McGill Physical Journal

Quantitative Analysis of the Radioactive Decay of Cs-137

Kevin Sohn (260782138), Lambert Francis (260861226)

McGill University Department of Physics

June 15, 2022

Abstract

In this experiment we determined the counting rate of a radioactive source of Cs-137 at a distance of 6.90 ± 0.05 cm and determined whether Gaussian or Poissonian distributions better modelled radioactive decay. Four experiments were performed with different polling rates and durations to record the radioactive decay. One additional dataset was provided by the lab technician. By plotting histograms of the experiment data, we determined the count rate of our source to be 12.53 ± 0.05 Bq at a distance of 6.9 ± 0.05 cm. By comparing χ^2 statistics between the Gaussian and Poisson fits to the histograms, we show that the Poisson distribution is a better model for discrete, random events. We note that larger histogram bin sizes allowed the Gaussian to fit much better due to greater data symmetry.

1 Introduction

In 1906, a French physicist named Henri Becquerel discovered radiation by accidentally burning himself from carrying radioactive materials in his pocket [1]. Although it was Becquerel that discovered this phenomenon, it was his doctorate student, Marie Curie, who named it radioactivity [2]. Both would go on to do pioneering work in the field of radiation, getting the unit of radioactive decay named after them for their efforts: becquerels (Bq) being the number of decays per second and curies (Ci) being equal to $3.7 \times 10^{10}\ Bq$ [3].

Geiger counters are instruments designed to detect ionizing radiation. The Geiger-Müller tube inside the Geiger counter is filled with an inert gasSince a high voltage is applied across the tube, when ionizing radiation penetrates through the tube and ionizes the inert gas molecules, it becomes conductive of electricity. This briefly generates a pulse of current that is detected and converted into a clicking sound outputted by an internal speaker [4].

Each radioactive decay that occurs within a material is independent of each other, and the decays happen at a constant half-life which vary from material to material. Because of this, radioactive decay is well described by a Poisson distribution, which is given by the equation

$$P(x;\mu) = \frac{\mu^x e^{-\mu}}{x!}, \tag{1}$$

where x is the number of events in a trial and μ is the average rate. Since the Poisson distribution is a special case of the binomial distribution, if the radioactive decay is measured for a long enough time, the Poisson distribution will transform into a Gaussian distribution due to the central limit theorem [5].

In this experiment, we explore the radioactivity of Cs-137. We fit a Poisson and Gaussian distribution to the raw data organized in different ways to determine the average count rate of Cs-137. We then examine χ^2 and the residuals to quantify the goodness of fit.

Here line breaks

are respected.

2 Materials and Methods

This experiment consisted of recording data at different sampling rates and duration as listed in Table 1. The radioactive source, Cs-137, was held above a Geiger counter with a clamp at a fixed distance. Data was recorded by a Geiger counter connected to a PASCO Interface. The schematic of the experimental setup can be found in Fig. 1. A Capstone program was used to plot histograms during the data taking process. Ambient activity was measured for each sampling rate before every experiment. For experiments E1 to E3, the positions of both the Geiger counter and the source remained unchanged. For experiment E4, the height of the source was adjusted until an average of 5-7 clicks per interval was observed.

Experiment	Duration (min)	Repetitions	Sampling Rate	Distance from Source (cm)
E1	5	3	5 Hz	6.90 ± 0.05
E2	5	3	2 s	6.90 ± 0.05
E3	1	20	10 Hz	6.90 ± 0.05
E4	5	5	2 s	19.8 ± 0.05
Tech	90	1	5 Hz	≈ 4

Table 1: List of settings used for each experiment. The distance from source remained unchanged for experiments E1-E3. The distance from source was increased to 19.8 ± 0.05 *cm* for experiment E4 to achieve 5-7 clicks per interval.

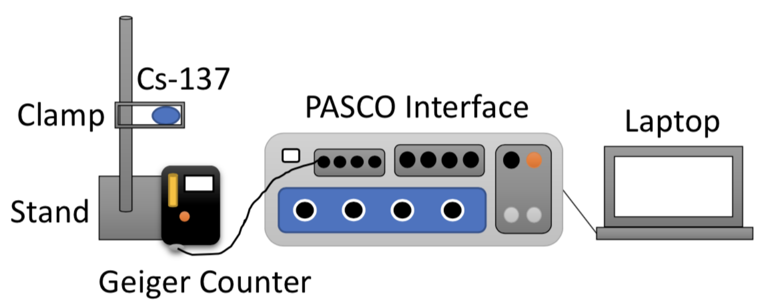


Figure 1: The experimental setup. The height of the clamp can be altered to change the distance from source. A ruler was used to measure the height.

3 Results

Data analysis was performed with Python in Jupyter notebook. The code can be found in Appendix C. The uncertainty of the distance from source was taken to be $\pm 0.05\ cm$, following the standard half a division rule. Both the Poisson and the Gaussian fits were performed with scipy.optimize.curve_fit; it returned a covariance matrix which we used to calculate the error in the fit parameters. The uncertainties in the means were calculated by $\sigma_\mu = \frac{\sigma}{\sqrt{N}}$, where σ is the standard deviation of the sample and N is the number of data points. The results for all of the experiments are complied in Table 2.

Experiment -	Gaussian mean $(\frac{counts}{bin})$	Poisson mean $(\frac{counts}{bin})$	Gaussian rate (Bq)	Poisson rate (Bq)	Count rate (Bq)
E1	2.22 ± 0.06	2.51 ± 0.02	11.1 ± 0.3	12.6 ± 0.1	12.6 ± 0.1
E1 h_3	7.23 ± 0.09	7.49 ± 0.06	12.1 ± 0.2	12.5 ± 0.1	12.6 ± 0.1
E2	24.6 ± 0.2	24.9 ± 0.2	12.3 ± 0.1	12.4 ± 0.1	12.4 ± 0.1
E2 h_3	75.6 ± 0.8	76 ± 1	12.6 ± 0.1	12.7 ± 0.2	12.4 ± 0.1
E3	0.86 ± 0.03	1.261 ± 0.006	8.6 ± 0.3	12.60 ± 0.06	12.6 ± 0.1
E3 h_{20}	24.9 ± 0.2	25.1 ± 0.2	12.5 ± 0.1	12.6 ± 0.1	12.6 ± 0.1
E3 v_{20}	746 ± 2	-	12.43 ± 0.03	-	12.63 ± 0.08
E4	4.5 ± 0.1	4.8 ± 0.1	2.25 ± 0.05	2.40 ± 0.05	2.44 ± 0.04
E4 h_5	24.2 ± 0.5	24.4 ± 0.5	2.42 ± 0.05	2.44 ± 0.05	2.44 ± 0.04
Tech.	1.18 ± 0.05	1.549 ± 0.006	5.9 ± 0.3	7.74 ± 0.03	7.76 ± 0.04
Tech. h_2	2.84 ± 0.06	3.10 ± 0.02	7.1 ± 0.2	7.75 ± 0.05	7.77 ± 0.04
Tech. h_5	7.55 ± 0.06	7.80 ± 0.04	7.55 ± 0.06	7.80 ± 0.04	7.77 ± 0.04
Tech. h_{20}	30.9 ± 0.2	31.2 ± 0.2	7.73 ± 0.05	7.80 ± 0.05	7.77 ± 0.04

Table 2: Gaussian and Poisson fitting parameters and the count rate of Cs-137. The h and v represent horizontally and vertically added data, respectively. The subscripts represent the amount the data was re-binned by. The Poisson mean and Poisson rate for E3 vertical data are not shown because the fit values did not converge.

The decay rate for each fit was determined by $r = \frac{\mu}{np}$, where r is the decay rate in becquerels, μ is the mean of the fit, n is the re-binning factor, and p is the bin width. The uncertainty in r was calculated by $\sigma_r = \frac{\sigma_\mu}{np}$.

By performing a weighted average on the count rates from E1, E2, and E3, we determined the count rates for the Gaussian and Poissonian fits to be 11.9 ± 0.9 Bq and 12.56 ± 0.05 Bq, respectively. Similarly, the count rate average of the data in E1-E3 was determined to be 12.53 ± 0.05 Bq. These values correspond to the count rate at 6.90 ± 0.05 cm, since E1-E3 were performed in position 1.

