PHYS 339 - Measurements Lab in Gen Phys

Lab 2: Counting Statistics - The Significance of Errors

Maclean Rouble (260423190), Michael Caouette-Mansour (260581053), Leo Thomas (260574268), Michael Totarella (260690583)

McGill University Department of Physics

January 30, 2017

Abstract

This experiment consists of measuring the decay of a Cesium-137 radioactive source by using a Geiger counter to record the number of "events" (decaying particles) per time interval. Several data sets are presented, each with a different size and mean number of occurrences. This data is analyzed in Python using Chi Squared distributions found by comparing the data to Poisson and Gaussian distributions.

1 Introduction

Handling random fluctuations in observed quantities is a key skill for an experimentalist, as they occur in nearly all experiments.In this experiment, a purely random process is analyzed to understand the significance of these fluctuations and how they present as error and variance in the data. The random process in question is radioactive decay of Cesium-137, measured by a Geiger-Müller counter. Probability distributions such as the Poisson and the Gaussian functions are used as comparisons to the behavior of the process and to quantify the randomness of its nature. χ^2 tests are used to evaluate the closeness of the gathered data to the theoretical distributions.

2 Theory

When handling processes containing randomness, comparing the gathered data to known probability distributions is a good way to separate random occurrences from experimental error, as well as gain insight into the characteristics of the data set. The following is a discussion of two such probability distributions (Poisson and Gaussian); and the nature of the random process measured in this experiment, radioactive decay.

2.1 Distribution

In order to describe the set x_i of n data by distributions, we define the mean (which is a good way to describe the average) by:

$$< x >= \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

The variance σ^2, which is the square of the standard deviation (a good way to describe the error) is defined as:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}[x_i - < x >]^2 \tag{2}$$

The Gaussian probability distribution describes the spreading of a variable ν by random

errors on it. If μ is the mean and σ is the standard deviation, the Gaussian probability distribution is defined as:

$$P(\nu, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\nu-\mu)^2}{2\sigma^2}} \tag{3}$$

The Poisson probability distribution applies when the occurring events are all independent and the average rate does not change over the period of interest. It is characterized by the mean μ, which is the average count per time interval. The Poisson probability distribution is defined as:

$$P(\nu, \mu) = \frac{\mu^\nu e^{-\mu}}{\nu!} \tag{4}$$

Here ν is the number of counts. One feature of this distribution is that the mean μ equals the variance σ^2. For large μ and large ν, one can show that the Poisson distribution approaches a Gaussian distribution. Since, for a Poisson distribution $\sigma^2 = \mu$, this gives the Gaussian limit:

$$\frac{\mu^\nu e^{-\mu}}{\nu!} \approx \frac{1}{\sqrt{2\pi\mu}} e^{-\frac{(\nu-\mu)^2}{2\mu}} \tag{5}$$

The χ^2 is a good way to test if the error one gives on a osberved set O_i is good or not. The χ^2 is defined by:

$$\chi^2 = \sum_{i=0}^{n} \frac{(O_i - E_i)^2}{\sigma_i^2} \tag{6}$$

where σ_i^2 is the variance and E_i is the expected value. Because of the σ^2 in the denominator, the χ^2 keeps track of how good the error estimate is. If we calculate the χ^2 over many observations of the same nature, the results should differ from one another because of the error. If the error estimate σ^2 is good enough, the χ^2's should follow a distribution which is given by:

$$P(\chi^2, n) = \frac{(\chi^2)^{\frac{n}{2}-1} e^{-\chi^2/2}}{2^{n/2}\Gamma(n/2)} \tag{7}$$

where n is the number of degrees of freedom and $\Gamma(x) = (x-1)!$ is the Gamma function.

