Johnson Noise 128AL

Madeleine Allen, Edward Piper

1/31/2019

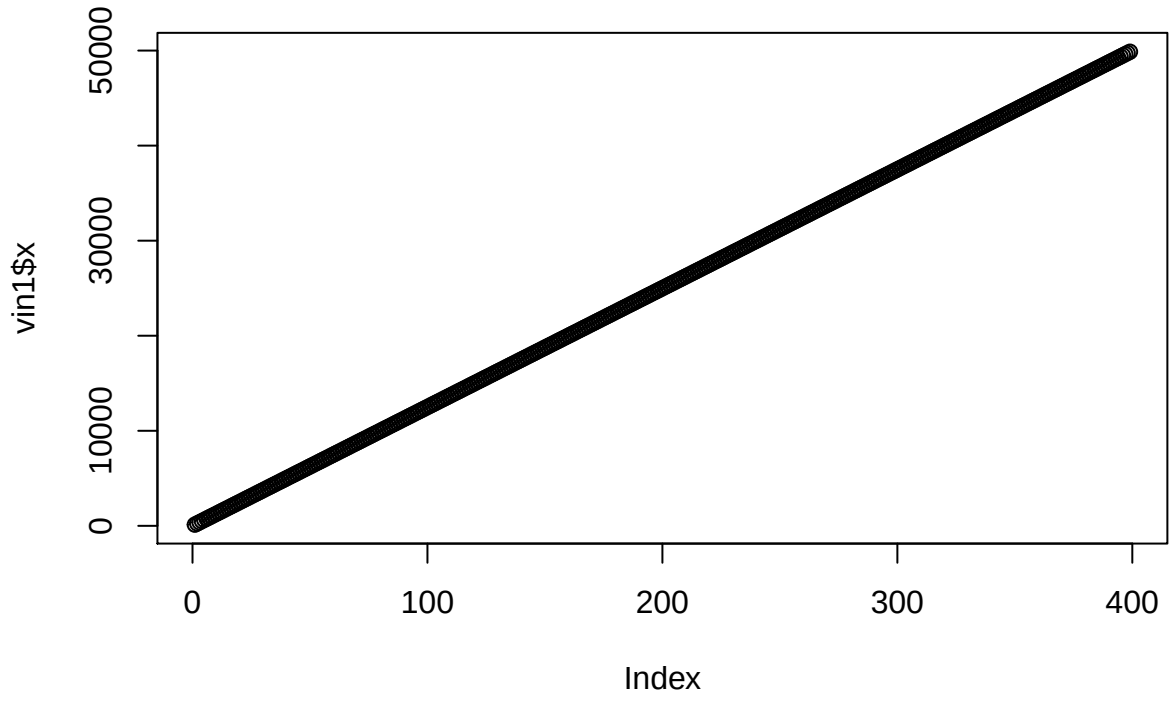
Analysis: Step 1: g(f)

Make sure that the data is in the same folder as the R-script (should be automatic if you clone the git repo) (btw I hate myself for saying "clone the git repo")