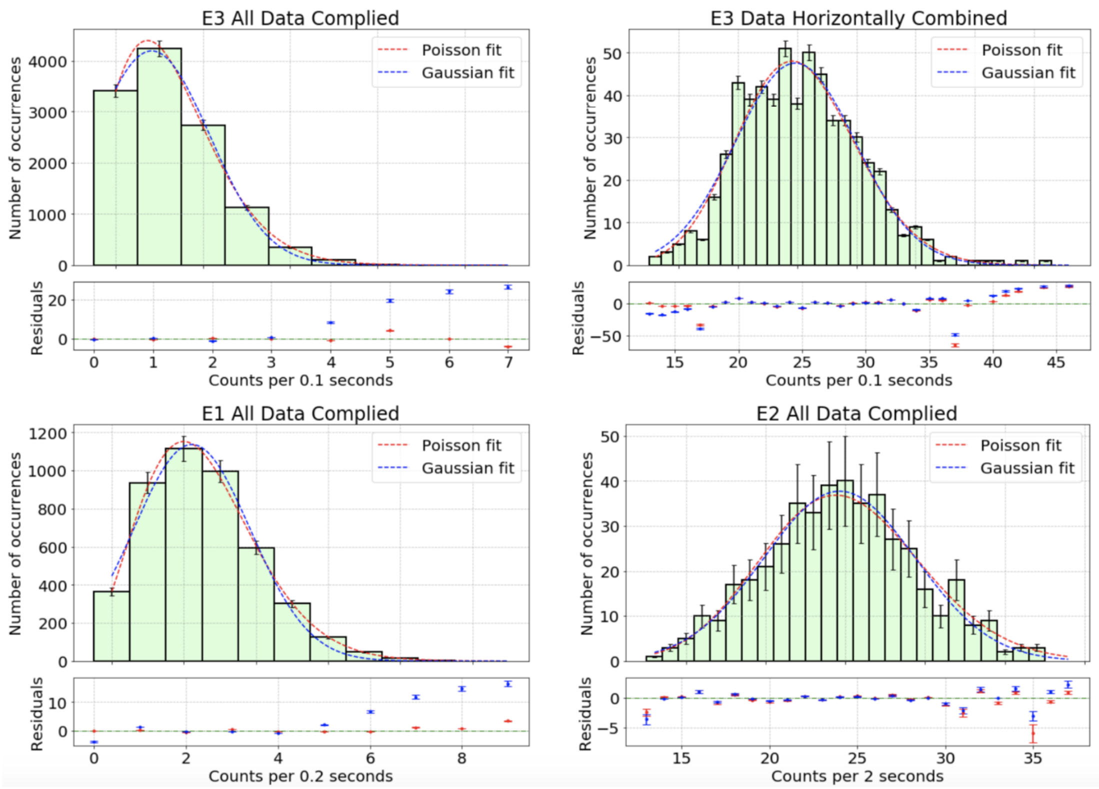


Figure 2: Histograms of the complied data for experiments E1-E3 plus an additional plot of E3 data horizontally added. Gaussian and Poisson fits are overlaid for each histogram.

The errors on the histograms were calculated by averaging the background radiation data and multiplying it by the number of occurrences for each bin. We do this because the error bars are uniform for each data point. Since each bin is the sum of all the data points for a given count, we can factor out the constant and treat it as a multiplication.

The χ^2 values were calculated by $\chi^2 = \sum \frac{(observed-expected)^2}{expected}$ to quantify the goodness of the Poisson and Gaussian fits. The χ^2 values for experiments E1-E4 are listed in Table 3:

Experiment	Poisson χ^2	Poisson df	Gaussian χ^2	Gaussian df
E1	3	8	345	7
E2	13	24	16	23
E3	5	7	494	6
E4	12	12	121	11

Table 3: χ^2 goodness of fit results for E1-E4. df represents the degrees of freedom.

4 Discussion

We determined the radioactivity (count rate) of our Cs-137 to be 12.53 ± 0.05 Bq at a distance of 6.90 ± 0.05 cm from the Geiger counter. Our expected Poisson and Gaussian values were 12.56 ± 0.05 Bq and 11.9 ± 0.9 Bq, respectively. Both of these values are within 3σ of the determined value. While both are consistent, we note that the Poisson value is more accurate and precise than the Gaussian value when compared to the determined value.

Examining the χ^2 values in Table 3, we observe that the Poisson values are much lower than the respective Gaussian values. Since lower χ^2 signifies better fit to the data, we conclude that the Poisson distribution is better at modelling discrete events. However, for experiment E2, we observe a low χ^2 value for the Gaussian as well. Looking at Fig. 2, we can attribute this to the fact that the E2 histogram is more symmetric than the histograms for experiments E1 and E3. Since the Gaussian is a symmetric distribution, it makes sense that the Gaussian is a good fit to the data in this case. By similar reasoning, it makes sense why the Gaussian χ^2 values for experiments E1, E3, and E4 (found in the appendix) are much higher than the Poisson χ^2 values; The Gaussian is not able to fit the skewed data as well as the Poisson. We note that experiments E1 and E3 had bin sizes of 0.1 s and 0.2 s, respectively, relative to the 2 s bin sizes of experiments E2 and E4. This indicates that larger bin sizes allow more symmetric distributions to arise, leading to better Gaussian fits.

5 Conclusions

In this paper, we investigated the radioactivity of Cs-137 and how different distributions model discrete events. We found the count rate of our source to be 12.53 ± 0.05 Bq at a distance of 6.90 ± 0.05 cm. By comparing χ^2 values, we found that the Poisson fit modelled discrete events significantly better than the Gaussian fit. However, the difference between the two fits could be mitigated by increasing bin sizes to allow for more symmetric distributions. Further analysis of the χ^2 values as a function of bin width should be performed to better show how the Poisson distribution becomes more Gaussian with larger bin widths. The experiment can be improved by addressing sources of error listed in Appendix B.

Author Contribution Statement: K.S and L.F contributed equally to the experiment and the report.

References

[1] "Radiation - Historical background — Britannica." [Online]. Available: https://www.britannica.com/science/radiation/Historical-background 1

[2] "The History of Radiation." [Online]. Available: https://www.mirion.com/learning-center/radiation-safety-basics/the-history-of-radiation 1

[3] "Radiation Units and Conversion Factors - Radiation Emergency Medical Management." [Online]. Available: https://www.remm.nlm.gov/radmeasurement.htm 1

[4] "What is a Geiger counter? - It's a Question of Physics - The Atomic Age - Linda Hall Library - Kansas City, MO." [Online]. Available: https://atomic.lindahall.org/what-is-a-geiger-counter.html 1

[5] "Central Limit Theorem." [Online]. Available: http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704{_}Probability/BS704{_}Probability12.html 1

A Bonus Analysis

Data was taken at 0.00 ± 0.05 cm, 6.90 ± 0.05 cm, and 15.00 ± 0.05 cm, to demonstrate the inverse square law of radiation. We would have taken more data points but we did not have enough time. However, the values we do have are consistent with an inverse square relationship.

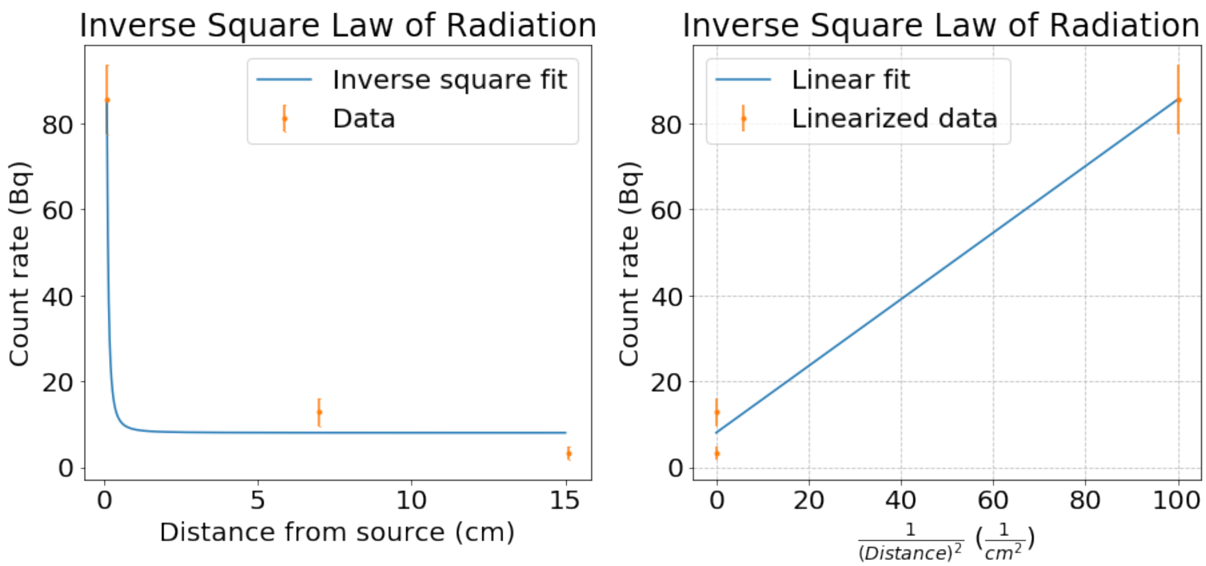


Figure 3: We demonstrate the inverse square law of radiation with two plots: One plot of count rate as a function of distance from source and one linearized fit.

If the inverse square relationship was true, we would expect a linear relationship when plotted against $\frac{1}{distances^2}$. Since the best fit line is within about 2σ of the error bars on the data, we conclude that the inverse square law of radiation is a reasonable model for count rate as a function of distance. More data points should be taken to further confirm this model's consistency.

We believe the reason for the inverse square law holding is because radiation is like flux, which we know is proportional to the inverse square of the distance from source. More specifically, since the decay detection is proportional to the surface area of the Geiger counter as a fraction of the total emission area, we expect an inverse square relationship with distance.

B Lab Notebook

Lab 6 Geiger Counter

* Measure background radiation to calibrate.

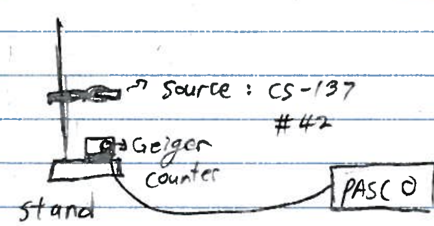
Source: ^{137}Cs or Cs-137. number 42.
↳ emits γ radiation.

* poisson if counts are rel. rare
  independent events
  average rate do not change sig.

