Johnson Noise 128AL

Madeleine Allen, Edward Piper

1/31/2019

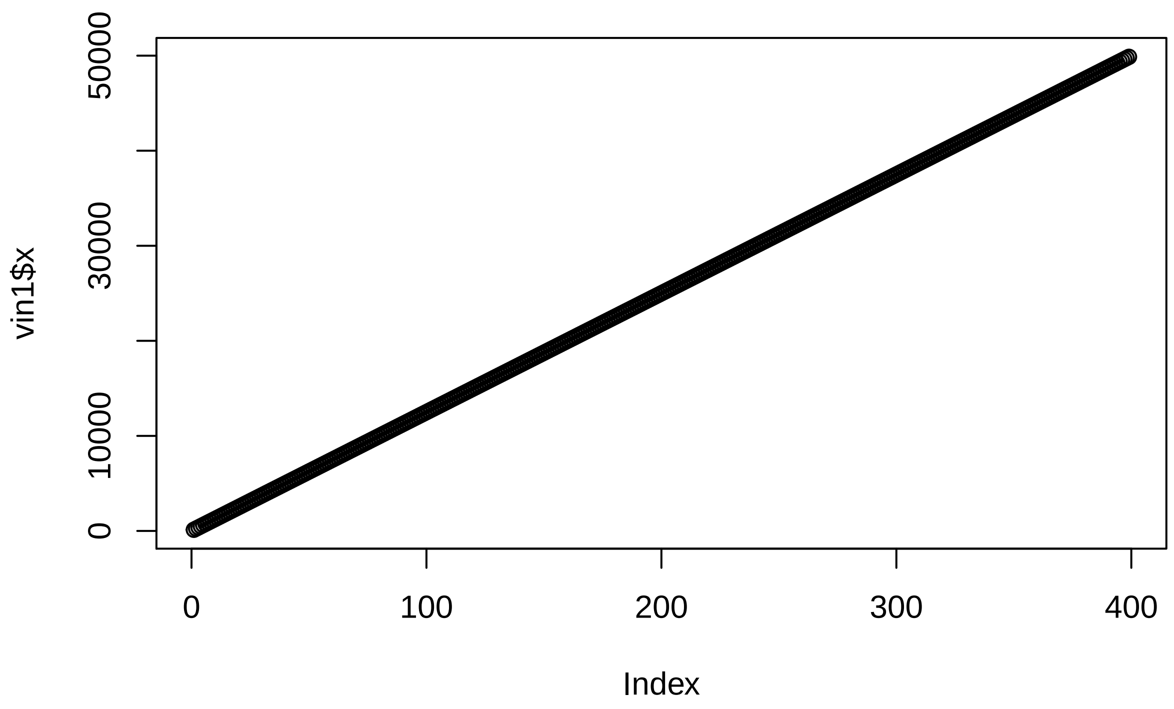
Experiment 1: Find g(f)

Analysis: Step 1: g(f)

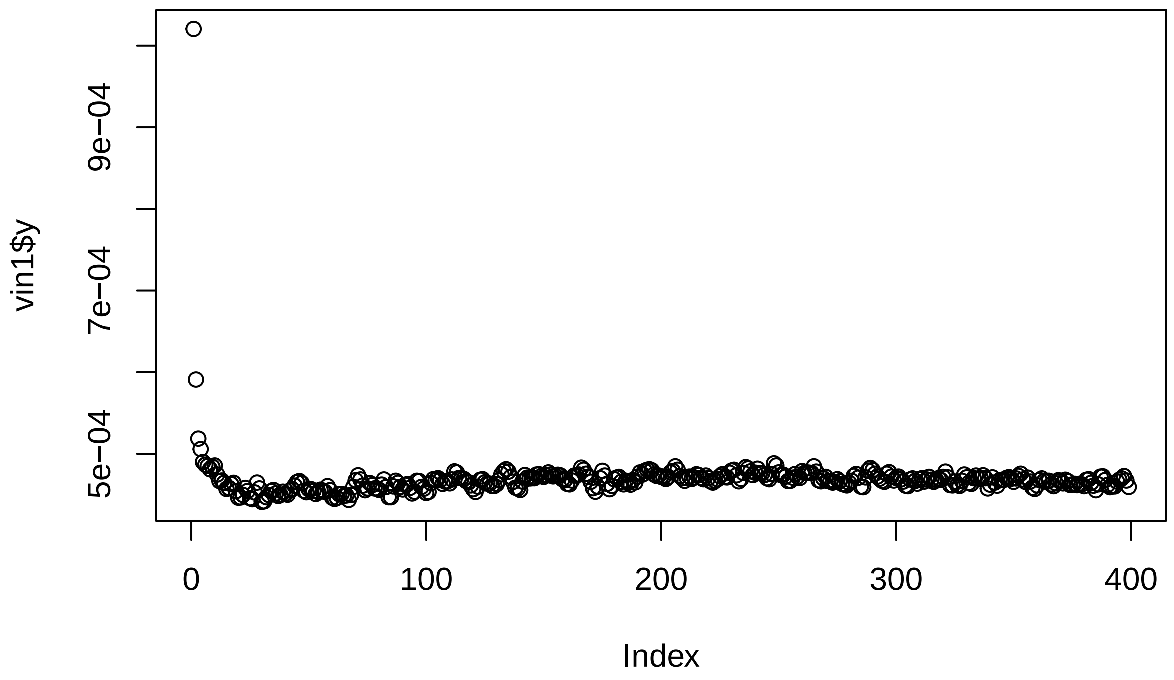
Make sure that the data is in the same folder as the R-script.