2.2 Radioactive Decay

Radioactive decay is a naturally random process that is unaffected by external factors, like temperature or electrical interference. Atoms undergo decay in various ways, releasing various

types of radiation. The type studied in this experiment is gamma (γ) radiation. Gamma rays are produced when a nucleus decays from a high-energy state (typically the result of a previous decay) to a low-energy state. The radiation produced is of high frequency ($f > 10^{19}$) and has enough energy to ionize other molecules [1].

The probability of these decay events occurring is described by:

$$\lambda = \frac{-\ln(1/2)}{t_{1/2}} \tag{8}$$

where $t_{1/2}$ is the half-life of the material. One can then consider as a function describing the likelihood, λ, of k events occurring in an interval:

$$f(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{9}$$

2.3 Geiger Counter

The Geiger counter is an instrument used to measure radiation in the environment. It operates by maintaining a tube filled with inert gas at a high voltage. Incoming high-energy particles cause an atom to ionize by freeing an electron, which starts a cascade reaction of ionization called a Townsend avalanche . This cascade creates a small voltage, which closes a switch and triggers the detector. This design does have some limitations, however, and can struggle to accurately measure very high radiation rates (due to the length of time required per cascade, which is typically on the order of several microseconds). [2]

3 Experimental Methods

The radioactive sources available were Cesium-137 ($^{137}_{55}Cs$) with a half-time of 30.1 years, and Barium-133 ($^{133}_{56}Ba$) with a half-time of 10.5 years. The source used for the experiment was Cesium-137. The source was placed directly on the detector and the mean rates of detection were made "by-hand" by moving the source with respect to the tube.

The detection of gamma rays was done with a Geiger-Müller tube, DX-1 model, manufactured by RDX Nuclear (see Figure 1). When a gamma ray hits the tube, a signal is sent

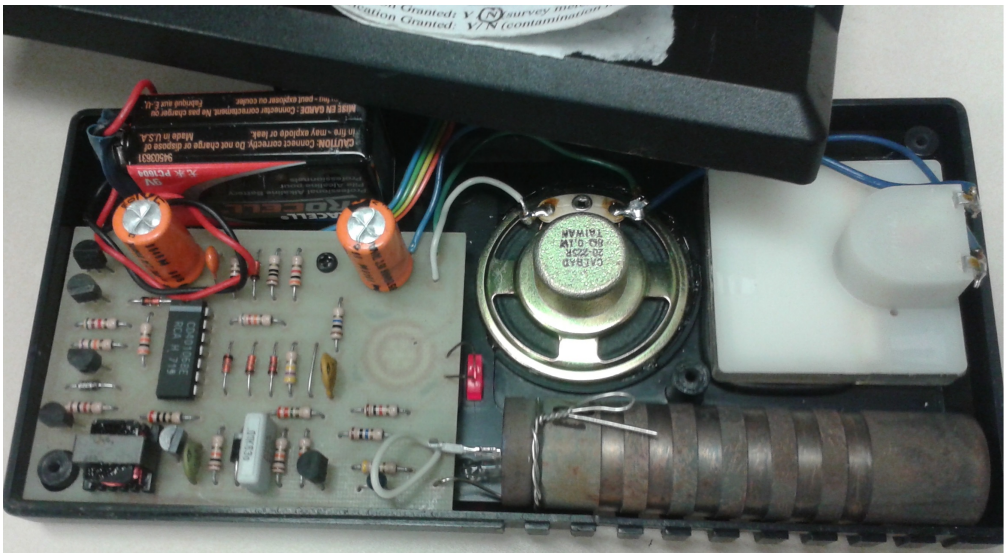

Figure 1: The Geiger-Müller tube used in the experiment. The tube is in the lower right part of the picture.

to the Arduino module through a coaxial wire.

A data-acquisition routine is then called by the computer connected to the Arduino module.