↳ $P(X;\mu) = \frac{\mu^x e^{-\mu}}{x!}$

X: events in a trial
μ: average rate

Sources of error:
- Radioactive source not far enough → background radiation biased.
- Source is not fresh
  ↳ radiates less frequently.
- Experiment disturbed during measurement
- Clamp blocking some radiation from the Geiger counter.
- Tilted sample → not flat



Experiment	Distance from Source (cm)	# Rep.	Runtime
5Hz : E1	6.90 ± 0.05 → manual	3	5min
2s : E2	" "	3	5min
10Hz : E3	" " maybe loosen to	20	1 min
2s : E4	19.8 ± 0.05 ±0.01 cm	5	5min

Bonus :

5Hz
5 min.

Trial	Distance (cm)
1	0.00 ± 0.05
2	2.10 ± 0.05
3	4.00 ± 0.05
4	6.90 ± 0.05
5	15.00 ± 0.05

C Python Code

In [30]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as opt
from mpl_toolkits.axes_grid1 import make_axes_locatable

plt.rc("font", size=20)
plt.rc("xtick", labelsize=20)
plt.rc("ytick", labelsize=20)
```

In [3]:

```
# Fit functions
def fish(n, lamb, norm):
    from scipy.special import factorial
    return norm * np.exp(-lamb)*lamb**n / factorial(n)

def goose(x, mew, sigma, norm):
    import numpy as np
    return norm/(sigma*np.sqrt(2*np.pi)) * np.exp(-.5 * ((x-mew)
/sigma)**2)

def chisq(observed, expected):
    chisq = 0
    for i in range(len(observed)):
        if observed[i] != 0:
            chisq += (observed[i]-expected[i])**2/expected[i]
    return chisq
```

In [4]:

```
def create_bins(data):
    x1 = int(min(data))
    x2 = int(max(data))+1
    bins = np.arange(x1, x2)
    #bins = [i for i in range(x1+1, x2+1)]
    counts, bin_edges = np.histogram(data, bins)
    bins = bins[:-1]
    return counts, bins, x1, x2
```

In [5]:

```
def plotHistogram(counts, bins, x1, x2, background, title, xlabe
l, mean=25, sigma=2, maxcalls=1000, width=1):
    # HISTOGRAM CREATION CENTER

    # Combined fit for possion and gauss
    params_p, cov_p = opt.curve_fit(fish, bins, counts, p0=[mean
, np.sum(counts)], maxfev=maxcalls)
    params_g, cov_g = opt.curve_fit(goose, bins, counts, p0=[mea
n, sigma, np.sum(counts)], maxfev=maxcalls)

    # Printing fit results and error
    print('Poisson parameters:', params_p)
    print('Poisson uncertainty:', np.sqrt(cov_p[0][0]))
    print('Gaussian parameters:', params_g)
    print('Gaussian uncertainty:', np.sqrt(cov_g[0][0]), np.sqrt
(cov_g[1][1]))

    # Axes for plotting
    fig = plt.figure(1, figsize=(9,6))
    ax = fig.add_axes([.1, .4, .9, .7])
    ax_res = fig.add_axes([.1, .15, .9, .2]) # left, bottom, wid
th height

    # Plotting the combined data
    ax.bar(bins, counts, edgecolor='black', width=width, linewid
th=2, fc=(0,1,0,.15))
    ax.errorbar(bins, counts, yerr = counts*background, fmt="non
e",  ecolor="xkcd:black", capsize=3)

    # Plotting the fits
    x = np.arange(x1, x2, step=0.01)
    ax.plot(x, fish(x, *params_p), linestyle='dashed', label='Po
isson fit', color="red")
    ax.plot(x, goose(x, *params_g), linestyle='dashed', label='G
aussian fit', color="blue")

    # Finding residuals and residual errors
    res_g = (counts - goose(bins, *params_g)) / (counts*backgrou
nd)
    res_p = (counts - fish(bins, *params_p)) / (counts*backgroun
d)

    # Plotting residuals
    ax_res.errorbar(bins, res_p, yerr = res_p*background, fmt='.
```

```
', color="red", capsize=5)
    ax_res.errorbar(bins, res_g, yerr = res_g*background, fmt='.
', color="blue", capsize=5)
    ax_res.axhline(y=0, linestyle="-.", color='green', linewidth
=1)

    # Aesthetics
    ax.set_title(title)
    ax.set_ylabel('Number of occurrences')
    ax.set_xticklabels([])
    ax.grid(color='grey', linestyle='--', alpha=.5)
    ax.legend()
    ax_res.grid(color='grey', linestyle='--', alpha=.5)
    ax_res.set_xlabel(xlabel)
    ax_res.set_ylabel('Residuals')

    return params_p, params_g
```

In [6]:

```
# Load background data
back_2s_c0, back_2s_c1 = np.loadtxt('Data/background_2s.csv', de
limiter=',').T # T/R
back_5hz_c0, back_5hz_c1 = np.loadtxt('Data/background_5hz.csv',
delimiter=',').T # T/R
back_10hz_c0, back_10hz_c1 = np.loadtxt('Data/background_10hz.cs
v', delimiter=',').T # T/R

# quantifying error
back_2s_avg = np.mean(back_2s_c1)
back_5hz_avg = np.mean(back_5hz_c1)
back_10hz_avg = np.mean(back_10hz_c1)
print(back_2s_avg, back_5hz_avg, back_10hz_avg)
```

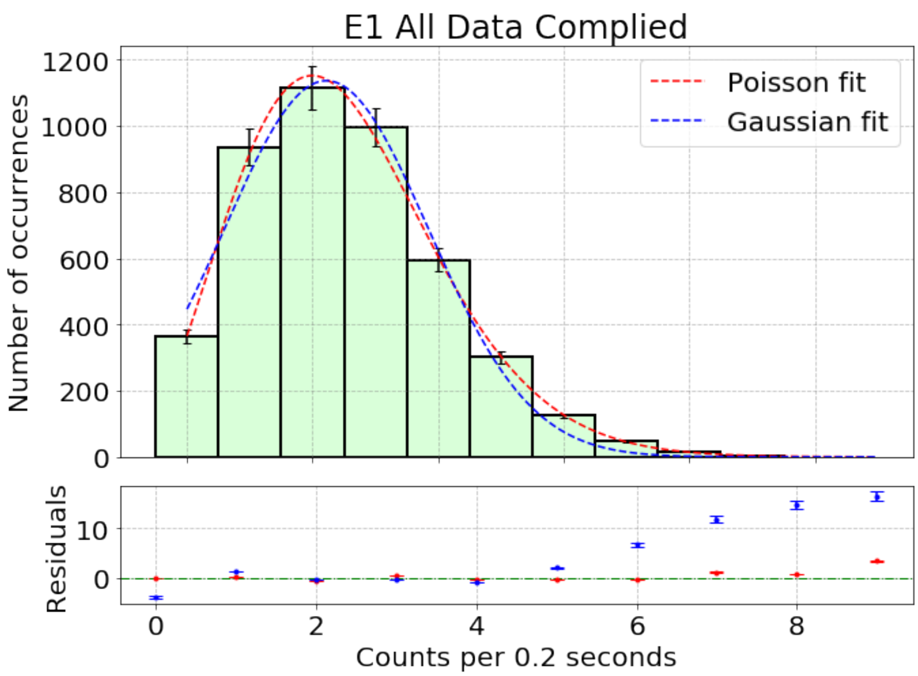
```
0.7866666666666666 0.05862758161225849 0.03660565723
7936774
```

In [7]:

```
# importing E1 data
E1_c0, E1_c1, E1_c2, E1_c3, E1_c4, E1_c5 = np.loadtxt('Data/E1.c
sv', delimiter=",").T

# combining data horizontally
E1 = np.concatenate([E1_c1, E1_c3, E1_c5])

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E1)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_5h
z_avg, title="E1 All Data Complied",
                                   xlabel="Counts per 0.2 second
s", mean=25, sigma = 5)

# count rate for both fits
print(params_p[0], params_g[0])

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

```
Poisson parameters: [2.51059025e+00 4.49870361e+03]
Poisson uncertainty: 0.02126698412328205
Gaussian parameters: [2.22069619e+00 1.62431400e+00
4.62597711e+03]
Gaussian uncertainty: 0.06499451906690284 0.07027837
692224728
2.510590245929664 2.2206961892695234
3.2287997972486764
345.08697681265795
```

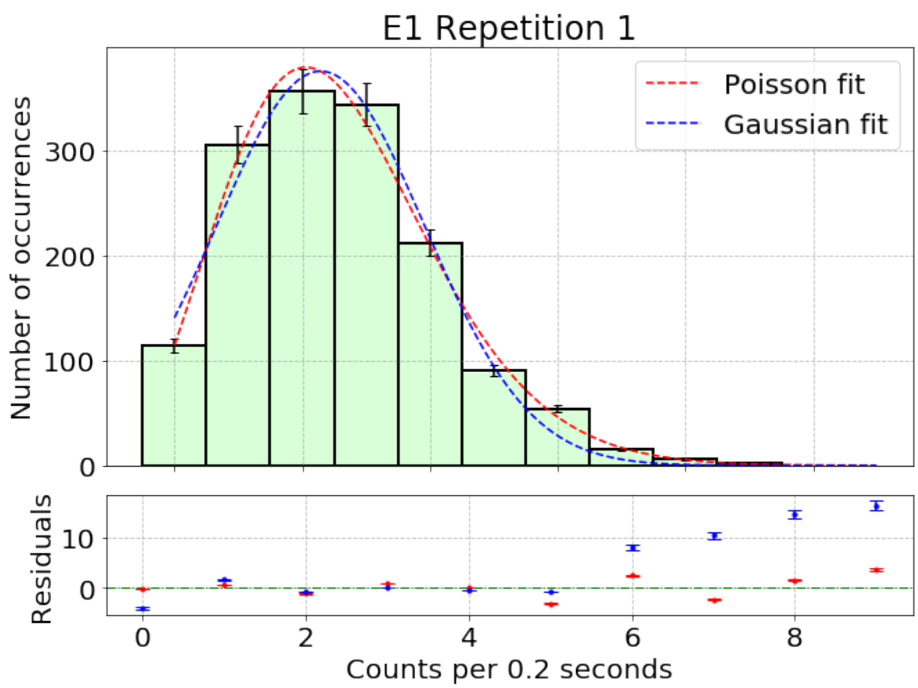
```
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:4: RuntimeWarning: over
flow encountered in exp
  after removing the cwd from sys.path.
```



