Project 1
PHGN 311 - Mathematical Physics
Galen Vincent

Get the data

Data was collected with a sound detector program run through a Raspberry Pi 3. The system was designed in the Physics Python course last semester by myself and a classmate (Caroline Ellis). The data collected for this project is volume level. Data points were taken once per second, for 100 seconds, in the middle of 50 songs. These 50 songs were all by the artist Logic.

Import Data - Make sure to have the file “DATA.txt” in the same directory as the notebook when running this part of the program.

In[192]:=
```
SetDirectory[NotebookDirectory[]]
```
Out[192]=
```
C:\Users\galen\Desktop\Fall 2017\Math Physics\Project 1
```

In[193]:=
```
data = Import["DATA.txt", "CSV"];
```

In[194]:=
```
Dimensions[data]
```
Out[194]=
```
{50, 100}
```

The dimensions of the table are correct with the data I have, so import was successful.

Analysis Part 1: Gross Features of the Data - Means and Variances

---

a.

Calculate the mean of each of the 50 sets of data. I store them in a list called “μ”.

In[195]:=
```
μ = Table[Mean[data[[n]]], {n, 1, 50}]
```

Out[195]=
```
{0.205762, 0.201885, 0.166882, 0.147926, 0.202237, 0.168291, 0.151049, 0.227297, 0.191426,
 0.165664, 0.188311, 0.147803, 0.184541, 0.172672, 0.209531, 0.129544, 0.176519,
 0.141036, 0.145966, 0.170284, 0.184511, 0.197656, 0.174713, 0.145715, 0.156037,
 0.167695, 0.184022, 0.190967, 0.16793, 0.185313, 0.172949, 0.164521, 0.157202, 0.17292,
 0.160019, 0.170698, 0.163516, 0.162392, 0.166377, 0.143371, 0.145254, 0.177998,
 0.161338, 0.165137, 0.131034, 0.181084, 0.156113, 0.186884, 0.168428, 0.160655}
```

Calculate the variance of each of the 50 sets of data. I store them in a list called “V”.

In[196]:=
```
V = Table[Sum[(data[[n, i]] - μ[[n]])^2 / Length[data[[n]]], {i, 1, Length[data[[n]]]}], {n, 1, 50}]
```

Out[196]=
```
{0.00458188, 0.00877966, 0.00549797, 0.0033283, 0.0038439, 0.00264144,
 0.00829703, 0.0112005, 0.00378329, 0.00462479, 0.00407571, 0.00610895,
 0.00339637, 0.00322478, 0.00706262, 0.0042757, 0.0049373, 0.00160903,
 0.00642848, 0.00358315, 0.00852065, 0.00548753, 0.00539064, 0.00248982,
 0.00388902, 0.00469888, 0.0031516, 0.00352949, 0.00239611, 0.00413036, 0.0029248,
 0.00291534, 0.0038157, 0.00410603, 0.00493898, 0.0057622, 0.003245, 0.00805677,
 0.00526319, 0.00344733, 0.00391009, 0.00428836, 0.00281957, 0.00272432,
 0.00382396, 0.00462547, 0.00297094, 0.00426426, 0.00587365, 0.00348925}
```

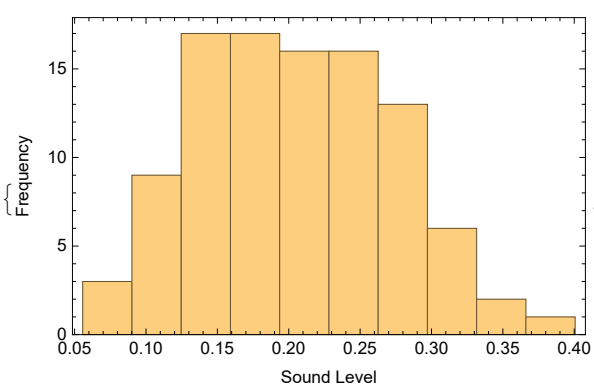
---

b.

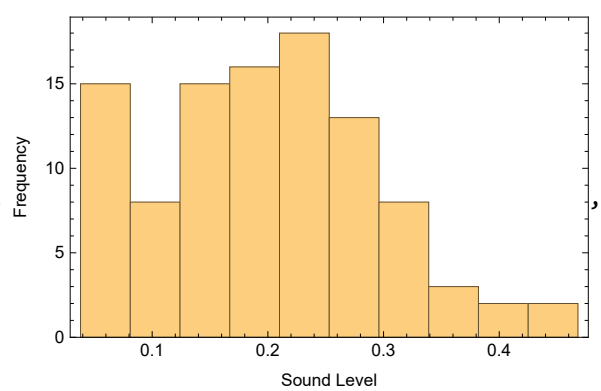
Plot each of the 50 data sets separately.

In[197]:=
```
hist1 = Table[Histogram[data[[n]],
   {Min[data[[n]]], Max[data[[n]]] + .01, (Max[data[[n]]] - Min[data[[n]]] + .01)/10},
   Frame → True, FrameLabel → {"Sound Level", "Frequency"}, ImageSize → 300], {n, 1, 50}]
hist1 = ArrayReshape[hist1, {25, 2}];
```

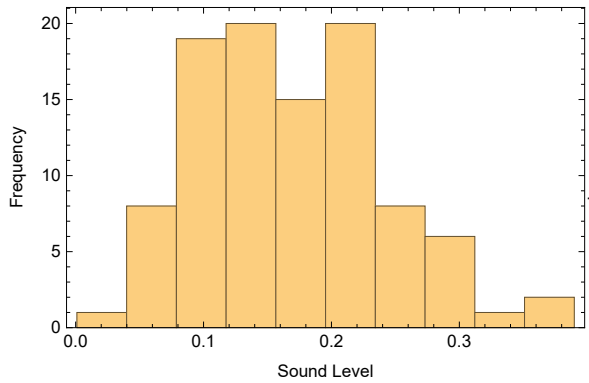
Out[197]=



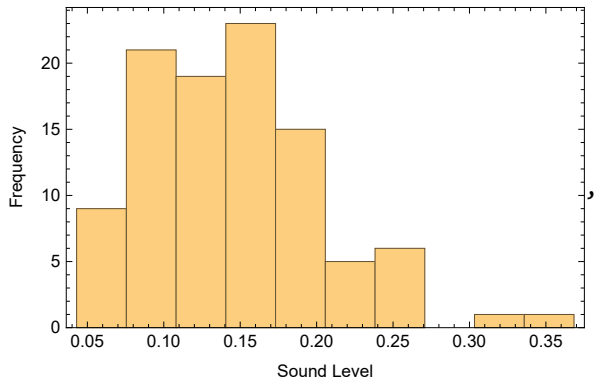
,



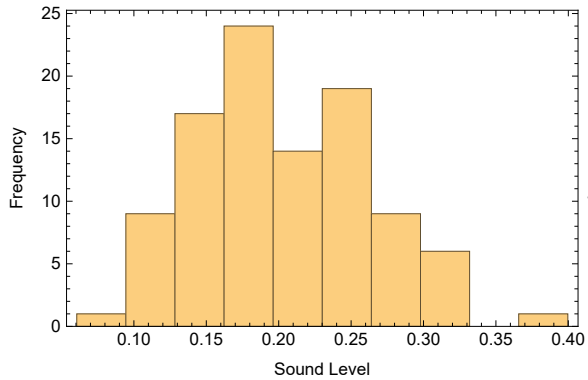
,


Frequency
0
5
10
15
20
0.0
0.1
0.2
0.3
Sound Level

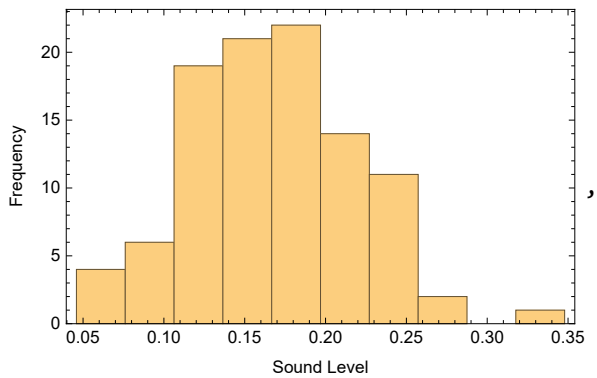
,


Frequency
0
5
10
15
20
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

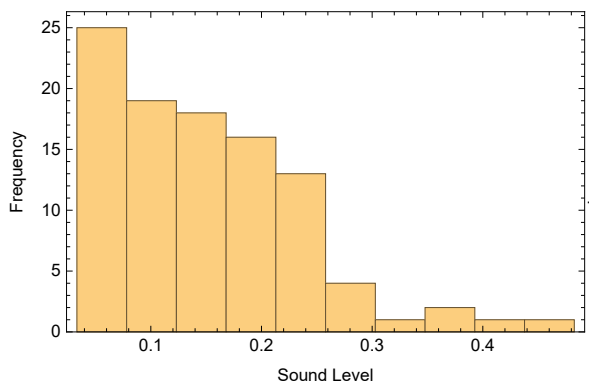
,


Frequency
0
5
10
15
20
25
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Sound Level

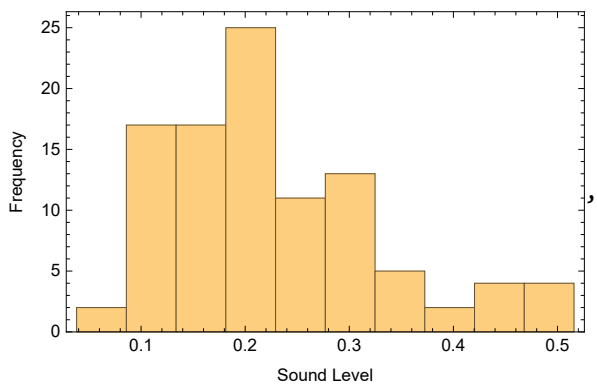
,


Frequency
0
5
10
15
20
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

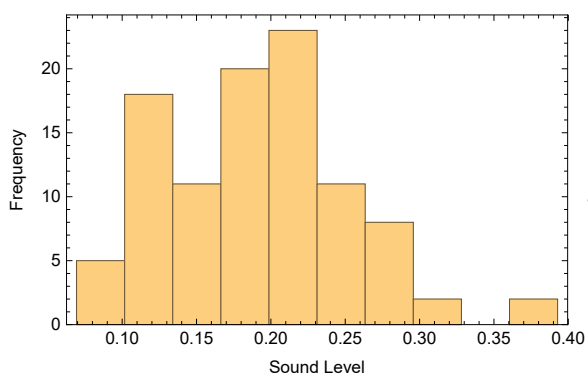
,


Frequency
0
5
10
15
20
25
0.1
0.2
0.3
0.4
Sound Level

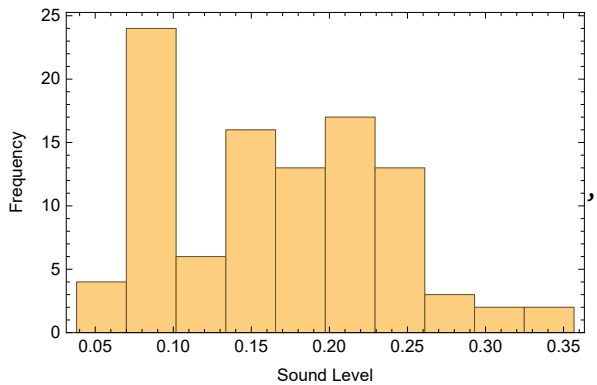
,


Frequency
0
5
10
15
20
25
0.1
0.2
0.3
0.4
0.5
Sound Level

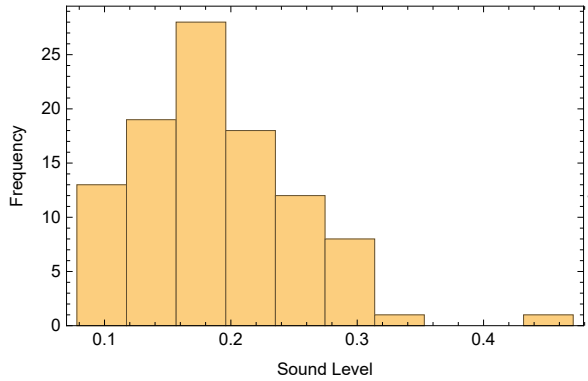
,


Frequency
0
5
10
15
20
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Sound Level

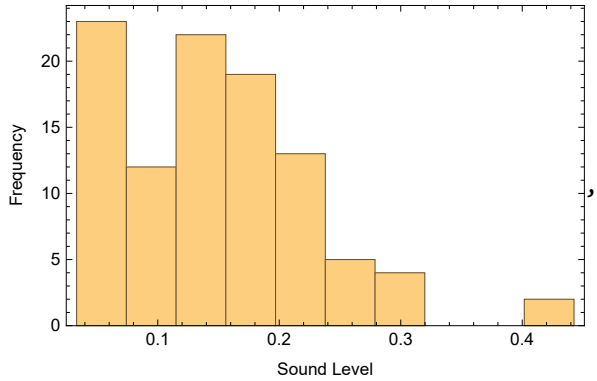
,


Frequency
0
5
10
15
20
25
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

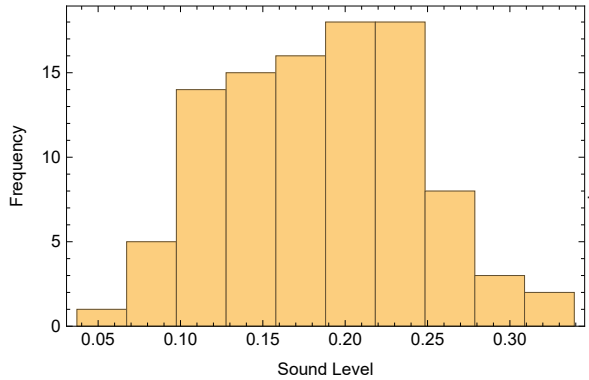
,


Frequency
0
5
10
15
20
25
0.1
0.2
0.3
0.4
Sound Level

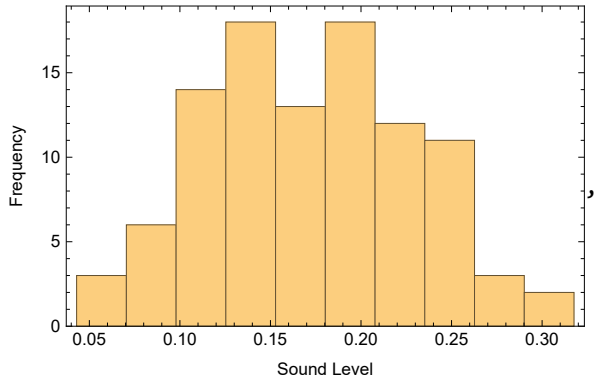
,


Frequency
0
5
10
15
20
0.1
0.2
0.3
0.4
Sound Level

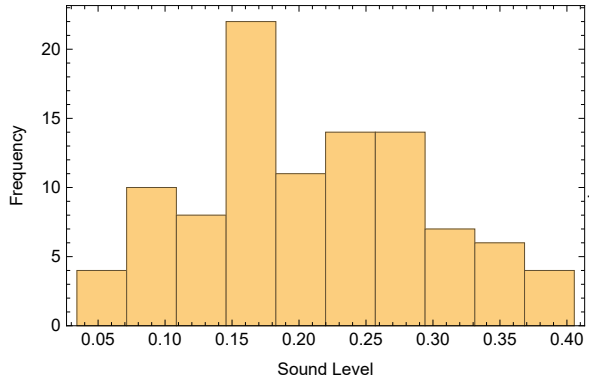
,


Frequency
0
5
10
15
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

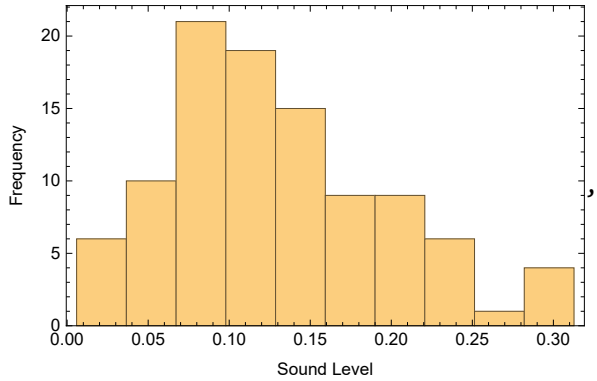
,


Frequency
0
5
10
15
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

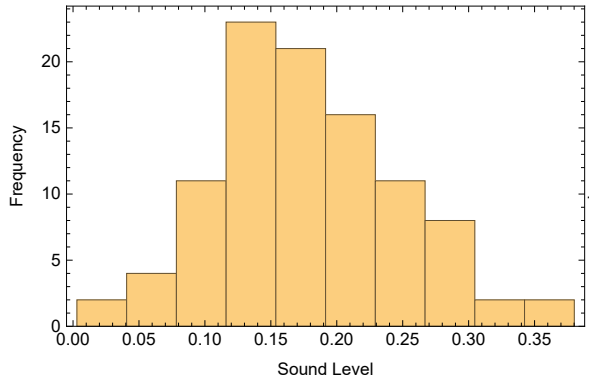
,


Frequency
0
5
10
15
20
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Sound Level