Plots to verify what the data looks like.

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

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

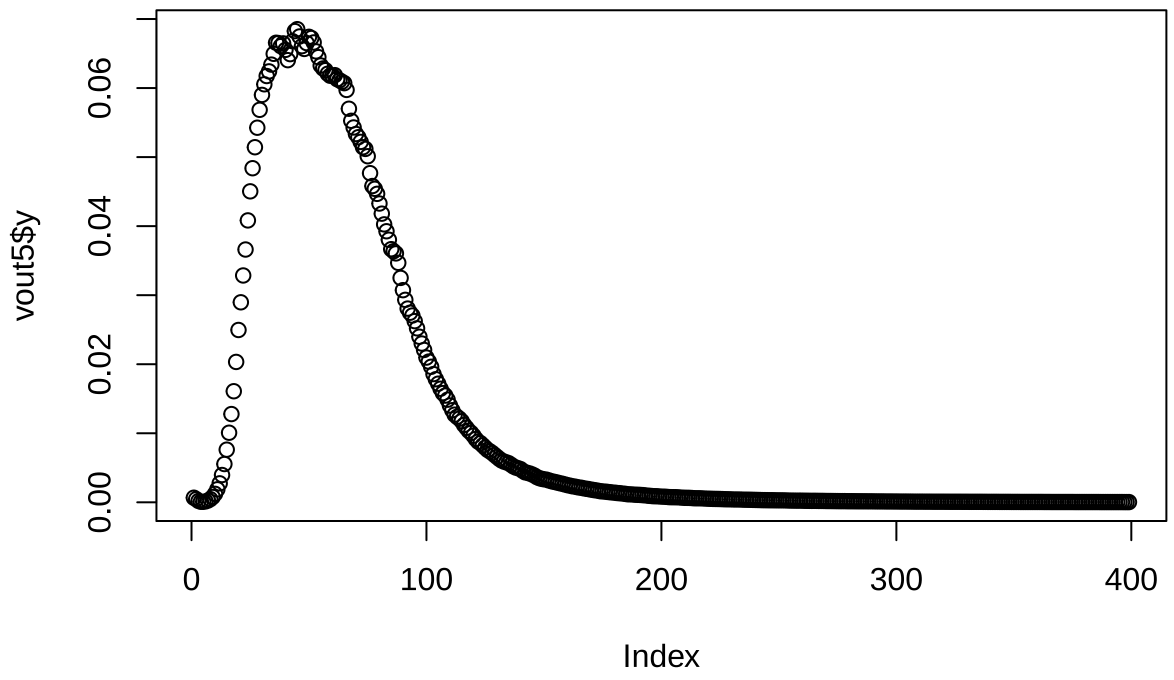


```
plot(vin1$y)
```



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

```
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):

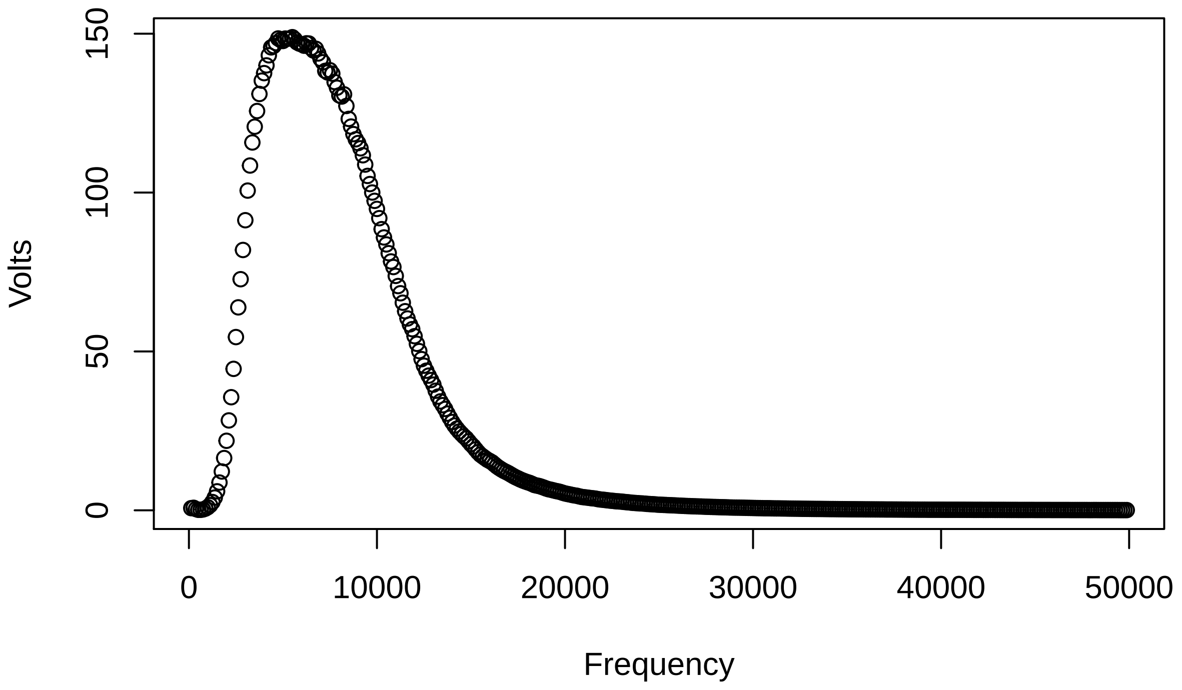
```
vin1_error<- sd(vin1$y, na.rm=TRUE)/sqrt(length(vin1$y[!is.na(vin1$y)]))
vin2_error<-sd(vin2$y, na.rm=TRUE)/sqrt(length(vin2$y[!is.na(vin2$y)]))
vin3_error<- sd(vin3$y, na.rm=TRUE)/sqrt(length(vin3$y[!is.na(vin3$y)]))
