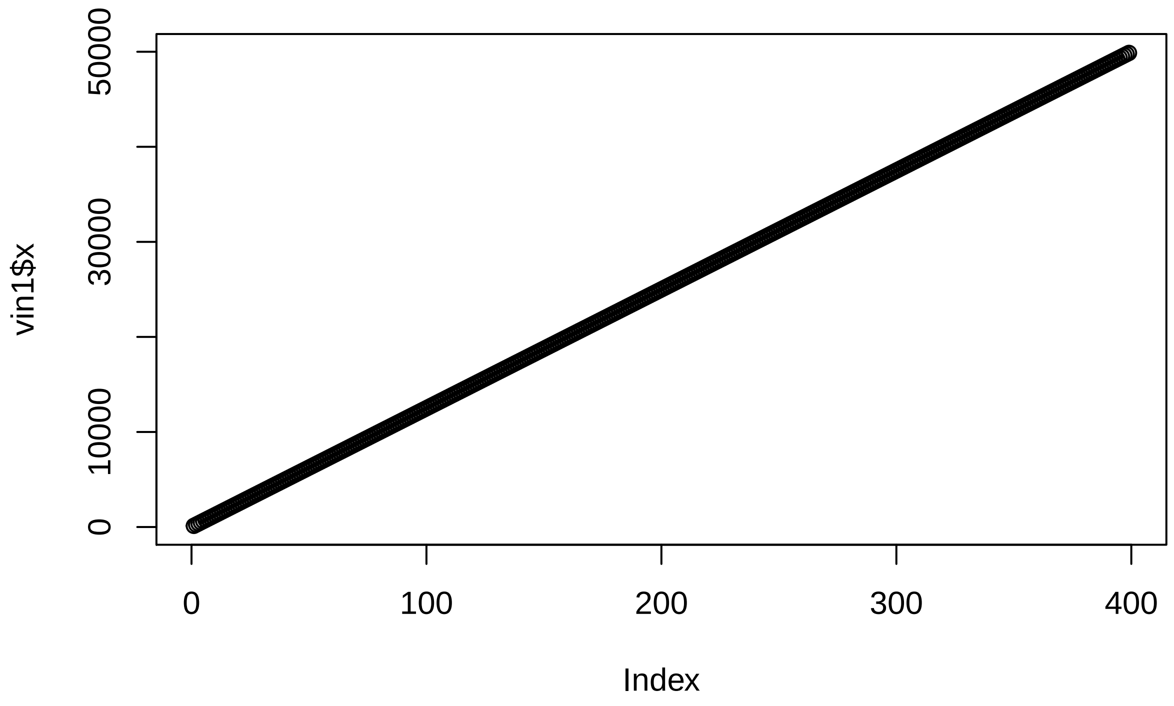
# analysisStep3

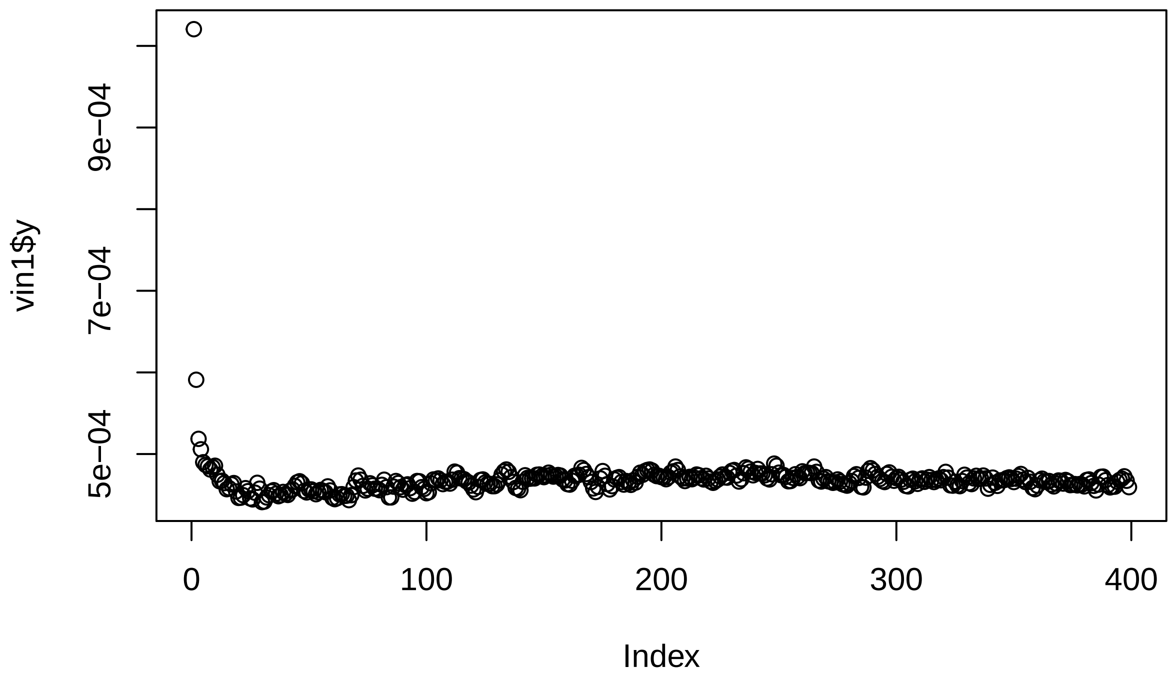
*Madeleine Allen*

*2/6/2019*

```
vin1<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/VIN1.CSV")

names(vin1)<-c("x", "y")
plot(vin1$x)
```