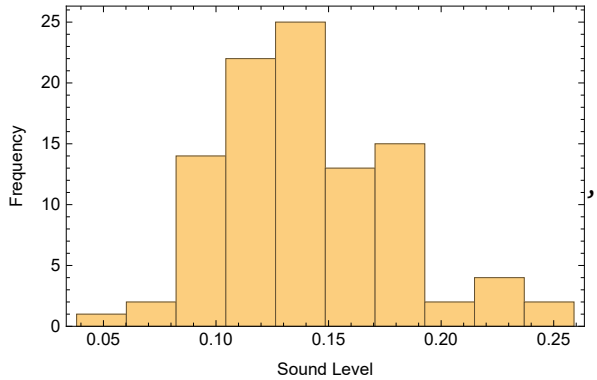
,


Frequency
0
5
10
15
20
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

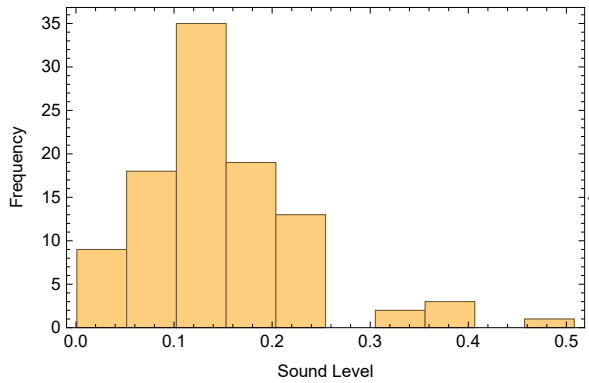
,


Frequency
0
5
10
15
20
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

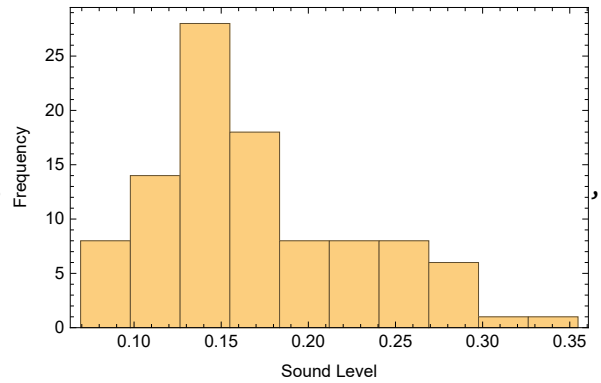
,


Frequency
0
5
10
15
20
25
0.05
0.10
0.15
0.20
0.25
Sound Level

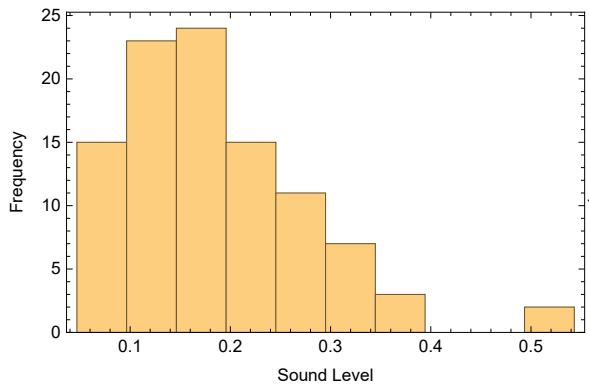
,


Frequency
0
5
10
15
20
25
30
35
0.0
0.1
0.2
0.3
0.4
0.5
Sound Level

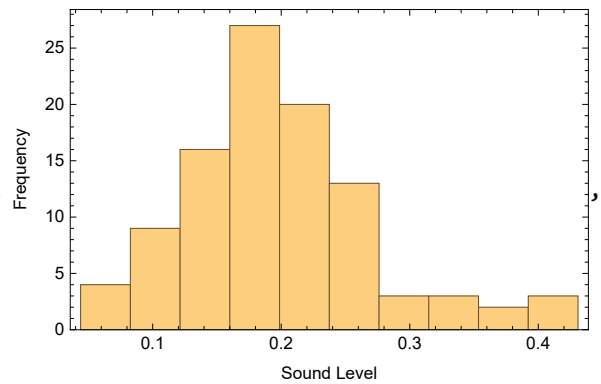
,


Frequency
0
5
10
15
20
25
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

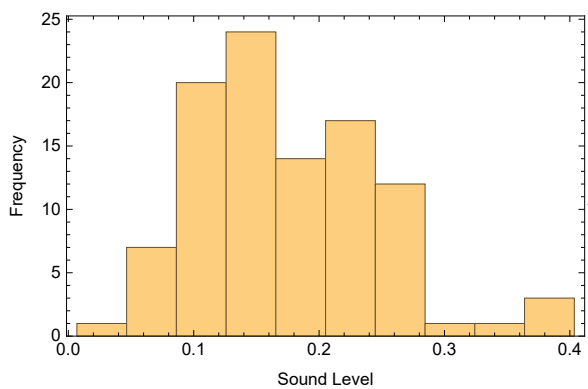
,


Frequency
0
5
10
15
20
25
0.1
0.2
0.3
0.4
0.5
Sound Level

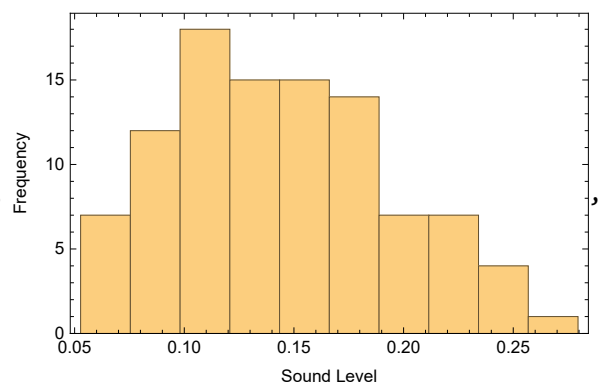
,


Frequency
0
5
10
15
20
25
0.1
0.2
0.3
0.4
Sound Level

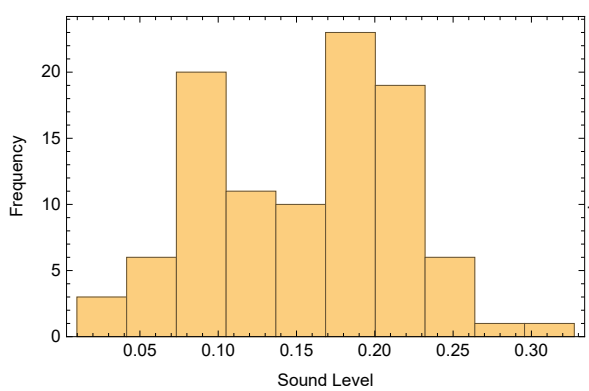
,


Frequency
0
5
10
15
20
25
0.0
0.1
0.2
0.3
0.4
Sound Level

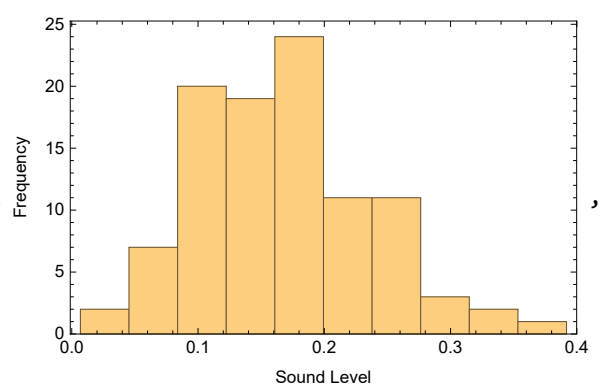
,


Frequency
0
5
10
15
0.05
0.10
0.15
0.20
0.25
Sound Level

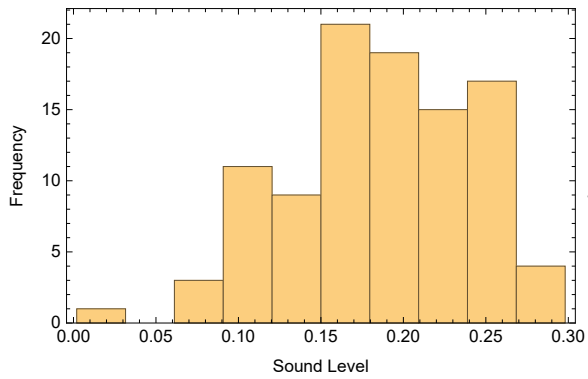
,


Frequency
0
5
10
15
20
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

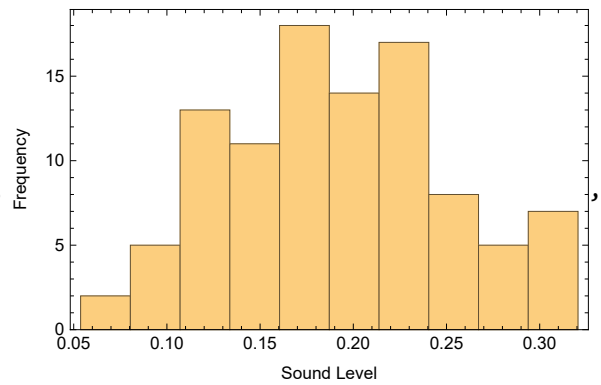
,


Frequency
0
5
10
15
20
25
0.0
0.1
0.2
0.3
0.4
Sound Level

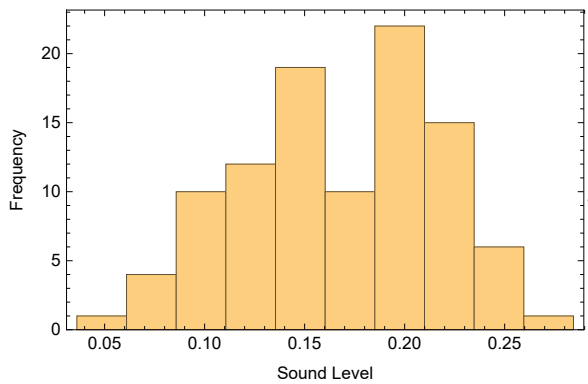
,


Frequency
0
5
10
15
20
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

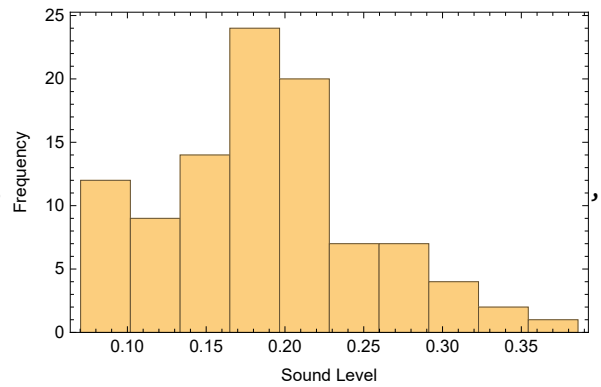
,


Frequency
0
5
10
15
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

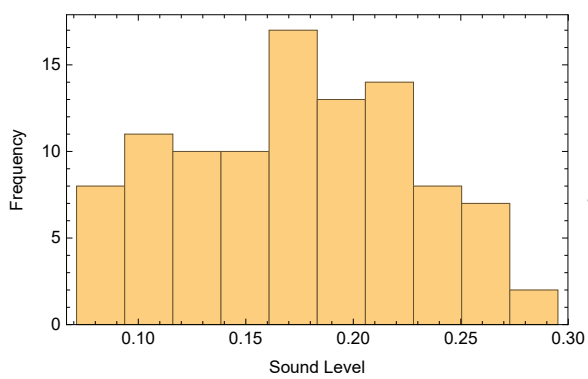
,


Frequency
0
5
10
15
20
0.05
0.10
0.15
0.20
0.25
Sound Level

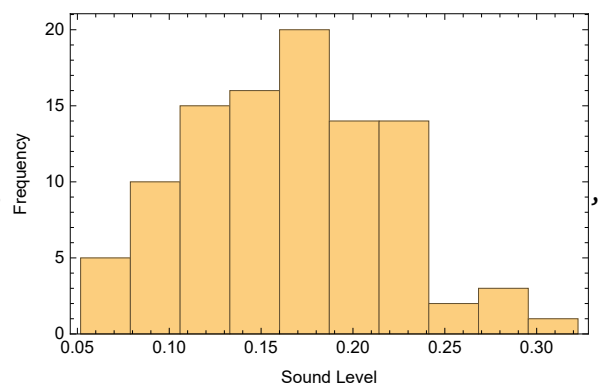
,


Frequency
0
5
10
15
20
25
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

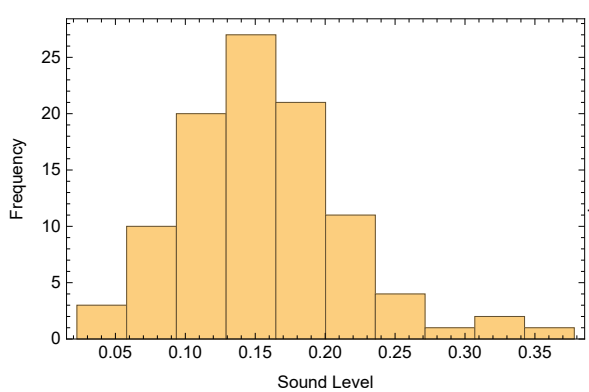
,


Frequency
0
5
10
15
0.10
0.15
0.20
0.25
0.30
Sound Level

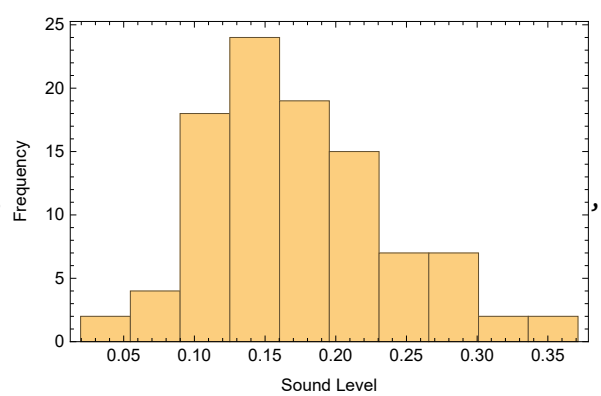
,


Frequency
0
5
10
15
20
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

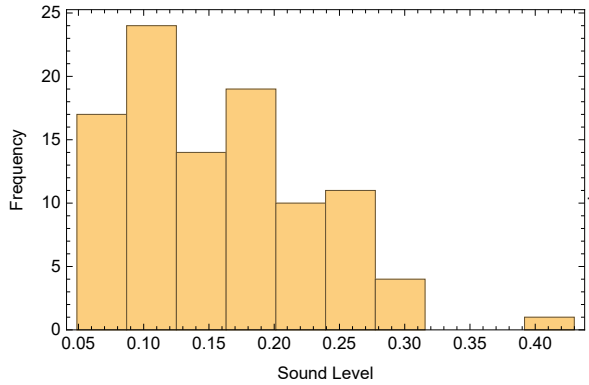
,


Frequency
0
5
10
15
20
25
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

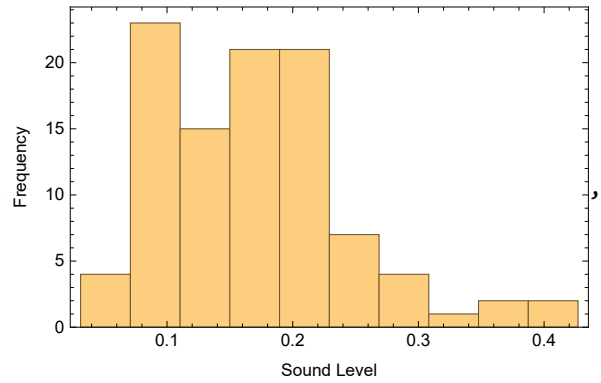
,


Frequency
0
5
10
15
20
25
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

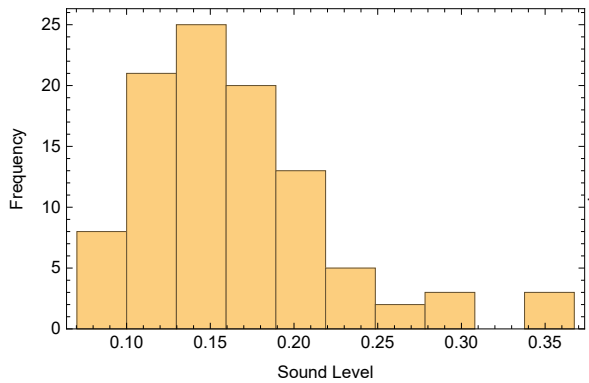
,


Frequency
Sound Level
0
5
10
15
20
25
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40

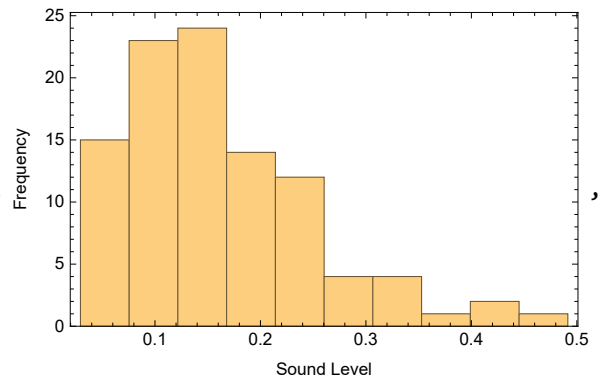
,


Frequency
Sound Level
0
5
10
15
20
0.1
0.2
0.3
0.4

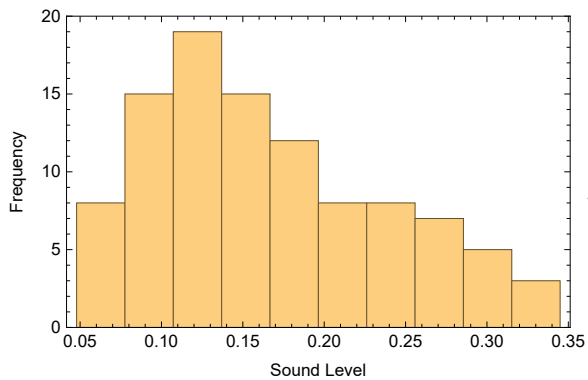
,


Frequency
Sound Level
0
5
10
15
20
25
0.10
0.15
0.20
0.25
0.30
0.35

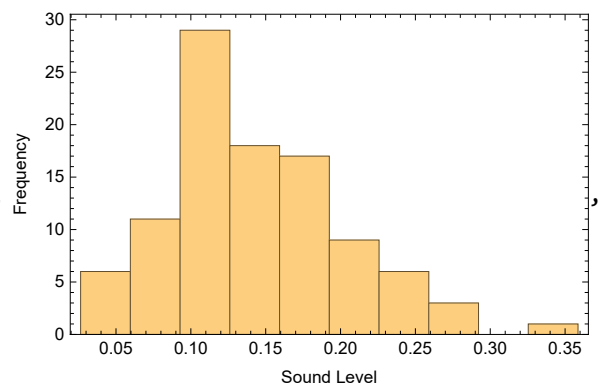
,


Frequency
Sound Level
0
5
10
15
20
25
0.1
0.2
0.3
0.4
0.5

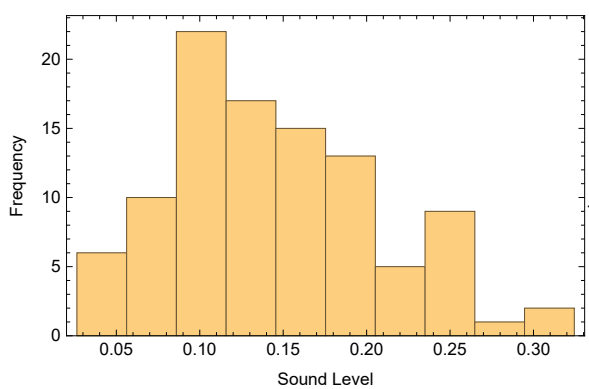
,


Frequency
Sound Level
0
5
10
15
20
0.05
0.10
0.15
0.20
0.25
0.30
0.35

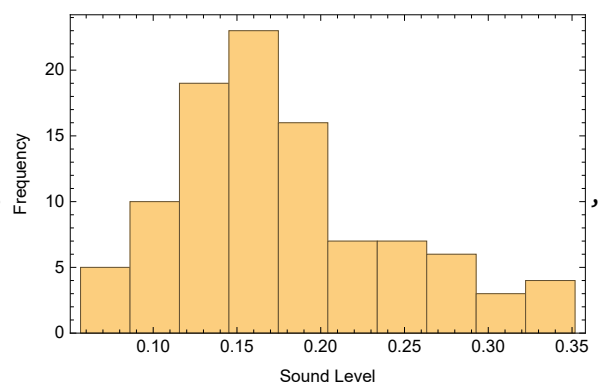
,


Frequency
Sound Level
0
5
10
15
20
25
30
0.05
0.10
0.15
0.20
0.25
0.30
0.35

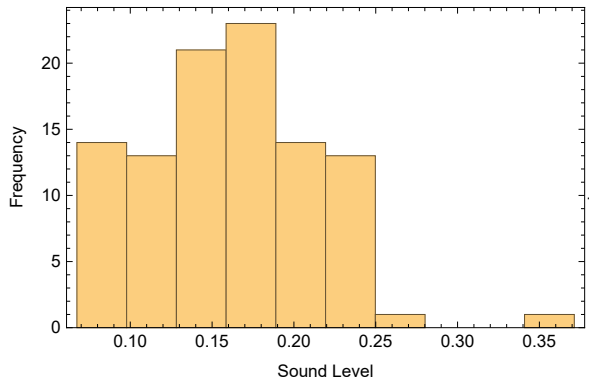
,


Frequency
Sound Level
0
5
10
15
20
0.05
0.10
0.15
0.20
0.25
0.30

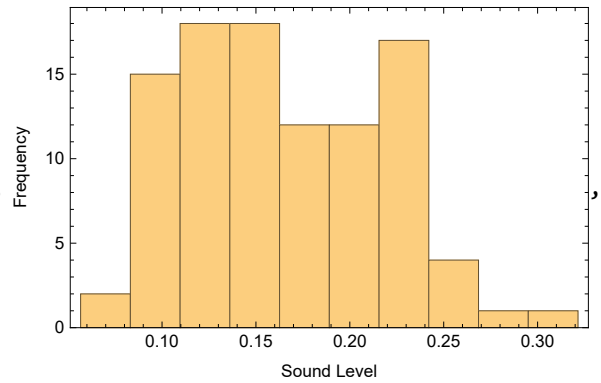
,


Frequency
Sound Level
0
5
10
15
20
0.10
0.15
0.20
0.25
0.30
0.35

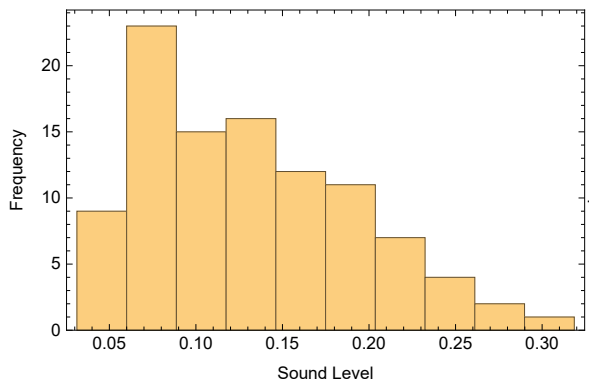
,


Frequency
0
5
10
15
20
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

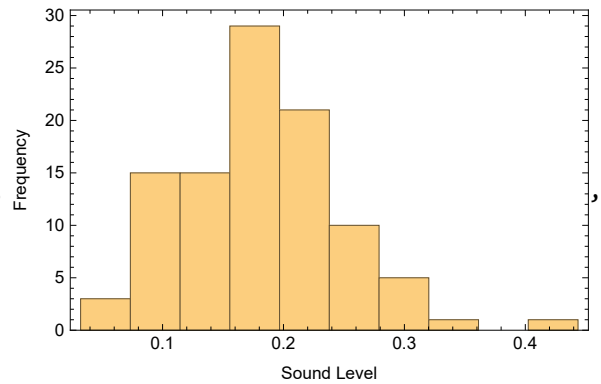
,


Frequency
0
5
10
15
0.10
0.15
0.20
0.25
0.30
Sound Level

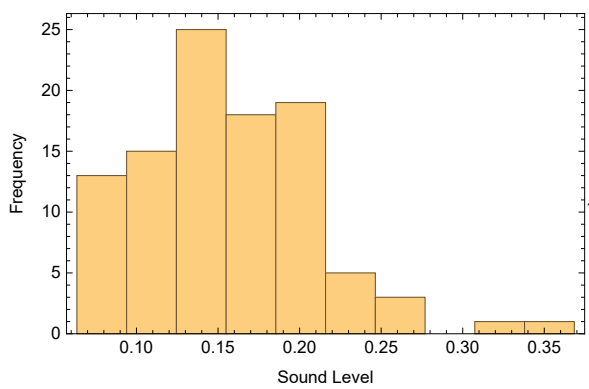
,


Frequency
0
5
10
15
20
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

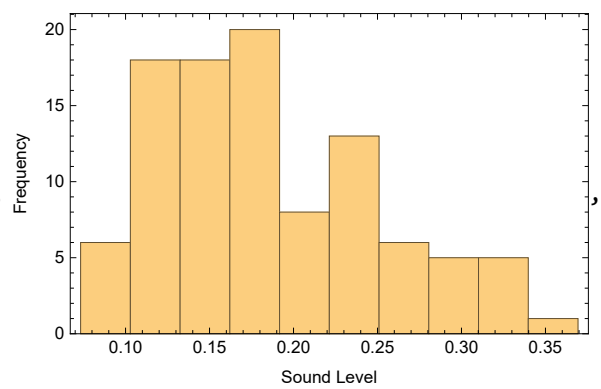
,


Frequency
0
5
10
15
20
25
30
0.1
0.2
0.3
0.4
Sound Level

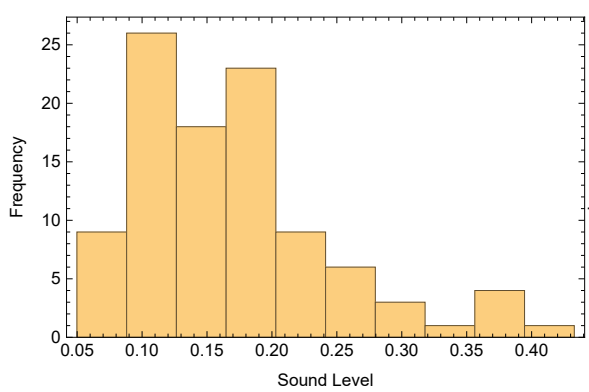
,


Frequency
0
5
10
15
20
25
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

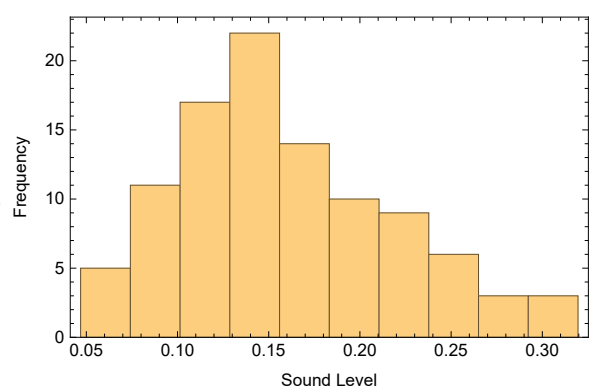
,


Frequency
0
5
10
15
20
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

,


Frequency
0
5
10
15
20
25
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Sound Level

,

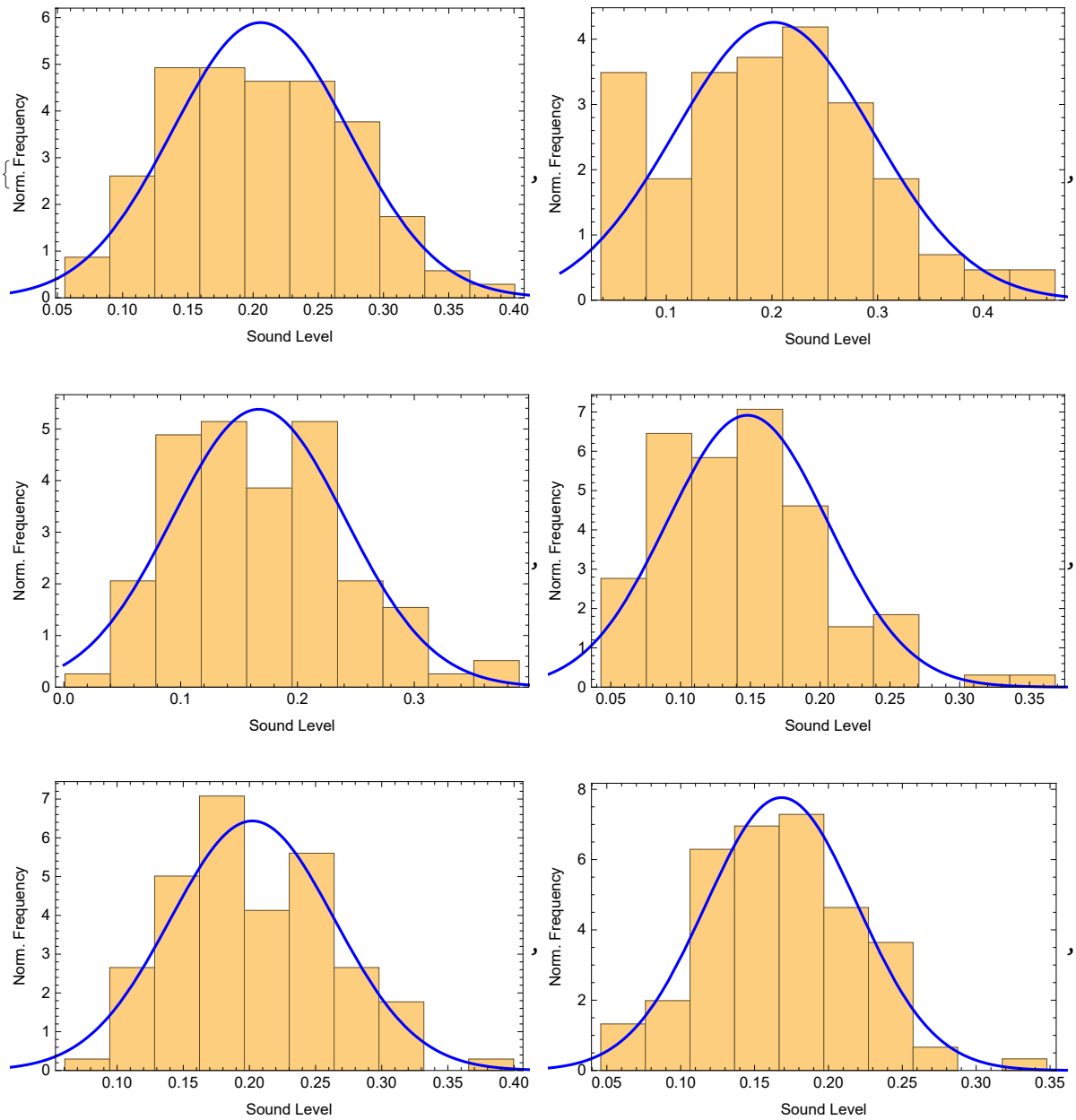

Frequency
0
5
10
15
20
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

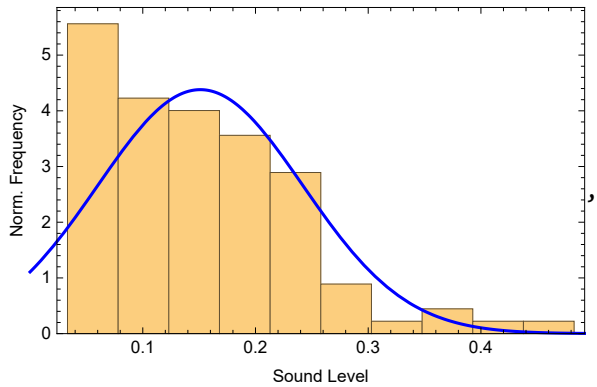
}

Plot a Gaussian on top of each of the data sets, based on the mean and variance. I made sure to normalize the histograms of each of the data sets, so that the Gaussians can be compared graphically.

In[199]:=
```
gaussian[m_, s_, x_] = 1/(Sqrt[2 π] s) Exp[-(x - m)^2/(2 s^2)];
hist2 = Table[Histogram[data[[n]], {Min[data[[n]]], Max[data[[n]]] + .01,
    (Max[data[[n]]] - Min[data[[n]]] + .01)/10}, "PDF", Frame → True,
   FrameLabel → {"Sound Level", "Norm. Frequency"}, ImageSize → 300], {n, 1, 50}];
gaussians = Table[Plot[gaussian[μ[[n]], Sqrt[V[[n]]], x],
   {x, 0, .5}, PlotStyle → Blue], {n, 1, 50}];
overlay = Table[Show[hist2[[n]], gaussians[[n]]], {n, 1, 50}]
```

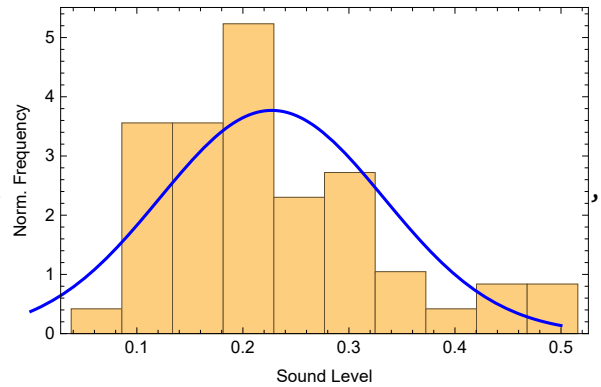
Out[202]=





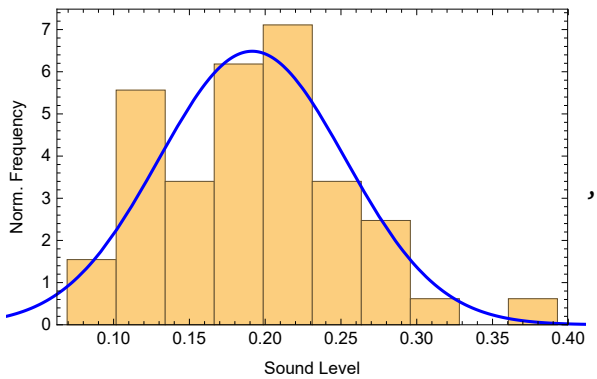
Norm. Frequency
0
1
2
3
4
5
0.1
0.2
0.3
0.4
Sound Level

,



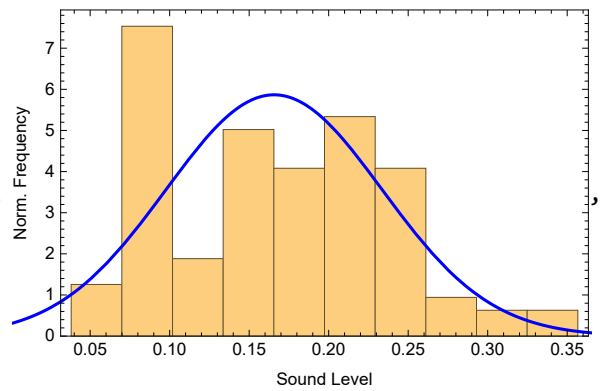
Norm. Frequency
0
1
2
3
4
5
0.1
0.2
0.3
0.4
0.5
Sound Level

,



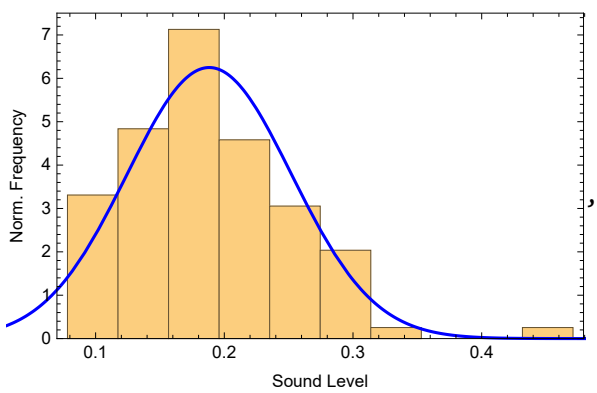
Norm. Frequency
0
1
2
3
4
5
6
7
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Sound Level

,



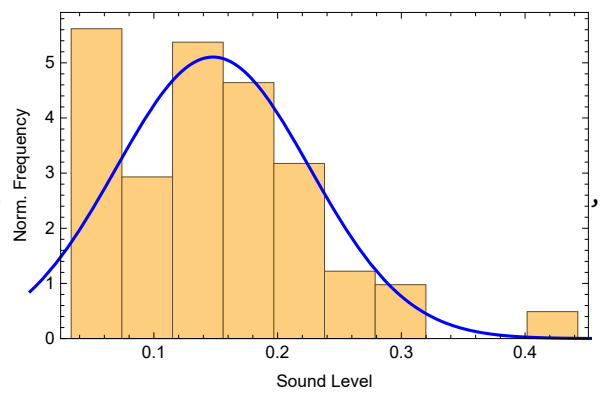
Norm. Frequency
0
1
2
3
4
5
6
7
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

,



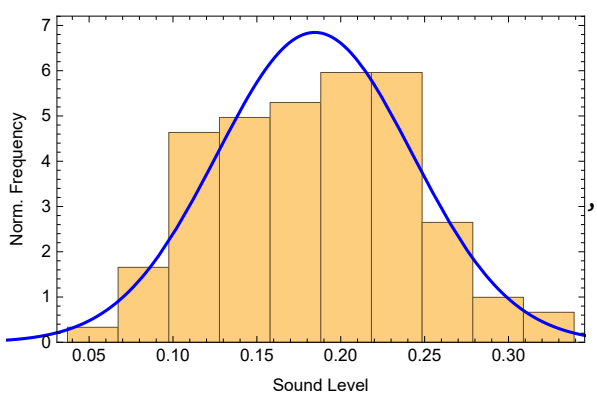
Norm. Frequency
0
1
2
3
4
5
6
7
0.1
0.2
0.3
0.4
Sound Level

,



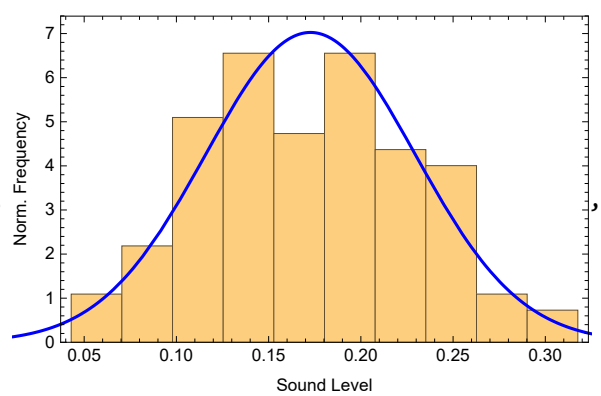
Norm. Frequency
0
1
2
3
4
5
0.1
0.2
0.3
0.4
Sound Level