In [8]:

```
# importing E1 data
E1_c0, E1_c1, E1_c2, E1_c3, E1_c4, E1_c5 = np.loadtxt('Data/E1.c
sv', delimiter=",").T

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E1_c1)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_5h
z_avg, title="E1 Repetition 1",
                                   xlabel="Counts per 0.2 second
s", mean=2.2, sigma=1)

# chi-square
#x = np.linspace(0, x2, len(counts))
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

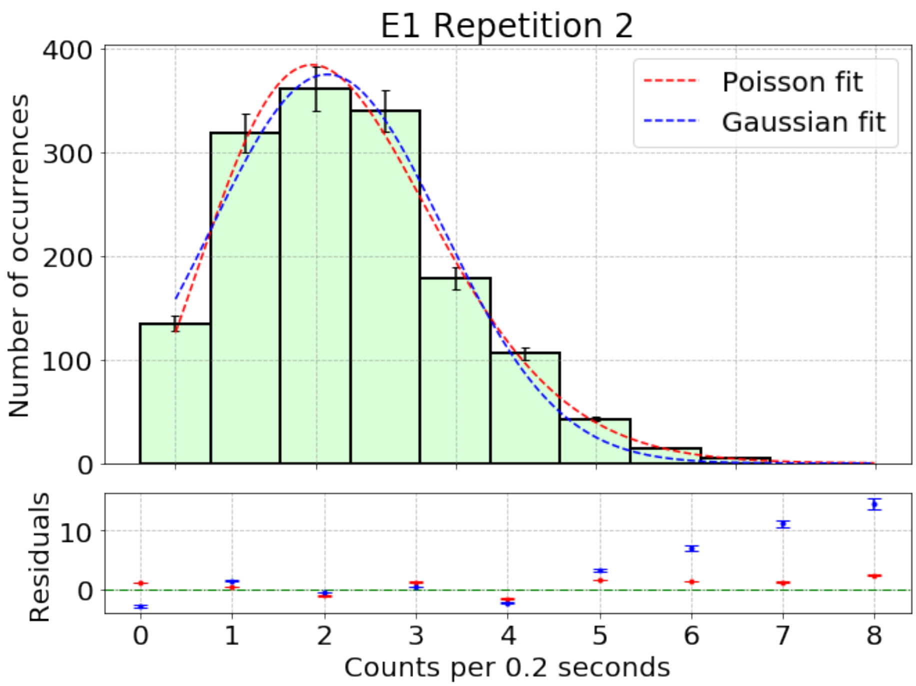
```
Poisson parameters: [   2.57296121 1501.49336456]
Poisson uncertainty: 0.04550402373180329
Gaussian parameters: [   2.28763361    1.63268837 15
38.14590707]
Gaussian uncertainty: 0.07259212739590987 0.07812086
306814216
7.16375932590027
125.18506516511256
```



In [9]:

```
# importing E1 data
E1_c0, E1_c1, E1_c2, E1_c3, E1_c4, E1_c5 = np.loadtxt('Data/E1.c
sv', delimiter=",").T


# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E1_c3)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_5h
z_avg, title="E1 Repetition 2",
                                   xlabel="Counts per 0.2 second
s", mean=2.2, sigma=1)

# chi-square
#x = np.linspace(0, x2, len(counts))
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

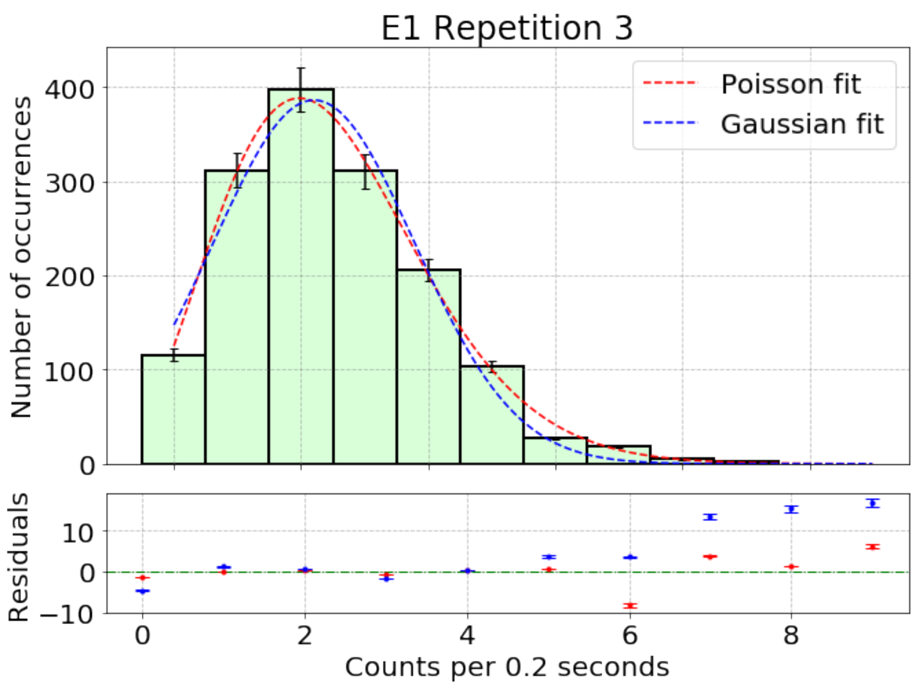
```
Poisson parameters: [   2.4687191 1487.155264 ]
Poisson uncertainty: 0.05280038596299464
Gaussian parameters: [   2.16933753    1.65153343 15
51.54083274]
Gaussian uncertainty: 0.08384690338652077 0.09144749
695642482
6.9519296210972366
69.22305596000967
```



In [10]:

```
# importing E1 data
E1_c0, E1_c1, E1_c2, E1_c3, E1_c4, E1_c5 = np.loadtxt('Data/E1.c
sv', delimiter=",").T


# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E1_c5)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_5h
z_avg, title="E1 Repetition 3",
                                   xlabel="Counts per 0.2 second
s", maxcalls=2000,
                                  mean = 2.2, sigma=1)

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

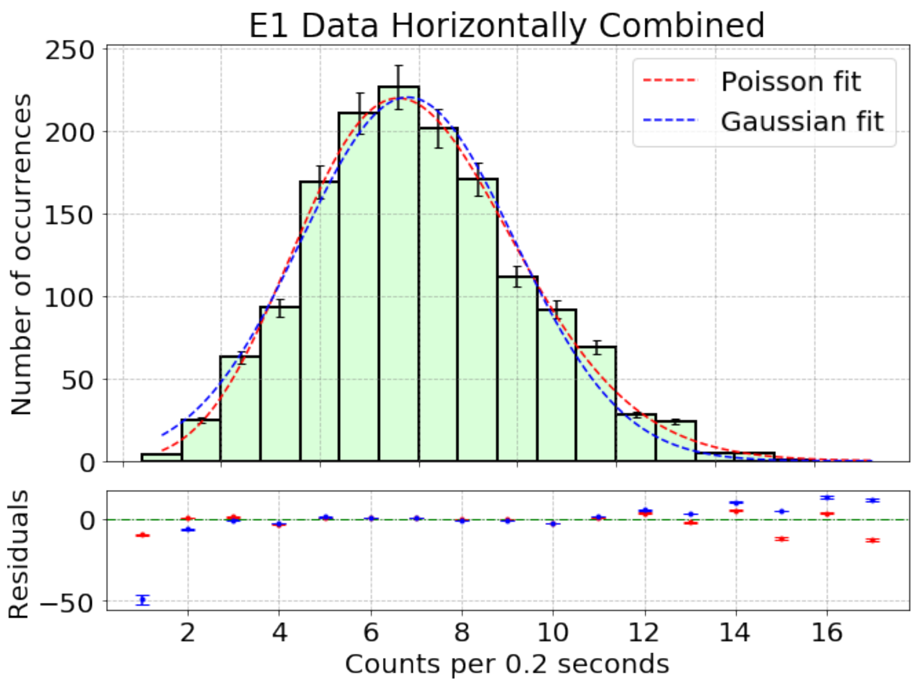
```
Poisson parameters: [   2.49058843 1510.83599245]
Poisson uncertainty: 0.028500884479569808
Gaussian parameters: [   2.20354848    1.58692378 15
36.5012758 ]
Gaussian uncertainty: 0.07849860105811239 0.08452449
146212408
7.676721940374009
212.37285727611103
```



In [11]:

```
# importing E1 data
E1_c0, E1_c1, E1_c2, E1_c3, E1_c4, E1_c5 = np.loadtxt('Data/E1.c
sv', delimiter=",").T

# combining data horizontally
E1 = E1_c1 + E1_c3 + E1_c5

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E1)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_5h
z_avg, title="E1 Data Horizontally Combined",
                                   xlabel="Counts per 0.2 second
s")