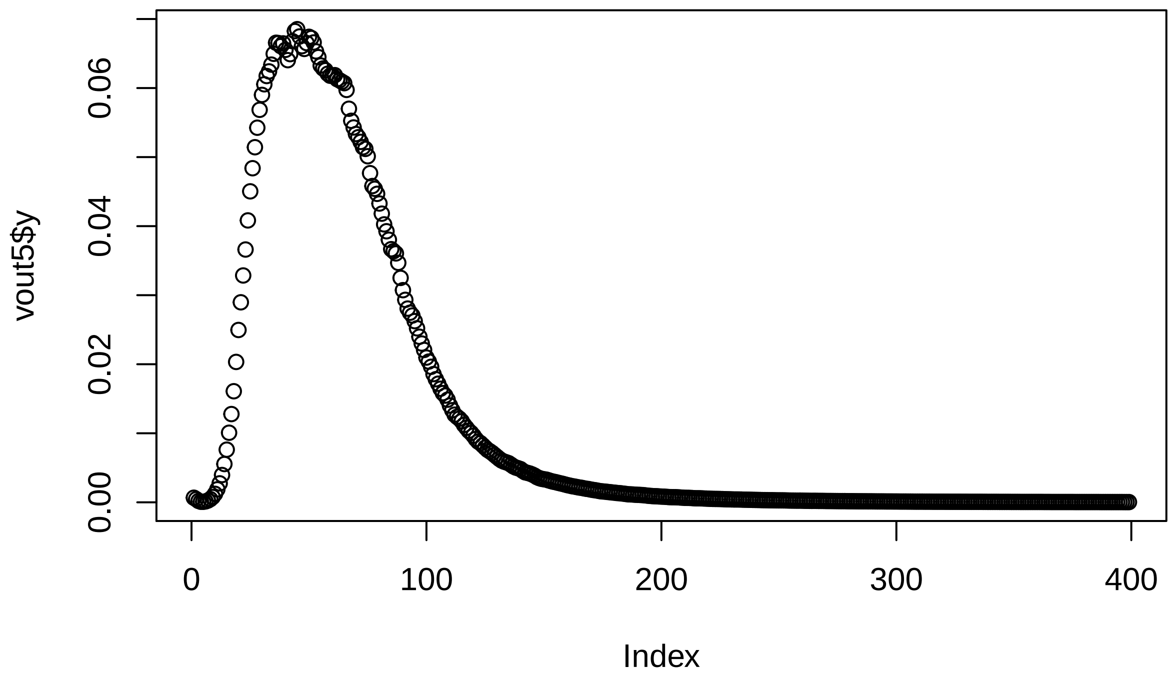


```
plot(vin1$y)
```



Now I'll upload the rest of the data. I'll plot an out graph for reference too.

```r
vin2<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/VIN2.CSV")
names(vin2)<-c("x", "y")

vin3<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/VIN3.CSV")
names(vin3)<-c("x", "y")
vin4<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/VIN4.CSV")
names(vin4)<-c("x", "y")
vin5<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/VIN5.CSV")
names(vin5)<-c("x", "y")

vout1<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/VOUT1.CSV")
names(vout1)<-c("x", "y")
vout2<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/VOUT2.CSV")
names(vout2)<-c("x", "y")
vout3<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/VOUT3.CSV")
names(vout3)<-c("x", "y")
vout4<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/VOUT4.CSV")
names(vout4)<-c("x", "y")
vout5<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/VOUT5.CSV")