vin4_error<- sd(vin4$y, na.rm=TRUE)/sqrt(length(vin4$y[!is.na(vin4$y)]))
vin5_error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

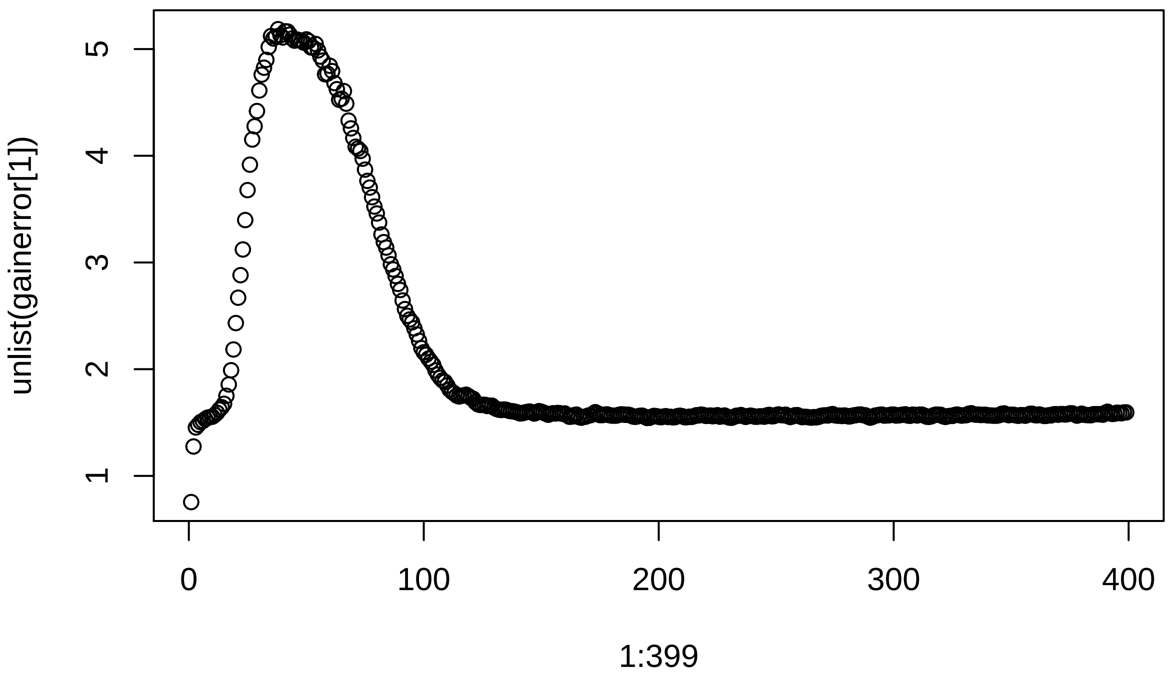


```
#this is consistent with the max gain of 150 calculated in class.
```

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
```

Import Band Voltage measurements from experiment 2

```
experiment2data<-read.csv("/Users/mallen/Documents/128AL/JohnsonNoise128AL/experiment2data1.csv")
```

Calculate Vmeas, V, and Vsystem

```
Vsys<- experiment2data[1,7] #first row 7th column
VsysError <- experiment2data[1,9]