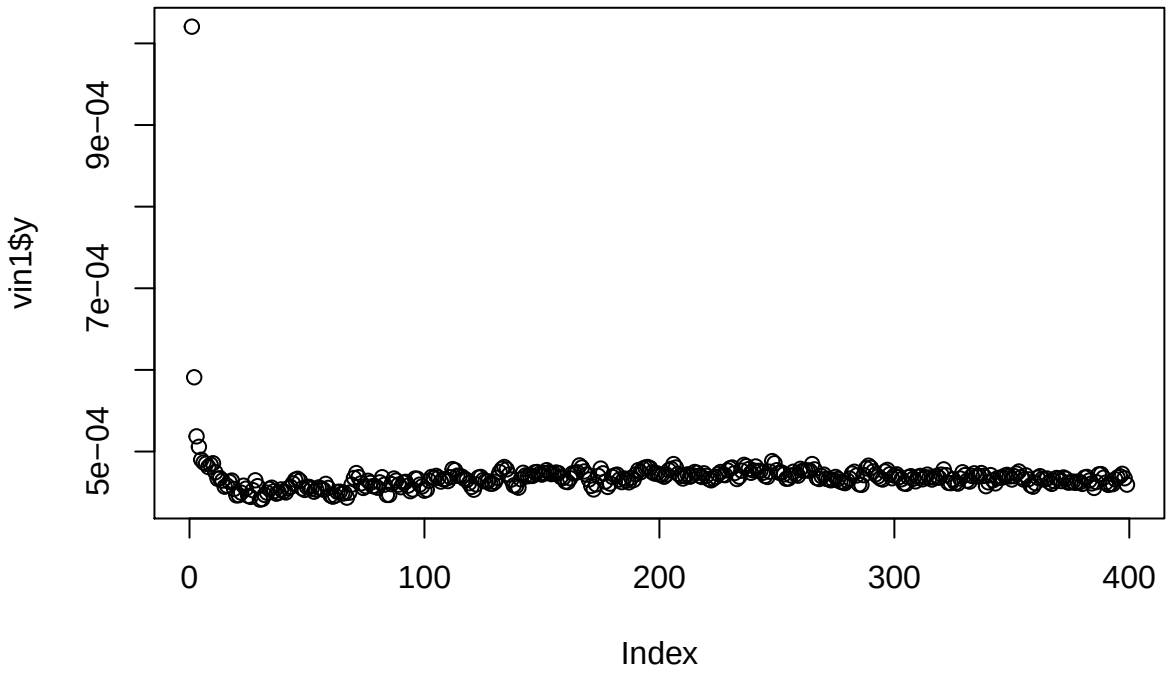
Plots to verify what the data looks like.

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

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




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

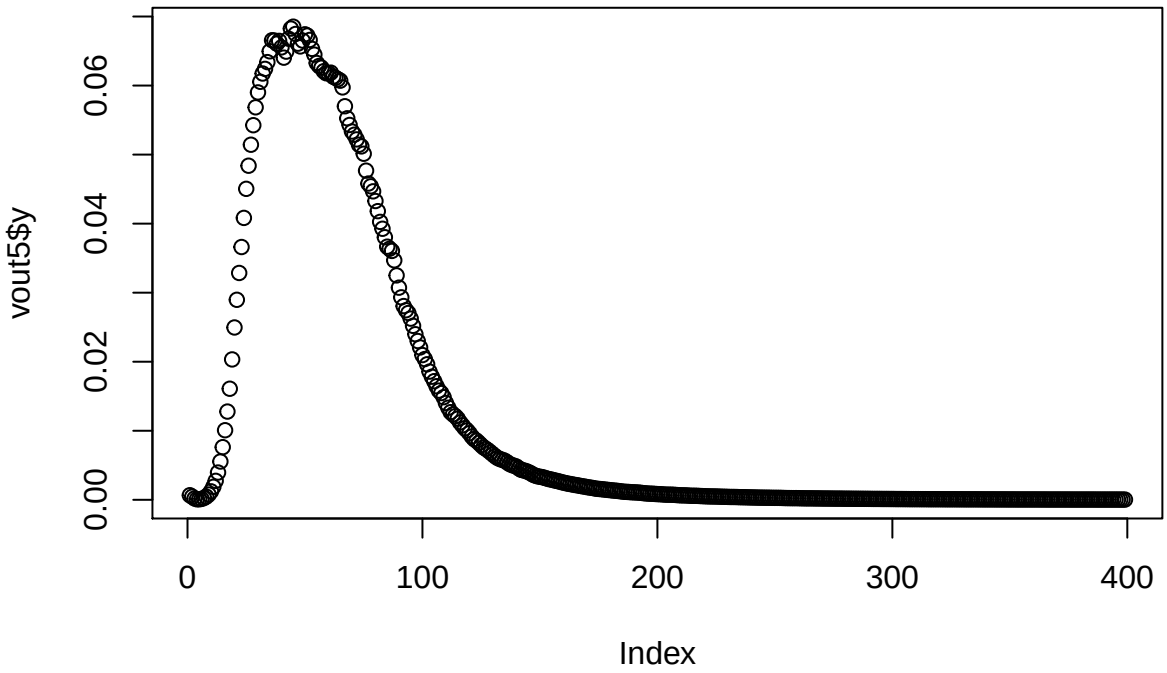


Now I'll upload the rest of the data. I'll plot an out graph for reference too.

```
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)
```