# Count rate
print(params_p[0], params_g[0])

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

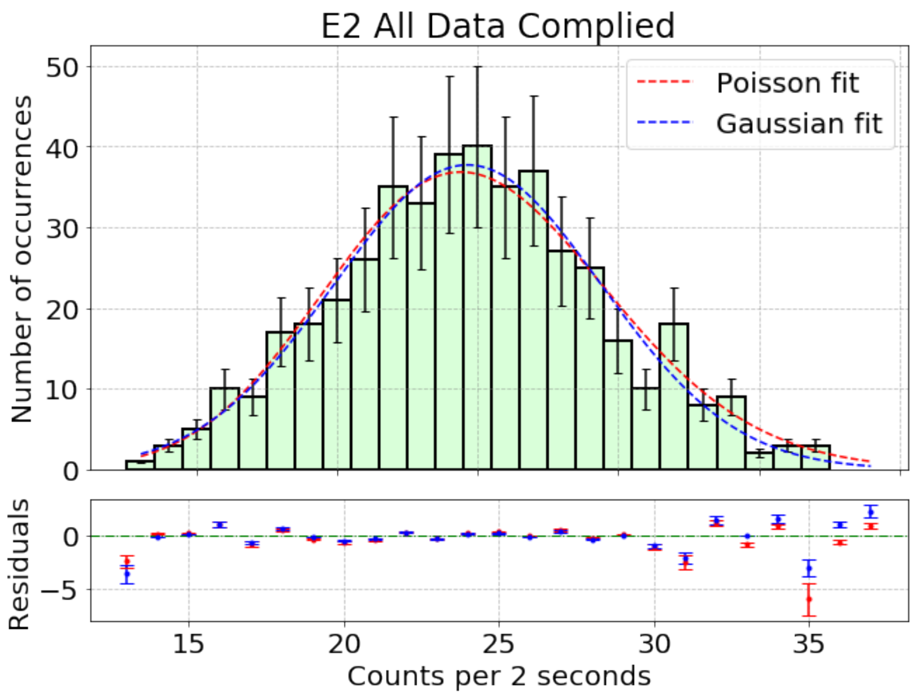
```
Poisson parameters: [   7.49207664 1500.63012299]
Poisson uncertainty: 0.0653076620116848
Gaussian parameters: [   7.22863082    2.70442708 14
94.67814656]
Gaussian uncertainty: 0.09732685735483922 0.09808447
796943795
7.492076641338146 7.228630818544421
15.922313595952103
68.00374312569399
```



In [12]:

```
# importing E2 data
E2_c0, E2_c1, E2_c2, E2_c3, E2_c4, E2_c5 = np.loadtxt('Data/E2.c
sv', delimiter=',').T # T/R T/R T/R

# combining data horizontally
E2 = np.concatenate([E2_c1, E2_c3, E2_c5])

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E2)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, 0.25, t
itle="E2 All Data Complied",
                                   xlabel="Counts per 2 seconds"
, mean=25, sigma = 5)

# count rate for both fits
print(params_p[0], params_g[0])

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

```
Poisson parameters: [ 24.8912728 459.8290804]
Poisson uncertainty: 0.18821915306656548
Gaussian parameters: [ 24.63006625   4.76046288 449.
99679842]
Gaussian uncertainty: 0.17559243365357652 0.17733691
348287905
24.891272801871796 24.630066251865756
12.73278187515748
15.991763397712926
```

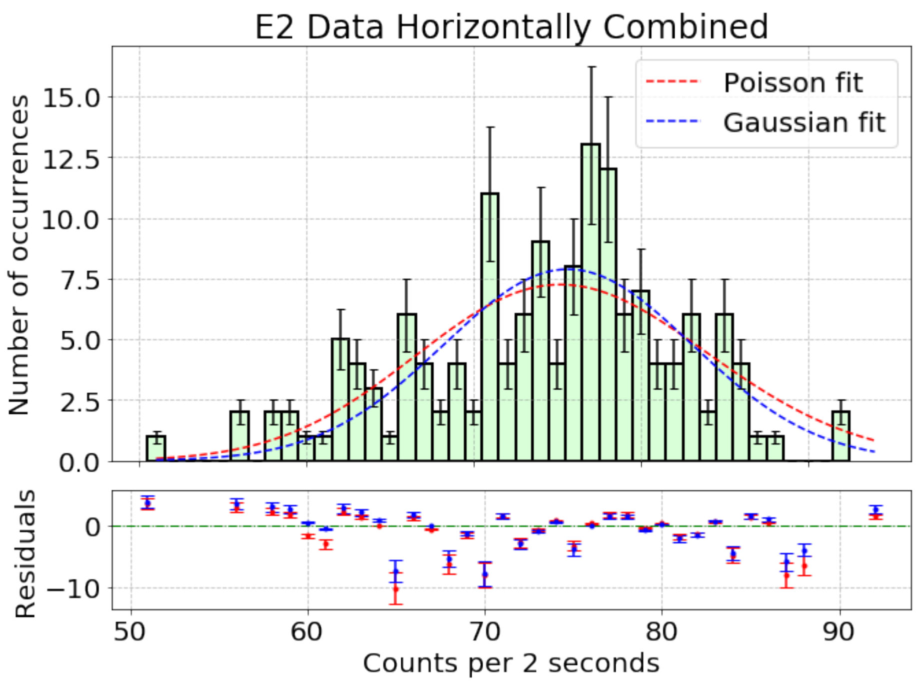


In [13]:

```
# importing E2 data
E2_c0, E2_c1, E2_c2, E2_c3, E2_c4, E2_c5 = np.loadtxt('Data/E2.c
sv', delimiter=',').T # T/R T/R T/R

# combining data horizontally
E2 = E2_c1 + E2_c3 + E2_c5

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E2)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, 0.25, t
itle="E2 Data Horizontally Combined",
                                   xlabel="Counts per 2 seconds"
, mean=75)

# count rate for both fits
print(params_p[0], params_g[0])

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

```
Poisson parameters: [ 75.69066076 158.03175497]
Poisson uncertainty: 0.9748991481471277
Gaussian parameters: [ 75.63904057   7.43213759 146.
8268669 ]
Gaussian uncertainty: 0.8063978348863786 0.820864904
0377604
75.69066076220639 75.63904057110966
50.21981804281228
90.06593036025447
```

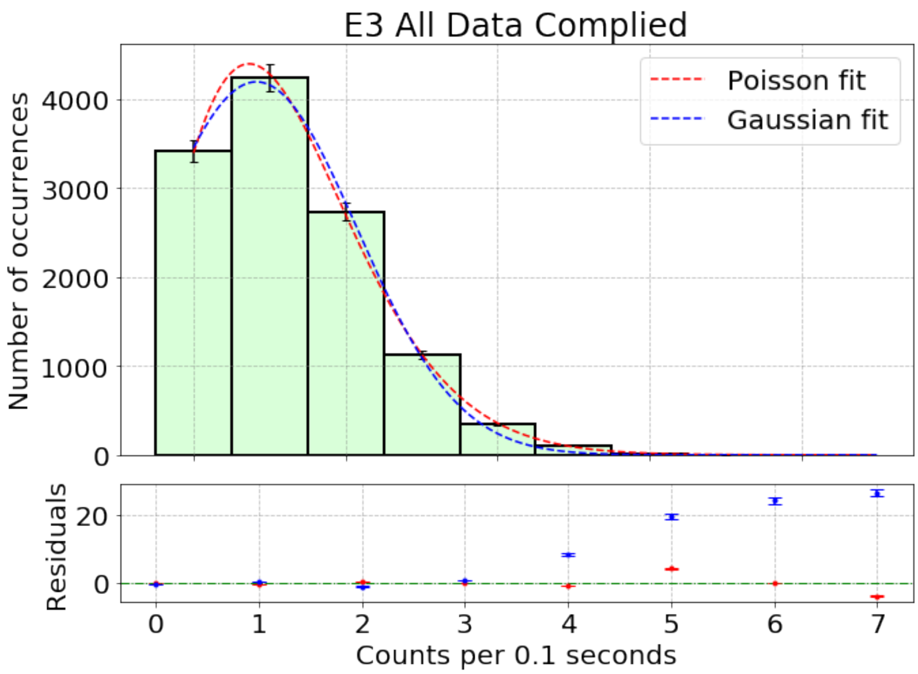
```
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:29: RuntimeWarning: div
ide by zero encountered in true_divide
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:30: RuntimeWarning: div
ide by zero encountered in true_divide
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/matplotlib/axes/_axes.py:3370: RuntimeWarning
: invalid value encountered in double_scalars
  low = [v - e for v, e in zip(data, a)]
```



In [14]:

```
# importing E3 data
E3_c0, E3_c1 = np.loadtxt('Data/E3.csv', delimiter=',').T # T/R

# Raw combined data
E3 = E3_c1

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E3)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_10
hz_avg, title="E3 All Data Complied",
                                   xlabel="Counts per 0.1 second
s", mean=2 ,sigma=1)

# count rate for both fits
print(params_p[0], params_g[0])

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