4 Results

Data is recorded over several runs, with varying numbers of replicas and intervals. The data from each run is plotted with x-axis bins corresponding to the number of radioactive decay occurrences per 0.2s time interval, and y-axis corresponding to the number of times a certain number of occurrences per interval were recorded. These plots are compared to theoretical Poisson and Gaussian distributions with the mean as calculated from the data. The goodness of these fits is evaluated by use of χ^2 values, calculated for each plot. Figures 2 through 5 show data with Poisson and Gaussian fits for several different means. The estimate count rate is done using equation 1 and the error by the square root of the variance from equation 2.

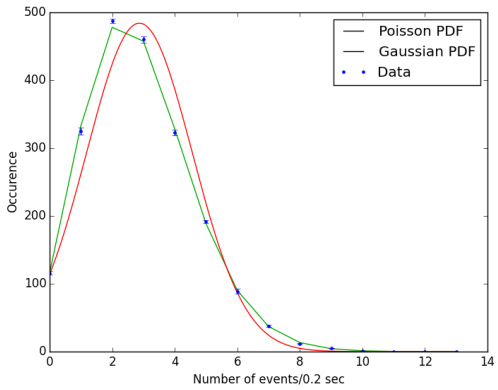

Figure 2: Mean = 2.9 ± 1.7 counts/0.2 sec.

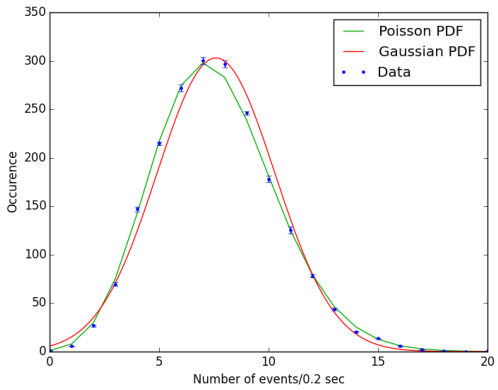

Figure 3: Mean = 5.5 ± 2.8 counts/0.2 sec.

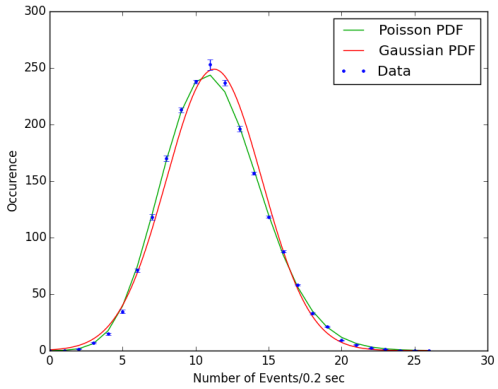

Figure 4: Mean = 11 ± 3 counts/0.2 sec.

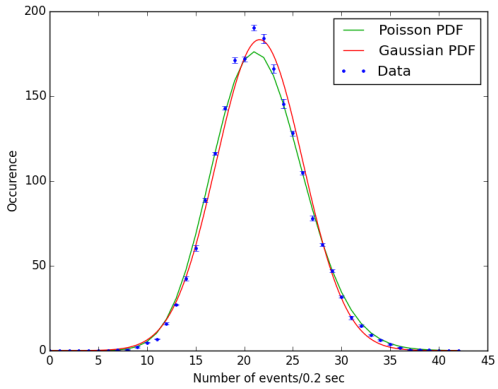

Figure 5: Mean = 22 ± 4 counts/0.2 sec.

Note that at low means, the Poisson function (green) is clearly a better fit than the Gaussian, however, as the mean number of occurrences increases the Poisson function tends towards the Gaussian. Also note, the fits for either function were calculated using the same number of data points. However the nature of the scipy.poisson and scipy.norm functions used to generated the poisson and gaussian plots, respectively, interpolates between the given values in the case of the gaussian, but not the poisson. This is why the poisson functions plotted are not as smooth and the gaussians.

The replica and interval numbers of runs with a mean number of occurrences of 10 can be seen in Table 1. The same for a mean of 20 can be seen in Table 2, and a mean of 4 in Table 3.