Find the mean of each value: It should be noted that the vin voltages are flat whereas the vout voltages are peaked, so only the peak region is selected (thus the ranges present in the y values for the vout values)

```
m_vin1<-mean(vin1$y)
m_vin2<-mean(vin2$y)
m_vin3<-mean(vin3$y)
m_vin4<-mean(vin4$y)
m_vin5<-mean(vin5$y)

m_in<- mean(m_vin1, m_vin2, m_vin3, m_vin4, m_vin5)

m_vout1<-mean(vout1$y[35:44])
m_vout2<-mean(vout2$y[37:54])
m_vout3<-mean(vout3$y[35:56])
m_vout4<-mean(vout4$y[35:53])
m_vout5<-mean(vout5$y[35:52])

m_out<- mean(m_vout1, m_vout2, m_vout3, m_vout4, m_vout5)
```

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

```
gF<- m_out/m_in
print(gF)
```

```
## [1] 142.857
```

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)]))
```

Now we have errors for every measurement. The averages used to calculate gain need errors too.

```
m_vin_error <- sqrt(vin1_error^2+vin2_error^2 + vin3_error^2+ vin4_error^2+vin5_error^2)
m_vout_error <- sqrt(vout1_error^2+vout2_error^2 + vout3_error^2+ vout4_error^2+vout5_error^2)
```

Now we can find the error in the gain function from the error in Vin and Vout

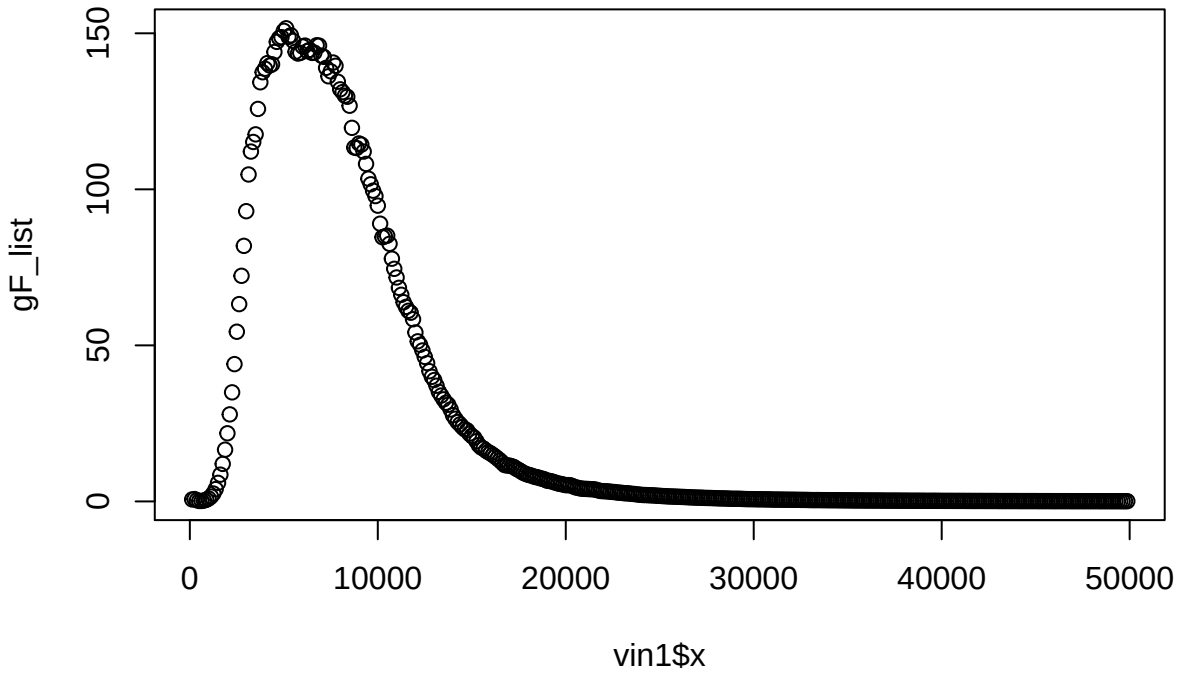
```
gF_error<-sqrt(m_vin_error^2 +m_vout_error^2)
print(gF_error)
```