,



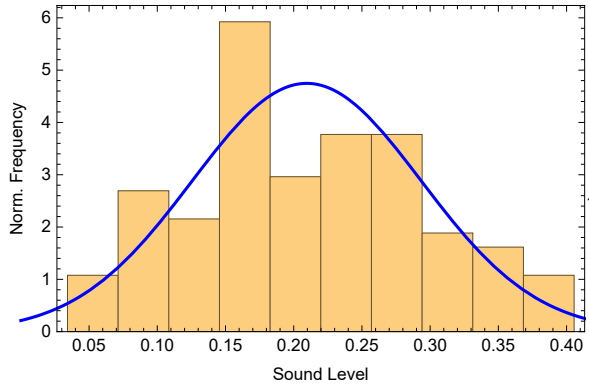
Norm. Frequency
0
1
2
3
4
5
6
7
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

,

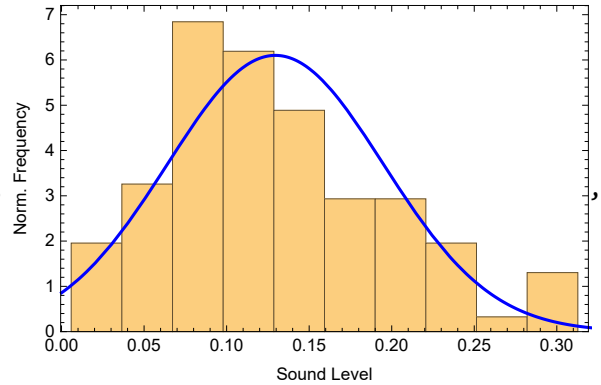


Norm. Frequency
0
1
2
3
4
5
6
7
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

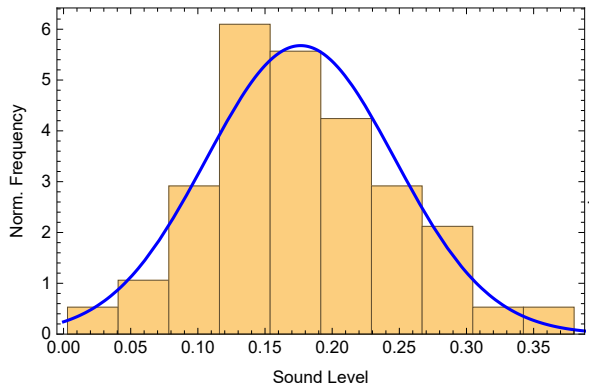
,


Norm. Frequency
0
1
2
3
4
5
6
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Sound Level

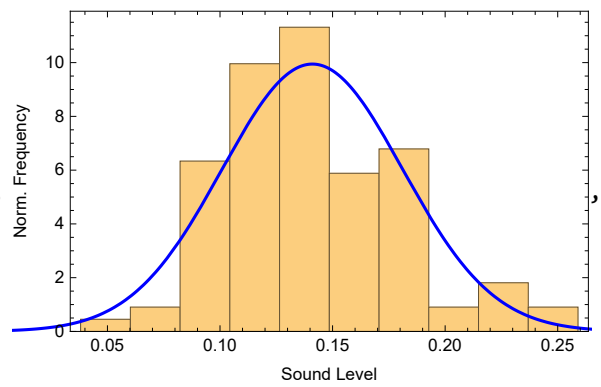
,


Norm. Frequency
0
1
2
3
4
5
6
7
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

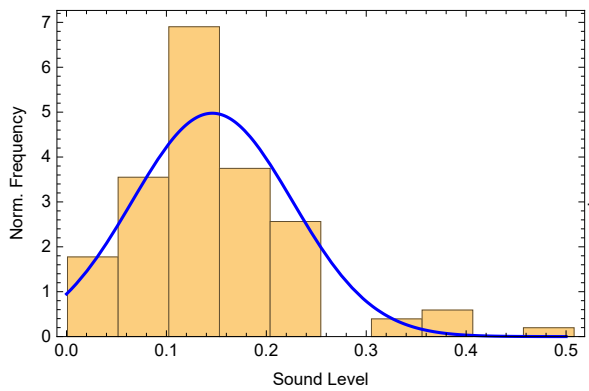
,


Norm. Frequency
0
1
2
3
4
5
6
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

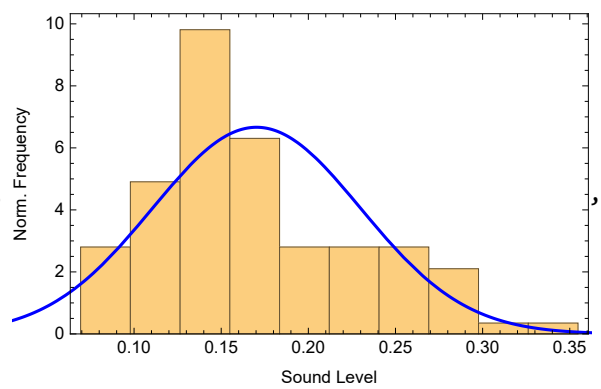
,


Norm. Frequency
0
2
4
6
8
10
0.05
0.10
0.15
0.20
0.25
Sound Level

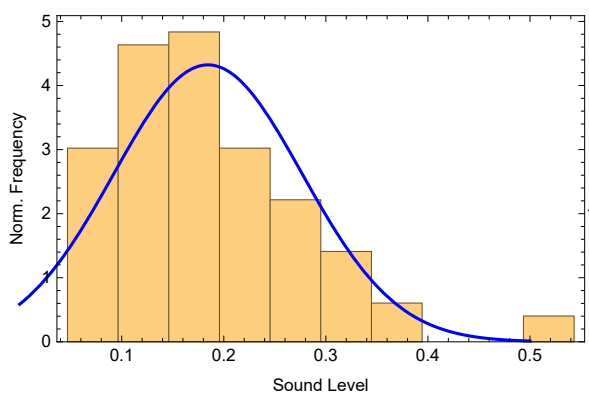
,


Norm. Frequency
0
1
2
3
4
5
6
7
0.0
0.1
0.2
0.3
0.4
0.5
Sound Level

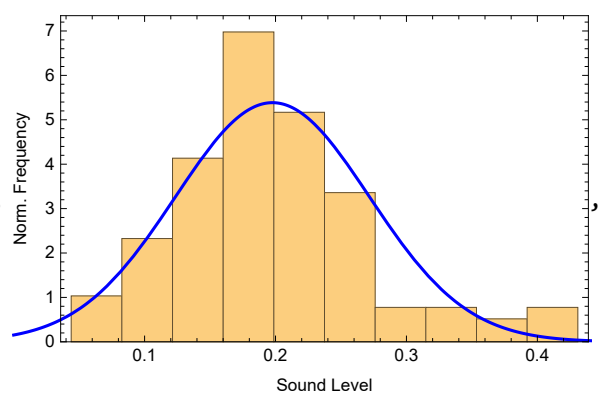
,


Norm. Frequency
0
2
4
6
8
10
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

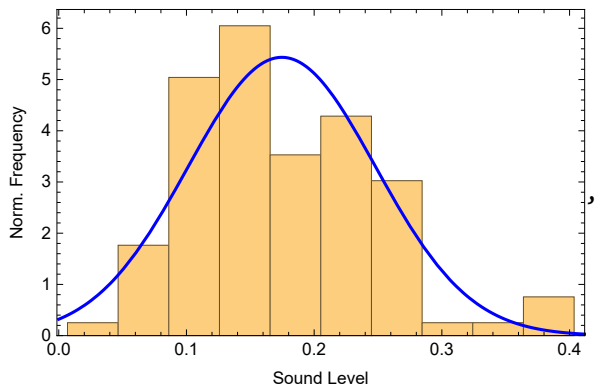
,


Norm. Frequency
0
1
2
3
4
5
0.1
0.2
0.3
0.4
0.5
Sound Level

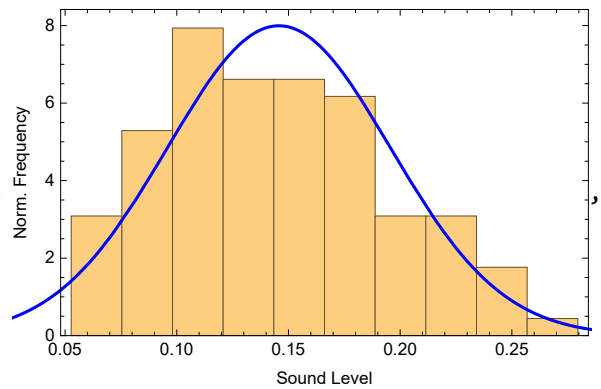
,


Norm. Frequency
0
1
2
3
4
5
6
7
0.1
0.2
0.3
0.4
Sound Level

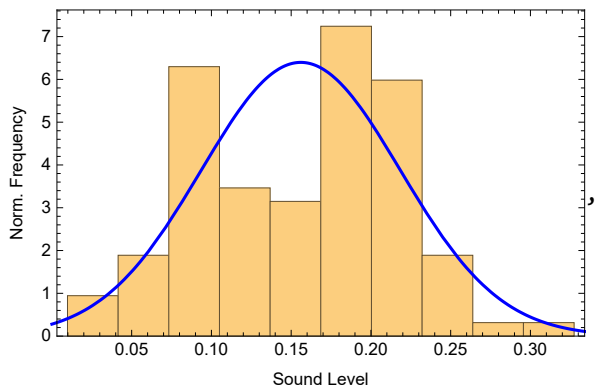
,


Norm. Frequency
0
1
2
3
4
5
6
0.0
0.1
0.2
0.3
0.4
Sound Level

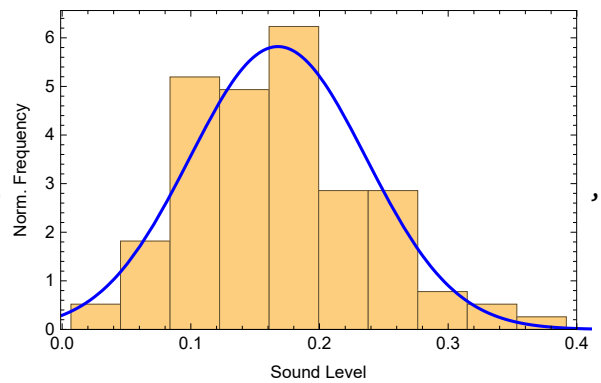
,


Norm. Frequency
0
2
4
6
8
0.05
0.10
0.15
0.20
0.25
Sound Level

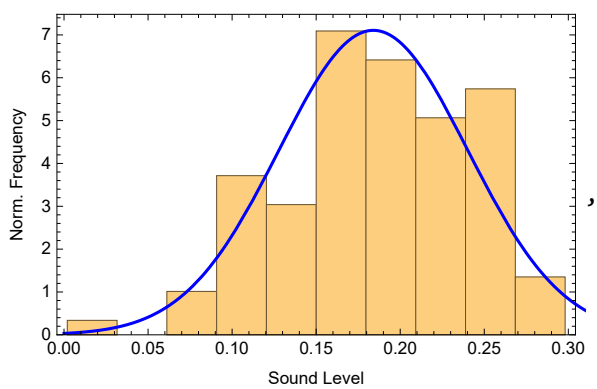
,


Norm. Frequency
0
1
2
3
4
5
6
7
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

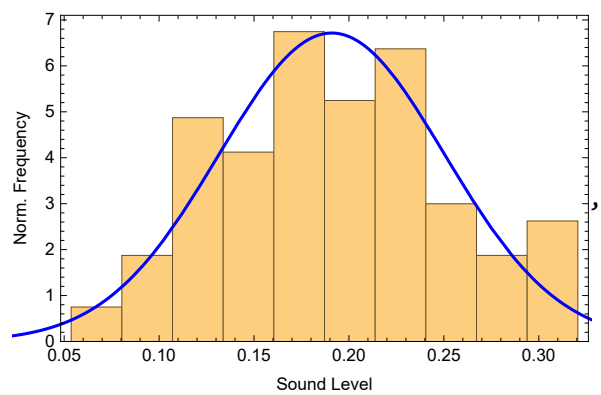
,


Norm. Frequency
0
1
2
3
4
5
6
0.0
0.1
0.2
0.3
0.4
Sound Level

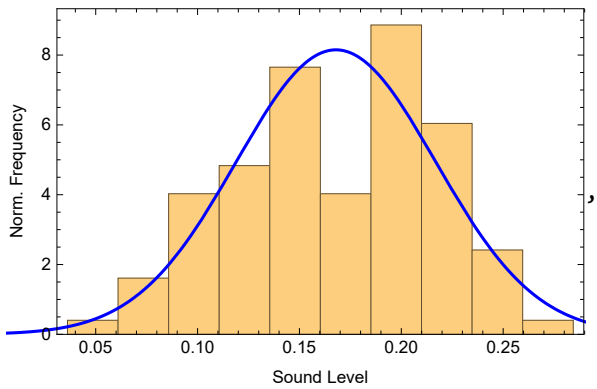
,


Norm. Frequency
0
1
2
3
4
5
6
7
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

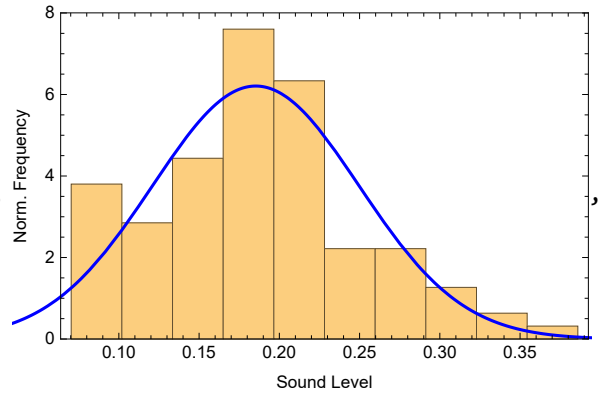
,


Norm. Frequency
0
1
2
3
4
5
6
7
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

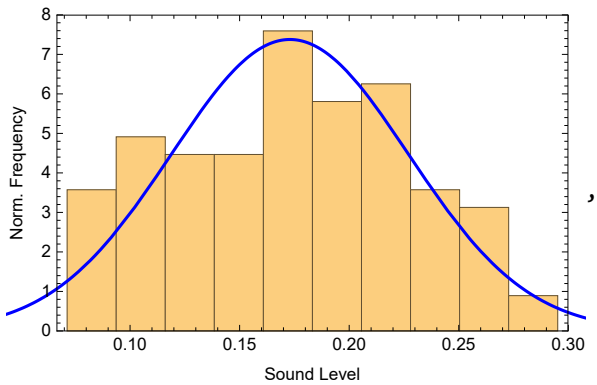
,


Norm. Frequency
0
2
4
6
8
0.05
0.10
0.15
0.20
0.25
Sound Level

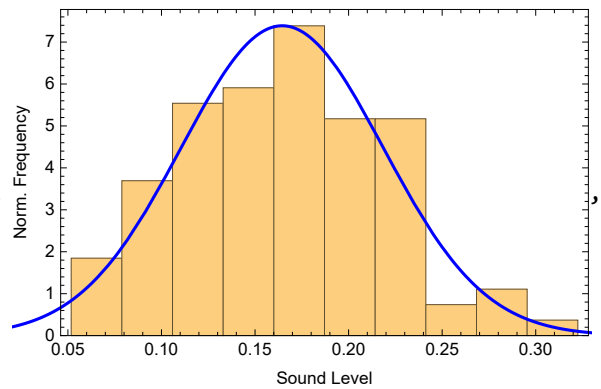
,


Norm. Frequency
0
2
4
6
8
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

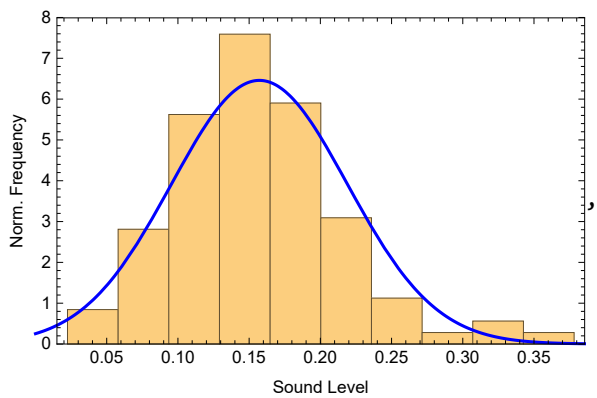
,


Norm. Frequency
0
1
2
3
4
5
6
7
8
0.10
0.15
0.20
0.25
0.30
Sound Level

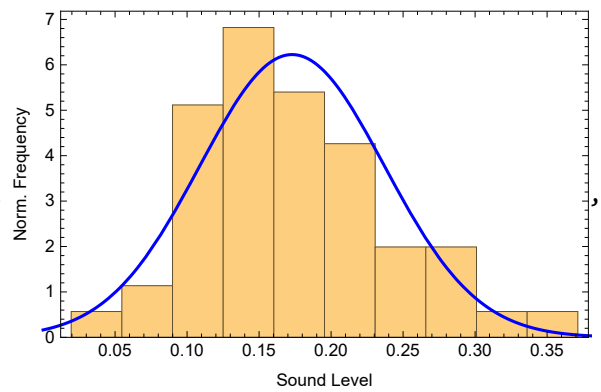
,


Norm. Frequency
0
1
2
3
4
5
6
7
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

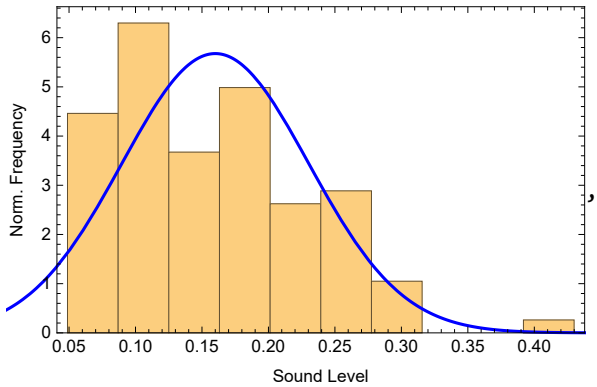
,


Norm. Frequency
0
1
2
3
4
5
6
7
8
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

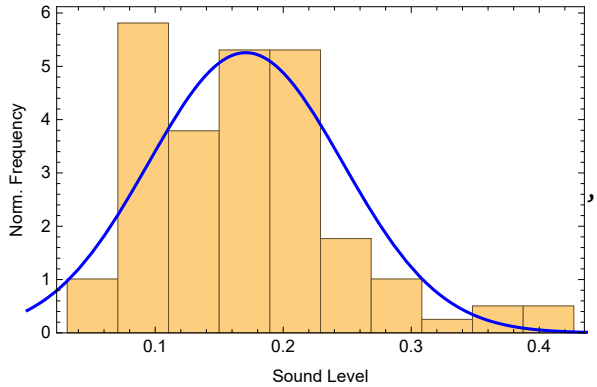
,


Norm. Frequency
0
1
2
3
4
5
6
7
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

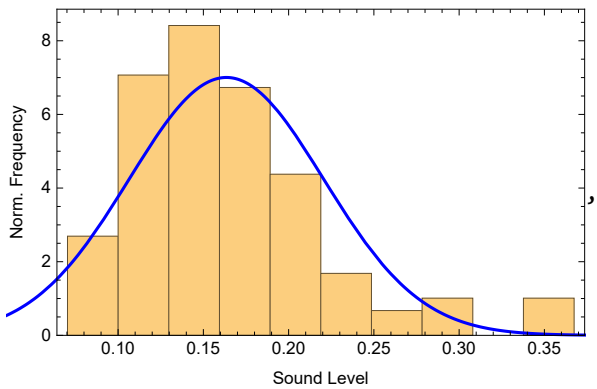
,


Norm. Frequency
0
1
2
3
4
5
6
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Sound Level

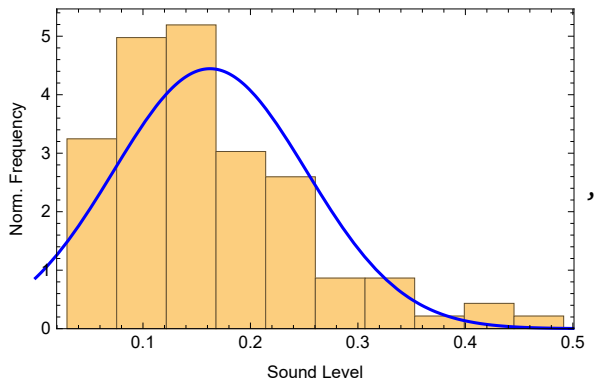
,


Norm. Frequency
0
1
2
3
4
5
6
0.1
0.2
0.3
0.4
Sound Level

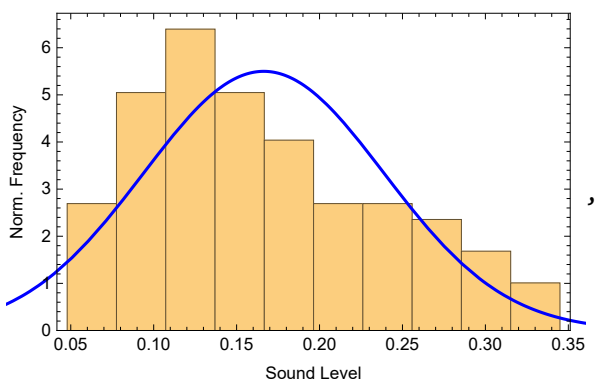
,


Norm. Frequency
0
2
4
6
8
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

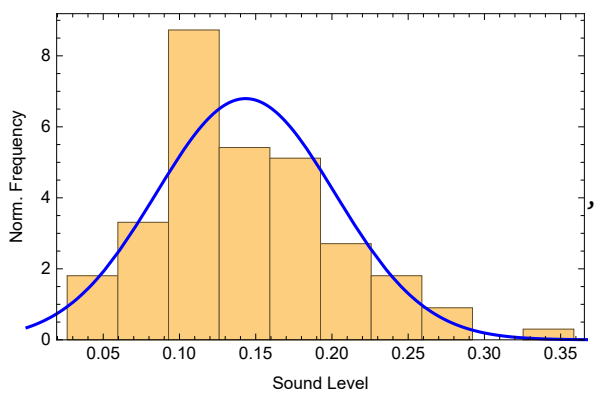
,


Norm. Frequency
0
1
2
3
4
5
0.1
0.2
0.3
0.4
0.5
Sound Level

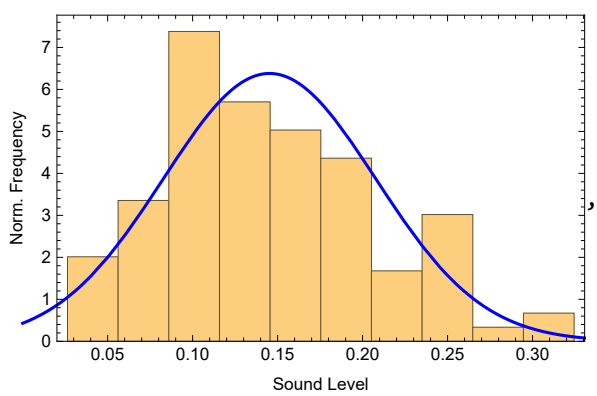
,


Norm. Frequency
0
1
2
3
4
5
6
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

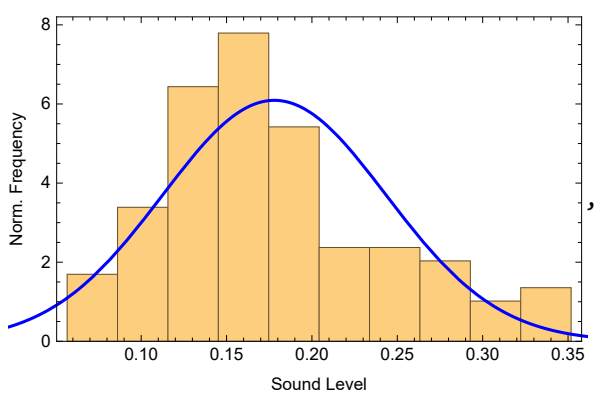
,


Norm. Frequency
0
2
4
6
8
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

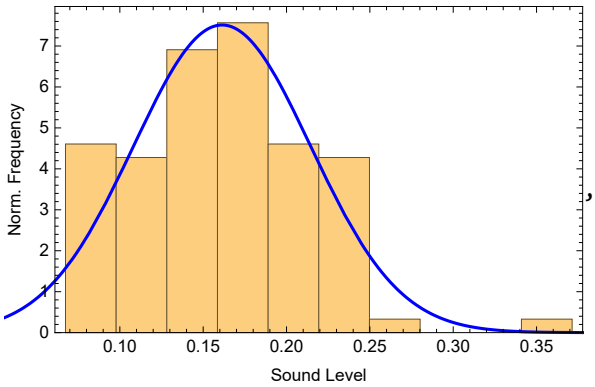
,

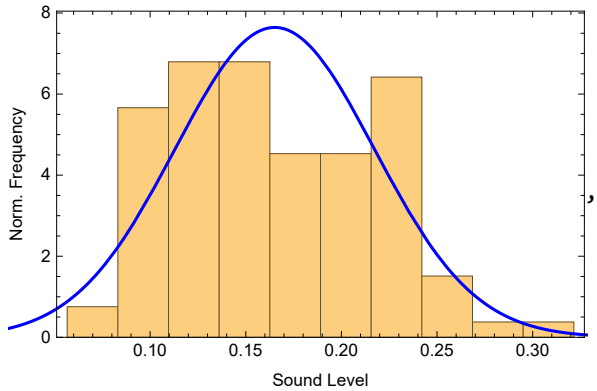

Norm. Frequency
0
1
2
3
4
5
6
7
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level

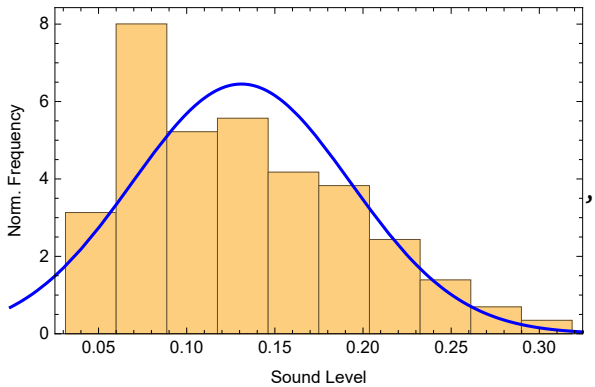
,

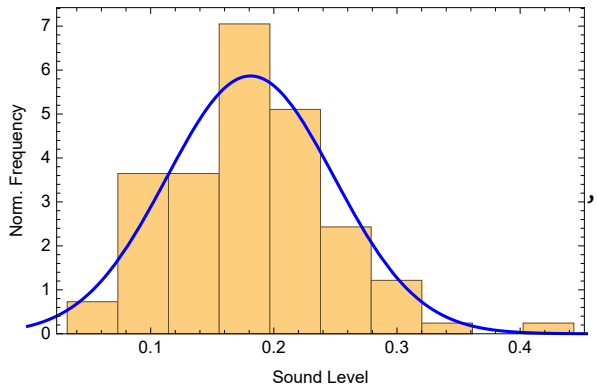

Norm. Frequency
0
2
4
6
8
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level

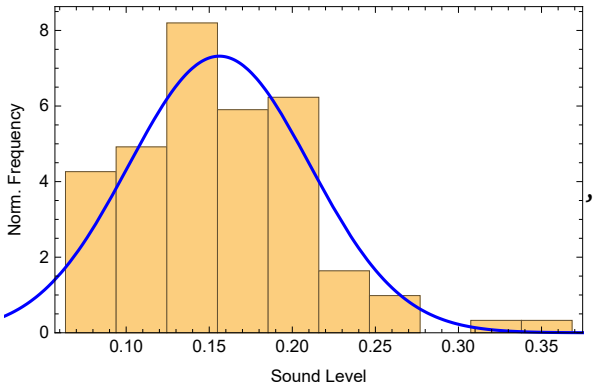
,

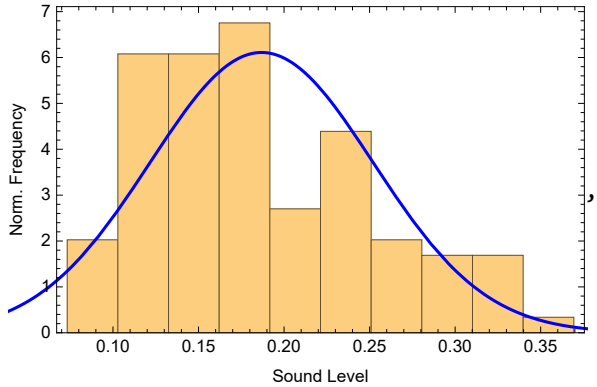

Norm. Frequency
0
1
2
3
4
5
6
7
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level
,

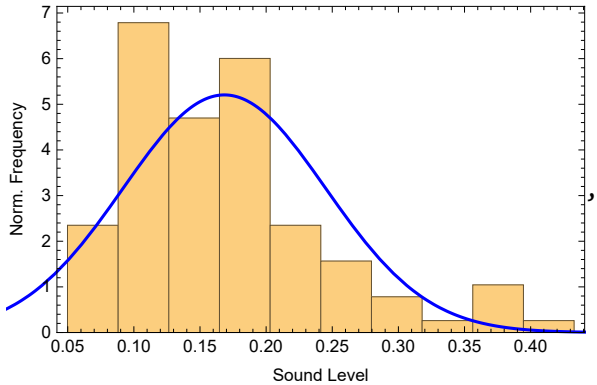

Norm. Frequency
0
2
4
6
8
0.10
0.15
0.20
0.25
0.30
Sound Level
,

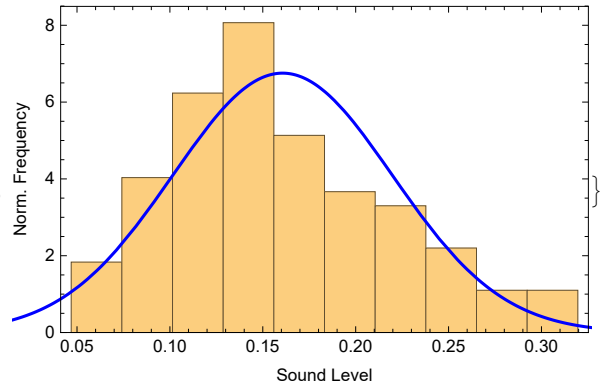

Norm. Frequency
0
2
4
6
8
0.05
0.10
0.15
0.20
0.25
0.30
Sound Level
,


Norm. Frequency
0
1
2
3
4
5
6
7
0.1
0.2
0.3
0.4
Sound Level
,


Norm. Frequency
0
2
4
6
8
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level
,


Norm. Frequency
0
1
2
3
4
5
6
7
0.10
0.15
0.20
0.25
0.30
0.35
Sound Level
,





I would say that the Gaussians have a moderate fit on the data sets. The trend in the majority of the sets is that there is a peak in the middle, with the data falling off on both sides. This is a basic characteristic of a Gaussian that is indeed satisfied for these data sets. On the other hand, many of the data sets are non-symmetric about the mean, which is non-Gaussian. Also, some of the sets are very “flat”, with not so much drop off around the mean. This is also non-Gaussian.

c.