Vmeask1<- (experiment2data[2,7])
Vmeask10<-experiment2data[3,7]
Vmeask20 <-experiment2data[4,7]
Vmeask32<-experiment2data[5,7]
Vmeask48<-experiment2data[6,7]
Vmeask100<-experiment2data[7,7]

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

#need to redo the error later (2/5)
VmeasError<-sqrt((sum(experiment2data[2:7,9])^2))

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

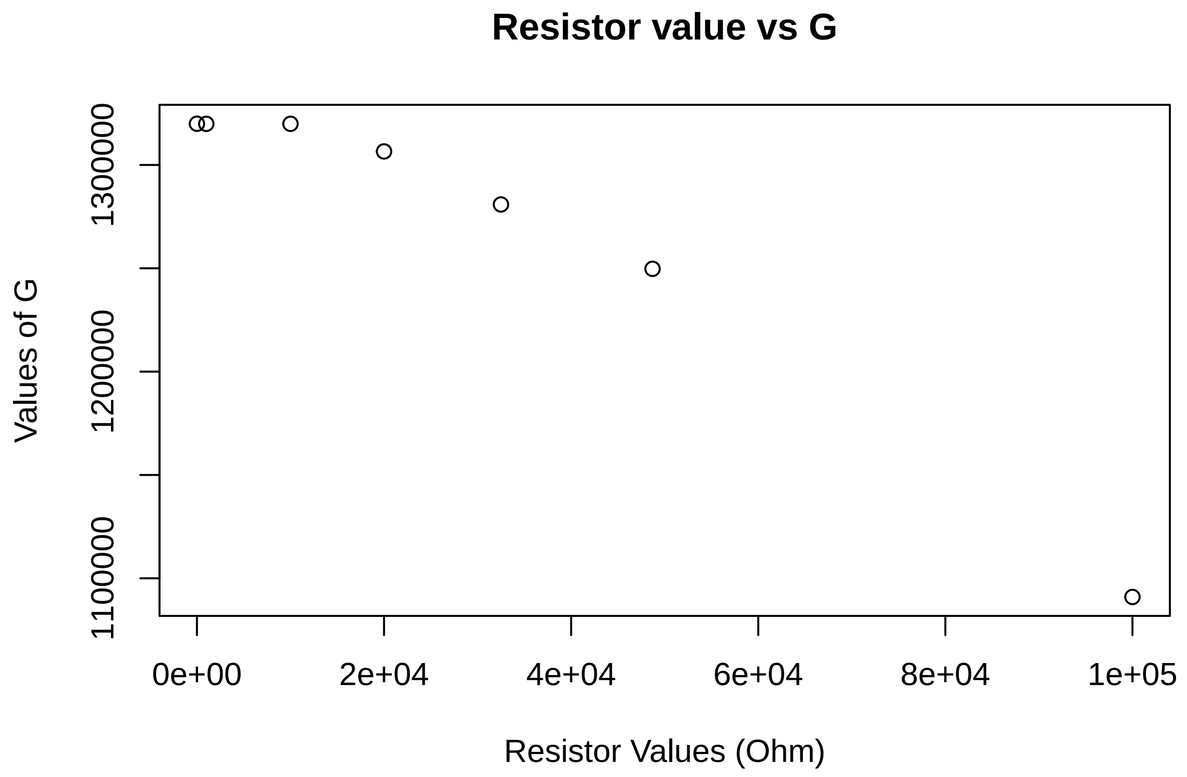
Calculating G

```
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("experiment2data1.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)")
```



##Plotting R as a function of V^2, kB, and G

IM VERY CONUFSED. I HAVE 4 GRAPHS HERE BECAUSE IM NOT SURE WHAT GOES ON THE Y AXIS AND WHAT KIND OF FIT LINE WE NEED SO I HAVE ALL 4 COMBOS (Y AND X SWITCHED AND EACH OF A LINEAR OR POLYNOMIAL FIT LINE)