names(vout5)<-c("x", "y")
plot(vout5$y)
```



Now to calculate the errors for each measurement (which will be propagated into the final error for the gain):

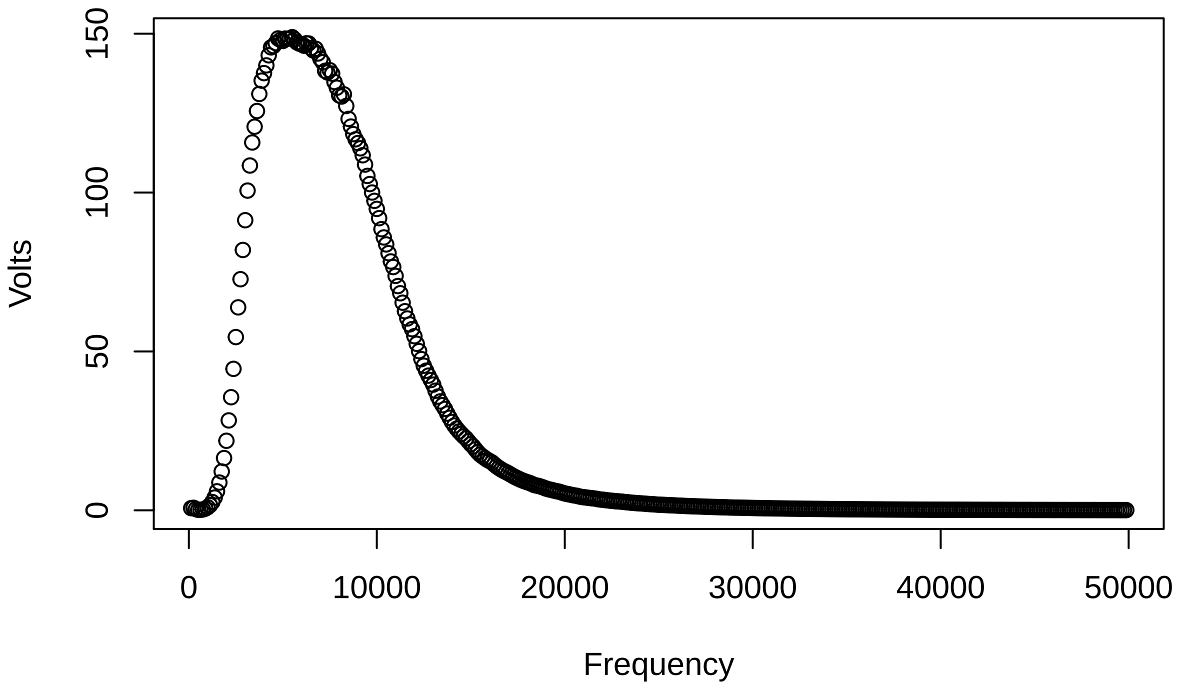
```
vin1_error<- sd(vin1$y, na.rm=TRUE)/sqrt(length(vin1$y[!is.na(vin1$y)]))
vin2_error<-sd(vin2$y, na.rm=TRUE)/sqrt(length(vin2$y[!is.na(vin2$y)]))
vin3_error<- sd(vin3$y, na.rm=TRUE)/sqrt(length(vin3$y[!is.na(vin3$y)]))
vin4_error<- sd(vin4$y, na.rm=TRUE)/sqrt(length(vin4$y[!is.na(vin4$y)]))
vin5_error<- sd(vin5$y, na.rm=TRUE)/sqrt(length(vin5$y[!is.na(vin5$y)]))

vout1_error<-sd(vout1$y[35:44], na.rm=TRUE)/sqrt(length(vout1$y[!is.na(vout1$y)]))
vout2_error<- sd(vout2$y[37:54], na.rm=TRUE)/sqrt(length(vout2$y[!is.na(vout2$y)]))
vout3_error<- sd(vout3$y[36:56], na.rm=TRUE)/sqrt(length(vout3$y[!is.na(vout3$y)]))
vout4_error<- sd(vout4$y[35:53], na.rm=TRUE)/sqrt(length(vout4$y[!is.na(vout4$y)]))
vout5_error<- sd(vout5$y[35:53], na.rm=TRUE)/sqrt(length(vout5$y[!is.na(vout5$y)]))
```