To analyze the validity of the gathered data, a value of the χ^2 function is chosen above

which 12.5% of the data should lie, if in perfect correspondence with the theoretical distribution (either Poisson or Gaussian). This point is then examined in the data set to test what percentage lies above it in reality. This allows for a quantitative evaluation of which distribution is most closely matched by the experimental data. These values are shown in Tables 1 through 3. For example, in Table 1, 12.5% of the χ^2 values of the Poisson fit (with 26 degrees of freedom, in accordance with the gathered data) should lie above 33.25. The numbers listed in the third column are the true percentages of data that lie above 33.25. Similarly, 12.5% of the χ^2 values for the Gaussian (of 25 degrees of freedom) fit falls above 32.08. The numbers listed in the fourth column are the percentages of the observed data that lie above this mark. This same structure is repeated in Tables 2 and 3.

Replicas	Intervals	% of $\chi_p^2 > \chi_{pc}^2$	% of $\chi_g^2 > \chi_{gc}^2$
1024	16	12	13
512	32	12	14
256	64	15	18
128	128	14	23
64	256	19	38
32	512	22	72
16	1024	31	100

Table 1: Mean = 10, d.o.f.= 26, $\chi_{pc}^2 = 33.247318$, $\chi_{gc}^2 = 32.080865$

Replicas	Intervals	% of $\chi_p^2 > \chi_{pc}^2$	% of $\chi_g^2 > \chi_{gc}^2$
128	128	11	11
64	256	8	11
32	512	22	22
16	1024	19	31

Table 2: Mean = 20, d.o.f. = 37, $\chi_{pc}^2 = 45.891209$, $\chi_{gc}^2 = 44.753475$

Replicas	Intervals	% of $\chi_p^2 > \chi_{pc}^2$	% of $\chi_g^2 > \chi_{gc}^2$
128	128	16	35
64	256	16	62
32	512	22	94
16	1024	25	100

Table 3: Mean = 4, d.o.f. = 17, $\chi_{pc}^2 = 17.703325$, $\chi_{gc}^2 = 16.456847$

5 Discussion

The goodness of fit of the gathered data to the theoretical Poisson and Gaussian distributions is measured by the use of a χ^2 distribution. As the χ^2 function depends only on the number of degrees of freedom for a data set, it can be generated to individually represent each experimentally-gathered set.

The original data contains too much noise to be meaningfully distinguished between a Poisson or a Gaussian distribution. A compression algorithm is used to combine replicas, lowering its associated variances. This algorithm divides the set of replicas in half and stacks them together, reducing the number of replicas and increasing the number of intervals per replica. For example, a set of four replicas with two intervals each would be compressed to a set of two replicas with four intervals each. The variance of this new replica is reduced, as outlying data points are now less significant, due to the doubling of the number of points in the set.

This compression is useful because of the way in which the χ^2 values are used to compare the data to the theoretical distribution. One χ^2 value is generated per replica, so reducing the number of replicas reduces the number of χ^2 values. These χ^2 values are also smaller, as the compressed data tends to follow more closely to the expected distribution, and its variances have been reduced.

However, as seen in tables 1, 2, and 3, this algorithm has some rather unexpected consequences at high levels of compression. The original data set contains several dramatic outliers created by instrument anomalies (discussed in section 2.3), which generate outliers in the χ^2 distribution.

These outliers have much larger χ^2 values than the rest of the data and shift the overall

set up from its expected range (based on its number of degrees of freedom), as they become more significant. This shift is demonstrated in figures 6, 7, 8, and 9.

Tables 4 and 5 show the probability of the maximum χ^2 value occurring naturally according to the χ^2 distribution of the uncompressed batch, for Poisson and Gaussian fits, respectively. The increase in the probabilities of occurrence throughout both tables indicates that the compression of the data dampens the effect of outliers in the original data, within the analysis.