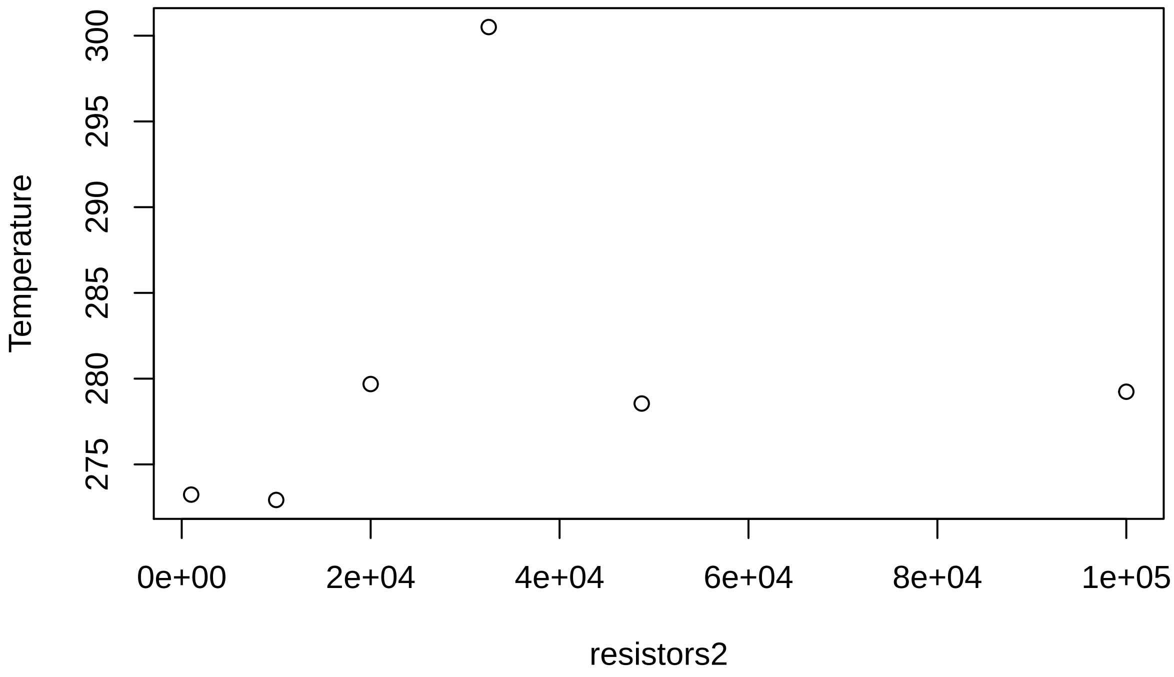
```
kb<- 1.38064852 *10^-23 #m2 kg s-2 K-1

area2<-area[2:7] #take away the short's data
y_value<- (V^2)/(4*kb*area2) #area is the vector that contains all G's
  #v^2

#prepare data for graphing
```

```r
resistors2 <-resistors[2:7] #take away the short
y<- unlist(y_value, use.names=FALSE)
```

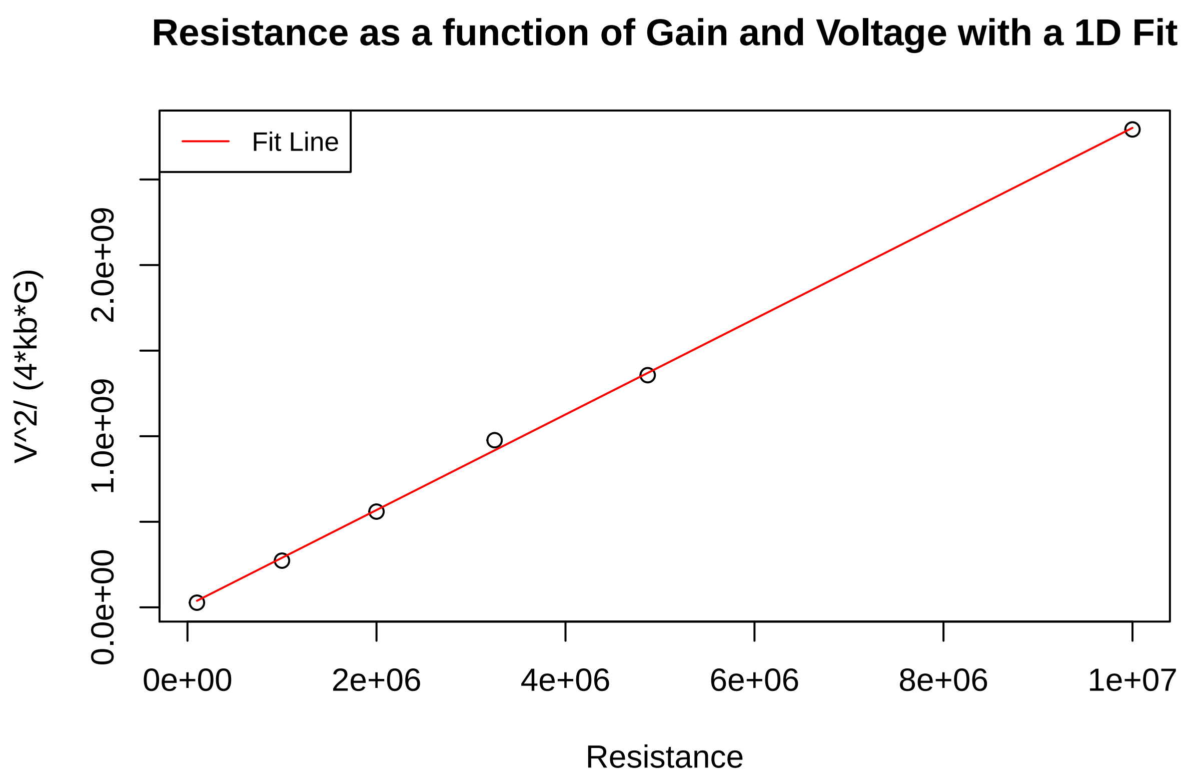
```r
#I'll try finding temperatures
Temperature<- ((V^2)/(4*kb*area2*resistors2))/100
summary(Temperature) #these are the correct values
```

```
##        G2              G3              G4              G5       
##  Min.   :273.2   Min.   :272.9   Min.   :279.7   Min.   :300.5  
##  1st Qu.:273.2   1st Qu.:272.9   1st Qu.:279.7   1st Qu.:300.5  
##  Median :273.2   Median :272.9   Median :279.7   Median :300.5  
##  Mean   :273.2   Mean   :272.9   Mean   :279.7   Mean   :300.5  
##  3rd Qu.:273.2   3rd Qu.:272.9   3rd Qu.:279.7   3rd Qu.:300.5  
##  Max.   :273.2   Max.   :272.9   Max.   :279.7   Max.   :300.5  
##        G6              G7       
##  Min.   :278.5   Min.   :279.2  
##  1st Qu.:278.5   1st Qu.:279.2  
##  Median :278.5   Median :279.2  
##  Mean   :278.5   Mean   :279.2  
##  3rd Qu.:278.5   3rd Qu.:279.2  
##  Max.   :278.5   Max.   :279.2
```

```r
plot(resistors2, Temperature)
```