I would say that overall, the Gaussian model is the best fit for the individual data sets. Volume data taken once per second over the course of a song is an interesting set of data to look at, as the distribution changes a bit with each song. Overall, however, the general trend of a high point at the mean and somewhat symmetrical drop-off on either side of the mean leads me to the conclusion that a Gaussian is the best fit (as opposed to a Binomial or Poisson Distribution).

Thinking about the source of the data, a Gaussian makes sense. Over the course of a song, there will be points of high volume, and low volume, but for the majority of the time, the song will be at some baseline volume. This “baseline” volume is shown as the peaks in the histograms, while the high and low volume points fall off on either side of it.

d.

Calculate the mean of means

In[203]:=
```
μofmeans = Mean[μ]
```

Out[203]= 0.170342

Calculate the variance of means

In[204]:=
```
Vofmeans = Sum[(μ[[n]] - μofmeans)^2 / Length[μ], {n, 1, Length[μ]}]
```

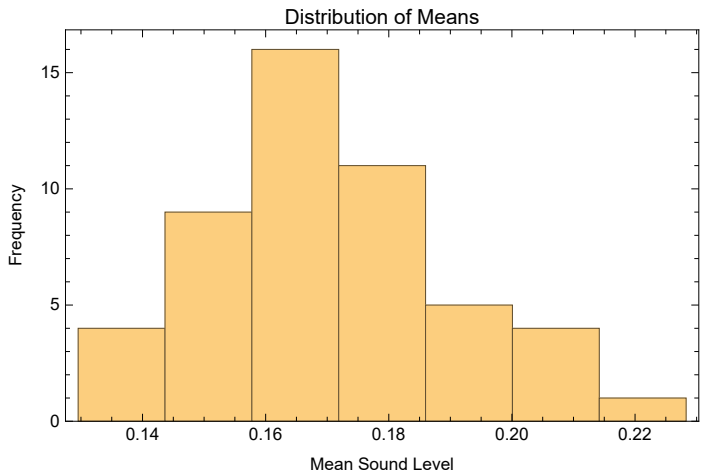
Out[204]= 0.00040868

e.

Plot the distribution over the means

In[205]:=
```
Histogram[μ, {Min[μ], Max[μ] + .001, (Max[μ] - Min[μ] + .001)/7}, Frame → True,
 FrameLabel → {"Mean Sound Level", "Frequency"}, PlotLabel → "Distribution of Means"]
```

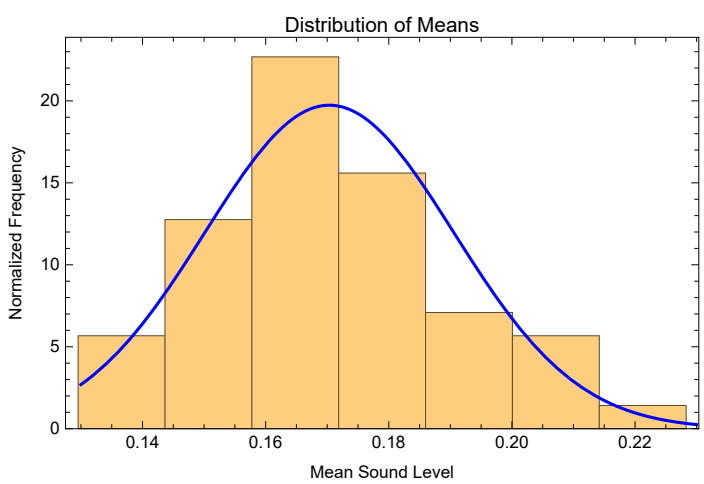
Out[205]=



Plot a Gaussian on top of the distribution. Once again, I make sure to normalize my histogram before plotting the Gaussian on top of it.

In[206]:=
```
hist3 = Histogram[μ, {Min[μ], Max[μ] + .001, (Max[μ] - Min[μ] + .001)/7}, "PDF",
   Frame → True, FrameLabel → {"Mean Sound Level", "Normalized Frequency"},
   PlotLabel → "Distribution of Means"];
gaussian1 = Plot[gaussian[μofmeans, √Vofmeans, x], {x, .13, .23}, PlotStyle → Blue];
Show[hist3, gaussian1]
```

Out[208]=



f.

While it may look from the plot above that the central limit theorem (CLT) does apply in my case, I make the argument that it should not apply to my data. The biggest assumption that is required for the CLT to apply is that each of the data sets are independent and identically distributed (IID). My data, however, is not IID.

Each set of volume level comes from a different song, which breaks the “identically distributed” part of IID, as the volume from each data set is not being pulled from the same big set of data. Each set of data is pulled from its own individual data set (the volume over the course of that specific song). As **songs significantly vary from one other**, I argue that this means my data is not identically distributed.

This points lead me to the conclusion that my data is not IID, and therefore the CLT cannot apply. This can sort of be seen in the plot above, as the data does not perfectly follow the normal distribution (although it is close, which is interesting).

g.

Combine data into one huge set.

In[209]:=
```
bigdata = Flatten[data];
```

Calculate the mean and variance

In[210]:=
```
μbig = Mean[bigdata]
```

Out[210]= 0.170342

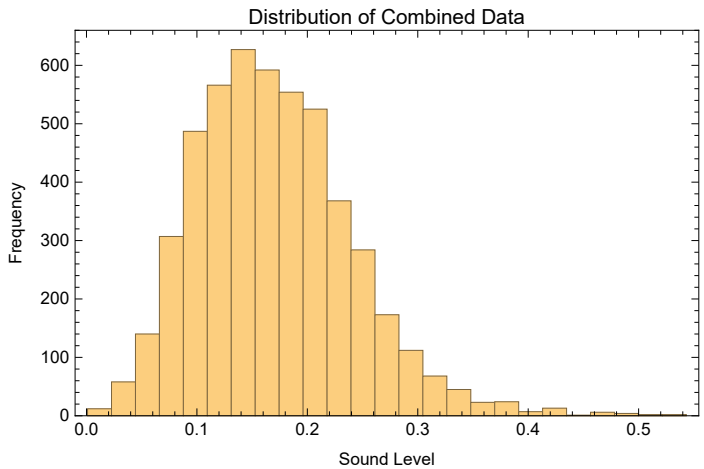
In[211]:=
```
Vbig = Sum[(bigdata[[n]] - μbig)^2/Length[bigdata], {n, 1, Length[bigdata]}]
```

Out[211]= 0.00497328

Plot the distribution of all data

In[212]:=
```
Histogram[bigdata, {Min[bigdata], Max[bigdata] + .01, (Max[bigdata] - Min[bigdata] + .01)/25},
 Frame → True, FrameLabel → {"Sound Level", "Frequency"},
 PlotLabel → "Distribution of Combined Data"]
```

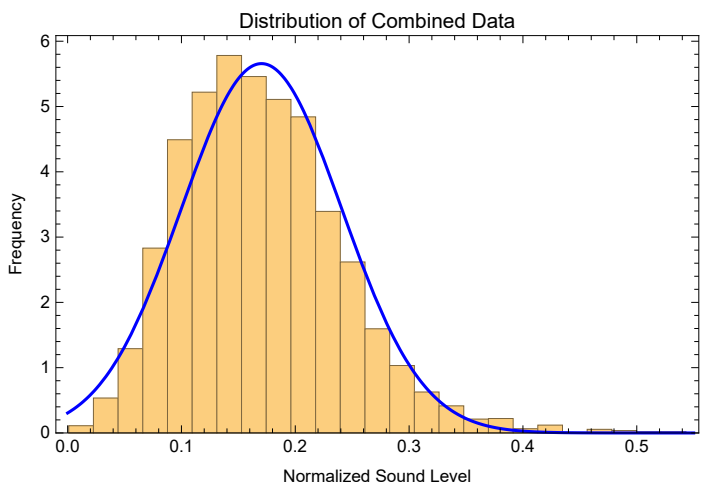
Out[212]=



Plot the distribution of data with a Gaussian on top of it, making sure to normalize the histogram.

In[213]:=
```
hist4 = Histogram[bigdata,
   {Min[bigdata], Max[bigdata] + .01, (Max[bigdata] - Min[bigdata] + .01)/25},
   "PDF", Frame → True, FrameLabel → {"Normalized Sound Level", "Frequency"},
   PlotLabel → "Distribution of Combined Data"];
gaussian2 = Plot[gaussian[μbig, Sqrt[Vbig], x], {x, 0, .55}, PlotStyle → Blue];
Show[hist4, gaussian2]
```

Out[215]=



You can see that the distribution of all of my data looks nice and smooth, but it is not symmetric about the mean. There is a much longer tail on the right side. It seems as though it would be almost perfectly Gaussian if there was not a hard cutoff at 0 (which is a result of the data I am looking at, as volume cannot be negative).

h.

Calculate the optimized mean, using inverse variance weighting. The optimized mean is defined by:

mean = $\sum x_i w_i$ where $w_i = \frac{1}{V_i}\left(\sum \frac{1}{V_j}\right)^{-1}$

In[216]:=
```
μopt = Sum[μ[[n]] 1/V[[n]] (Sum[1/V[[m]], {m, 1, Length[V]}])^-1, {n, 1, Length[μ]}]
```

Out[216]= 0.167681

Compare this to the definitional mean, obtained previously.

In[217]:=
```
μPercentDiff = Abs[μopt - μbig] / ((μbig+μopt)/2) * 100
```

Out[217]= 1.57429

So there is only a 1.574% difference between the optimized and actual mean of all of the data.

i.

Calculate the optimized variance, using inverse variance weighting. The optimized variance is defined by:

variance (of the means) = $\left(\sum \frac{1}{V_j}\right)^{-1}$

In[218]:=
```
Vopt = (Sum[1/V[[m]], {m, 1, Length[V]}])^-1
```

Out[218]= 0.0000793443

Compare this to the definitional variance of means, obtained previously.

In[219]:=
```
VPercentDiff = Abs[Vopt - Vofmeans] / ((Vofmeans+Vopt)/2) * 100
```

Out[219]= 134.967

So there is an 134.967% difference between the optimized and actual variance of the means for my 50 data sets. The actual value is larger than the optimized value. This is a result of the fact that the means of my 50 data sets are more spread out than they theoretically could be based upon the variance of each of these data sets.

j.

The probability that the optimized mean (or more than the optimized mean) is a valid measurement of

the definitional mean, based on the variance of the definitional mean, is:

In[220]:=
```
Integrate[gaussian[μofmeans, √Vofmeans , x], {x, -∞, μopt}]
```
Out[220]= 0.447644

So there is a 44% chance that the optimized mean (or less) is a valid measurement of the definitional mean. This is assuming a Gaussian fits my distribution of means, however, which may not be the most accurate assumption. As a result of this, I take this percentage with a grain of salt.

---

k.

I will do this analysis assuming that my data is IID, as it is a requirement to make a lot of these assumptions about sample vs population data. Know that I understand that these assumptions are not 100% valid for my data.

If it is assumed that my big data set was pulled from some much bigger set of data, then I have taken a sample of 5000 data points from the “population”, where the population is the bigger data set that I have theoretically pulled from.

We derived in class that as the number of data points in the sample increases, the mean of the samples converges to the mean of the population. So our mean of means, which is derived from 5000 individual data points, is most likely a very good estimate of the mean of the population data set.

We also derived in in class the following result for an estimation of the population variance:

$$\text{Variance}_{\text{population}} = \frac{n}{n-1}\,\text{Mean}(\text{Variance}_{\text{sample}})$$

Where n is the number of data points in each sample. For my data, n = 50, as there are 50 sets of data with defined variances. The equation above tells us that if we find the average of the variances of my 50 data sets, it will be equivalent to $\frac{49}{50} * \text{Variance}_{\text{population}}$.This calculation is done below.

In[221]:=
```
Vsampleavg = Mean[V]
```
Out[221]= 0.0045646

In[222]:=
```
Vpop = 50/49 * Vsampleavg
```
Out[222]= 0.00465776

So we have an idea of what the variance of the population data is. it is important to remember that this is just an estimation. So what can I conclude? That becasue of my very large number of data points, my sample of the overall population is statistically very close to accurately representing the overall population, both its mean and variance.

Analysis Part 2: Finer Features of your Data - Moments, Tails, and Bad Data Points

a.

I will approximate the moment integral as a sum. Changed from an integral to a sum, the moment equation becomes:

nth moment of x = $\Delta x \sum (x - c)^n P(x)$

where Δx is the spacing of the bins in the histogram, the sum is over the x values of the centers of the bins in a histogram, and P(x) is the normalized histogram height at the associated bin. For the 3rd and 4th moments, we will center the sum at the mean of means (definitional).

First, I find the un-normalized heights of each of the bins in the histogram:

In[223]:=
```
counts =
 BinCounts[bigdata, {Min[bigdata], Max[bigdata] + .01, (Max[bigdata] - Min[bigdata] + .01)/25}]
```

Out[223]=
```
{12, 58, 140, 307, 487, 566, 627, 592, 554,
 525, 368, 284, 173, 112, 68, 45, 23, 24, 7, 13, 1, 6, 4, 2, 2}
```

Next, I normalize the bin count, by finding the total area in the histogram, and dividing each un-normalized height by this value. This will make it so the area in the histogram adds to 1.

In[224]:=
```
binSpacing = (Max[bigdata] - Min[bigdata] + .01)/25
```

Out[224]=
```
0.0216889
```

In[225]:=
```
totalArea = Sum[binSpacing * counts[[n]], {n, 1, Length[counts]}]
```

Out[225]=
```
108.445
```

In[226]:=
```
normalCounts = counts/totalArea;
```

Check to make sure area adds to 1:

In[227]:=
```
Sum[binSpacing * normalCounts[[n]], {n, 1, Length[counts]}]
```

Out[227]=
```
1.
```

Now that I know the histogram is normalized, I create table of midpoint x values of each of the bins:

In[228]:=
```
xmids = Table[Min[bigdata] + binSpacing/2 + binSpacing * n, {n, 0, Length[counts] - 1}];
```

Now we have all of the information to do the sums for the moments:

In[229]:=
```
m0 = binSpacing * Sum[(xmids[[n]])^0 * normalCounts[[n]], {n, 1, Length[normalCounts]}]
```
Out[229]= 1.

In[230]:=
```
m1 = binSpacing * Sum[(xmids[[n]])^1 * normalCounts[[n]], {n, 1, Length[normalCounts]}]
```
Out[230]= 0.170324

In[231]:=
```
m2 =
 binSpacing * Sum[(xmids[[n]] - μbig)^2 * normalCounts[[n]], {n, 1, Length[normalCounts]}]
```
Out[231]= 0.00501303

In[232]:=
```
m3 =
 binSpacing * Sum[(xmids[[n]] - μbig)^3 * normalCounts[[n]], {n, 1, Length[normalCounts]}]
```
Out[232]= 0.000257449

In[233]:=
```
m4 =
 binSpacing * Sum[(xmids[[n]] - μbig)^4 * normalCounts[[n]], {n, 1, Length[normalCounts]}]
```
Out[233]= 0.000104791

I calculated the first four moments, just to check and see that the sums were working, and we see that the 0th moment is 1, as it should be, the second moment is .170, which should be the mean, which is very close to what we calculated it to be. The second moment is .005, which should be the variance, which is also very close to what we calculated it to be. This means that the moment sums are working well. Now I can confidently analyze the third and fourth moments.

The third moment is almost consistent with a normal distribution. It should be zero if perfectly Gaussian, and it is pretty close: 0.00026. It is a order of magnitude smaller than the variance, so it is small, but not zero. The third moment is a measure of the skew, or the symmetry around the mean, and we can see that our data has a long tail on the right side, so that is probably where this discrepancy from zero comes from.

The fourth moment is theoretically (assuming a Gaussian): $3\,\sigma^4$

In[234]:=
```
3 * (Sqrt[Vbig])^4
```
Out[234]= 0.0000742006

We can see that the Gaussian 4th moment is 0.7×10^{-4}, which is quite close to the theoretical value, 1×10^{-4}. So it is fair to say that this 4th moment is also close to consistent with a Gaussian. In conclusion, moments 0-2 were pretty much spot on with a Gaussian, and moments 3-4 were also very close to what a Gaussian predicts, but not perfect.

To quantitatively analyze the differences in 3rd and 4th moments, I look at percent differences between

Gaussian predictions and calculated values. This doesn't work so well for the 3rd moment, as the Gaussian prediction is zero, but I do it for the fourth:

In[235]:=
```
pdiff[a_, b_] = Abs[a - b] / ((a+b)/2) * 100;
```

In[236]:=
```
pdiff[3 * (Sqrt[Vbig])^4, m4]
```

Out[236]= 34.1808

Only a 34% difference. Not bad!

---

b.

For the distribution of means, I go through the same process that i did in pat a:

In[237]:=
```
counts1 = BinCounts[μ, {Min[μ], Max[μ] + .001, (Max[μ] - Min[μ] + .001)/7}]
```

Out[237]= {4, 9, 16, 11, 5, 4, 1}

In[238]:=
```
binSpacing1 = (Max[μ] - Min[μ] + .001)/7;
```

In[239]:=
```
totalArea1 = Sum[binSpacing1 * counts1[[n]], {n, 1, Length[counts1]}];
```

In[240]:=
```
normalCounts1 = counts1/totalArea1;
```

In[241]:=
```
xmids1 = Table[Min[μ] + binSpacing1/2 + binSpacing1 * n, {n, 0, Length[counts1] - 1}];
```

Third moment:

In[242]:=
```
m31 = binSpacing1 *
  Sum[(xmids1[[n]] - μbig)^3 * normalCounts1[[n]], {n, 1, Length[normalCounts1]}]
```

Out[242]= 3.43867×10^{-6}

Fourth moment:

In[243]:=
```
m41 = binSpacing1 *
  Sum[(xmids1[[n]] - μbig)^4 * normalCounts1[[n]], {n, 1, Length[normalCounts1]}]
```

Out[243]= 4.3936×10^{-7}

These moments are much closer to gaussian than in part a. The third moment is theoretically zero, and we can see that it is indeed very close to zero (two order of magnitudes lower than the variance of the means). So this moment is close to being Gaussian.

The fourth moment is theoretically (assuming a Gaussian):

In[244]:= $3 * \left(\sqrt{\text{Vofmeans}}\right)^4$

Out[244]= 5.01059×10^{-7}

And we see that the calculated value of 4.39×10^{-7} is very close to the theoretical Gaussian moment.

To quantitatively analyze the differences in the 4th moment, I will look at percent difference again:

In[245]:= $\text{pdiff}\left[3 * \left(\sqrt{\text{Vofmeans}}\right)^4, \text{m41}\right]$

Out[245]= 13.1215

Only a 13% difference. A much better fit to being Gaussian than in part a.

As the data in this distribution of means is a closer fit to a Gaussian (based on the 3rd and 4th moments) than the distribution of all data, these moments are somewhat consistent with the CLT. It tells us that as we look at a distribution of means, versus the raw data, our data gets closer to being normally distributed, which is the essence of the CLT. It is not a perfect fit to the CLT, as the moments do not perfectly match, but they are quite close. This is interesting, as I pointed out earlier that the CLT may not actually apply to my data, but it seems to be working for the most part regardless of the fact that the data is not IID.

C.

The third moment tells us all about the skew of our data (how non symmetrical), while the fourth moment tells us about kurtosis, or how “peaked” the data is. The kurtosis is really information about the tails of the data. It will tell us about if there are too many, or too few data points in the tails of the data, compared to a normal distribution.

In the code that follows, I just repeat the moment calculations I did in parts a and b, but I generalize it into functions, where you pass in the number of bins (q) and data set (n).

In[246]:= $\text{binsizeFunc}[q\_, n\_] := \frac{(\text{Max}[\text{data}[[n]]] - \text{Min}[\text{data}[[n]]] + .001)}{q};$

In[247]:= $\text{countsFunc}[q\_, n\_] := \text{BinCounts}\left[\text{data}[[n]], \left\{\text{Min}[\text{data}[[n]]], \text{Max}[\text{data}[[n]]] + .001, \frac{(\text{Max}[\text{data}[[n]]] - \text{Min}[\text{data}[[n]]] + .001)}{q}\right\}\right]$

In[248]:=
```
totalAreaFunc[q_, n_] :=
  Sum[binsizeFunc[q, n] * countsFunc[q, n][[i]], {i, 1, Length[countsFunc[q, n]]}];
```

In[249]:= $\text{normalcountsFunc}[q\_, n\_] := \frac{\text{countsFunc}[q, n]}{\text{totalAreaFunc}[q, n]}$

In[250]:= $\text{midbinsFunc}[q\_, n\_] := \text{Table}\left[\text{Min}[\text{data}[[n]]] + \frac{\text{binsizeFunc}[q, n]}{2} + \text{binsizeFunc}[q, n] * i, \{i, 0, \text{Length}[\text{countsFunc}[q, n]] - 1\}\right];$

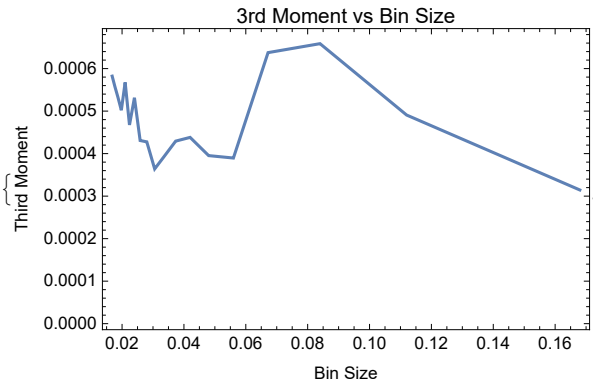
```
In[251]:= moment3[q_, n_] :=
            binsizeFunc[q, n] * Sum[(midbinsFunc[q, n][[i]] - μ[[i]])^3 * normalcountsFunc[q, n][[i]],
              {i, 1, Length[normalcountsFunc[q, n]]}];

In[252]:= moment4[q_, n_] :=
            binsizeFunc[q, n] * Sum[(midbinsFunc[q, n][[i]] - μ[[i]])^4 * normalcountsFunc[q, n][[i]],
              {i, 1, Length[normalcountsFunc[q, n]]}];
```

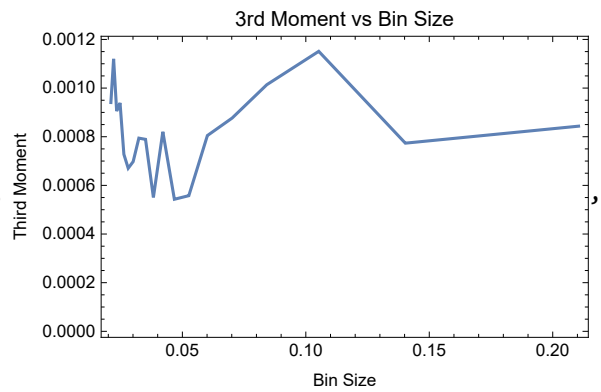
Below is the plot of the third moments as a function of bin size:

```
In[253]:= m3Plots = Table[ListLinePlot[Table[{binsizeFunc[q, n], moment3[q, n]}, {q, 2, 20}],
             Frame -> True, FrameLabel → {"Bin Size", "Third Moment"},
             PlotLabel → "3rd Moment vs Bin Size", PlotRange → All, ImageSize → 300], {n, 1, 50}]
```