At the last compression level, the probability of the largest χ^2 value occurring naturally is $\approx 9\%$ when compared to a Poisson. We conclude that the compression reduces the effect of statistical anomalies on the analysis, and that in the absence of these anomalies the Poisson distribution is a more accurate function to describe our data.

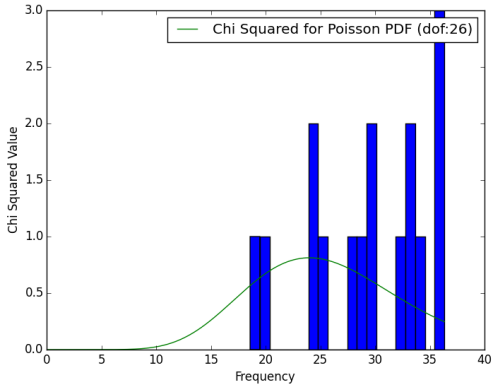

Figure 6: The fully compressed χ^2 data shown with the expected χ^2 for the Poisson distribution.

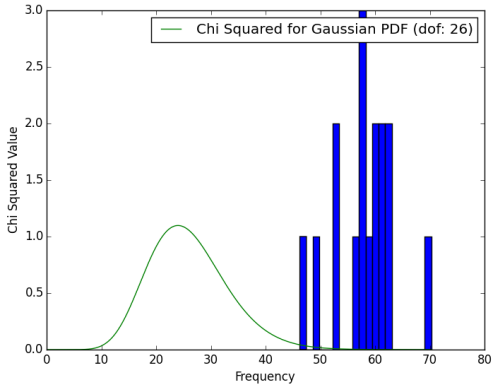

Figure 7: The fully compressed χ^2 data shown with the expected χ^2 for the Gaussian distribution.

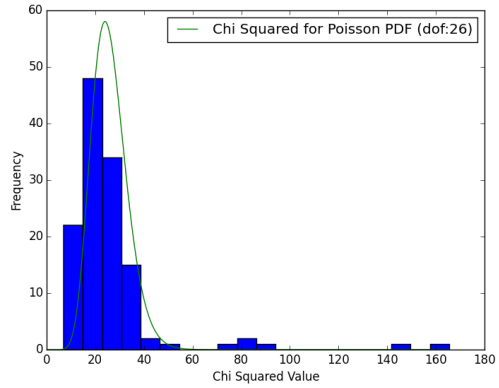

Figure 8: The less compressed χ^2 data shown with the expected χ^2 for the Poisson distribution.

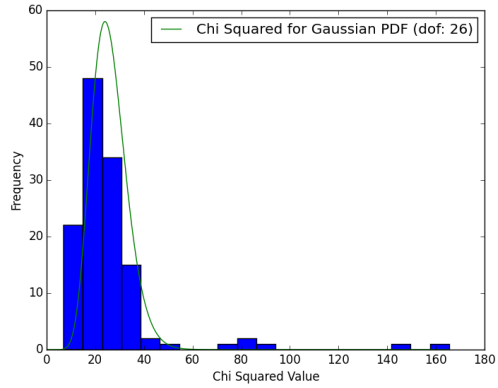

Figure 9: The less compressed χ^2 data shown with the expected χ^2 for the Gaussian distribution.

Accordingly, these outliers in the χ^2 distribution increase the percentage of the data that falls beyond the point tested in tables 1, 2, and 3 (the point which 12.5% of the data lies above). For this reason, the sets with higher numbers of replicas tend to fit more reliably with the Poisson distribution than those with lower numbers. This shifting does not affect the expected outcome of the comparison to the Gaussian distribution, however, because even before the outliers gain large significance, the data's χ^2 values are higher than those of the theoretical.

Despite the discrepancies at high compression, it is still possible to compare the fits

between the Poisson and the Gaussian distributions. The three data sets examined in detail are those featured in 1, 2, and 3, with means of 10, 20, and 4, respectively.