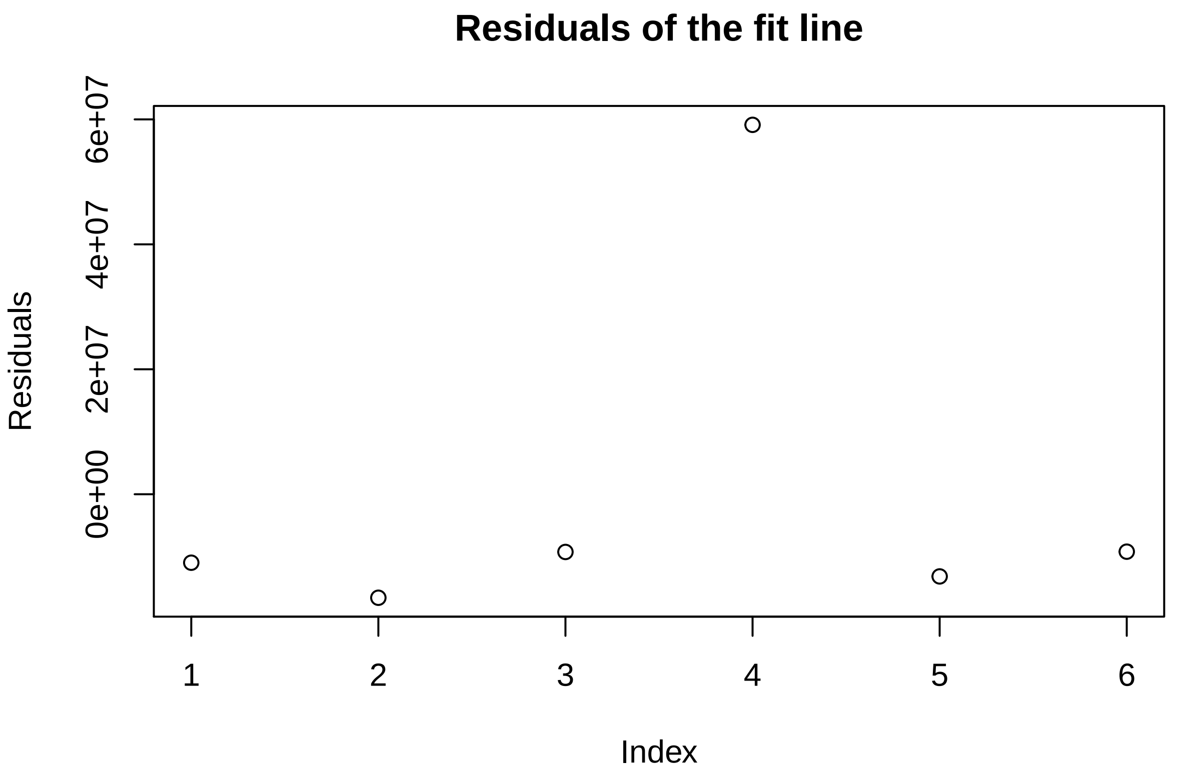
```r
#
#resistor as the x axis and the other term as the y axis.
#
resistors3<-resistors2*100
fit  <- lm(y~resistors3)
plot(resistors3,y_value, main= "Resistance as a function of Gain and Voltage with a 1D Fit", xlab= "Res
lines(resistors3, predict(fit, data.frame(resistors3)), col="red")
legend("topleft", legend=c("Fit Line"),
       col=c("red"), lty=1:2, cex=0.8)
```