Find the mean of each value: It should be noted that the vin voltages are flat whereas the vout voltages are peaked, so dont just take average

```
m_in<- (vin1$y+vin2$y+vin3$y+vin4$y+vin5$y)/5
#take average of vouts
m_vout <- data.frame(Frequency = vout1$x, Volts = (vout1$y+vout2$y+vout3$y+vout4$y+vout5$y)/5)
```

Now we have the mean in and the mean out so we can find the gain:

```
#compute gain using the average vouts and m_in
gain <- data.frame(Frequency = vout1$x, Gain = (m_vout[2]/m_in))
plot(gain)
```

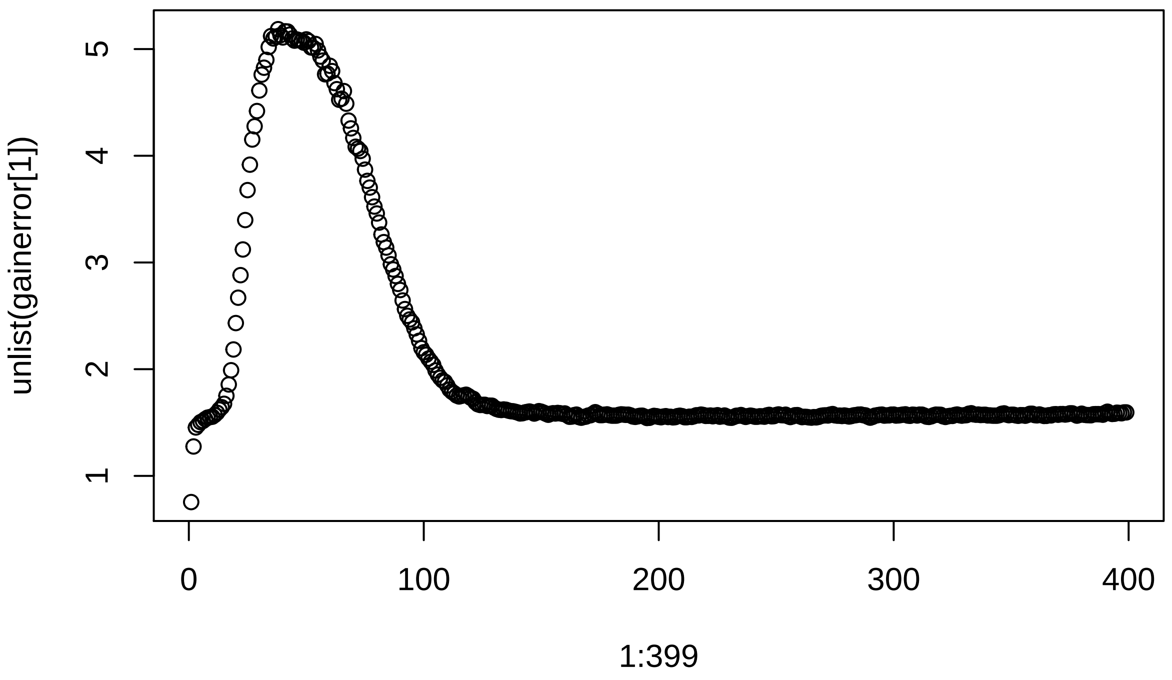


```
#this is consistent with the max gain of 150 calculated in class.
```