Comparing the mean 20 results to the mean 4 results gives a striking illustration of the effect of the mean on the shape of the distribution. The high mean of 20 occurrences per 0.2s interval results in a distribution which is nearly equally characteristic of a Poisson or Gaussian, with similar amounts of the data set falling beyond the expected range for each function. The results of the mean 4 trial, however, are much more distinct. While there is still more than expected of the data falling above the 12.5% mark on the Poisson distribution, that 100% of the data does not agree with the Gaussian distribution points strongly towards it following the Poisson. This type of difference is demonstrated in figures 2, 3, 4, and 5.

6 Conclusion

From this experiment we can conclude that statistical anomalies have a significant impact when analyzing data, even in large quantities. A few outliers caused by simple events such as the geiger counter registering multiple events within too short of a time frame can make it impossible to distinguish which distribution the data should be fit to. We also conclude that compressing, and in a more general sense, averaging, large quantities of small data collections into a smaller number of larger collections can reduce the noisiness of the data. This allows for a more meaningful analysis to be recovered since the inaccuracies due to statistical outliers are dampened. We can finally conclude that radioactive gamma (γ) decay of Cesium-137, a random process, falls under a Poisson distribution. However this conclusion is weakened by the presence of the above mentioned statistical outliers. Data sets with equally large number of replicas, but with larger collection periods or much smaller means could be taken to avoid the glitches due to multiple events registering too close to each other, and lead to a more meaningful analysis, and stronger conclusion.

References

[1] "Gamma radiation." [Online]. Available: https://www.nde-ed.org/EducationResources/CommunityCollege/Radiography/Physics/gamma.htm 3

[2] G. F. Knoll, *Radiation Detection and Measurement*. John Wiley and Sons Inc., 2000. 3