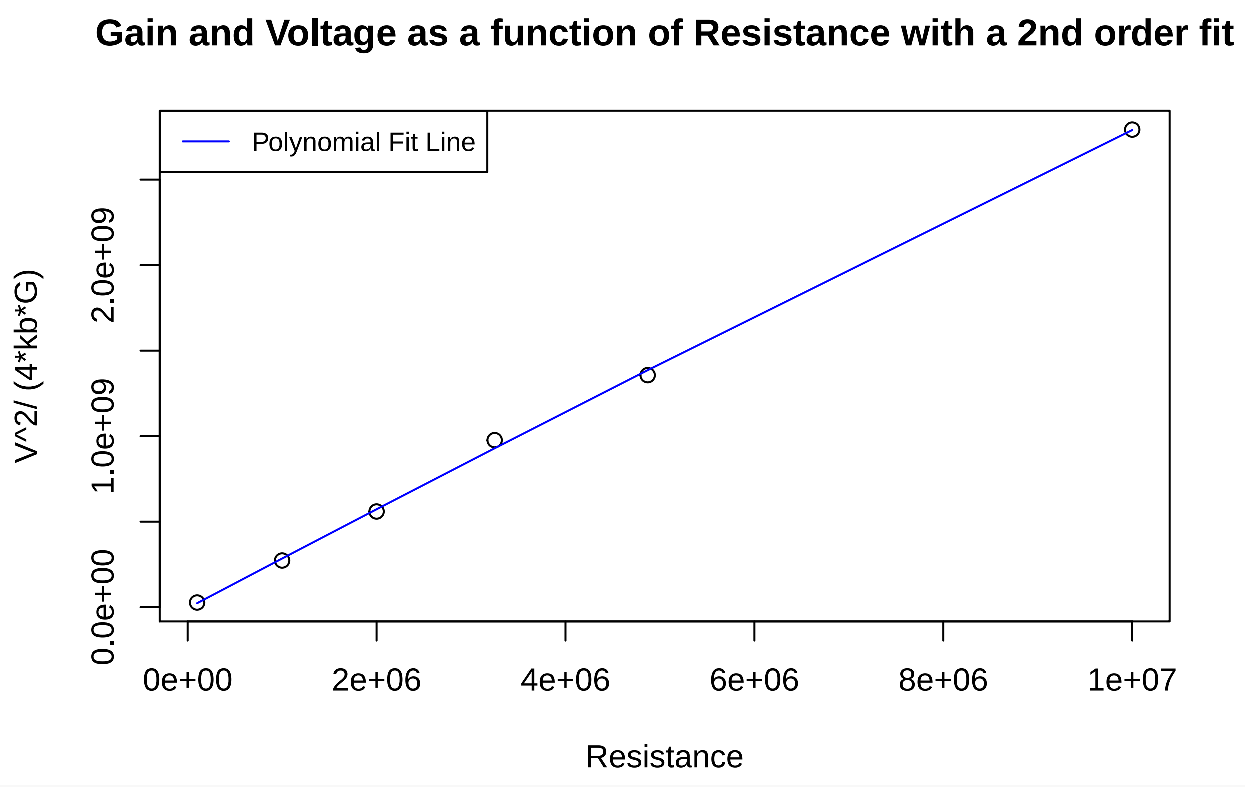
```r
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ resistors3)
##
## Residuals:
##         1         2         3         4         5         6
## -10963011 -16569202  -9243729  59124250 -13154432  -9193877
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.037e+07  1.956e+07   0.531    0.624
## resistors3  2.791e+02  4.059e+00  68.767 2.68e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32530000 on 4 degrees of freedom
## Multiple R-squared:  0.9992, Adjusted R-squared:  0.9989
## F-statistic:  4729 on 1 and 4 DF,  p-value: 2.679e-07
```