## Calculating Error

not really sure how to calculate error of a function. going to take RMSE for each.

```
rmserrors <- sqrt(data.frame(
  vin1 = sum(((m_in-vin1$y)^2))/399,
  vin2 = sum(((m_in-vin2$y)^2))/399,
  vin3 = sum(((m_in-vin3$y)^2))/399,
  vin4 = sum(((m_in-vin4$y)^2))/399,
  vin5 = sum(((m_in-vin5$y)^2))/399,
  vout1 = sum((m_vout$Volts-vout1$y)^2)/399,
  vout2 = sum((m_vout$Volts-vout2$y)^2)/399,
  vout3 = sum((m_vout$Volts-vout3$y)^2)/399,
  vout4 = sum((m_vout$Volts-vout4$y)^2)/399,
  vout5 = sum((m_vout$Volts-vout5$y)^2)/399
))
#error in gain, adding in quadrature:
vinerror <-sqrt(sum(rmserrors[1:5]^2))
vouterror <- sqrt(sum(rmserrors[6:10]^2))
gainerror <- gain[2]*sqrt((vinerror/m_in)^2+(vouterror*(m_vout[2])^-1)^2)
#g(f) error
plot(1:399,unlist(gainerror[1]))
```



## Experiment 2: Johnson Noise 1

Recording the resistor values measured during lab.

```
#everything will be in ohms
short<- .03
shortError<-.001

k20<-20090
k20error<-1

k35 <- 35230 #secretly 35.2 but that would be an ugly variable name
k35error<-1

k100 <- 100700
k100error <- 1

k10 <- 999.05
k10error<- .01

k1 <- 998.17
k1error <- .01

k48 <- 48650 #secretly 48.7k but again that would be an ugly variable name
k48error<- 1
```

### Experiment 3: Johnson Noise - Boltzmann Constant

```
experiment3data1<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/experiment3data1.csv")
```

Calculate Vmeas, V, and Vsystem

```r
Vsys<- experiment3data1[1,7] #first row 7th column
VsysError <- experiment3data1[1,9]

Vmeask1<- (experiment3data1[2,7])
Vmeask10<-experiment3data1[3,7]
Vmeask20 <-experiment3data1[4,7]
Vmeask32<-experiment3data1[5,7]
Vmeask48<-experiment3data1[6,7]
Vmeask100<-experiment3data1[7,7]

Vmeas<-c(Vmeask1, Vmeask10, Vmeask20, Vmeask32, Vmeask48, Vmeask100)

#need to redo the error later (2/5)
VmeasError<-sqrt((sum(experiment3data1[2:7,9])^2))

V<- sqrt(-Vsys^2+Vmeas^2)
```

```
## Warning in sqrt(-Vsys^2 + Vmeas^2): NaNs produced
```