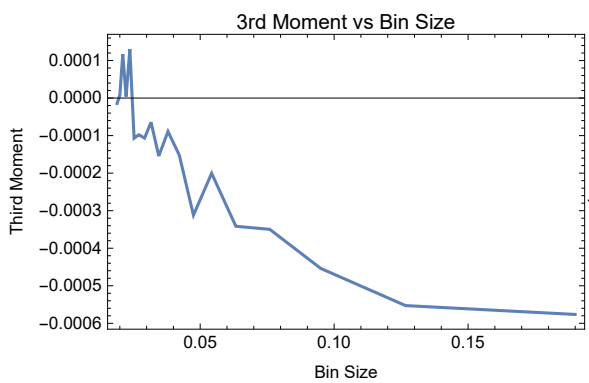
Out[253]= {



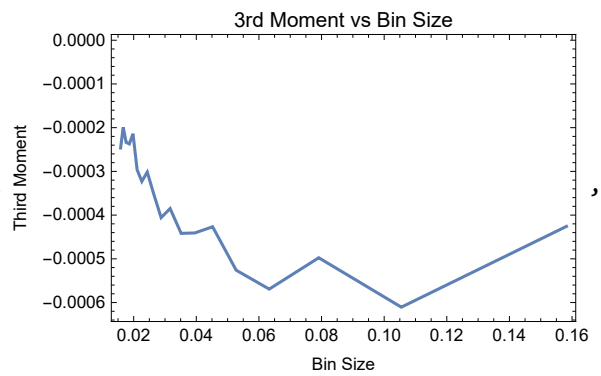
,



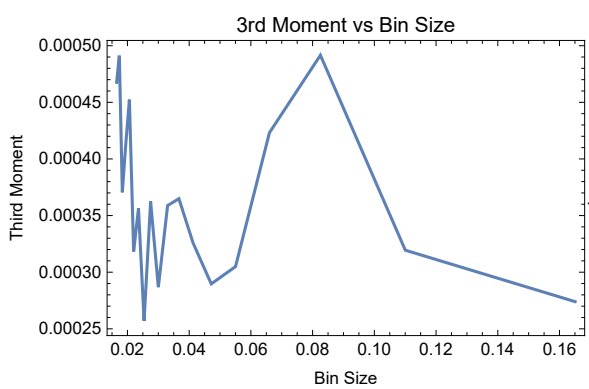
,



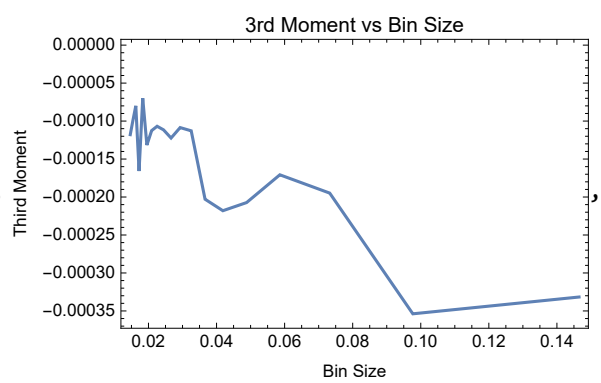
,



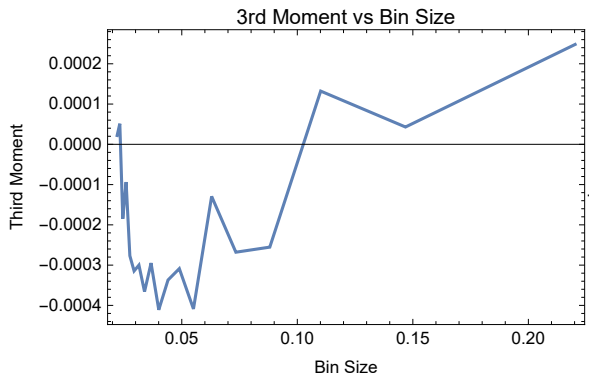
,



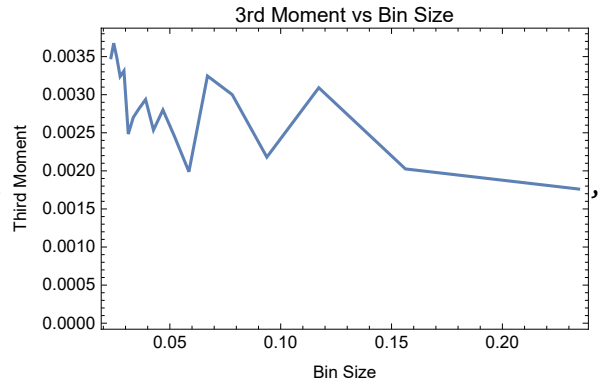
,



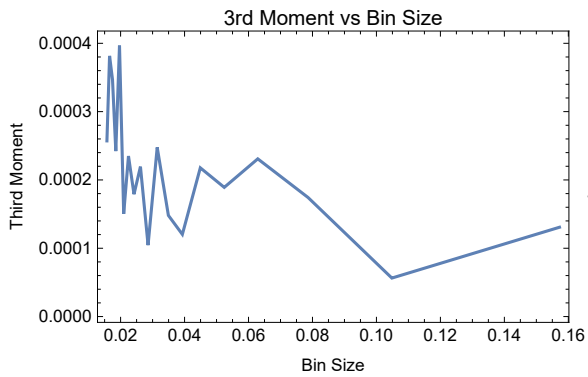
,


3rd Moment vs Bin Size
Third Moment
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
0.05
0.10
0.15
0.20
Bin Size

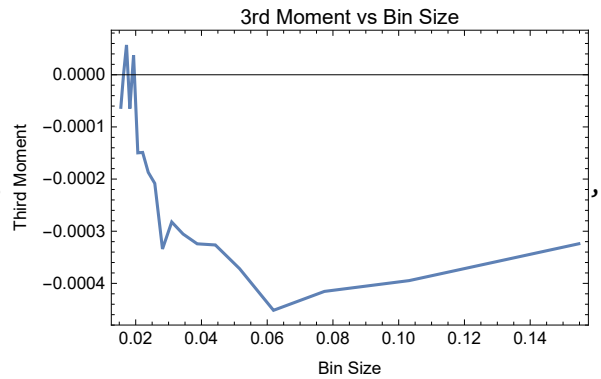
,


3rd Moment vs Bin Size
Third Moment
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0.05
0.10
0.15
0.20
Bin Size

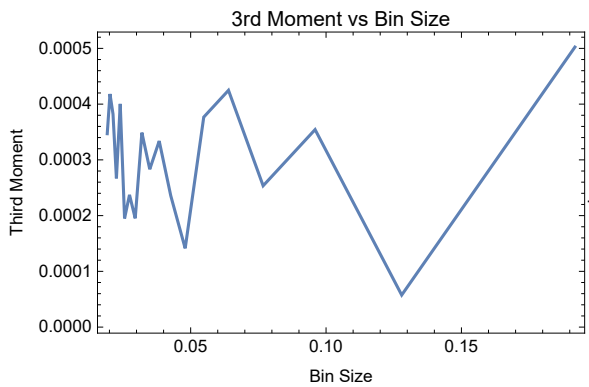
,


3rd Moment vs Bin Size
Third Moment
0.0004
0.0003
0.0002
0.0001
0.0000
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Bin Size

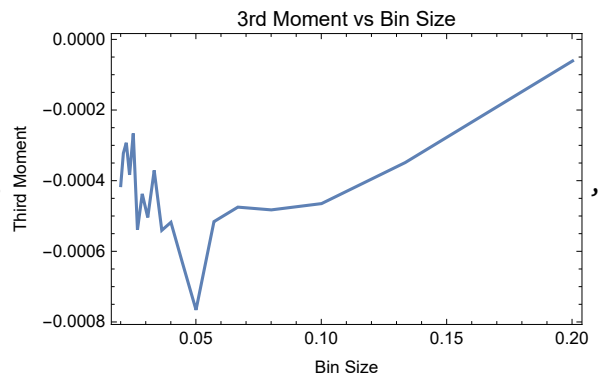
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Bin Size

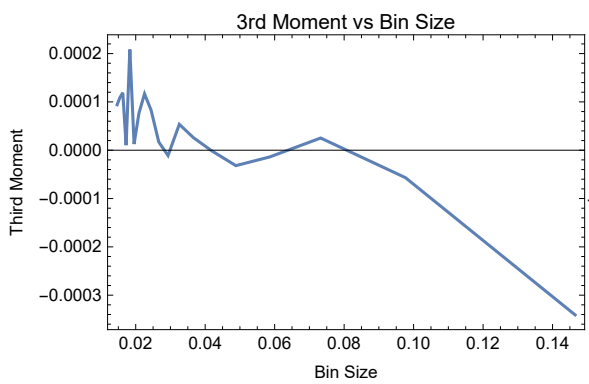
,


3rd Moment vs Bin Size
Third Moment
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
0.05
0.10
0.15
Bin Size

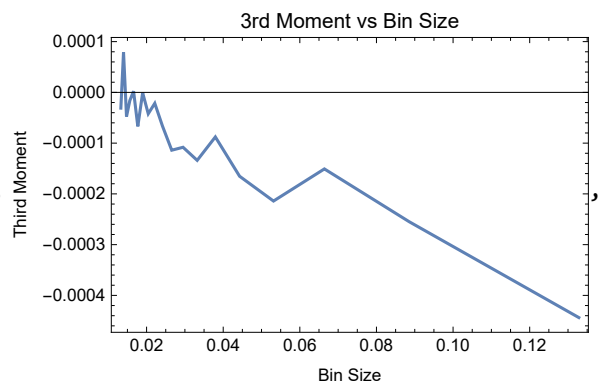
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
0.05
0.10
0.15
0.20
Bin Size

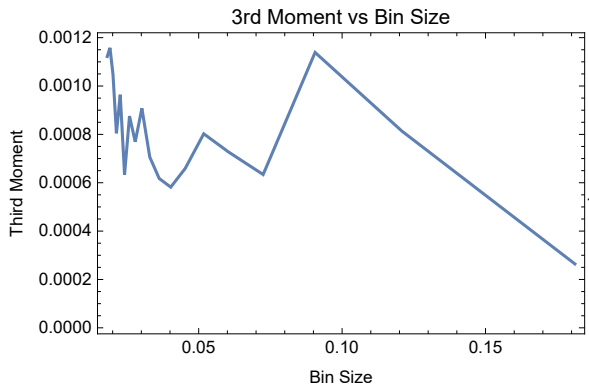
,


3rd Moment vs Bin Size
Third Moment
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Bin Size

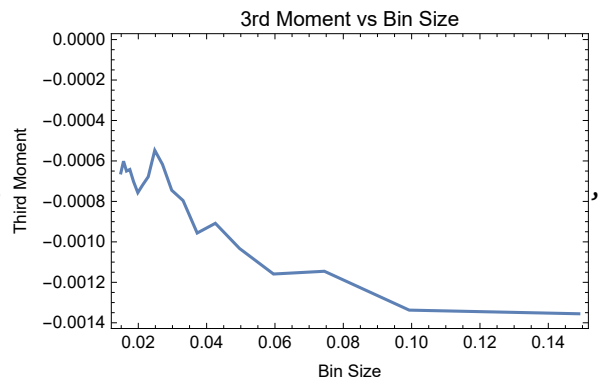
,


3rd Moment vs Bin Size
Third Moment
0.0001
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
0.02
0.04
0.06
0.08
0.10
0.12
Bin Size

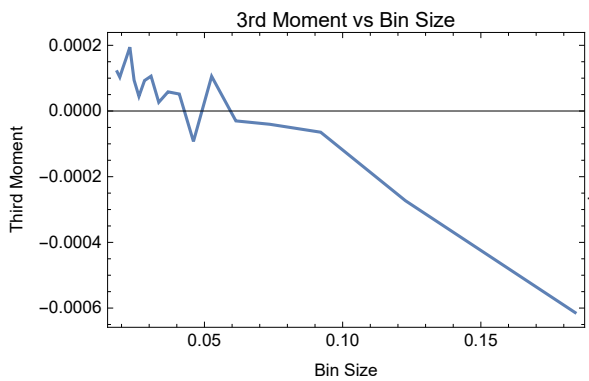
,


3rd Moment vs Bin Size
Third Moment
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0.05
0.10
0.15
Bin Size

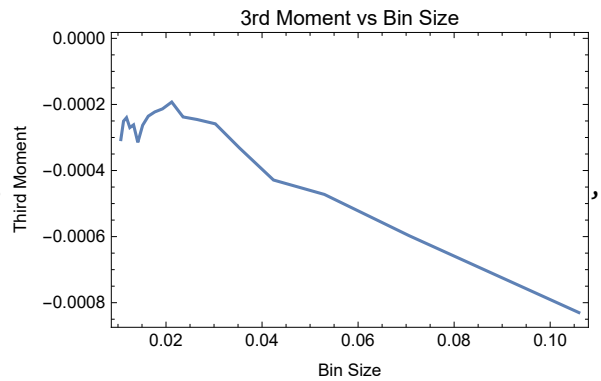
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
−0.0012
−0.0014
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Bin Size

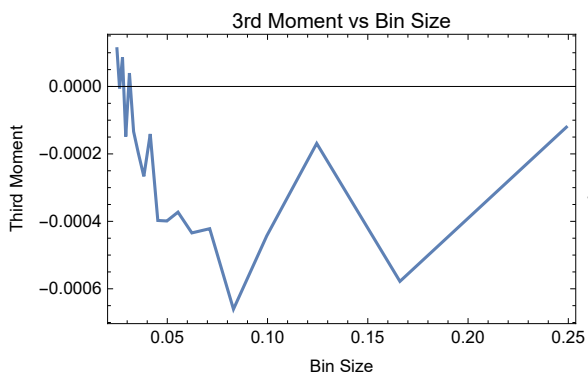
,


3rd Moment vs Bin Size
Third Moment
0.0002
0.0000
−0.0002
−0.0004
−0.0006
0.05
0.10
0.15
Bin Size

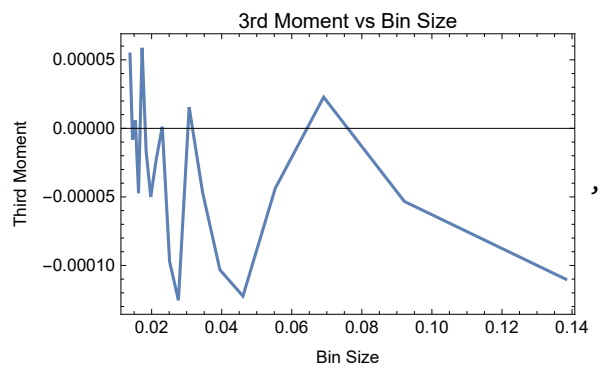
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
0.02
0.04
0.06
0.08
0.10
Bin Size

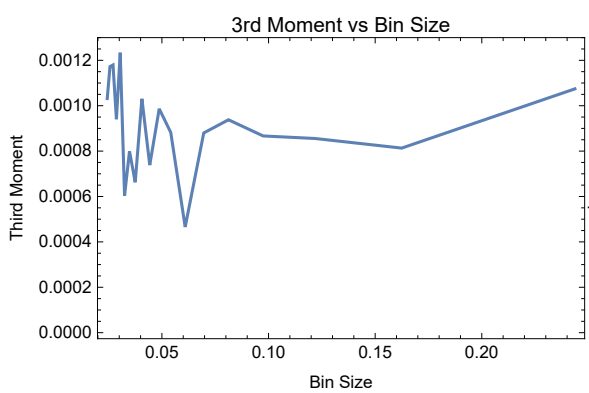
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0002
−0.0004
−0.0006
0.05
0.10
0.15
0.20
0.25
Bin Size

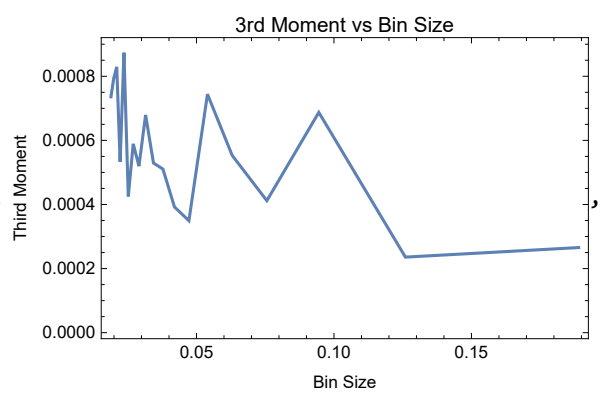
,


3rd Moment vs Bin Size
Third Moment
0.00005
0.00000
−0.00005
−0.00010
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Bin Size

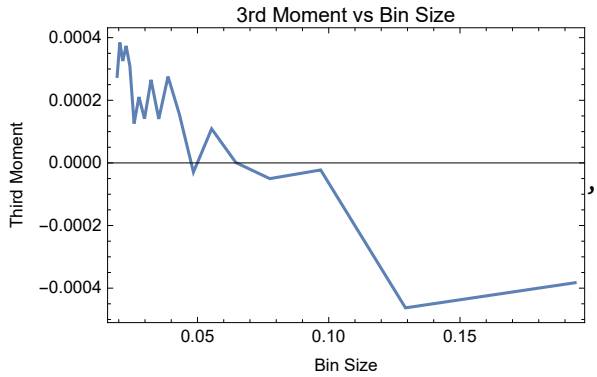
,


3rd Moment vs Bin Size
Third Moment
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0.05
0.10
0.15
0.20
Bin Size

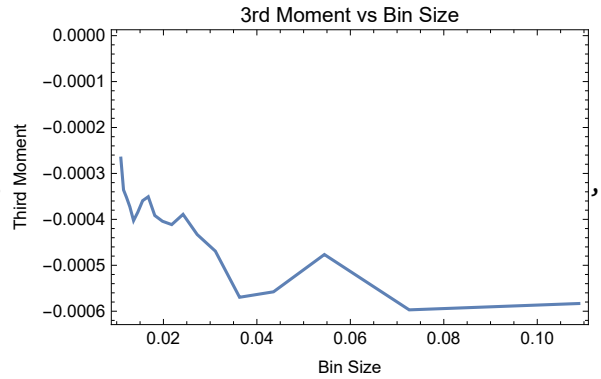
,


3rd Moment vs Bin Size
Third Moment
0.0008
0.0006
0.0004
0.0002
0.0000
0.05
0.10
0.15
Bin Size

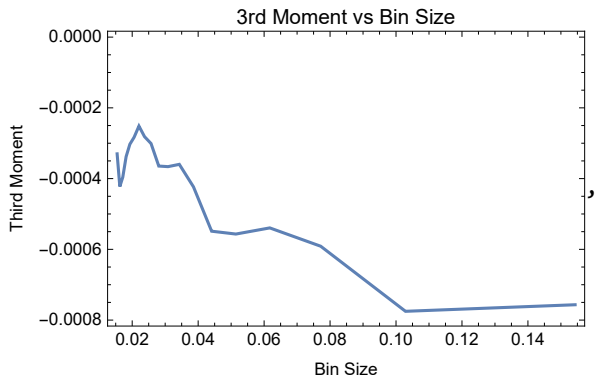
,


3rd Moment vs Bin Size
Third Moment
0.0004
0.0002
0.0000
−0.0002
−0.0004
0.05
0.10
0.15
Bin Size

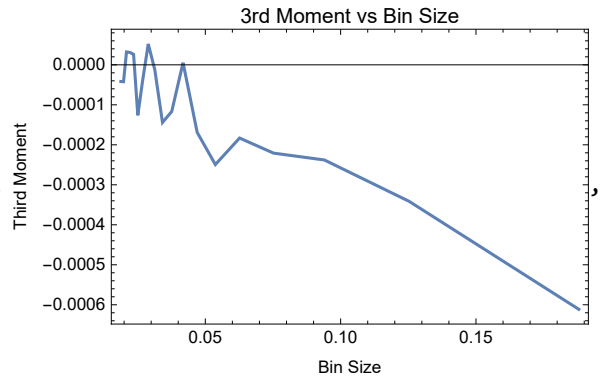
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
−0.0005
−0.0006
0.02
0.04
0.06
0.08
0.10
Bin Size

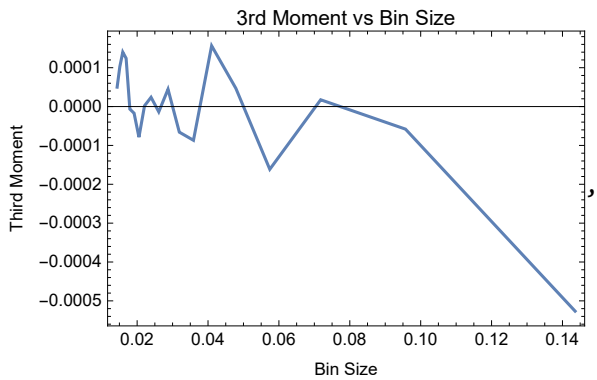
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Bin Size

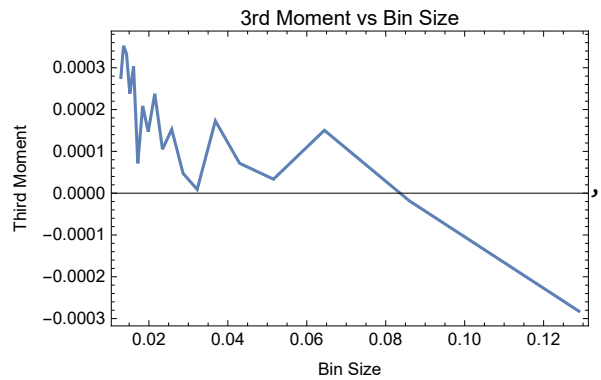
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
−0.0005
−0.0006
0.05
0.10
0.15
Bin Size

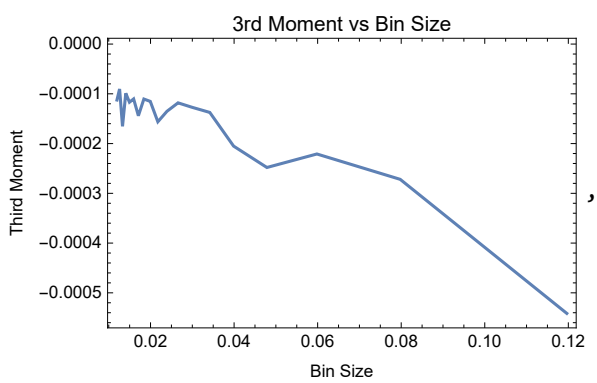
,


3rd Moment vs Bin Size
Third Moment
0.0001
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
−0.0005
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Bin Size

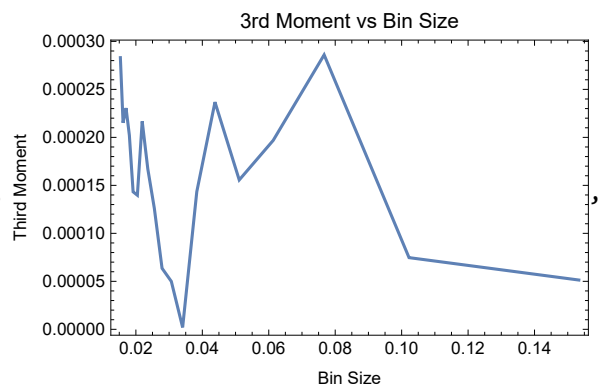
,


3rd Moment vs Bin Size
Third Moment
0.0003
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003
0.02
0.04
0.06
0.08
0.10
0.12
Bin Size

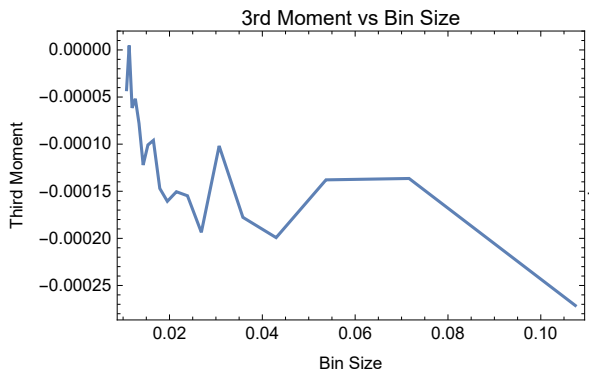
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
−0.0005
0.02
0.04
0.06
0.08
0.10
0.12
Bin Size

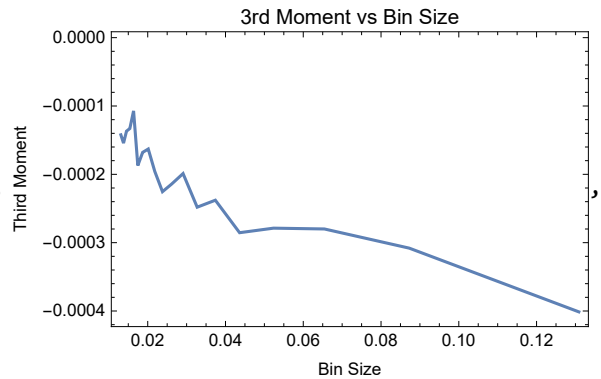
,


3rd Moment vs Bin Size
Third Moment
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Bin Size

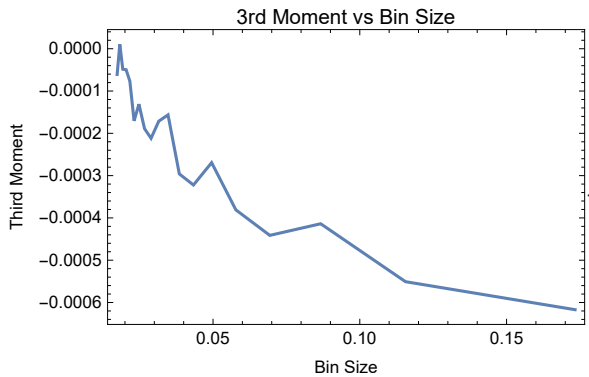
,


3rd Moment vs Bin Size
Third Moment
0.00000
−0.00005
−0.00010
−0.00015
−0.00020
−0.00025
0.02
0.04
0.06
0.08
0.10
Bin Size

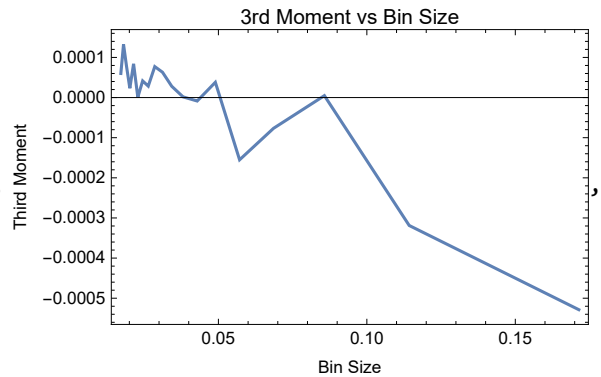
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
0.02
0.04
0.06
0.08
0.10
0.12
Bin Size

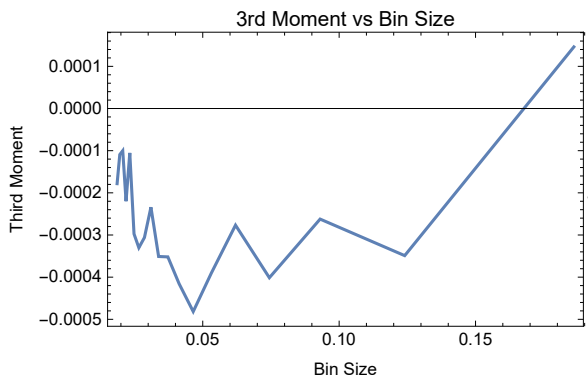
,


3rd Moment vs Bin Size
Third Moment
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
−0.0005
−0.0006
0.05
0.10
0.15
Bin Size

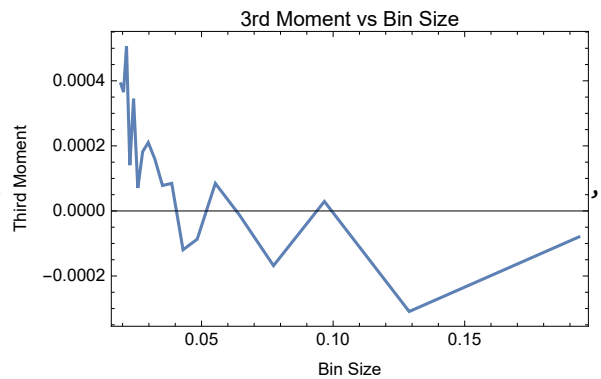
,


3rd Moment vs Bin Size
Third Moment
0.0001
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
−0.0005
0.05
0.10
0.15
Bin Size

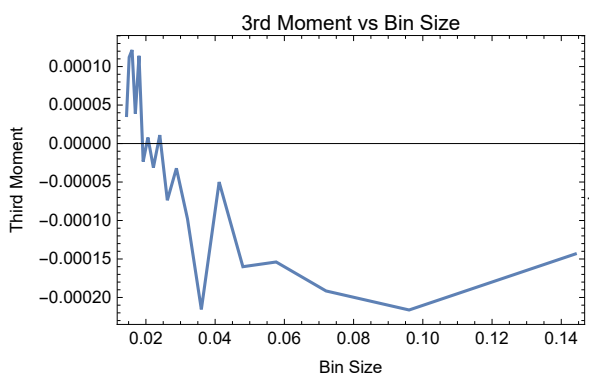
,


3rd Moment vs Bin Size
Third Moment
0.0001
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
−0.0005
0.05
0.10
0.15
Bin Size

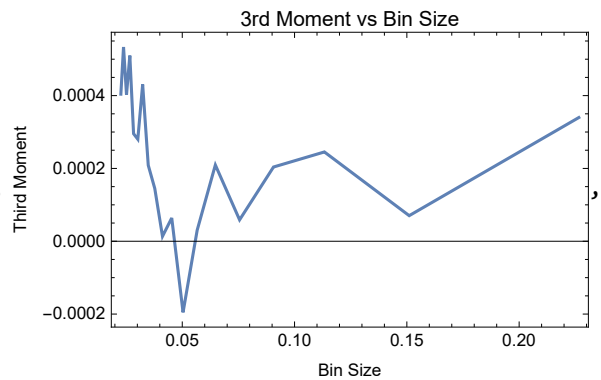
,


3rd Moment vs Bin Size
Third Moment
0.0004
0.0002
0.0000
−0.0002
0.05
0.10
0.15
Bin Size