```
Poisson parameters: [1.26111102e+00 1.19827237e+04]
Poisson uncertainty: 0.006303769315925155
Gaussian parameters: [8.26569021e-01 1.32645376e+00
1.39481157e+04]
Gaussian uncertainty: 0.036420175918340354 0.0403795
1277964953
1.261111021975436 0.8265690205117144
4.568506565060981
494.069078225684
```

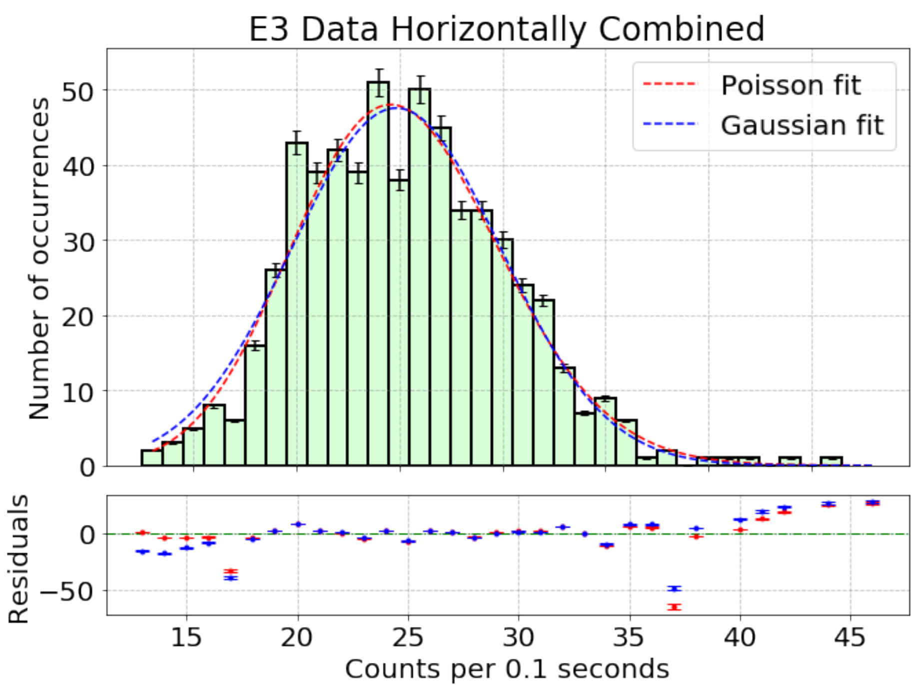


In [15]:

```
# importing E3 data
E3_c0, E3_c1 = np.loadtxt('Data/E3.csv', delimiter=',').T # T/R

# splitting a 20 min run into 20 1 min runs
E3_c1 = np.array_split(E3_c1, 20)
E3_c1[0] = np.delete(E3_c1[0], 0) # deleting extra entry

# E3 horizontally combined
E3_h = np.zeros(600)
for i in range(0,20):
    E3_h += E3_c1[i]

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E3_h)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_10
hz_avg, title="E3 Data Horizontally Combined",
                                   xlabel="Counts per 0.1 second
s")

# count rate for both fits
print(params_p[0], params_g[0])

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

```
Poisson parameters: [ 25.08515463 601.83847277]
Poisson uncertainty: 0.1880803024958489
Gaussian parameters: [ 24.85058599   5.09115391 606.
99702275]
Gaussian uncertainty: 0.20335484411990262 0.20544635
401495367
25.085154629991994 24.850585989061386
56.09989357185331
166.92157368933127
```

```
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:29: RuntimeWarning: div
ide by zero encountered in true_divide
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:30: RuntimeWarning: div
ide by zero encountered in true_divide
```

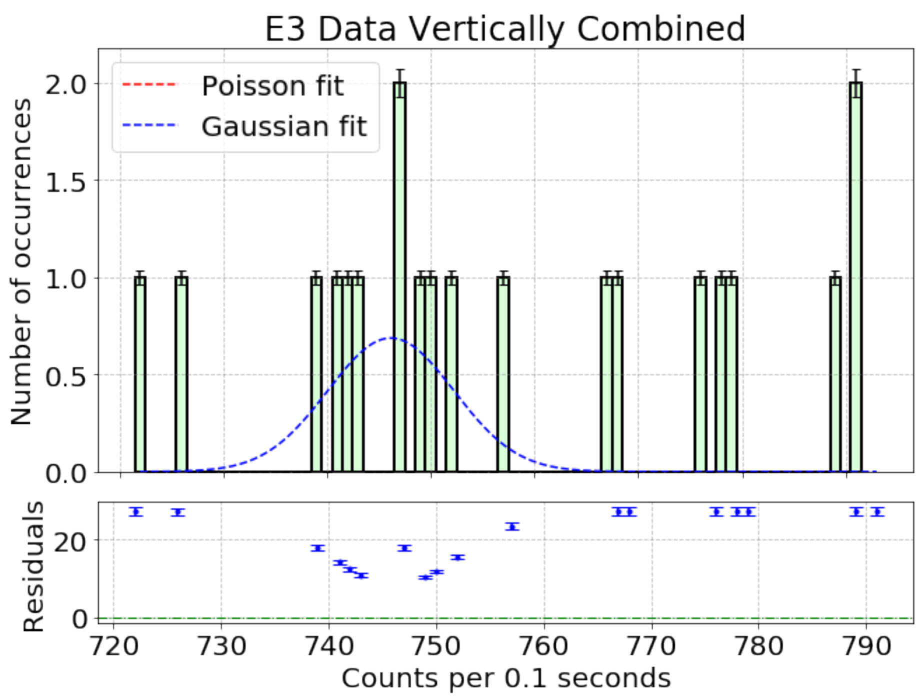


In [16]:

```
# E3 vertically combined
E3_v = np.zeros(0)
for c in E3_c1:
    E3_v = np.append(E3_v, np.sum(c))
print(E3_v)

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E3_v)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_10
hz_avg, title="E3 Data Vertically Combined",
                                   xlabel="Counts per 0.1 second
s", mean=750)

# count rate for both fits
print(params_p[0], params_g[0])

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

```
[739. 789. 779. 768. 776. 752. 741. 778. 792. 722. 7
67. 747. 726. 743.
 757. 749. 747. 742. 791. 750.]
Poisson parameters: [750.  20.]
Poisson uncertainty: inf
Gaussian parameters: [746.16485072   6.07367073  10.
46346395]
Gaussian uncertainty: 2.023926383166391 2.0239288108
151476
750.0 746.1648507188406
nan
4052622148688.2095
```

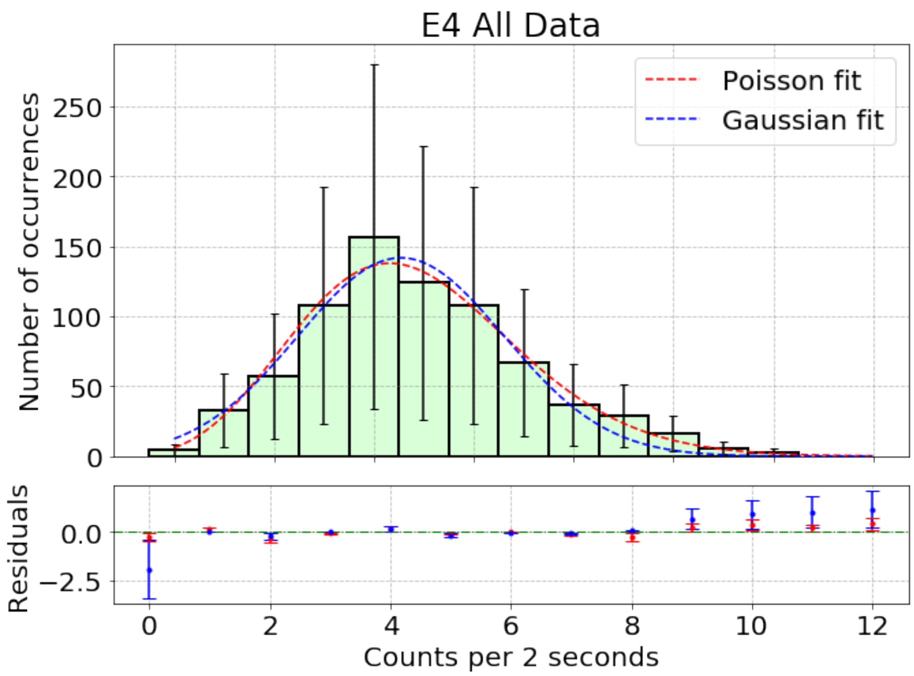
```
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:4: RuntimeWarning: over
flow encountered in power
  after removing the cwd from sys.path.
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:4: RuntimeWarning: inva
lid value encountered in multiply
  after removing the cwd from sys.path.
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/scipy/optimize/minpack.py:795: OptimizeWarnin
g: Covariance of the parameters could not be estimat
ed
  category=OptimizeWarning)
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:29: RuntimeWarning: div
ide by zero encountered in true_divide
```



In [17]:

```
E4 = np.loadtxt('Data/E4.csv', delimiter=',').T # T/R T/R T/R T/
R T/R

# E4 horizontally combined
E4 = np.concatenate([E4[1], E4[3], E4[5], E4[7], E4[9]])

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E4)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_2s
_avg, title="E4 All Data",
                                   xlabel="Counts per 2 seconds"
)

# count rate for both fits
print(params_p[0], params_g[0])

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

```
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:4: RuntimeWarning: over
flow encountered in exp
  after removing the cwd from sys.path.

Poisson parameters: [  4.82525422 753.26827372]
Poisson uncertainty: 0.09998733339718004
Gaussian parameters: [  4.54395265   2.06143657 733.
16742881]
Gaussian uncertainty: 0.11661176440543765 0.11762437
907667075
4.825254220774733 4.54395265491743
12.427090584984583
121.16740418055284
```