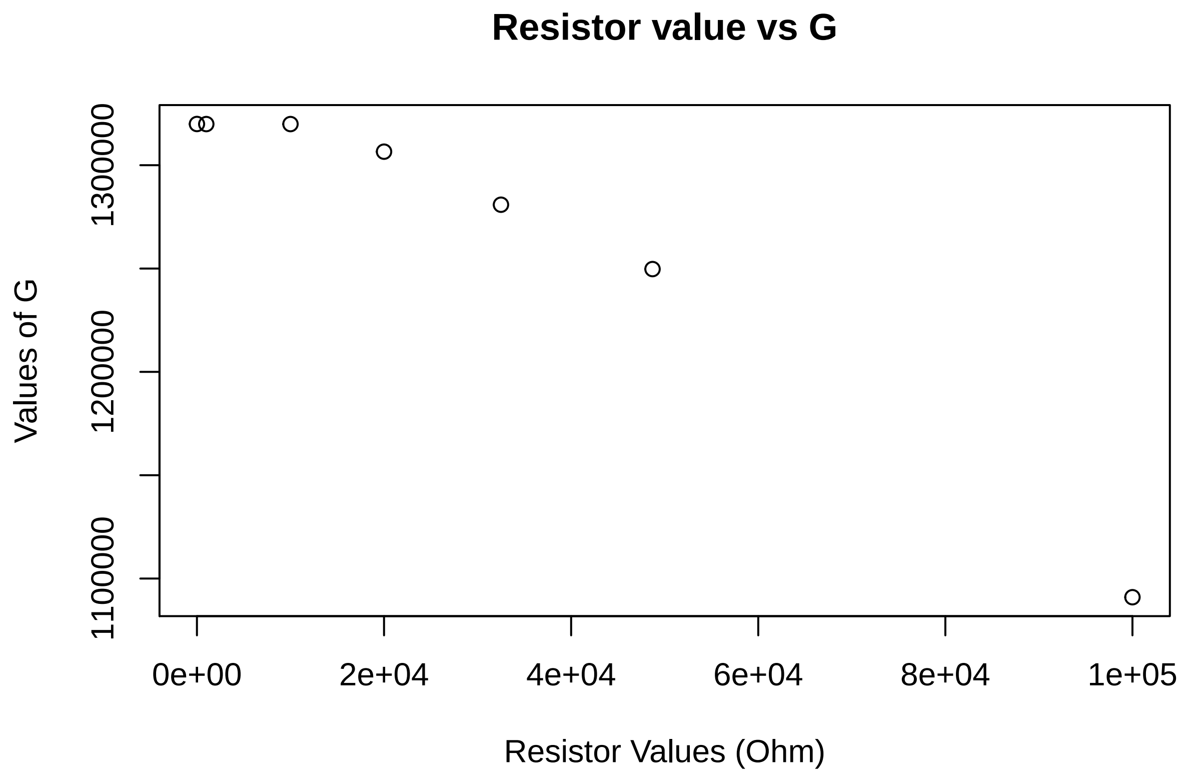
```r
Verror<- sqrt(VmeasError^2+ VsysError^2)
```

## Calculating G

```r
gain <- data.frame(Frequency = vout1$x, Gain = (m_vout[2]/m_in))

capacitance <-87.875*(10^-12)
capacitanceError <-.594*(10^-12)
#df is just the x componenent

riemanSum <- function(fa,fb){
  area <-0.5*(125)*(fb-fa)+fa*125
  return(area)
}

resistors<-read.csv("experiment3data1.csv")

C = capacitance
integrand <- data.frame(
  gain[2]/(1+(2*pi*C*vin1$x*short)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k1)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k10)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k20)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k35)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k48)^2),
  gain[2]/(1+(2*pi*C*vin1$x*k100)^2)
  )
area <- data.frame(
  G1 =0,
  G2 =0,
  G3 =0,
  G4 =0,
  G5 =0,
  G6 =0,
  G7 =0

  )
for(i in 1:length(integrand))
{
    for(l in 1:398)
    {
      if(is.na(integrand[l+1,i]))
      {
        break
      }
      else
      {
        area[i] <- area[i]+ riemanSum(integrand[l,i],integrand[l+1,i])
      }
    }
}
```

So this returns a gain value G for each resistor (called "area")

```
resistors<-c(0,1000,10000,20000,32500, 48700,100000)
plot(resistors,area, main= "Resistor value vs G", ylab = "Values of G", xlab = "Resistor Values (Ohm)")
```