```
## [1] 0.0001223323
```

```
#seems a little small. not sure if I did it right then?
```

Graph g(f) against frequency

```
gF_list= vector()
for (i in 1:length(vin1$x))
{
  mean_vin = mean(vin1$y[i], vin2$y[i], vin3$y[i], vin4$y[i], vin5$y[i])
  mean_vout= mean(vout1$y[i], vout2$y[i], vout3$y[i], vout4$y[i], vout5$y[i])
  gF_list[i]=mean_vout/mean_vin
}
plot(vin1$x, gF_list) #is this what they mean by plot? Is this what it should look like?
```

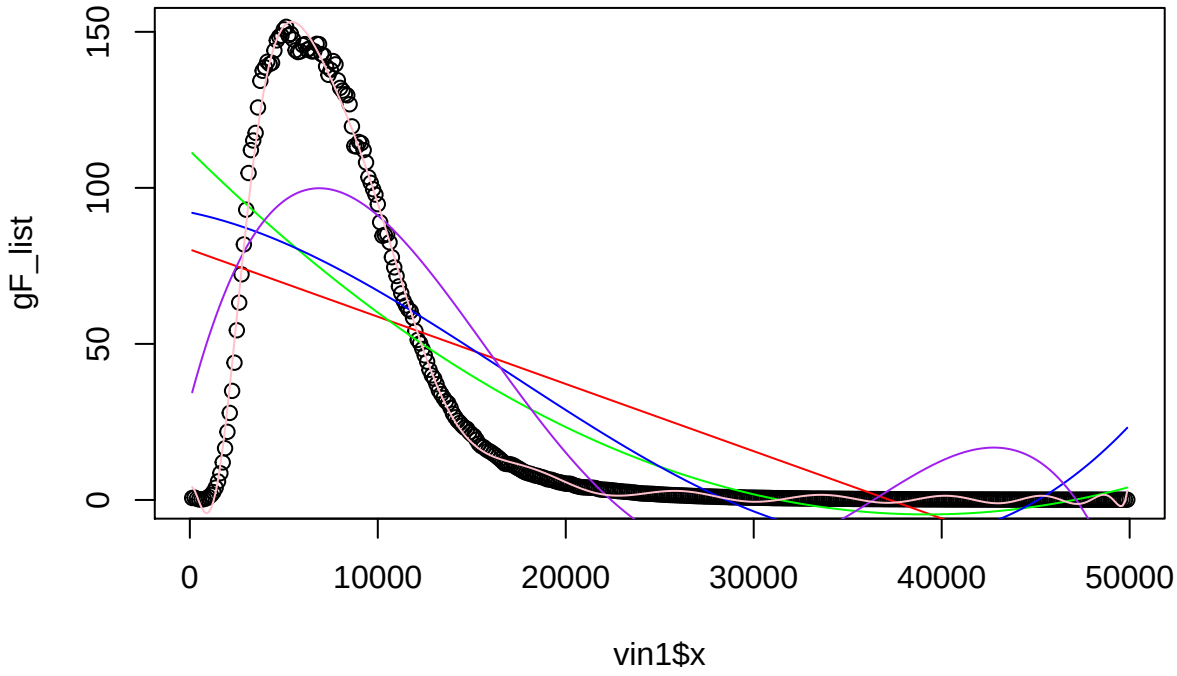


Apply a fit to g(f). (Pretty sure this is wrong but that's just showbiz baby)

```
#fit first degree polynomial equation:
fit  <- lm(gF_list~vin1$x)
#second degree
fit2 <- lm(gF_list~poly(vin1$x,2,raw=TRUE))
```

```
#third degree
fit3 <- lm(gF_list~poly(vin1$x,3,raw=TRUE))
#fourth degree
fit4 <- lm(gF_list~poly(vin1$x,4,raw=TRUE))
fit10 <-lm(gF_list~poly(vin1$x, 30, raw=TRUE))
#generate range of 50 numbers starting from 30 and ending at 160
#xx <- seq(30,160, length=50)
plot(vin1$x,gF_list,pch=1) #1 through 24 (i think) are all different cool shapes!
lines(vin1$x, predict(fit, data.frame(vin1$x)), col="red")
lines(vin1$x, predict(fit2, data.frame(vin1$x)), col="green")
lines(vin1$x, predict(fit3, data.frame(vin1$x)), col="blue")
lines(vin1$x, predict(fit4, data.frame(vin1$x)), col="purple")
lines(vin1$x, predict(fit10, data.frame(vin1$x)), col = "pink")
```