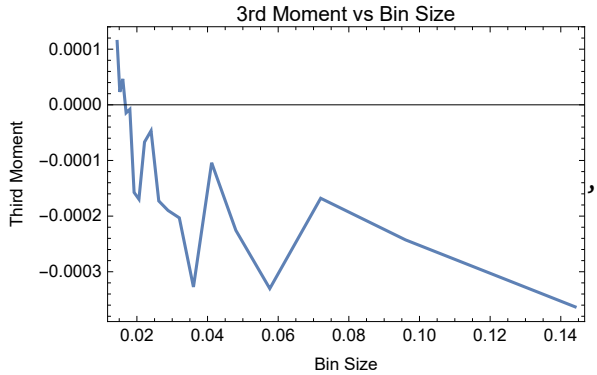
,


3rd Moment vs Bin Size
Third Moment
0.00010
0.00005
0.00000
−0.00005
−0.00010
−0.00015
−0.00020
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Bin Size

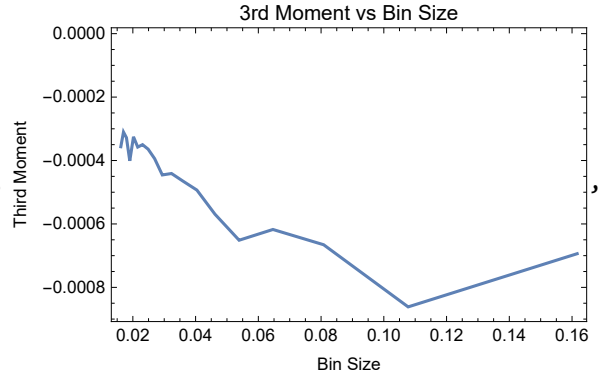
,


3rd Moment vs Bin Size
Third Moment
0.0004
0.0002
0.0000
−0.0002
0.05
0.10
0.15
0.20
Bin Size

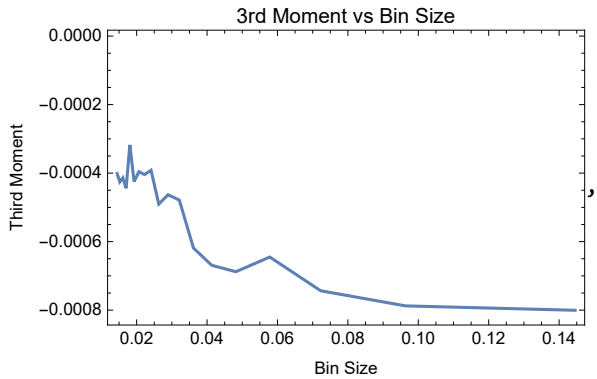
,


3rd Moment vs Bin Size
Third Moment
Bin Size
0.0001
0.0000
−0.0001
−0.0002
−0.0003
0.02
0.04
0.06
0.08
0.10
0.12
0.14

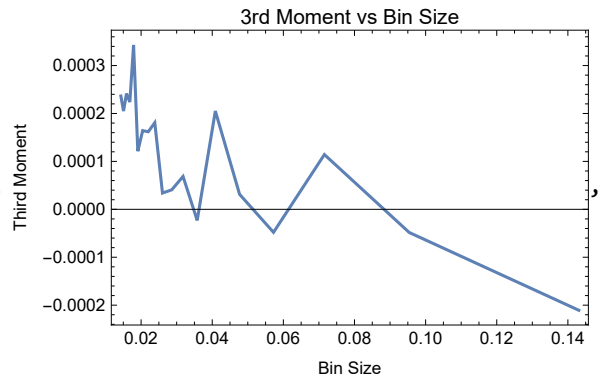
,


3rd Moment vs Bin Size
Third Moment
Bin Size
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16

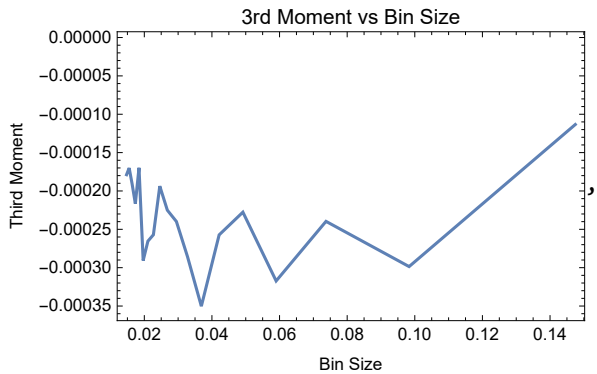
,


3rd Moment vs Bin Size
Third Moment
Bin Size
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
0.02
0.04
0.06
0.08
0.10
0.12
0.14

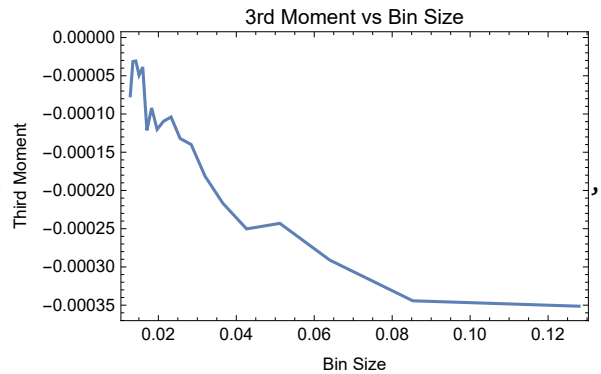
,


3rd Moment vs Bin Size
Third Moment
Bin Size
0.0003
0.0002
0.0001
0.0000
−0.0001
−0.0002
0.02
0.04
0.06
0.08
0.10
0.12
0.14

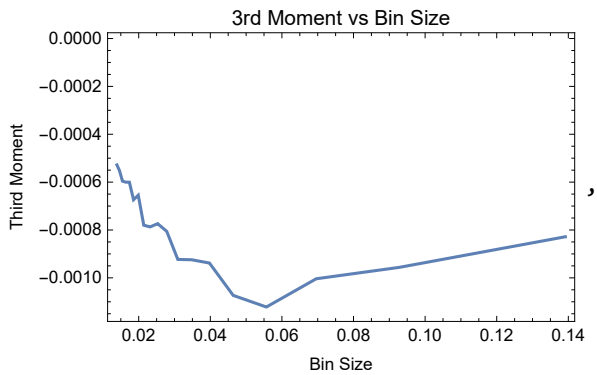
,


3rd Moment vs Bin Size
Third Moment
Bin Size
0.00000
−0.00005
−0.00010
−0.00015
−0.00020
−0.00025
−0.00030
−0.00035
0.02
0.04
0.06
0.08
0.10
0.12
0.14

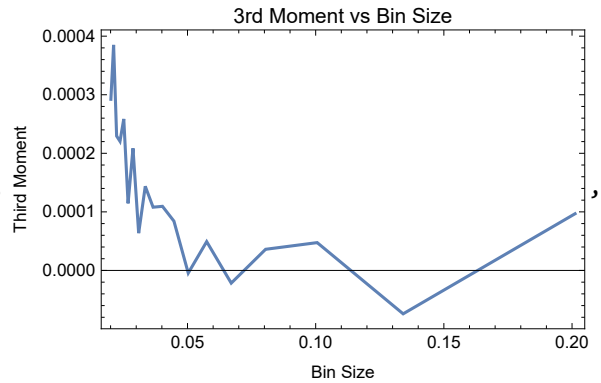
,


3rd Moment vs Bin Size
Third Moment
Bin Size
0.00000
−0.00005
−0.00010
−0.00015
−0.00020
−0.00025
−0.00030
−0.00035
0.02
0.04
0.06
0.08
0.10
0.12

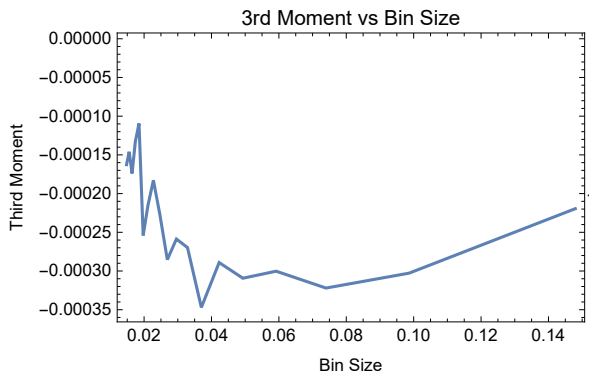
,


3rd Moment vs Bin Size
Third Moment
Bin Size
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
0.02
0.04
0.06
0.08
0.10
0.12
0.14

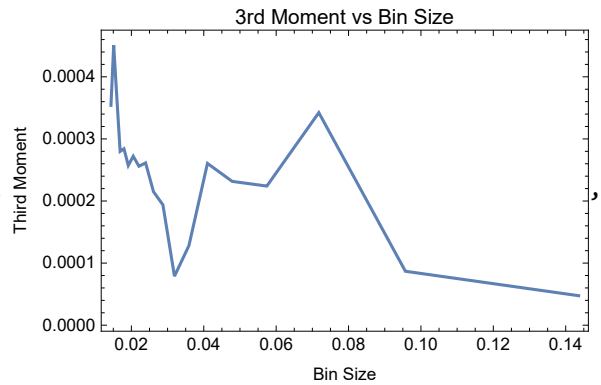
,


3rd Moment vs Bin Size
Third Moment
Bin Size
0.0004
0.0003
0.0002
0.0001
0.0000
0.05
0.10
0.15
0.20

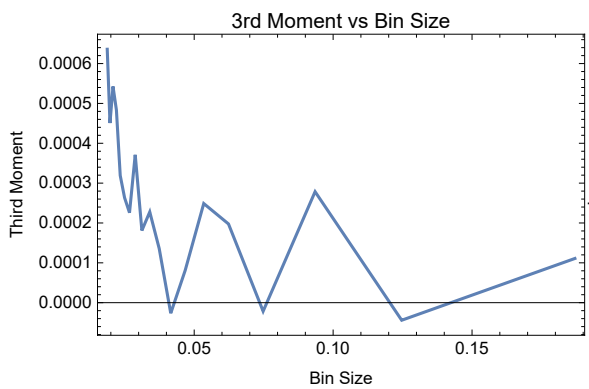
,



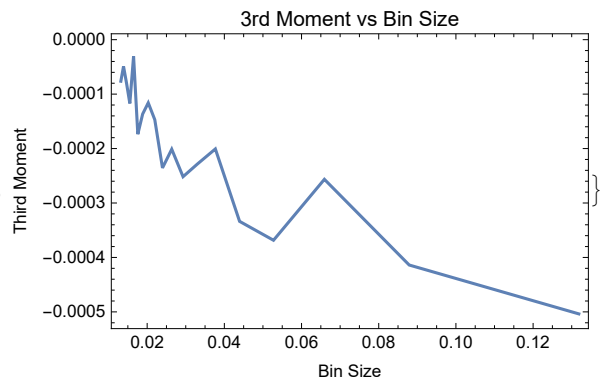
,



,



,



}

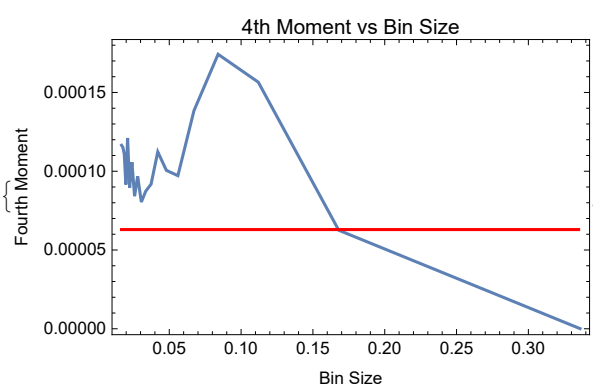
Below is the plot of the fourth moments as a function of bin size. Also included on each plot is a red line which represents the theoretical 4th moment of the data, if a Gaussian is assumed.

In[254]:=
```
m4Plots = Table[ListLinePlot[Table[{binsizeFunc[q, n], moment4[q, n]}, {q, 1, 20}],
    Frame -> True, FrameLabel → {"Bin Size", "Fourth Moment"},
    PlotLabel → "4th Moment vs Bin Size", PlotRange → All, ImageSize → 300], {n, 1, 50}];
```

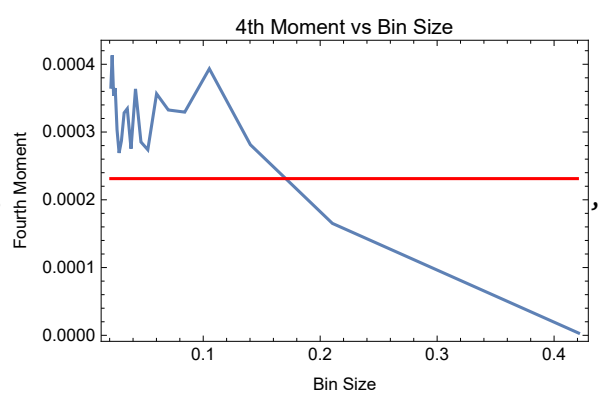
In[255]:=
```
m4predicts = Table[Plot[3 (√V[[n]])^4, {x, (Max[data[[n]]] - Min[data[[n]]])/20,
     (Max[data[[n]]] - Min[data[[n]]])/1}, PlotStyle → Red], {n, 1, 50}];
```

In[256]:=
```
all = Table[Show[m4Plots[[n]], m4predicts[[n]]], {n, 1, 50}]
```

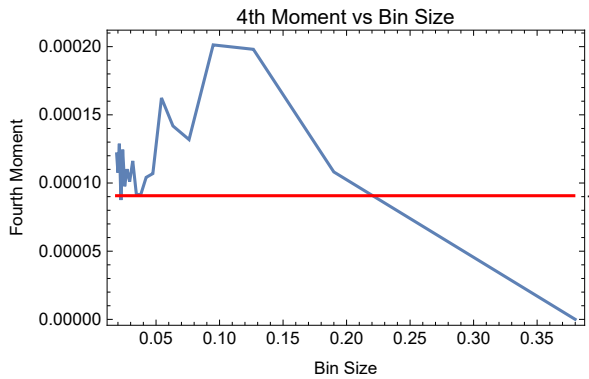
Out[256]= {



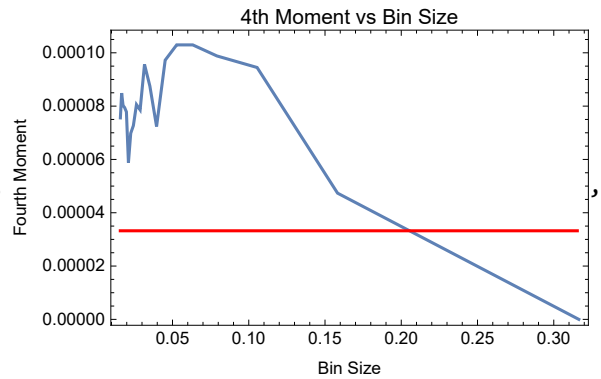
,



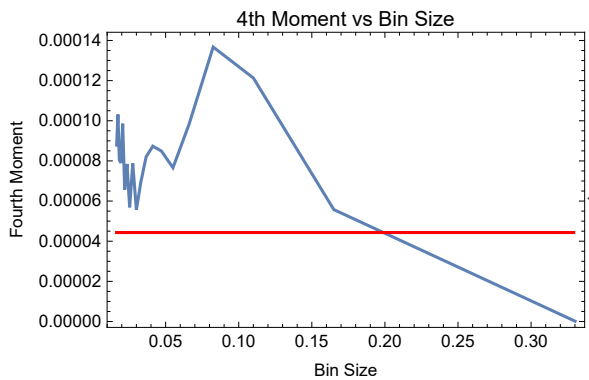
,


4th Moment vs Bin Size
Fourth Moment
0.00020
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Bin Size

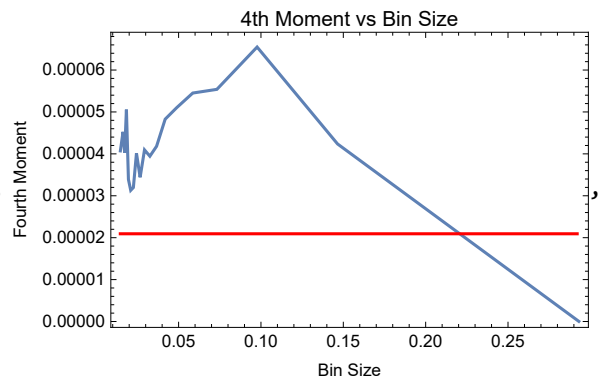
,


4th Moment vs Bin Size
Fourth Moment
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
Bin Size

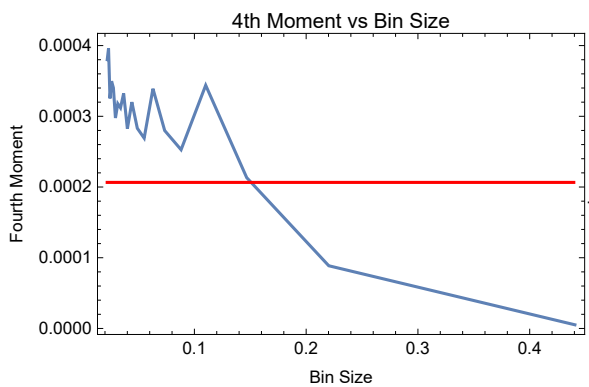
,


4th Moment vs Bin Size
Fourth Moment
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
Bin Size

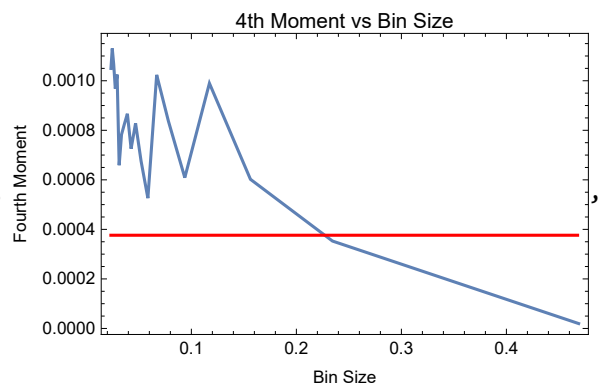
,


4th Moment vs Bin Size
Fourth Moment
0.00006
0.00005
0.00004
0.00003
0.00002
0.00001
0.00000
0.05
0.10
0.15
0.20
0.25
Bin Size

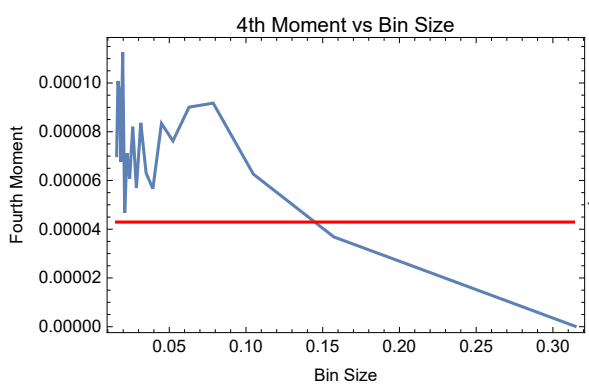
,


4th Moment vs Bin Size
Fourth Moment
0.0004
0.0003
0.0002
0.0001
0.0000
0.1
0.2
0.3
0.4
Bin Size

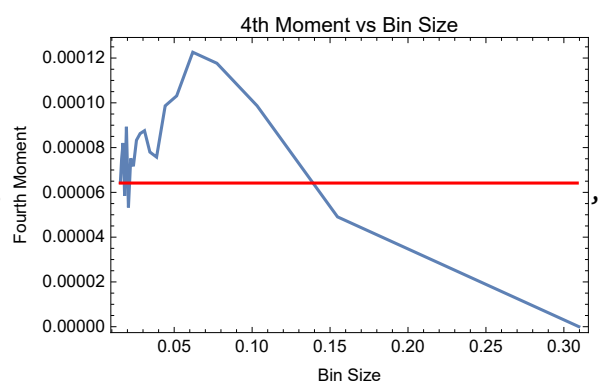
,


4th Moment vs Bin Size
Fourth Moment
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0.1
0.2
0.3
0.4
Bin Size

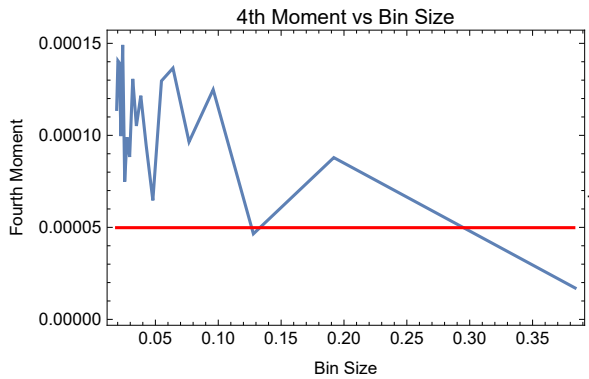
,


4th Moment vs Bin Size
Fourth Moment
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
Bin Size

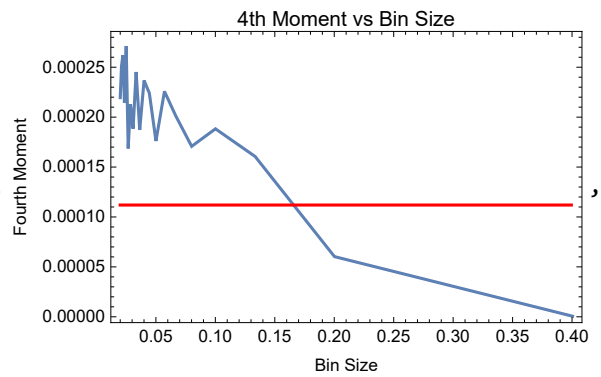
,


4th Moment vs Bin Size
Fourth Moment
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
Bin Size

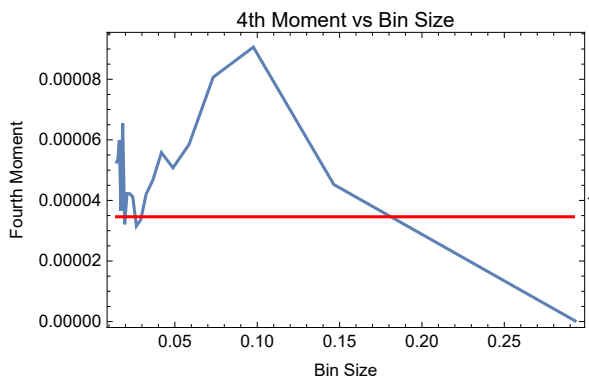
,


4th Moment vs Bin Size
Fourth Moment
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Bin Size

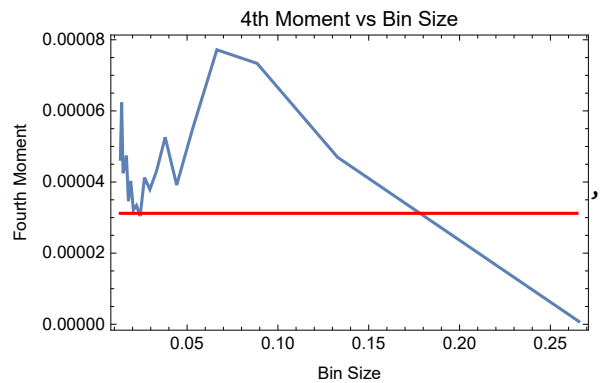
,


4th Moment vs Bin Size
Fourth Moment
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Bin Size

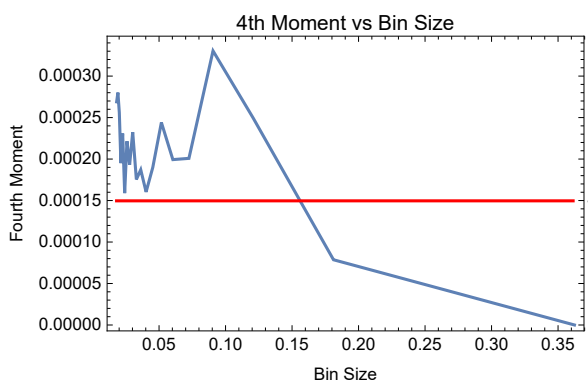
,


4th Moment vs Bin Size
Fourth Moment
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
Bin Size

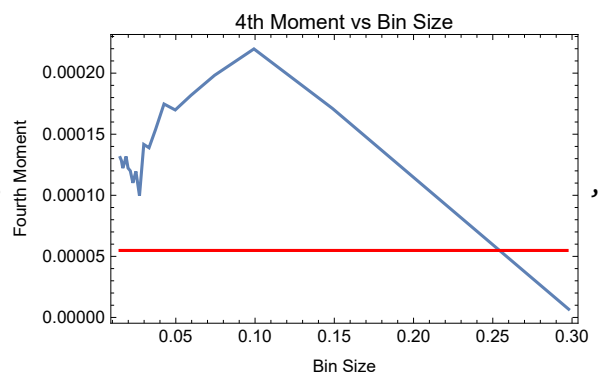
,


4th Moment vs Bin Size
Fourth Moment
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
Bin Size

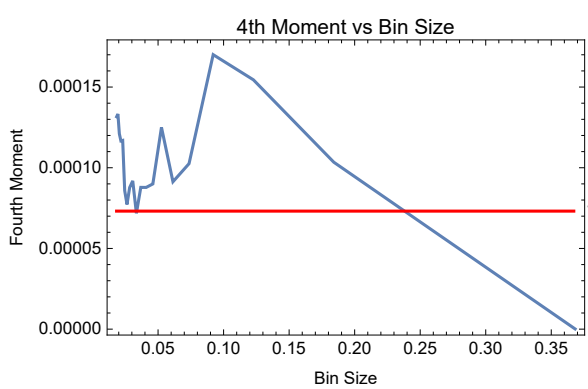
,


4th Moment vs Bin Size
Fourth Moment
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Bin Size

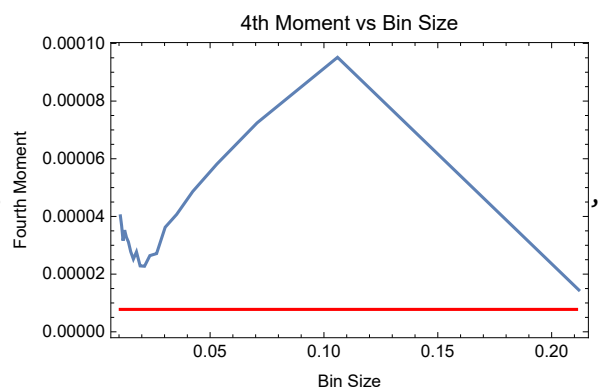
,


4th Moment vs Bin Size
Fourth Moment
0.00020
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
Bin Size

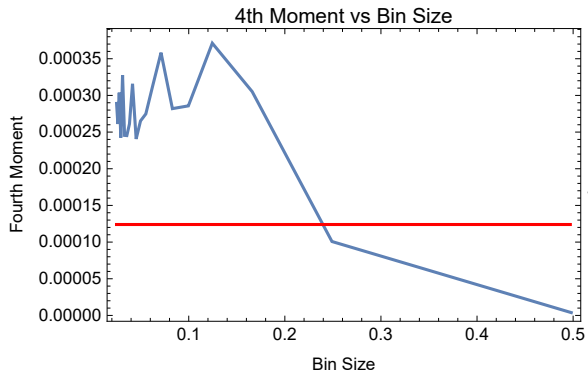
,


4th Moment vs Bin Size
Fourth Moment
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Bin Size

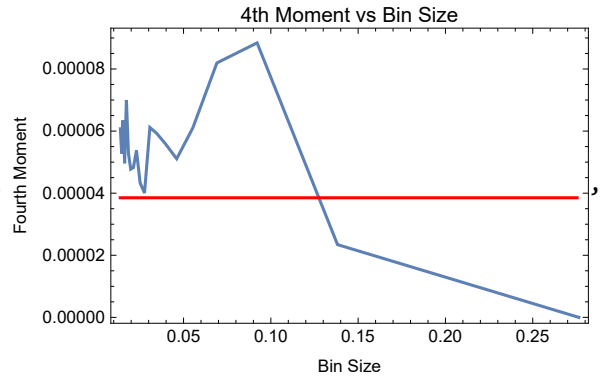
,


4th Moment vs Bin Size
Fourth Moment
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
Bin Size