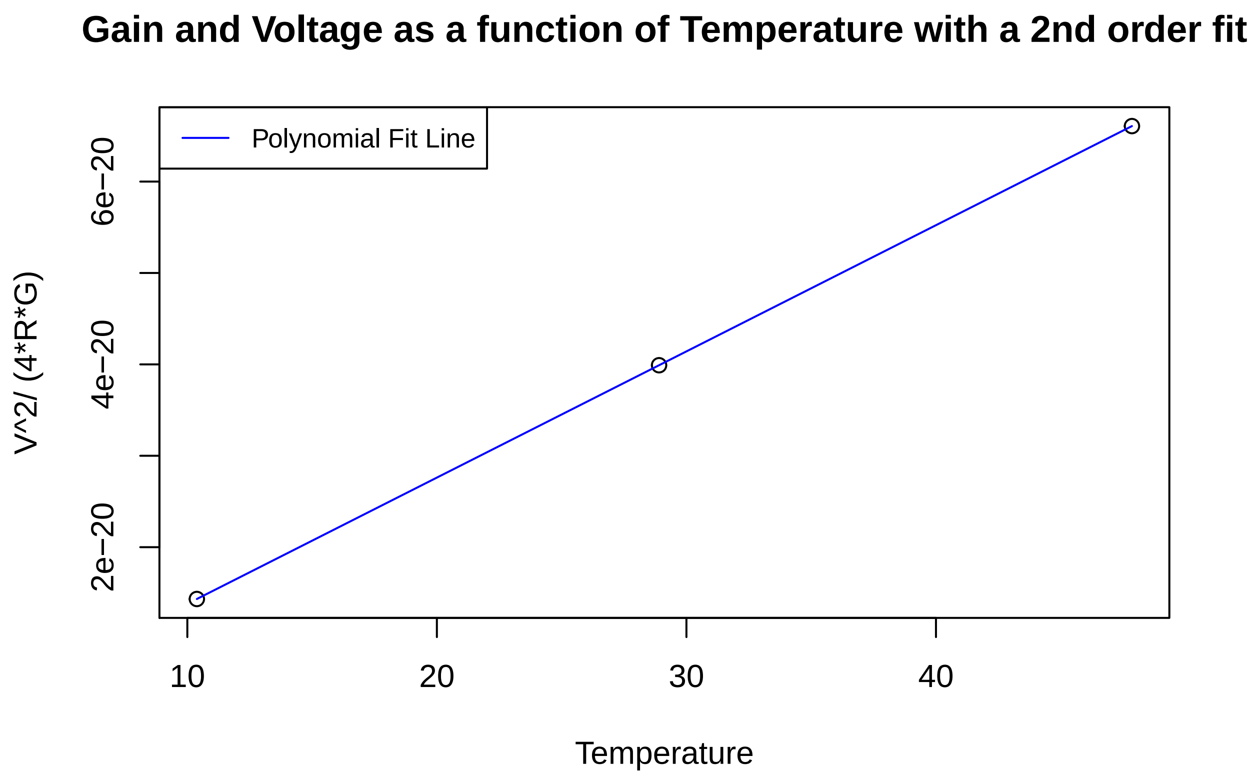


```
#prepare the data for graphing
kb<- 1.38064852 *10^-23 #m2 kg s-2 K-1

area2<-area[2:7] #take away the short's data
resistors2 <-resistors[2:7] #take away the short
```

```
y_value2<- (V^2)/(4*resistors2*area2)
y2<- unlist(y_value2, use.names=FALSE)


Temperature<- ((V^2)/(4*kb*area2*resistors2))/100
Temperature2<-unlist(Temperature, use.names = FALSE)

summary(Temperature2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   10.38   19.64   28.90   29.05   38.38   47.85       3
```

```
#2nd degree fit
fit <- lm(y2~Temperature2)

#plot same as above this time with a 2d fit line
plot(Temperature2,y2, main= "Gain and Voltage as a function of Temperature with a 2nd order fit", ylab=
lines(Temperature2, predict(fit, data.frame(Temperature2)), col="blue")
legend("topleft", legend=c("Polynomial Fit Line"),
       col=c("blue"), lty=1:2, cex=0.8)
```