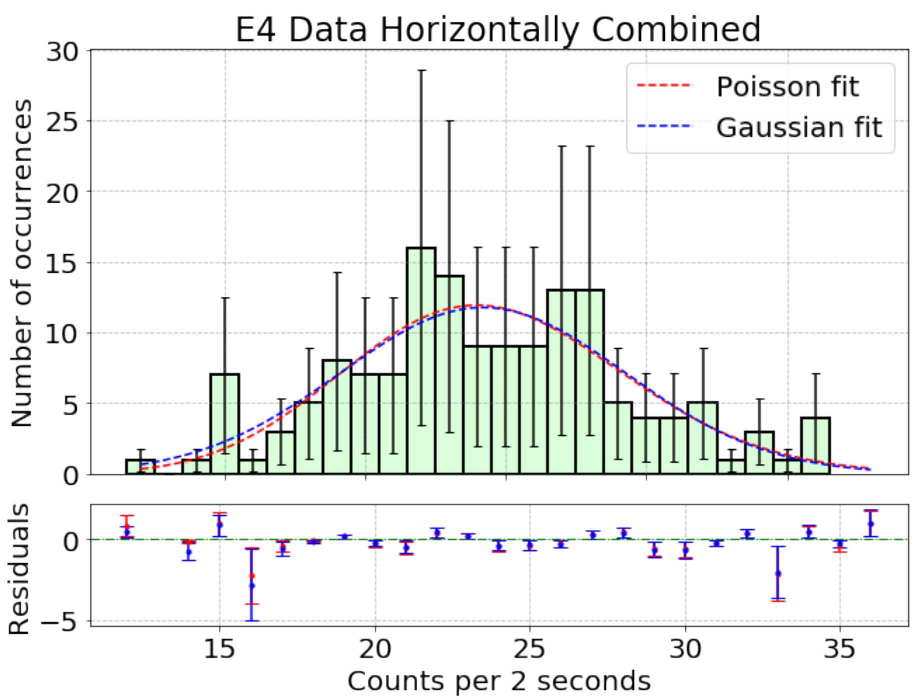


In [18]:

```
E4 = np.loadtxt('Data/E4.csv', delimiter=',').T # T/R T/R T/R T/
R T/R

# E4 horizontally combined
E4_h = E4[1]+E4[3]+E4[5]+E4[7]+E4[9]

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E4_h)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_2s
_avg, title="E4 Data Horizontally Combined",
                                   xlabel="Counts per 2 seconds"
)

# count rate for both fits
print(params_p[0], params_g[0])

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

```
Poisson parameters: [ 24.45560253 147.4673175 ]
Poisson uncertainty: 0.5031305127911868
Gaussian parameters: [ 24.18080747   5.07870072 149.
64257066]
Gaussian uncertainty: 0.5278811263415544 0.537647334
0061113
24.455602528701654 24.18080746876673
40.67604119613295
37.919445372144374
```

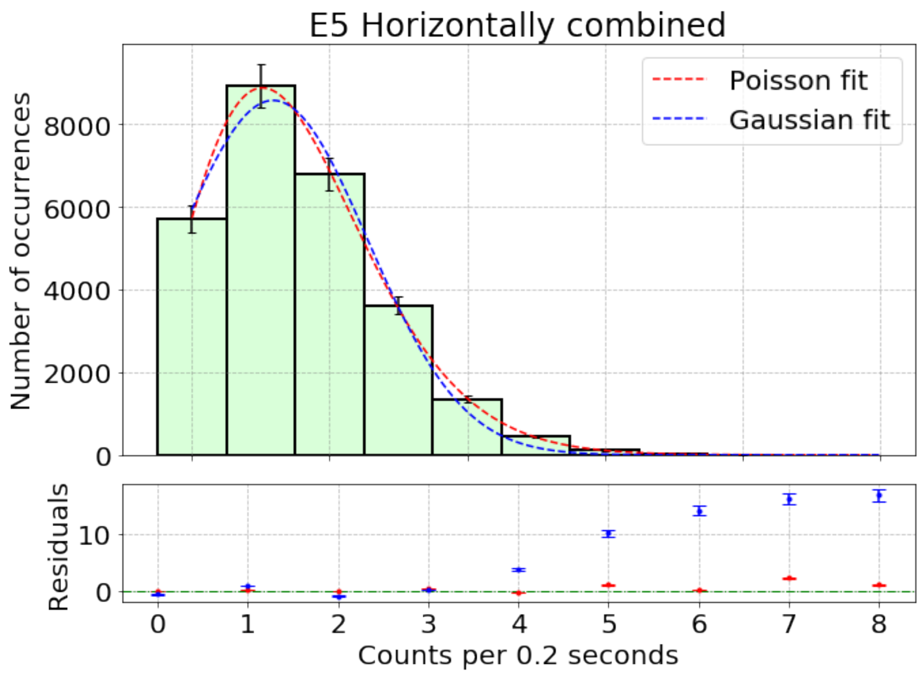
```
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:29: RuntimeWarning: div
ide by zero encountered in true_divide
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:30: RuntimeWarning: div
ide by zero encountered in true_divide
```



In [19]:

```
# E5 plotting with bin width of 1
E5_c0, E5_c1 = np.loadtxt('Data/techtue2020.csv', delimiter=',')
.T

# getting parameters for histogram
counts, bins, x1, x2 = create_bins(E5_c1)

# plotting histogram
params_p, params_g = plotHistogram(counts, bins, x1, x2, back_5h
z_avg, title="E5 Horizontally combined",
                                   xlabel="Counts per 0.2 second
s", mean=1)

# count rate for both fits
print(params_p[0], params_g[0])

# chi-square
exp_p = fish(bins, *params_p)
exp_g = goose(bins, *params_g)
print(chisq(counts, exp_p))
print(chisq(counts, exp_g))
```

```
Poisson parameters: [1.54892679e+00 2.69823270e+04]
Poisson uncertainty: 0.006720390045282371
Gaussian parameters: [1.18033023e+00 1.37632019e+00
2.95938388e+04]
Gaussian uncertainty: 0.05403518210152215 0.0615284
2104089673
1.5489267885719213 1.1803302290487665
5.823910597913101
2253.8063257189806
```



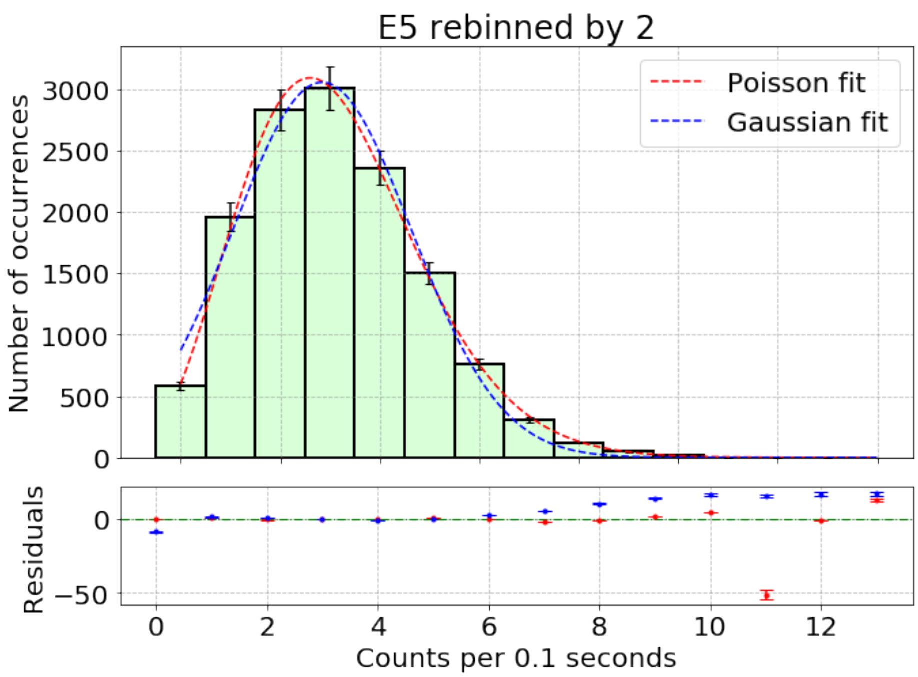
In [20]:

```
# # E5 plotting with twice bin width
# print(bins)
# rebin = np.array_split(E5_c1, len(bins)/2)
# E5 = np.zeros(0)
# for elem in rebin:
#     E5 = np.append(E5, np.sum(elem))
# print(E5)

# # plotting histogram
# params_p, params_g = plotHistogram(counts, E5, x1, x2, title="
E5 Data Horizontally Combined",
#                                     xlabel="Counts per 0.2 seco
nds")

# # count rate for both fits
# print(params_p[0], params_g[0])

arr = np.array_split(E5_c1, 2)
print(arr)
arr[0] = np.delete(arr[0], 0)
ci = arr[0] + arr[1]

print(max(ci))
print(min(ci))

counts, bins = np.histogram(ci, bins=14)
bins = bins[0:-1]
plotHistogram(counts, bins, min(ci), max(ci), back_5hz_avg, titl
e="E5 rebinned by 2", xlabel="Counts per 0.1 seconds", mean=3, s
igma=.8, maxcalls=2000,
             width=1)
```

```
[array([2., 1., 1., ..., 1., 1., 3.]), array([2., 0.
, 1., ..., 0., 0., 2.])]
14.0
0.0
Poisson parameters: [3.11327585e+00 1.34811699e+04]
Poisson uncertainty: 0.017582487586878325
Gaussian parameters: [2.84201216e+00 1.79534623e+00
1.37400622e+04]
Gaussian uncertainty: 0.05753732872394929 0.06064049
858617605
```

Out[20]:

```
(array([3.11327585e+00, 1.34811699e+04]),
 array([2.84201216e+00, 1.79534623e+00, 1.37400622e+
04]))
```



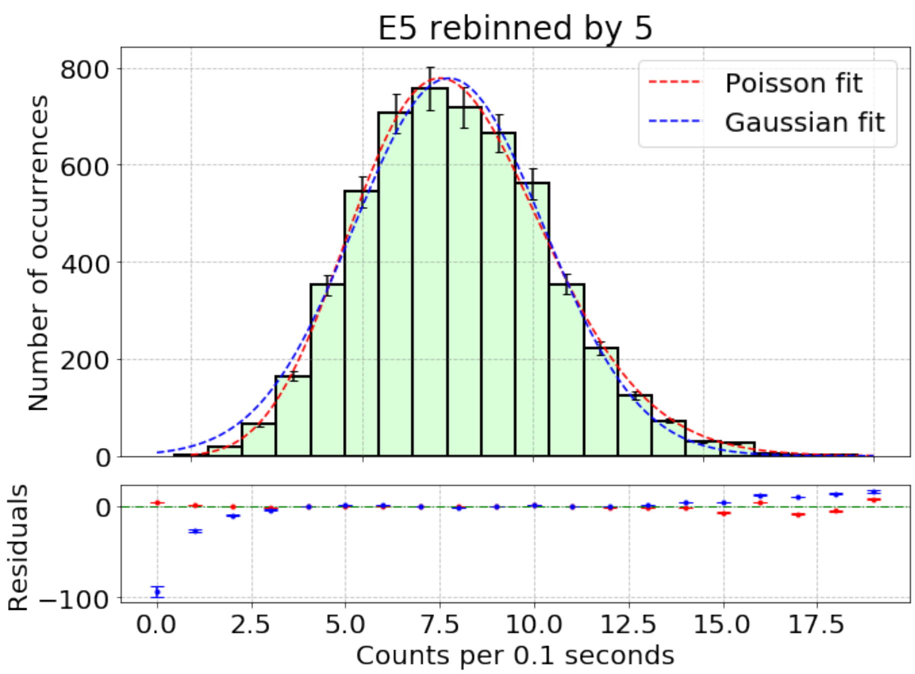
In [21]:

```
# E5 plotting with five times times bin width
E5 = np.loadtxt('Data/techtue2020.csv', delimiter=',').T

arr = np.array_split(E5[1], 5)
arr[0] = np.delete(arr[0], 0)

ci = np.zeros(len(arr[0]))
for x in arr:
    ci += x

print(max(ci))
print(min(ci))
size = int(max(ci) - min(ci))
counts, bins = np.histogram(ci, bins=size)
bins = bins[0:-1]
plotHistogram(counts, bins, -1, max(ci), back_5hz_avg, title="E5
rebinned by 5", xlabel="Counts per 0.1 seconds", mean=3, sigma=.
8, maxcalls=2000,
             width=1)
```

```
20.0
0.0
Poisson parameters: [   7.79826116 5419.24266415]
Poisson uncertainty: 0.03962528590795676
Gaussian parameters: [7.55324410e+00 2.78205186e+00
5.42575948e+03]
Gaussian uncertainty: 0.06007513918504601 0.06015496
010508763

/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:4: RuntimeWarning: divi
de by zero encountered in true_divide
  after removing the cwd from sys.path.
```

Out[21]:

```
(array([   7.79826116, 5419.24266415]),
 array([7.55324410e+00, 2.78205186e+00, 5.42575948e+
03]))
```



In [22]:

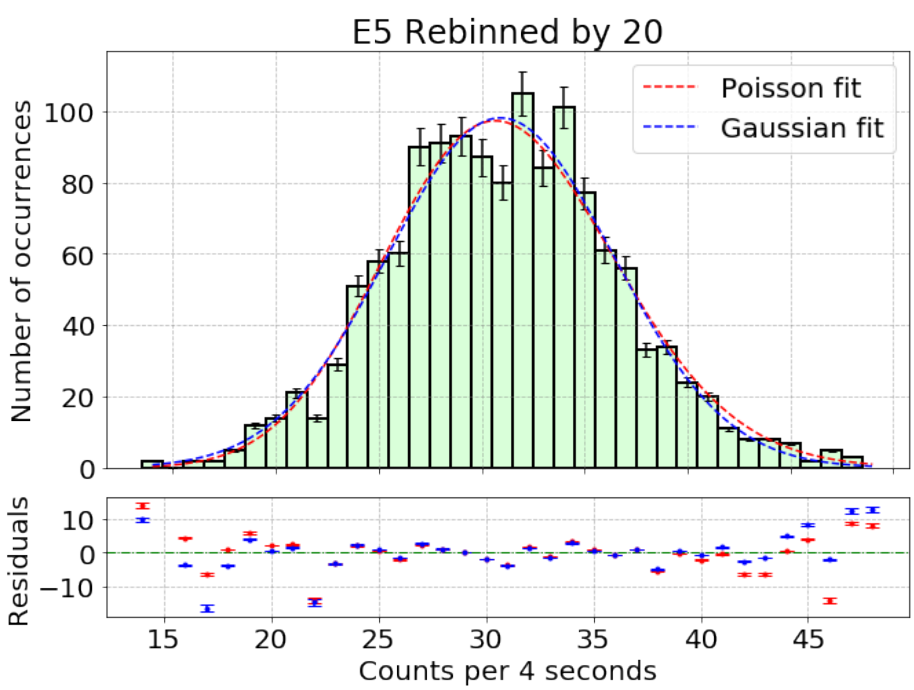
```
# E5 plotting with 20 times times bin width
E5 = np.loadtxt('Data/techtue2020.csv', delimiter=',').T

arr = np.array_split(E5[1], 20)
arr[0] = np.delete(arr[0], 0)

ci = np.zeros(len(arr[0]))
for x in arr:
    ci += x

print(max(ci))
print(min(ci))
size = int(max(ci) - min(ci))
counts, bins = np.histogram(ci, bins=size)
bins = bins[0:-1]
plotHistogram(counts, bins, min(ci), max(ci), back_5hz_avg, titl
e="E5 Rebinned by 20", xlabel="Counts per 4 seconds", mean=np.me
an(ci), sigma=np.std(ci), maxcalls=2000,
             width=1)
```

```
49.0
14.0
Poisson parameters: [  31.16270082 1358.86791341]
Poisson uncertainty: 0.17674744850611665
Gaussian parameters: [  30.91728598    5.48817688 13
48.85812811]
Gaussian uncertainty: 0.17525415206117306 0.17536733
94613068
```

```
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:29: RuntimeWarning: div
ide by zero encountered in true_divide
/Users/kevinsohn/opt/anaconda3/lib/python3.7/site-pa
ckages/ipykernel_launcher.py:30: RuntimeWarning: div
ide by zero encountered in true_divide
```

Out[22]:

```
(array([  31.16270082, 1358.86791341]),
 array([  30.91728598,    5.48817688, 1348.85812811]
))
```



In [62]:

```
E1 = np.loadtxt('Data/E1.csv', delimiter=',').T # time/counts T/
```

```
R T/R

E5 = np.loadtxt('Data/techtue2020.csv', delimiter=',').T

# Load bonus data (All with 5Hz)
bonus_0 = np.loadtxt('Data/bonus_0.csv', delimiter=',').T # T/R
bonus_2_1= np.loadtxt('Data/bonus_2_1.csv', delimiter=',').T # T
/R
bonus_15 = np.loadtxt('Data/bonus_15.csv', delimiter=',').T # T/
R

bonuses = [bonus_0[1], bonus_2_1[1], E5[1], E1[1], bonus_15[1]]
print(np.mean(bonus_0[1]))
print(np.mean(E1[1]))
print(np.mean(bonus_15[1]))

y1 = np.std(bonus_0[1], ddof=1)
y2 = np.std(E1[1], ddof=1)
y3 = np.std(bonus_15[1], ddof=1)
yerr = np.array([y1,y2,y3])*2

means = np.array([17.112, 2.566289, 0.66888])
rates = means/.2
distances = np.array([0, 6.9, 15])+.1
inv_dsq = 1/distances**2

def f1(x, a, b):
    return a/(x)**2 + b

def f2(x, a, b):
    return a*x + b

params1, cov1 = opt.curve_fit(f1, distances, rates)
params2, cov2 = opt.curve_fit(f2, 1/distances**2, rates)

fig, (ax1, ax2) = plt.subplots(1,2,figsize=(15, 6))

x = np.linspace(.1, 15, 500)

ax1.plot(x, f1(x, *params), label='Inverse square fit')
ax1.errorbar(distances, rates, yerr=yerr, capsize = .5, fmt='.',
label='Data')
ax2.plot(1/distances**2, f2(1/distances**2, *params2), label="Li
near fit")
ax2.errorbar(1/distances**2, rates, yerr = yerr, fmt=".", label=
```

```
"Linearized data")

ax1.set_title("Inverse Square Law of Radiation")
ax1.set_ylabel("Count rate (Bq)")
ax1.set_xlabel(r'Distance from source (cm)')
ax2.set_title("Inverse Square Law of Radiation ")
ax2.set_ylabel("Count rate (Bq)")
ax2.set_xlabel(r'$\frac{1}{(Distance)^2}$ ($\frac{1}{cm^2}$)')

plt.grid(linestyle='dashed', color='silver')
ax1.legend()
ax2.legend()
plt.show()
```

```
17.111925383077795
2.5662891405729513
0.6688874083944037
```