```
## Warning in predict.lm(fit10, data.frame(vin1$x)): prediction from a rank-
## deficient fit may be misleading
```

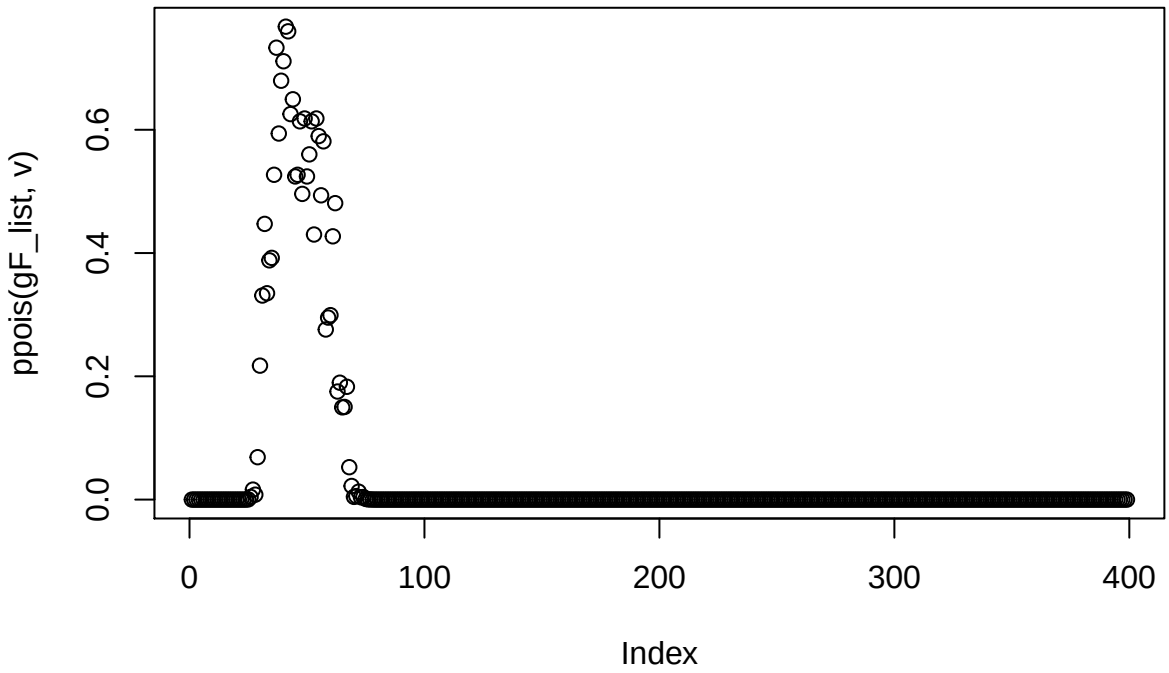



```
#this is nice but it also sucks so im gonna try something that may suck less
```

```
#added the library "fitdistrplus"
#fit.gamma <- fitdist(gF_list, distr = "gamma", method = "mle")
#summary(fit.gamma)
#plot(fit.gamma)

#curve(dweibull(x, scale=2.5, shape=1.5),from=0, to=15, main="Weibull distribution")
#looks similar but I'm confused