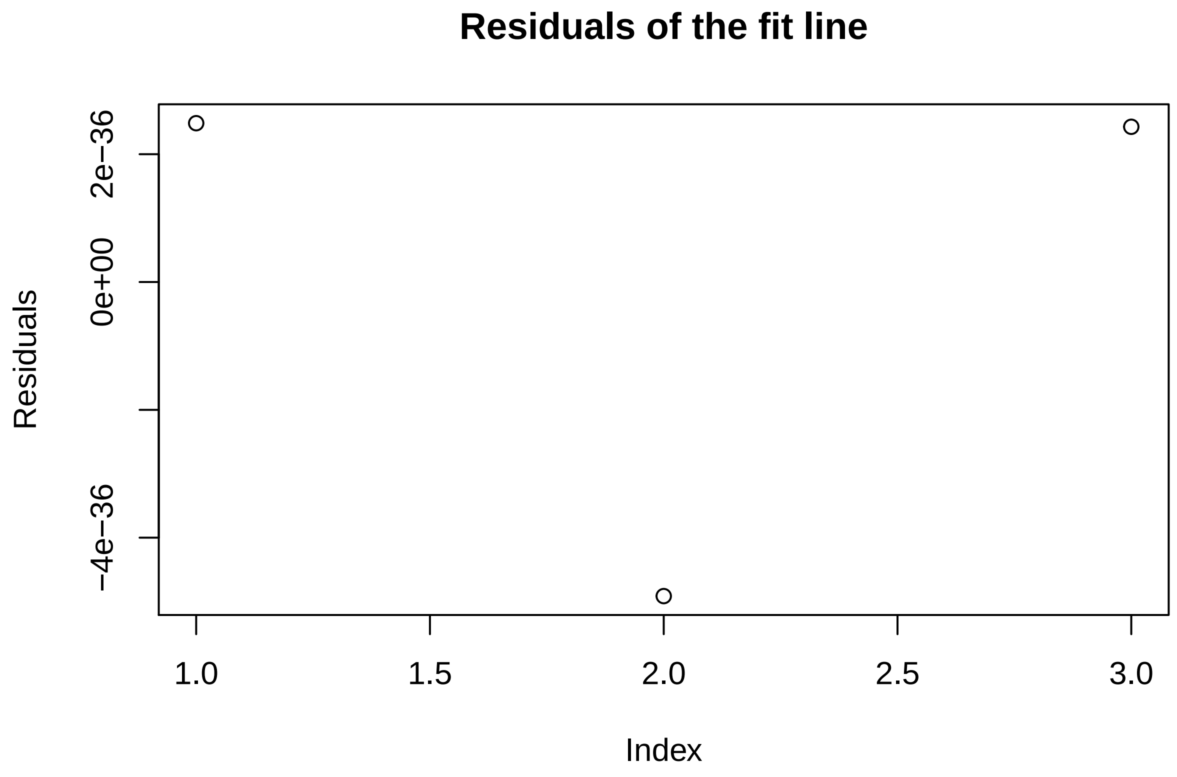


```
summary(fit)
```

```
## Warning in summary.lm(fit): essentially perfect fit: summary may be
## unreliable

##
## Call:
## lm(formula = y2 ~ Temperature2)
##
## Residuals:
```

```
##           4           5           6
##  2.485e-36 -4.914e-36  2.429e-36
##
## Coefficients:
##                Estimate Std. Error    t value Pr(>|t|)
## (Intercept)  -6.950e-36  7.456e-36 -9.320e-01    0.522
## Temperature2  1.381e-21  2.271e-37  6.079e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.019e-36 on 1 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 3.695e+31 on 1 and 1 DF,  p-value: < 2.2e-16
```

```r
plot(fit$residuals, main = "Residuals of the fit line", ylab= "Residuals")
```



Seems small for kb