A: Python Code

```
#returns column stack of mean and variance by replica
def stats_by_batch (h):

    means= numpy.zeros(len(h))
    variances = numpy.zeros(len(h))

    for n in range(len(h)):
        means[n] = hist_mean(h[n])
        variances[n] = hist_var(h[n], means[n])

    return numpy.column_stack((means, variances))

#returns column stack of mean and variance by column
def stats_by_column(h):

    h = numpy.transpose(h)
    means = numpy.array([numpy.mean(n) for n in h])

    variances = numpy.array([numpy.var(n) for n in h])
    return numpy.column_stack((means,variances))



# helper function returns mean value of a replica, given its histogram
# (ie: average of column index times column value)
def hist_mean(data):
    pmean=0.0
    for i in range(0, len(data)):
        pmean = pmean + (i*data[i])
```

```
    pmean = pmean/sum(data)
    return pmean

# helper function returns variance of a replica given its histogram
def hist_var(data, mean):
    pvar = 0.0
    temp_var = numpy.zeros(len(data))
    for i in range(0, len(data)):
        temp_var[i] = ((i-mean)**2)*data[i]
    pvar = sum(temp_var)/sum(data)
    return pvar

# Returns a single chi squared value given a replica, a function
# the replica is being fitted to, and array of variances per column
def chi_squared(O,E,err):
    temp = numpy.zeros(numpy.count_nonzero(err))
    for i in range(len(temp)):
        # filter out 0 variance columns to avoid division by zero, or
        # can increase value to filter out small variances
        if err[i]>0.000:
            temp[i] = (O[i]-E[i])**2/err[i]

        # The following was more pythonic, but we were forced to change
        # to a loop to avoid division by zero/ filter out vary small
        # variances:
        # sum(numpy.divide((O-E)**2, err))

    return sum(temp)

# Returns a chi squared value for a replica being fitted to either a
```

```
# Poisson or Gaussian. The Poisson fit requires the replica mean and
# the Gaussian fit requires both the replica mean and variance.
# The Poisson and Gaussian functions used to compare each replica
# to are gnerated once, with the mean values of replica means and
# replica variances
# The column variances are used to calculate the chi squared value, as
# per Mark's recomendation

def chi_squared_byBatch(replica, mv, var, typ):
    x = numpy.linspace(0,len(replica)-1,len(replica))

    pfit = sum(replica)*(poisson.pmf(x, mv[0]))
    gfit = sum(replica)*(norm.pdf(x, mv[0], numpy.sqrt(mv[1])))

    csqPoisson = chi_squared(replica, pfit, var)
    csqGauss = chi_squared(replica, gfit, var)

    if typ == 'G': return csqGauss
    elif typ == 'P': return csqPoisson



#data = 1 replica from geiger or compressed data
def poisson_fit(data, x):
    mean = hist_mean(data)
    x = numpy.linspace(0, len(data)-1, len(data))

    y = sum(data)*(poisson.pmf(x, mean))
    return x,y

#data = 1 replica from geiger or compressed data
```

```
def gaussian_fit(data,x):
    mean = hist_mean(data)
    var = hist_var(data, mean)
    y = sum(data)*(norm.pdf(x, mean, numpy.sqrt(var)))
    return x,y

# Adds second half of the data to the first half, replica by replica
def compress_data(original):
    L = numpy.size(original, axis=0)/2
    new_array = original[:L,:] + original[L:]
    return new_array

# prints a histogram of chi squared values overlayed with the
# chi squared PDF corresponding to the degress of freedom determined
# by the amount of non-zero variance columns (ie: significant columns)
def show_chisq(chisqs, dof):
    p.figure()
    hist, bins, patch = p.hist(chisqs,20)
    x = numpy.linspace(0, bins[len(bins)-1],  1000)
    y = sum(hist)*chi2.pdf(x,dof)*(bins[1]-bins[0])
    p.plot(x,y)

#statRows = stats_by_batch(data)
def findDOF(data):
    statCols = stats_by_column(data)
    col_vars_geiger = statCols[:,1]
    dof = 0
    for i in col_vars_geiger:
        if i != 0: dof = dof+1
    return dof
```

```
# calculates and prints the probability of the maxiumum chi squared value
# for the given replicas, according to the chi squared PDF.
# This is used to show that the count registering glitch of the
# geiger counter leads to statistically impossible results.
def outlier_prob(chisq_p, chisq_g):
    maxchisq = max(chisq_p)
    tdof = numpy.count_nonzero(col_vars)
    prob = chi2.sf(maxchisq, tdof)*100