#gamma anad weibull both didnt work i will try poisson
v<-c(m_vout1/m_vin1, m_vout2/m_vin2, m_vout3/m_vin3, m_vout4/m_vin4, m_vout5/m_vin5)
#this is the only one that looks ~vaguely~ like a fit
plot(ppois(gF_list,v))
```

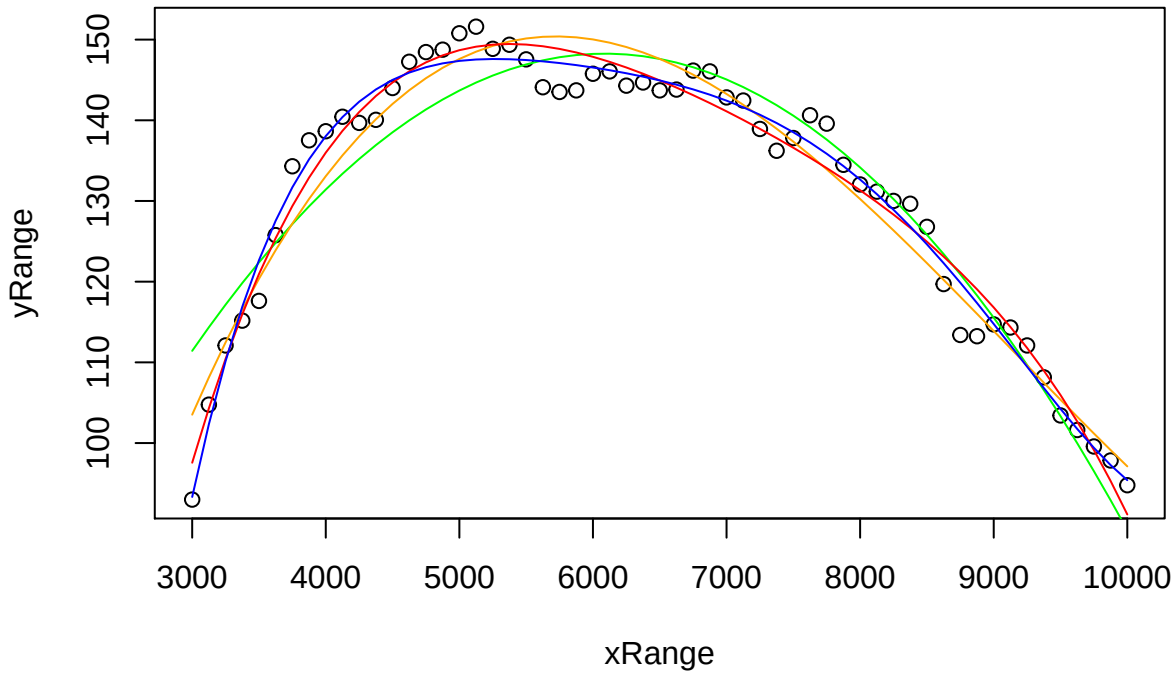



```
stupidFit<-ppois(gF_list, v)
#this could be useful but im too tired to search:
#https://stat.ethz.ch/R-manual/R-devel/library/stats/html/glm.html
```

```
#maybe we only need 3-10 kHz range???
xRange <- vin1$x[24:80] #place between 3 and 10 kHz
yRange <- gF_list[24:80]

fit2 <- lm(yRange~poly(xRange,2,raw=TRUE))
fit3 <- lm(yRange~poly(xRange,3,raw=TRUE))
fit4 <- lm(yRange~poly(xRange,4,raw=TRUE))
fit5 <- lm(yRange~poly(xRange,5,raw=TRUE))

plot(xRange, yRange) #looks like it could be reasonably well fit with a 2nd or 3rd degree polynomial
lines(xRange, predict(fit2, data.frame(xRange)), col="green")
lines(xRange, predict(fit3, data.frame(xRange)), col="orange")
lines(xRange, predict(fit4, data.frame(xRange)), col="red")
lines(xRange, predict(fit5, data.frame(xRange)), col="blue")
```



Analysis Step 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
```

Capacitance for the circuit.

```
capacitance <-87.875
capacitanceError <-.594
```

Import measurements from experiment 2

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

#View(experiment2data)
```

Calculate Vmeas

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

VsysError <- experiment2data[1,9]

Vmeas<- mean(experiment2data[2:7,7])
VmeasError<-sqrt((sum(experiment2data[2:7,9])^2))

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

Calculate a value for G

```
#definitely confused about this
#G= integrate((gF_list)^2/1+(2*pi*), lower=3,upper = 10)
#integrate()

#G1k=integrate(((stupidFit^2)/(1+2*pi*capacitance*k1*f)^2), lower=3, upper=10)

#http://spa-mxpweb.spa.umn.edu/resources/ExpWriteups/LabManNoise.pdf
```