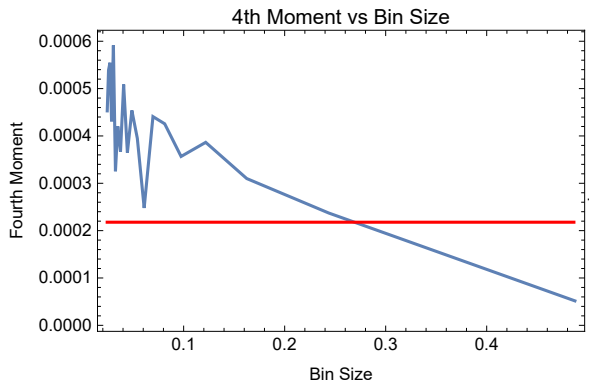
,


4th Moment vs Bin Size
Fourth Moment
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0.1
0.2
0.3
0.4
0.5
Bin Size

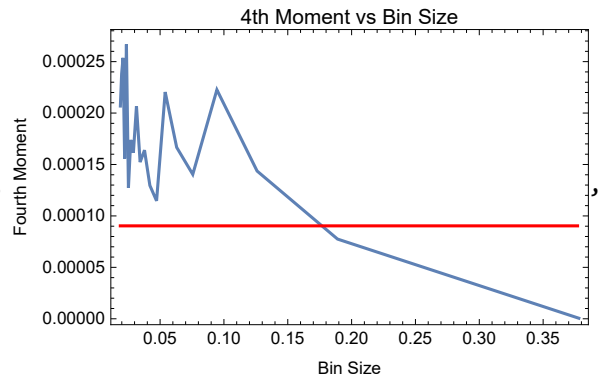
,


4th Moment vs Bin Size
Fourth Moment
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
Bin Size

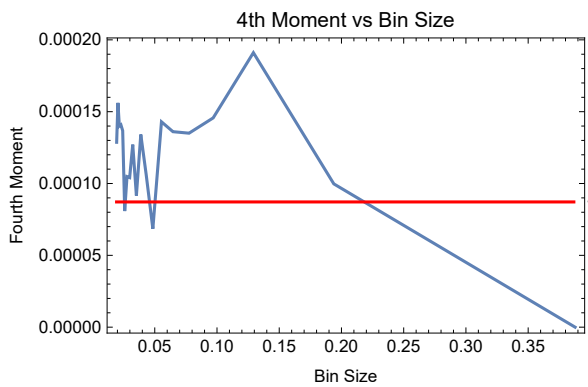
,


4th Moment vs Bin Size
Fourth Moment
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
0.1
0.2
0.3
0.4
Bin Size

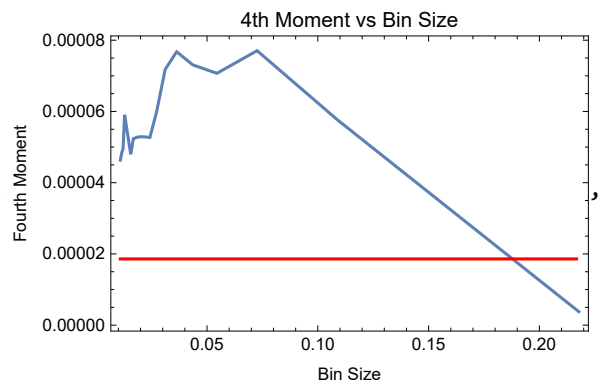
,


4th Moment vs Bin Size
Fourth Moment
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Bin Size

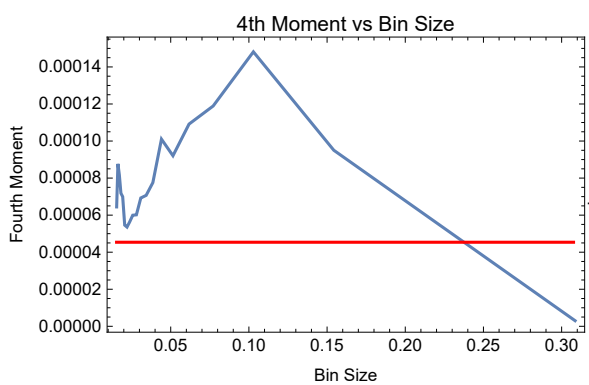
,


4th Moment vs Bin Size
Fourth Moment
0.00020
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Bin Size

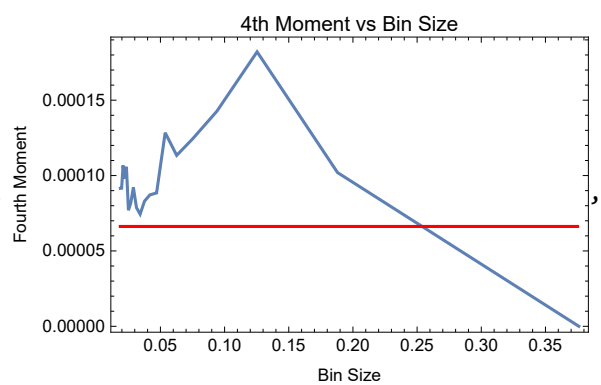
,

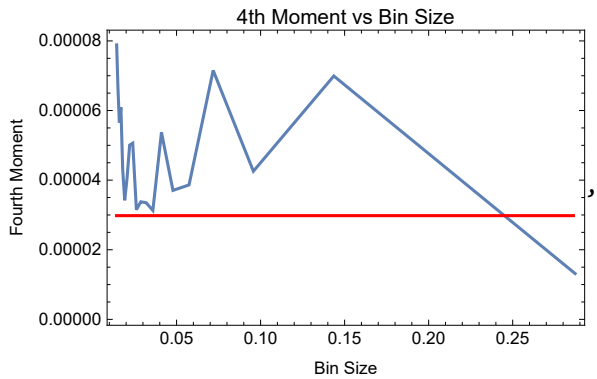

4th Moment vs Bin Size
Fourth Moment
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
Bin Size

,

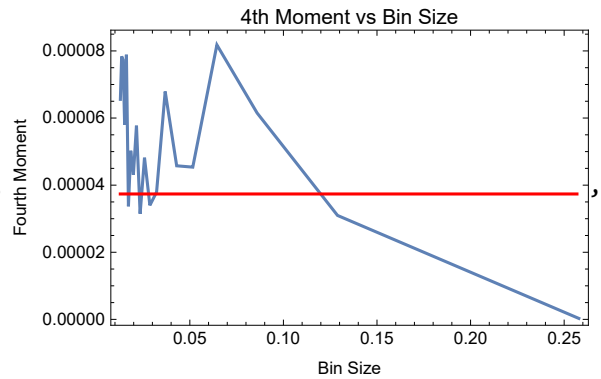

4th Moment vs Bin Size
Fourth Moment
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
Bin Size

,

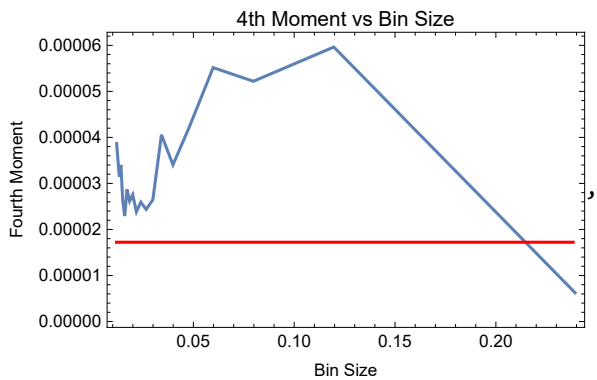

4th Moment vs Bin Size
Fourth Moment
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Bin Size


4th Moment vs Bin Size
Fourth Moment
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
Bin Size

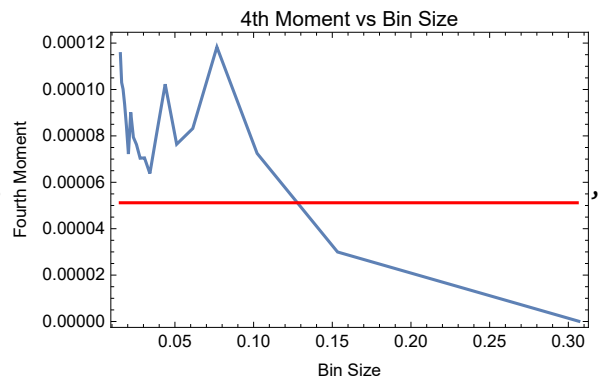
,


4th Moment vs Bin Size
Fourth Moment
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
Bin Size

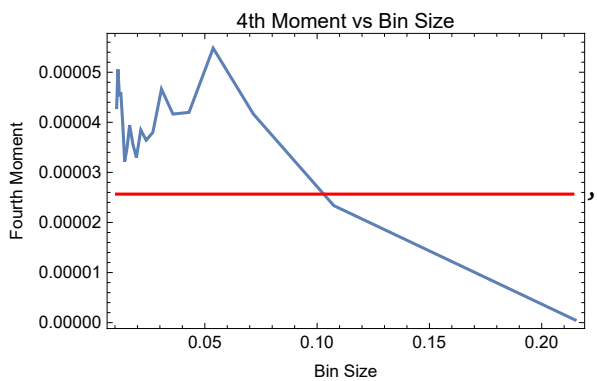
,


4th Moment vs Bin Size
Fourth Moment
0.00006
0.00005
0.00004
0.00003
0.00002
0.00001
0.00000
0.05
0.10
0.15
0.20
Bin Size

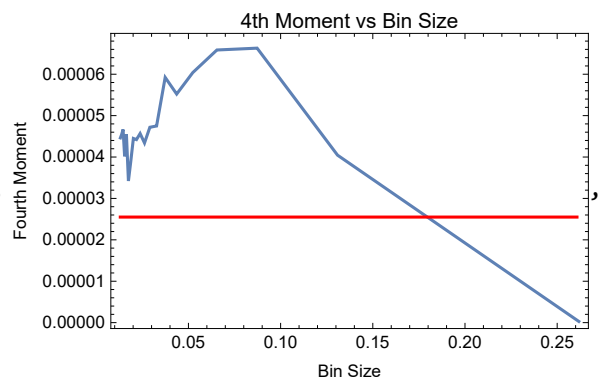
,


4th Moment vs Bin Size
Fourth Moment
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
Bin Size

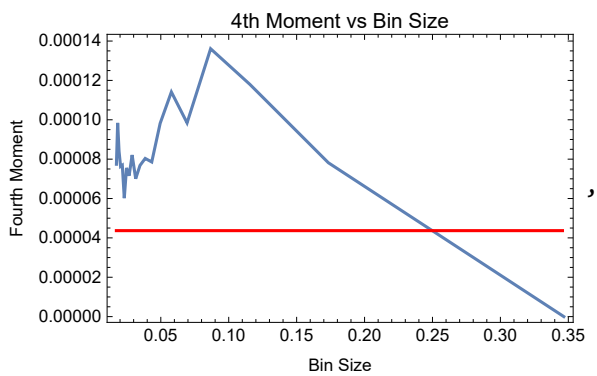
,


4th Moment vs Bin Size
Fourth Moment
0.00005
0.00004
0.00003
0.00002
0.00001
0.00000
0.05
0.10
0.15
0.20
Bin Size

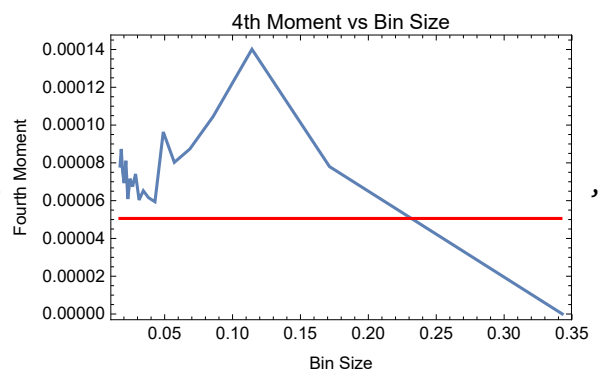
,


4th Moment vs Bin Size
Fourth Moment
0.00006
0.00005
0.00004
0.00003
0.00002
0.00001
0.00000
0.05
0.10
0.15
0.20
0.25
Bin Size

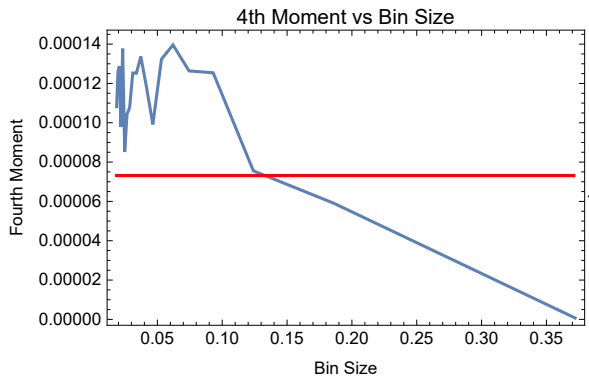
,


4th Moment vs Bin Size
Fourth Moment
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Bin Size

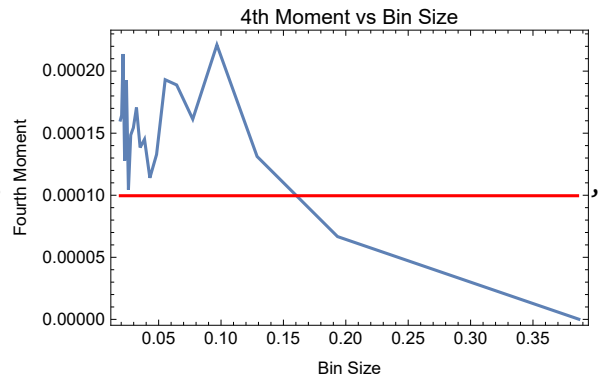
,


4th Moment vs Bin Size
Fourth Moment
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Bin Size

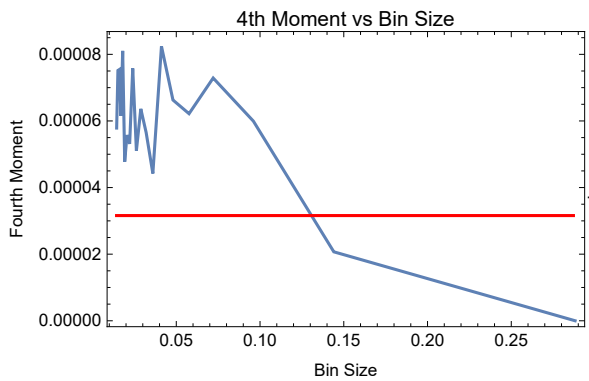
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35

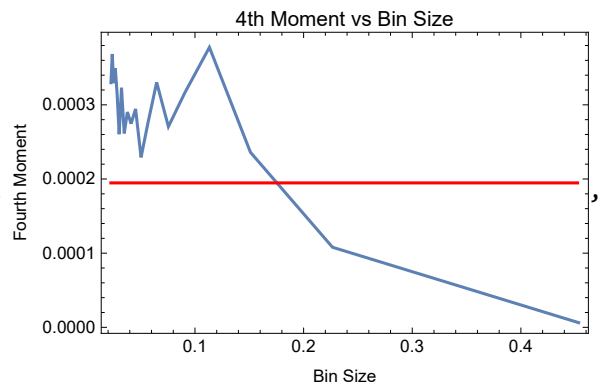
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00020
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25
0.30
0.35

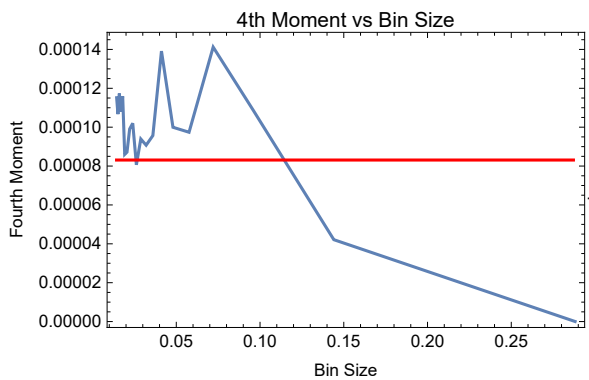
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25

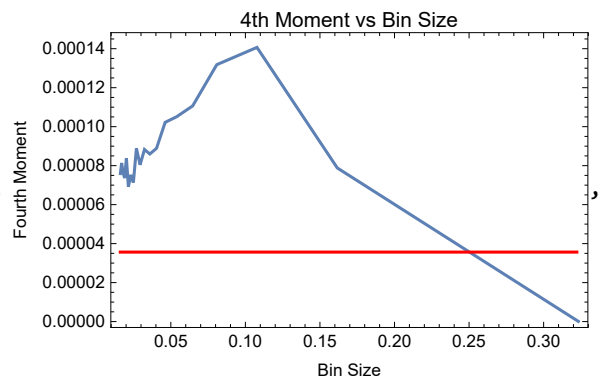
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.0003
0.0002
0.0001
0.0000
0.1
0.2
0.3
0.4

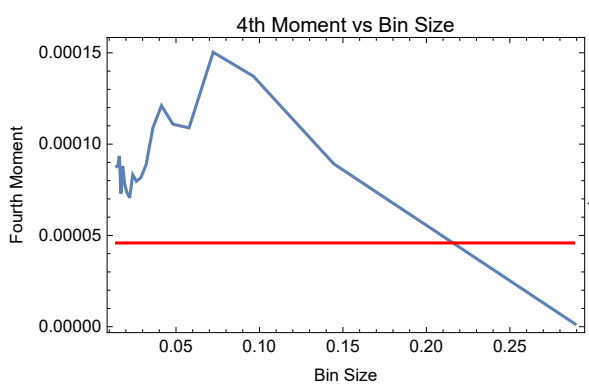
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25

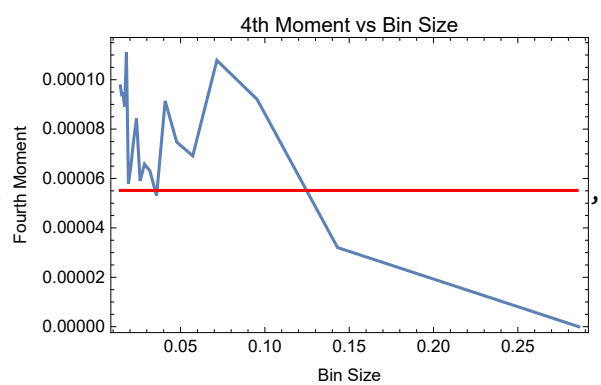
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25
0.30

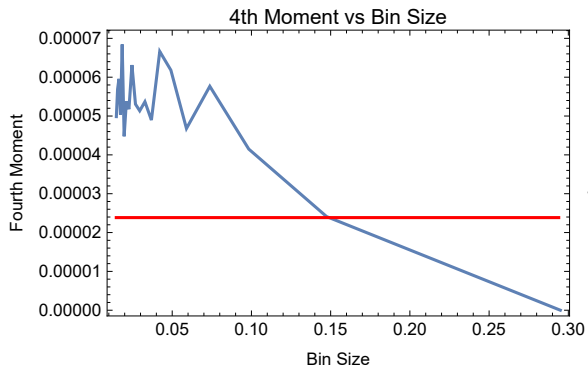
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00015
0.00010
0.00005
0.00000
0.05
0.10
0.15
0.20
0.25

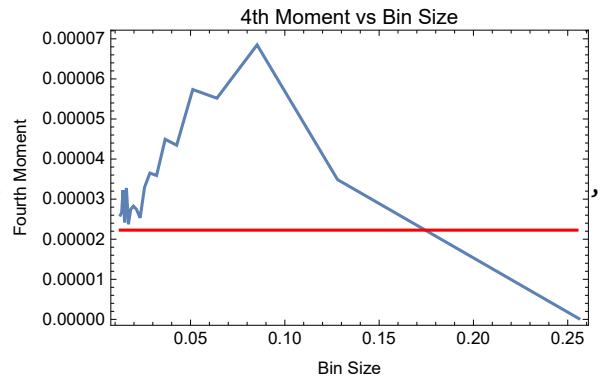
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.05
0.10
0.15
0.20
0.25

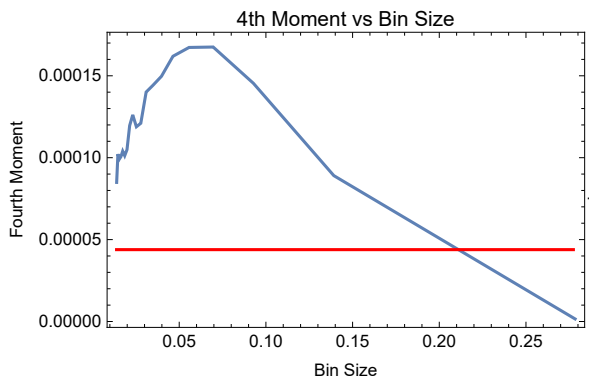
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00000
0.00001
0.00002
0.00003
0.00004
0.00005
0.00006
0.00007
0.05
0.10
0.15
0.20
0.25
0.30

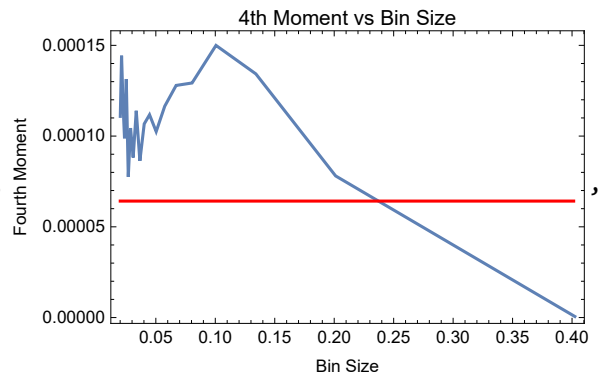
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00000
0.00001
0.00002
0.00003
0.00004
0.00005
0.00006
0.00007
0.05
0.10
0.15
0.20
0.25

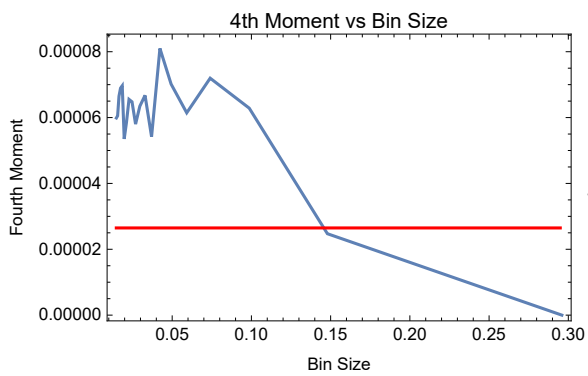
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00000
0.00005
0.00010
0.00015
0.05
0.10
0.15
0.20
0.25

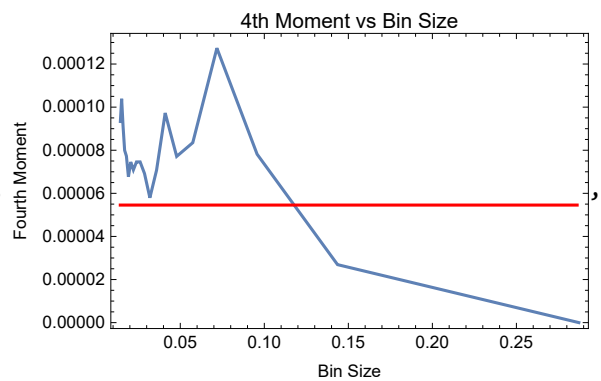
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00000
0.00005
0.00010
0.00015
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40

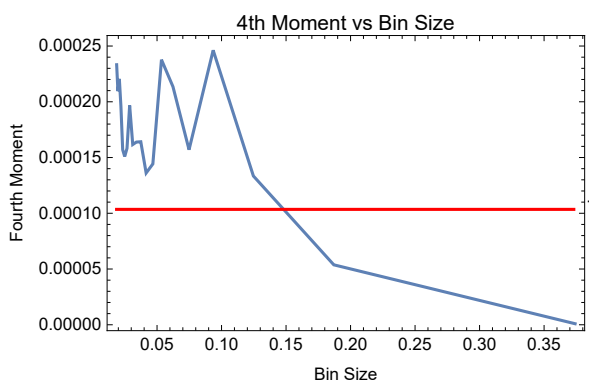
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00000
0.00002
0.00004
0.00006
0.00008
0.05
0.10
0.15
0.20
0.25
0.30

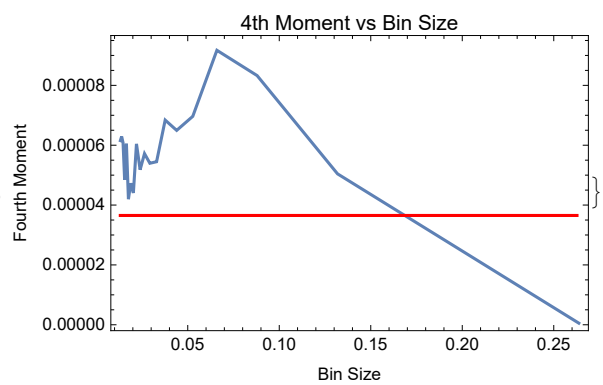
,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
0.00012
0.05
0.10
0.15
0.20
0.25

,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.05
0.10
0.15
0.20
0.25
0.30
0.35

,


4th Moment vs Bin Size
Fourth Moment
Bin Size
0.00000
0.00002
0.00004
0.00006
0.00008
0.05
0.10
0.15
0.20
0.25

}

Notice that my plots look jagged becasue of how I defined my moment calculations. How I wrote my code, I input number of bins, not bin size, so my bin sizes are discrete, and linked to number of bins that I tell the code to make. So I looked at the moments for 1-20 bins for each data set, and then calculated the bin size for each of these number of bins, and plotted the moments vs those bin sizes.

So what conclusions do I make from these plots? First off, I like to look at the 4th moments, as I feel that the 3rd moment graphs do not hold a whole lot of significance. Why? The 3rd moment of my data describes the skew, or how non-symmetrical it is. This varies fairly wildly as a function of bin size for my data. However, the 4th moment, or the kurtosis, is quite a meaningful way to look at how the binning affects the tails of the distributions. The moments of the data only seem to be meaningful for bin sizes < 0.10. This is when most of the 4th moments of my data begin to fall off (indicating very fat tails, which is a result of too big of bin size). This is a big generalization for all of my data sets, but for the most part, it hold true.

The fourth moment is the moment that tells us about the tails of the data. The fourth moment is called the kurtosis. A high kurtosis means that there are “thin tails”, and a low kurtosis means that there are “fat tails ” to the data. We can see from the 4th moment plots above that as the bin size increases, the kurtosis falls (for the most part), especially once the bin sizes get very large. This means that as the bin size gets larger, the tails of the data get fatter. This makes sense, as more data is clumped into fewer bins, bins grow in height, and the tails of the data appear to be bigger. As the bin size decreases, we see a lot of variation of 4th moment, but it seems to be varying around a constant value. This indicates that the bin sizes are meaningful, as it is telling us that the tails are about the same for each bin grouping. If I were to make bin spacing even smaller (which I didn’t do, as my computer can’t handle many more points), we would likely see the 4th moment rise, as data would be spread out into too many bins, and the tails would be tiny.

I conclude that the tails of each of my data sets are not perfectly Gaussian. This stems from looking at the red lines plotted above. These are what the 4th moments of each data set would be, if we assume a Gaussian. I see a systematically larger value for 4th moment than what the Gaussian predicts. The calculated value only drops to the predicted value when the bin size gets so big that the calculation is meaningless. The moments are not all that much larger, but it is evident. This means that my tails are slightly thinner than a Gaussian’s, as higher 4th moment corresponds to thinner tails. This is somewhat of a visual approach, but we did a lot of analytical calculations to produce each plot. My final conclusion is that the tails to each of my individual data sets are thinner than a Gaussian, by a little bit.

d.

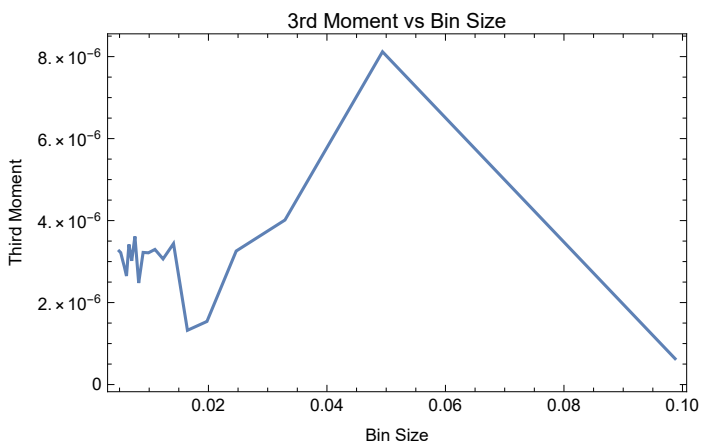
Here, I go through the same process as I did in part c, above, but for the data set of my means.