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



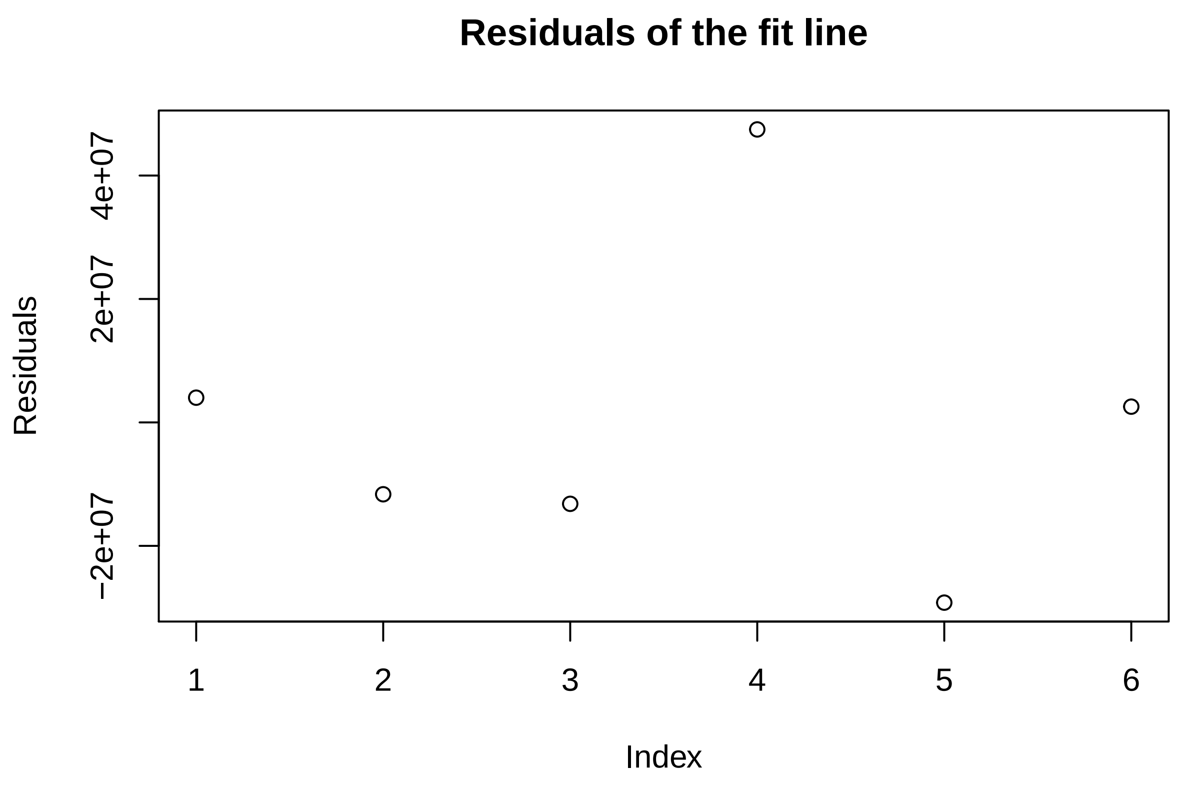
```
#2nd degree fit
fit2 <- lm(y~poly(resistors3,2,raw=TRUE))

#plot same as above this time with a 2d fit line
plot(resistors3,y_value, main= "Gain and Voltage as a function of Resistance with a 2nd order fit", ylal
lines(resistors3, predict(fit2, data.frame(resistors3)), col="blue")
legend("topleft", legend=c("Polynomial Fit Line"),
       col=c("blue"), lty=1:2, cex=0.8)
```



```r
summary(fit2)
```

```
##
## Call:
## lm(formula = y ~ poly(resistors3, 2, raw = TRUE))
##
## Residuals:
##         1         2         3         4         5         6
##   4009219 -11642891 -13191842  47469522 -29198790   2554781
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -5.834e+06  2.816e+07  -0.207 0.849115
## poly(resistors3, 2, raw = TRUE)1  2.916e+02  1.557e+01  18.725 0.000332
## poly(resistors3, 2, raw = TRUE)2 -1.204e-06  1.445e-06  -0.833 0.465938
##
## (Intercept)
## poly(resistors3, 2, raw = TRUE)1 ***
## poly(resistors3, 2, raw = TRUE)2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33850000 on 3 degrees of freedom
## Multiple R-squared:  0.9993, Adjusted R-squared:  0.9989
## F-statistic:  2184 on 2 and 3 DF,  p-value: 1.798e-05
```

```r
plot(fit2$residuals, main = "Residuals of the fit line", ylab= "Residuals")
```



So I have fit a 1st order polynomial line and a 2nd order polynomial line. Both give reasonable predictions, both have reasonably random residuals. If I didn't know better, I'd pick the 1D line. But Nate told me to look for 2nd order relationships so I will check to see if the data is normally distributed so I can pick based on their p-values.

```
shapiro.test(fit$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit$residuals
## W = 0.58894, p-value = 0.0003366
```

```
shapiro.test(fit2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit2$residuals
## W = 0.89195, p-value = 0.3285
```

The 2nd order fit is better. The Shapiro-Wilk normality test tells us the assumptions of normality and homogeniety of variance were met, as there weren't any obvious abnormalities on the residual plots and the Shapiro-Wik p-value for residuals was p=.147 (fail to reject H0 that data is normal for the 2D line but not for the 1D line).

So because of this we will go with the 2nd order polynomial fit line.

@EDWARD HELLO

I NEED TO GET THE VALUE OF ABSOLUTE 0 OUT OF THIS BUT I GOT THE TEMPERATURES OF EACH RESISTOR AND THEY HONESTLY MAKE SENSE.

I WILL NEED TO FIGURE OUT HOW TO GET ABSOLUTE 0, THEN ILL DO THE ANALYSIS FOR THE NEXT PART.

ON THURSDAY WE CAN TAKE TURNS WORKING ON THE REPORT AND DOING DATA ANALYSIS FOR THE 2ND DATA SET WE TOOK