    print ("(POISSON) %d replicas, highest chi squared value: %.0f with probability %e%%
                    maxchisq, prob))

    maxchisq = max(chisq_g)
    prob = chi2.sf(maxchisq, tdof)*100
    print ("(GAUSSIAN) %d replicas, highest chi squared value: %.0f with probability %e%
                    maxchisq, prob))

# graph historgram of given data set overlayed with corresponding
# poisson and gaussian fits
def graph_datafits(cd):

    cd_cols = stats_by_column(cd)
    x = numpy.linspace(0,len(cd_cols)-1, 200*len(cd_cols))
    p.figure()
    # This is the best I can do. For some reason, the scipy poisson function
    # needs to be generated over a small number of bins. I don't know why.
    xp, pfit = poisson_fit(cd_cols[:,0], x)
    p.plot(xp, pfit, "g", label='Poisson PDF')

    xg, gfit = gaussian_fit(cd_cols[:,0], x)
```

```
    p.plot(xg, gfit, "r", label='Gaussian PDF')

    xplot = numpy.linspace(0, len(cd_cols)-1, len(cd_cols) )
    p.errorbar(xplot, cd_cols[:,0], numpy.sqrt(cd_cols[:,1]/len(cd_cols)), fmt = '.')



# note: I LOVE LEGACY CODE...

# calculate number of degrees of freedom
dof = findDOF(geiger)

new_big = geiger
p.figure()
fraction = 1./8.
chisq_cut_p = chi2.isf(fraction,dof-1)
chisq_cut_g= chi2.isf(fraction,dof-2)
print("Dataset has %d degrees of freedom"%(dof))
print("%.1f%% of Poisson chisq should be larger than X_pc = %f"%(100*fraction,chisq_cut_
print("%.1f%% of Gaussian chisq should be larger than X_gc = %f"%(100*fraction,chisq_cut
p.loglog() # for the scatter plot of column stats
p.xlabel("column means")
p.ylabel("column variances")
cd = new_big

# saves values (just in case)
save_intervals = []
save_fraction_ps = []
save_fraction_gs = []

while fraction*numpy.size(cd,axis=0) > 1:
```

```
rows = numpy.size(cd,axis=0)
cols = numpy.size(cd,axis=1)
intervals = cd.sum()/rows

# means and variances by replica
row_means,row_vars = stats_by_batch(cd)[:,0],  stats_by_batch(cd)[:,1]
# mans and variances by column
col_means,col_vars = stats_by_column(cd)[:,0], stats_by_column(cd)[:,1]

#plots column variances as a function of column means
p.scatter(col_means,col_vars)
p.pause(1e-3)

chisq_p = numpy.zeros(len(row_means))
chisq_g = numpy.zeros(len(row_means))

# mean of replica means
tmean= numpy.mean(row_means)

#mean of replica variances
vmean= numpy.mean(row_vars)

for i in range(0, len(row_means)):
    # chi squared values (one per replica) of each replica compared to
    # a poisson distribution, using overall data set mean
    chisq_p[i] = chi_squared_byBatch(cd[i],[tmean, vmean], col_vars, 'P' )

    # chi squared values (one per replica) of each replica compared to
    # a gaussian distribution, using overall data set mean and variances
    chisq_g[i] = chi_squared_byBatch(cd[i],[tmean, vmean], col_vars, 'G' )
```

```
# print probability of our highest chi squared value occuring
# according to the given chi squared distribution
outlier_prob(chisq_p, chisq_g)

# following 2 lines produce a histogram of chi squared values overlayed
# with corresponding chi squared PDF, generated using the number of
# degrees of freedom found above.

# chi squared compared to poisson expected values
show_chisq(chisq_p, numpy.count_nonzero(col_vars))

#chi squared compared to gaussian expected values
show_chisq(chisq_g, numpy.count_nonzero(col_vars))

fraction_p = (chisq_p > chisq_cut_p).mean()
fraction_g = (chisq_g > chisq_cut_g).mean()
print("%d replicas with %d intervals each: %.0f%% of X_p > x_pc, %.0f%% of X_g > X_g
(rows,intervals,100*fraction_p,100*fraction_g))

save_intervals.append(intervals)

save_fraction_ps.append(fraction_p)
save_fraction_gs.append(fraction_g)

cd = compress_data(cd)
```

B: Output of outlier probability function

Replicas	Intervals	Max χ^2 value	Probability (%)
1024	16	1042	5.08×10^{-201}
512	32	546	1.23×10^{-96}
256	64	287	6.80×10^{-44}
128	128	166	2.89×10^{-20}
64	256	89	8.00×10^{-07}
32	512	62	8.62×10^{-03}
16	1024	31	8.59

Table 4: Max chi squared value when comparing data to a Poisson fit. Mean = 10, d.o.f.= 26

Replicas	Intervals	Max χ^2 value	Probability (%)
1024	16	1036	1.06×10^{-199}
512	32	527	7.87×10^{-93}
256	64	278	4.60×10^{-42}
128	128	150	1.84×10^{-17}
64	256	83	7.19×10^{-06}
32	512	71	4.92×10^{-04}
16	1024	70	6.08×10^{-04}

Table 5: Max chi squared value when comparing data to a Gaussian fit. Mean = 10, d.o.f.= 26