In[257]:=
```
binsizeMeansFunc[q_] := (Max[μ] - Min[μ] + .001)/q;
countsMeansFunc[q_] := BinCounts[μ, {Min[μ], Max[μ] + .001, (Max[μ] - Min[μ] + .001)/q}];
totalAreaMeansFunc[q_] :=
  Sum[binsizeMeansFunc[q] * countsMeansFunc[q][[i]], {i, 1, Length[countsMeansFunc[q]]}];
normalcountsMeansFunc[q_] := countsMeansFunc[q]/totalAreaMeansFunc[q];
midbinsMeansFunc[q_] := Table[Min[μ] + binsizeMeansFunc[q]/2 + binsizeMeansFunc[q] * i,
   {i, 0, Length[countsMeansFunc[q]] - 1}];
momentMeans3[q_] := binsizeMeansFunc[q] * Sum[(midbinsMeansFunc[q][[i]] - μbig)^3 *
     normalcountsMeansFunc[q][[i]], {i, 1, Length[normalcountsMeansFunc[q]]}];
momentMeans4[q_] := binsizeMeansFunc[q] * Sum[(midbinsMeansFunc[q][[i]] - μbig)^4 *
     normalcountsMeansFunc[q][[i]], {i, 1, Length[normalcountsMeansFunc[q]]}];
```

Below is a plot of the third moments as a function of bin size.

In[264]:=
```
m3MeansPlot = ListLinePlot[Table[{binsizeMeansFunc[q], momentMeans3[q]}, {q, 1, 20}],
  Frame -> True, FrameLabel → {"Bin Size", "Third Moment"},
  PlotLabel → "3rd Moment vs Bin Size", PlotRange → All]
```

Out[264]=



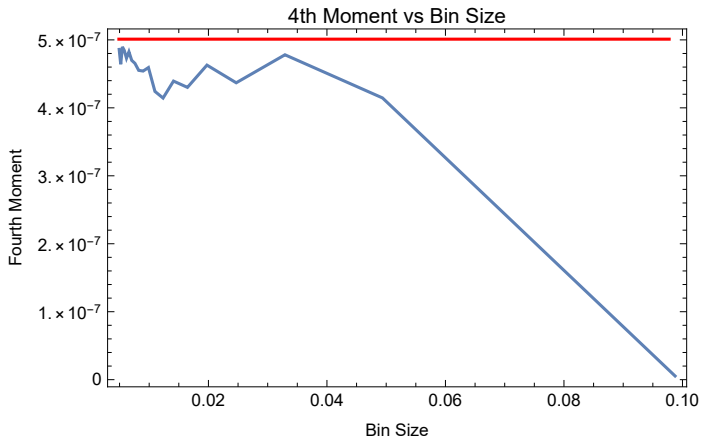
So we see that the third moments fall somewhat close to zero for the bin sizes where an analysis is actually useful. The theoretical 3rd moment, assuming a Gaussian, is zero. Our values fall around 3×10^{-6}. There is a systematic error that puts the 3rd moment slightly above zero.

In[265]:=
```
m4MeansPlot = ListLinePlot[Table[{binsizeMeansFunc[q], momentMeans4[q]}, {q, 1, 20}],
   Frame -> True, FrameLabel → {"Bin Size", "Fourth Moment"},
   PlotLabel → "4th Moment vs Bin Size", PlotRange → All];
m4MeansPredicted = Plot[3 (Sqrt[Vofmeans])^4,
   {x, (Max[μ] - Min[μ])/20, (Max[μ] - Min[μ])/1}, PlotStyle → Red];
```

Below is a plot of the 4th moment as a function of bin size, along with a red line to show what the predicted (assuming a Gaussian) 4th moment is.

In[267]:=
```
Show[m4MeansPlot, m4MeansPredicted]
```

Out[267]=



So, to do the same analysis on the set of means, we look at the above graph. We see that the actual 4th moment is systematically below the Gaussian 4th moment, but only by a very small amount. The bin sizes look to be meaningful for bin size < 0.04 or so. After this, the bin size gets too large, and the kurtosis drops, indicating once again that all of the data is being forced into a few bins, causing very fat tails.

In the range where bin size is useful to look at, we see a 4th moment slightly lower than the Gaussian predicts. This points to a slightly fat-tailed distribution. So I come to the conclusion that my distribution of means is indeed slightly fat-tailed.

Based on this plot, and specifically the systematic error that I see in the 4th and 3rd moments, I come to the conclusion that my data is close to following the CLT, but does not follow it perfectly. This small systematic error is shown in the plots above. it is obvious that the moments are close to being Gaussian, but are not exactly there. The fact that for each of the different bin sizes, the moments are above (for the third moment), or below (for the fourth moment), the Gaussian prediction leads me to believe that the CLT is not perfectly working, with certainty. Once again, however, it is interesting to me that the CLT appears to be close to working, even though my data is not IID.

---

e.

Below is a set of nested for loops that searches each set of data for the index of the data point that is the furthest away from the mean. This is what I classify as the worst data point in each set.

In[268]:=
```
worstPointIndexArray = Table[1, {n, 1, 50}];
```

In[269]:=
```
For[j = 1, j ≤ 50, j++,
 For[i = 1, i ≤ 100, i++,
  If[Abs[data[[j, i]] - μ[[j]]] > Abs[data[[j, worstPointIndexArray[[j]]]] - μ[[j]]],
   worstPointIndexArray[[j]] = i]
 ]]
```

In[270]:=
```
worstPointIndexArray
```

Out[270]=
```
{84, 46, 73, 20, 42, 35, 92, 7, 73, 62, 27, 79, 49, 53, 66, 83, 1, 20, 45, 11, 95, 57, 84, 54, 51,
 93, 76, 96, 32, 59, 79, 27, 42, 52, 6, 28, 43, 66, 6, 45, 53, 38, 90, 93, 54, 26, 11, 48, 45, 73}
```

Calculate how many standard deviations away from the mean each of these worst points are.

In[271]:=
```
σFromMeans = Table[Abs[data[[n, worstPointIndexArray[[n]]]] - μ[[n]]] / Sqrt[V[[n]]], {n, 1, 50}]
```

Out[271]=
```
{2.73112, 2.73346, 2.87259, 3.64828, 3.02283, 3.29991, 3.53078, 2.63212, 3.11151, 2.66178,
 4.27042, 3.64404, 2.52979, 2.37639, 2.21299, 2.64856, 2.75526, 2.69202, 4.39129, 2.91427,
 3.77748, 3.01364, 2.98058, 2.48135, 2.5872, 3.124, 3.24318, 2.31034, 2.69248, 2.96675,
 2.07484, 2.74066, 3.41515, 2.94031, 3.69819, 3.24456, 3.40393, 3.55458, 2.32375, 3.49592,
 2.70581, 2.50134, 3.76635, 2.80453, 2.8713, 3.71274, 3.71125, 2.64153, 3.31972, 2.52101}
```

Calculate the probability of measuring the worst point (or worse) for each data set, assuming they are Gaussian.

In[272]:=
```
probablilites =
 Table[Integrate[PDF[NormalDistribution[0, 1], x], {x, σFromMeans[[n]], ∞}], {n, 1, 50}]
```

Out[272]=
```
{0.00315601, 0.00313363, 0.00203559, 0.000132002, 0.00125212, 0.000483575, 0.00020717,
 0.00424275, 0.000930662, 0.00388647, 9.75514 × 10^-6, 0.000134197, 0.00570657,
 0.00874151, 0.0134491, 0.00404177, 0.00293231, 0.00355103, 5.63409 × 10^-6,
 0.00178262, 0.0000792126, 0.00129068, 0.0014385, 0.00654424, 0.00483804,
 0.000892066, 0.000591019, 0.0104346, 0.00354618, 0.00150483, 0.0190005, 0.00306578,
 0.000318735, 0.00163941, 0.000108572, 0.000588166, 0.000332122, 0.00018929,
 0.0100695, 0.000236216, 0.0034069, 0.00618622, 0.0000828249, 0.00251946,
 0.00204394, 0.000102515, 0.000103118, 0.00412659, 0.000450545, 0.00585101}
```

Calculate the probability that with 100 measurements, the worst data point will be one of the measurements.

In[273]:=
```
probOfMeasurement = probablilites * 100
```

Out[273]=
```
{0.315601, 0.313363, 0.203559, 0.0132002, 0.125212, 0.0483575, 0.020717, 0.424275, 0.0930662,
 0.388647, 0.000975514, 0.0134197, 0.570657, 0.874151, 1.34491, 0.404177, 0.293231,
 0.355103, 0.000563409, 0.178262, 0.00792126, 0.129068, 0.14385, 0.654424, 0.483804,
 0.0892066, 0.0591019, 1.04346, 0.354618, 0.150483, 1.90005, 0.306578, 0.0318735, 0.163941,
 0.0108572, 0.0588166, 0.0332122, 0.018929, 1.00695, 0.0236216, 0.34069, 0.618622,
 0.00828249, 0.251946, 0.204394, 0.0102515, 0.0103118, 0.412659, 0.0450545, 0.585101}
```

Below is a list of all of the sets of data for which it is okay to throw out the worst point.

In[274]:=
```
okayToThrow = {0};
```

In[275]:=
```
For[i = 1, i <= 50, i++,
 If[probOfMeasurement[[i]] < .5, AppendTo[okayToThrow, i]]]
```

In[276]:=
```
okayToThrow
```

Out[276]=
```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 16, 17, 18, 19, 20, 21, 22, 23, 25,
 26, 27, 29, 30, 32, 33, 34, 35, 36, 37, 38, 40, 41, 43, 44, 45, 46, 47, 48, 49}
```

Why is it okay to throw out the worst point from each of these sets? Because there is less than a 50% chance that they were measured legitimately during the experiment. This is a low enough probability to

throw them out of the set.

I made the original conclusion in Analysis Part 1, problem c, that Gaussians were the best fit for my individual data sets. As a result, it is indeed acceptable to use Chauvenet's criterion to throw out these points.

Analysis Part 3: Modeling your Data

a.

Binomial: This is the basic overarching distribution that we derived in class. It gives the probability of a given number of success out of a number of trials, based on the probability of success for each trial. This distribution works well away from crazy limits. This means that the probability of success shouldn't be very close to zero, and the number of trials shouldn't be enormous. The number of successes can run from 0 to number of trials.

Normal: This distribution is found often in data collection. It is found by looking at a limit of the Binomial distribution. Specifically, it occurs when you push the bounds of the Binomial distribution to -∞ and ∞. This is done by making the number of trials huge, putting the number of successes somewhere in the middle of those trials, and making the standard deviation of the distribution much less than the number of trials. This basically "zooms in" on the binomial distribution, and makes it a continuous distribution (becasue of the large number of trials).

Poisson: The Poisson distribution describes rare events. It is another limit of the Binomial distribution. This occurs when p is very small, and the number of trials is very big. This makes the bounds on the distribution 0 to ∞. Also, as the events being looked at are rare, it creates an exponential decay from a low number of successes (0, 1, 2, etc.) to higher numbers.

b.

Binomial: $P_B(r) = {}^N C_r \, p^r (1-p)^{N-r}$
Here, N is the number of trials performed, r is the number of successes, and p is the probability of a success. For example, if we roll a dice 6 times, and are looking for three 5s, N = 6, p = $\frac{1}{6}$, and r is 3.

Normal: $P_N(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{\frac{-(x-\bar{x})^2}{2\sigma^2}}$
$\bar{x}$ is the mean, or average, of the data set. This can be though of as the center of this distribution, and is the point at which $P_N(x)$ is the highest.

σ is the standard deviation. This is a measure of the width of the distribution, with the same units as the

data collected. Specifically, if we look at the probability of finding a data point in between $\bar{x} + \sigma$ and $\bar{x} - \sigma$, it will be about 68%.

Poisson: $P_P(r) = \frac{a^r}{r!} e^{-a}$

r is number of successes, just as it was in the Binomial distribution. For this distribution, the important r's will be small. r does need to be an integer value as r is still number of successes. a is the center of this distribution. It is calculated as: a = Np, where N and p represent the same things that they do for the binomial distribution. It is often called the "expected outcome".

C.

I decided to find another distribution to fit my data, other than the three described above. This was mostly based on the fact that the Binomial and Poisson distributions are both discrete in nature, while my data is not discrete at all. My volume level could take on any value. The main reason that I wanted to fit something that isn't a Gaussian is that my data has a lower bound of 0. A Gaussian is defined on -∞ to ∞, which is not accurate for my data. I also wanted to take into account the fact that my data has a slightly longer tail on the right than the left.

After searching around for different distributions, I settled on a Gamma Distribution, which seems to have just the properties that I was looking for. This distribution has the equation:

$$P_G(x) = \frac{\lambda^\alpha x^{\alpha-1} e^{-\lambda x}}{\int_0^\infty x^{\alpha-1} e^{-x} \, dx}$$

Where α and λ are the parameters of the equation. After researching for a while, I discovered that these parameters are quite difficult to calculate by hand from a set of data, so I instead used Mathematica's built in FindDistributionParameters function to find the best possible fit. This is done below.

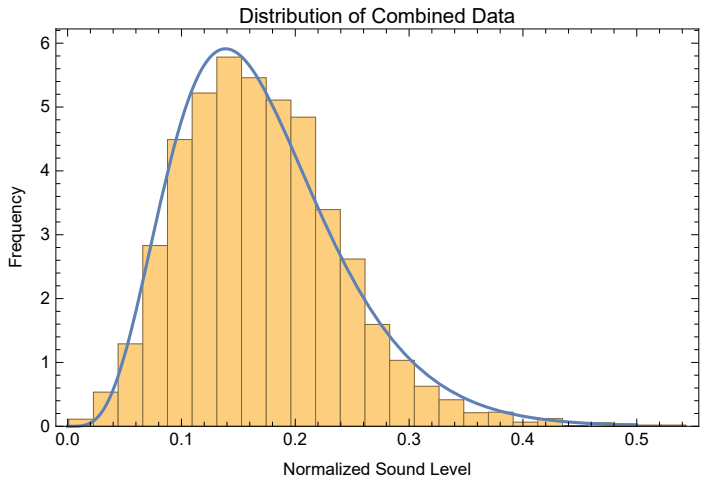
In[277]:=
```
sub1 = FindDistributionParameters[bigdata, GammaDistribution[α, β]]
```

Out[277]=
```
{α → 5.39048, β → 0.0316004}
```

In[278]:=
```
p1 = Plot[PDF[GammaDistribution[α, β] /. sub1, x], {x, 0, .5}];
```

In[279]:=
```
Show[hist4, p1]
```

Out[279]=



We can see that this distribution seems to be a very good fit for my data, qualitatively. It is bound at zero, and follows the slightly longer right tail of the data very well. To analyze how well the fit is quantitatively, I will look at the moments of the distribution and compare it to my data. 0th moment is trivial, so I will ignore that one.

1st Moment:

In[280]:=
```
mGamma1 = Integrate[(x)^1 PDF[GammaDistribution[α, β] /. sub1, x], {x, 0, ∞}]
```

Out[280]= 0.170342

And I pull the 1st moment from the calculation I did in Analysis Part 2 a.

In[281]:=
```
m1
```

Out[281]= 0.170324

In[282]:=
```
pdiff[mGamma1, m1]
```

Out[282]= 0.0103988

We see that there is a .01% difference in the 1st moment (using a percent difference function that I wrote for an earlier section of the project).

2nd Moment:

In[283]:=
```
mGamma2 = NIntegrate[(x - mGamma1)^2 PDF[GammaDistribution[α, β] /. sub1, x], {x, 0, ∞}]
```

Out[283]= 0.00538287

In[284]:=
```
m2
```

Out[284]= 0.00501303

In[285]:=
```
pdiff[mGamma2, m2]
```

Out[285]= 7.11495

We see a 7% difference for the second moment.

In[286]:=
```
mGamma3 = NIntegrate[(x - mGamma1)^3 PDF[GammaDistribution[α, β] /. sub1, x], {x, 0, ∞}]
```
Out[286]= 0.000340202

In[287]:=
```
m3
```
Out[287]= 0.000257449

In[288]:=
```
pdiff[mGamma3, m3]
```
Out[288]= 27.6927

We see a 28% difference for the third moment.

In[289]:=
```
mGamma4 = NIntegrate[(x - mGamma1)^4 PDF[GammaDistribution[α, β] /. sub1, x], {x, 0, ∞}]
```
Out[289]= 0.000119177

In[290]:=
```
m4
```
Out[290]= 0.000104791

In[291]:=
```
pdiff[mGamma4, m4]
```
Out[291]= 12.8468

And lastly, we see a 13% difference for the fourth moment. If we recall from earlier in the project, I reported a 34% difference in fourth moment between the Gaussian fit and my data. Based on this fact, and also the fact that the other moment's have low percent difference, I conclude that this Gamma distribution is a much better fit for my data than the Gaussian.

---

d.

If I look into the parameters that went into the measurement of my data, I can find some interesting simultaneous limits. My data is measured in some unit of linear volume, which is directly proportional to sound intensity. We can define this as a function:

In[292]:=
```
dataVolume[intensity_] = const * intensity;
```

Where const is some constant that relates the two values. Sound intensity is determined by the distance away from the source it is being measured, assuming a point source (which for my case, would have been the speaker I was using to play songs). It is also dependent on the power of this point source, which in my case, would be related to the volume I was playing the music from the speakers at. Intensity is related to power and distance by:

In[293]:=
```
inten[p_, r_] = p/(4 π r^2);
```

Where p is the power of the point source, and r is the radial distance from the source. This leads to a final equation for my data points which is:

In[294]:=
```
dataVolume[inten[p, r]]
```

Out[294]= $\frac{p}{4\pi r^2}$

So what simultaneous limits can we take from this equation? We see that if we increase the distance from the source for which we are taking the volume measurement in a linear fashion, at a rate of dr, and we also increase the power at a rate of dr^2, that the volume will stay constant. This simultaneous limit stays true as p and r go to infinity (or zero), as long as the power continues to increase quadratically compared to the radius. This idea is shown graphically below.

Say we assume a volume constant of:

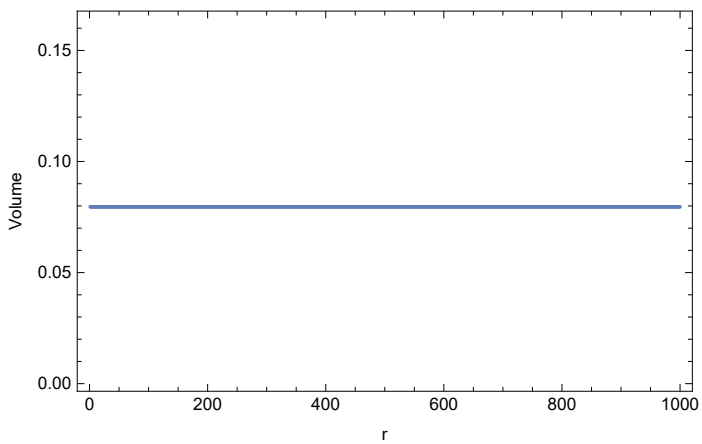
In[295]:=
```
const = 1;
```

Then we can look at a plot of multiple volume data points, as p and r increase towards infinity.

In[296]:=
```
volume = Table[dataVolume[inten[r^2, r]], {r, 1, 1000}];
```

In[297]:=
```
ListPlot[volume, Frame → True, FrameLabel → {"r", "Volume"}]
```

Out[297]=



We see that indeed, the volume level stays constant, as we take r and p out to infinity.

The volume equation makes sense physically when you think about it. Sound moves out spherically from the source, so it makes sense that the intensity of the sound drops off at a factor of $\frac{1}{4\pi r^2}$ which is $\frac{1}{\text{Surface Area}}$.

---

e.

Below, I find the two most different individual data sets that I collected. I do this by finding the maximum and minimum means from all of the sets.

In[298]:=
```
Position[μ, Max[μ]]
```

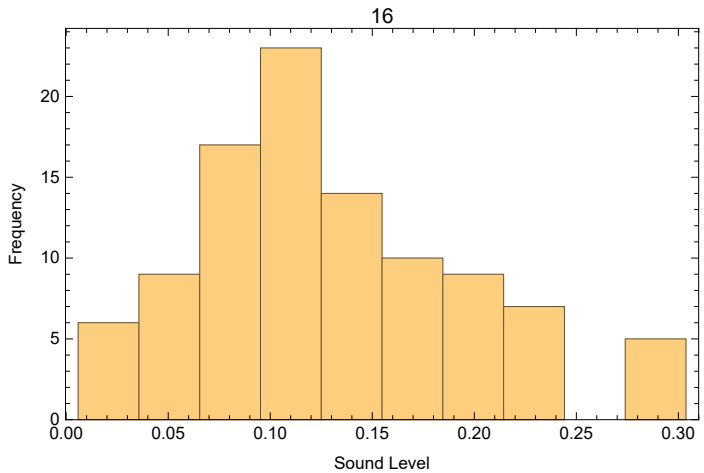
Out[298]= {{8}}

In[299]:=
```
Position[μ, Min[μ]]
```
Out[299]= {{16}}

So data sets 8 and 16 are the two most different sets. I plot them below.

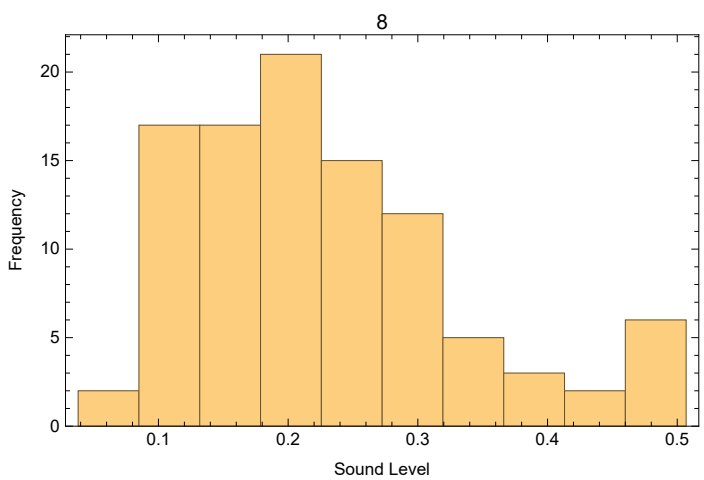
In[300]:=
```
h16 = Histogram[data[[16]],
  {Min[data[[16]]], Max[data[[16]]] + .001, (Max[data[[16]]] - Min[data[[16]]] + .001)/10},
  Frame → True, FrameLabel → {"Sound Level", "Frequency"}, PlotLabel → "16"]
```

Out[300]=



In[301]:=
```
h8 = Histogram[data[[8]],
  {Min[data[[8]]], Max[data[[8]]] + .001, (Max[data[[8]]] - Min[data[[8]]] + .001)/10},
  Frame → True, FrameLabel → {"Sound Level", "Frequency"}, PlotLabel → "8"]
```

Out[301]=



I can now use the student's t-test to determine if these sets are significantly different from one another. Each data set has 100 data points, so the overall degrees of freedom for the two sets is 198.

In[302]:= $\sigma diff = \sqrt{\frac{V[[16]] + V[[8]]}{198}}$

Out[302]= 0.00884095

In[303]:= $t = \frac{Abs[\mu[[16]] - \mu[[8]]]}{\sigma diff}$

Out[303]= 11.0569

The below calculation will give me the confidence that our data sets are from the same data set. Essentially, it gives us the area under the student t-distribution from our calculated t value to ∞. This gives the probability of getting a t value equal to or worse than ours. I did not use the table in the notes, as my number of degrees of freedom was too big. Instead, I used a built in Mathematica function:

```
In[304]:= Needs["HypothesisTesting`"];

In[305]:= StudentTPValue[t, 198, TwoSided → True]

Out[305]= TwoSidedPValue → 1.92501 × 10^-22
```

We get essentially 0%. This tells me that the two data sets statistically were **not** pulled from the same big parent data set. They are indeed significantly different from one another. What does this say about my data? It confirms my previous analysis that each song is significantly different from one another. I made this conclusion without any quantitative reasoning in Analysis Part 1 f, but now I can say with quantitative evidence that different song's volume levels are significantly different from one another.

---

f.

We use the χ^2 analysis by comparing the heights of my binned histogram to the height predicted by the Gamma distribution. To do this, I have to add a scaling factor to the distribution. I scale the distribution by the total area in in my histogram. I have actually already calculated this area earlier in the project, when calculating the moment of this data. It is:

```
In[306]:= totalArea

Out[306]= 108.445
```

So I scale the Gamma Distribution:

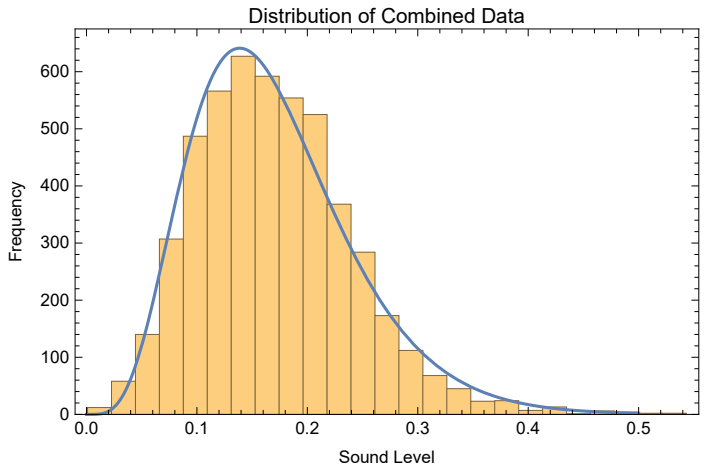
```
In[307]:= scaledGamma[x_] = totalArea * PDF[GammaDistribution[α, β] /. sub1, x];
```

I plot this distribution on top of my data, just to make sure that it checks out:

```
In[308]:= hist5 = Histogram[bigdata,
    {Min[bigdata], Max[bigdata] + .01, (Max[bigdata] - Min[bigdata] + .01)/25}, Frame → True,
    FrameLabel → {"Sound Level", "Frequency"}, PlotLabel → "Distribution of Combined Data"];

In[309]:= gammaPlot = Plot[scaledGamma[x], {x, 0, 0.5}];
```

In[310]:=
```
Show[hist5, gammaPlot]
```

Out[310]=


So it all checks out.

Next, we need the heights of each of the bins in the above histogram. I already have these found from the moment calculations done earlier in the project:

In[311]:=
```
counts
```

Out[311]=
```
{12, 58, 140, 307, 487, 566, 627, 592, 554,
 525, 368, 284, 173, 112, 68, 45, 23, 24, 7, 13, 1, 6, 4, 2, 2}
```

Next, I need the x values for the center of each of my bins in the histogram, so that I can get out what the Gamma distribution predicts for each of those values. Once again, I already calculated these values earlier in the project:

In[312]:=
```
xmids
```

Out[312]=
```
{0.0118211, 0.03351, 0.0551989, 0.0768879, 0.0985768, 0.120266, 0.141955, 0.163644,
 0.185333, 0.207021, 0.22871, 0.250399, 0.272088, 0.293777, 0.315466, 0.337155,
 0.358844, 0.380533, 0.402222, 0.423911, 0.4456, 0.467289, 0.488978, 0.510667, 0.532356}
```

I now can calculate the χ^2 value for this data vs the Gamma distribution:

In[313]:=
```
χ2 = Sum[(counts[[n]] - scaledGamma[xmids[[n]]])^2 / scaledGamma[xmids[[n]]], {n, 1, Length[xmids]}]
```

Out[313]=
```
266.106
```

And we can find the percent confidence

In[314]:=
```
N[1 - CDF[ChiSquareDistribution[24], χ2]]
```

Out[314]=
```
0.
```

So the χ^2 test tells me that the model does not actually fit my data. Not at all. There is a 0% probability that my Gamma function fits my data. Why the failure? From what I can tell, the Chi Square distribution is very sensitive, especially when the counts that exist in each bin of a histogram are so large. Because there are counts of around 600 at the peak of my histogram, this has a large impact on the χ^2 value being terrible.

One of the main assumptions of the χ^2 distribution is that the data that it analyzes is randomly selected from one big overall distribution. I just proved this to not be statistically possible for my data in the previous section, so right off the bat, I can say that this criterion is incorrect for my data.