EP219: Data Analysis and Interpretation

Assignment Report 4

By Team: *Sigificantly Different*

October/28/2018 - November/05/2018

Contents

Problem Statement

Consider a dark matter direct detection exper- iment that is designed to measure the recoil energy of nuclei being scattered by dark matter particles. The measured recoil energies (E R) range from 0 40 KeV and the total number of events are reported in 1 KeV bins..

- Make a clearly labelled histogram of the data.
- Assuming background only processes, calculate the mean number of events that you would expect to see in each bin. Make a histogram of this expected background.
- Assuming cross-sections of 0.01 fb, 0.1 fb, 1 fb, 10 fb, 100 fb, calculate the mean number of events that you would expect to see in each bin assuming background and signal. Make histograms for each of these cases. In which cases do you expect to tell by eye whether or not you have a dark matter signal?
- Find the log likelihood function of the cross-section log L() and plot it. Describe in detail the process used to arrive at this log likelihood function.
- Use this log likelihood function to find the maximum likelihood estimate (MLE) of the cross-section. Also report a 1- interval of cross-sections that are consistent with the data.

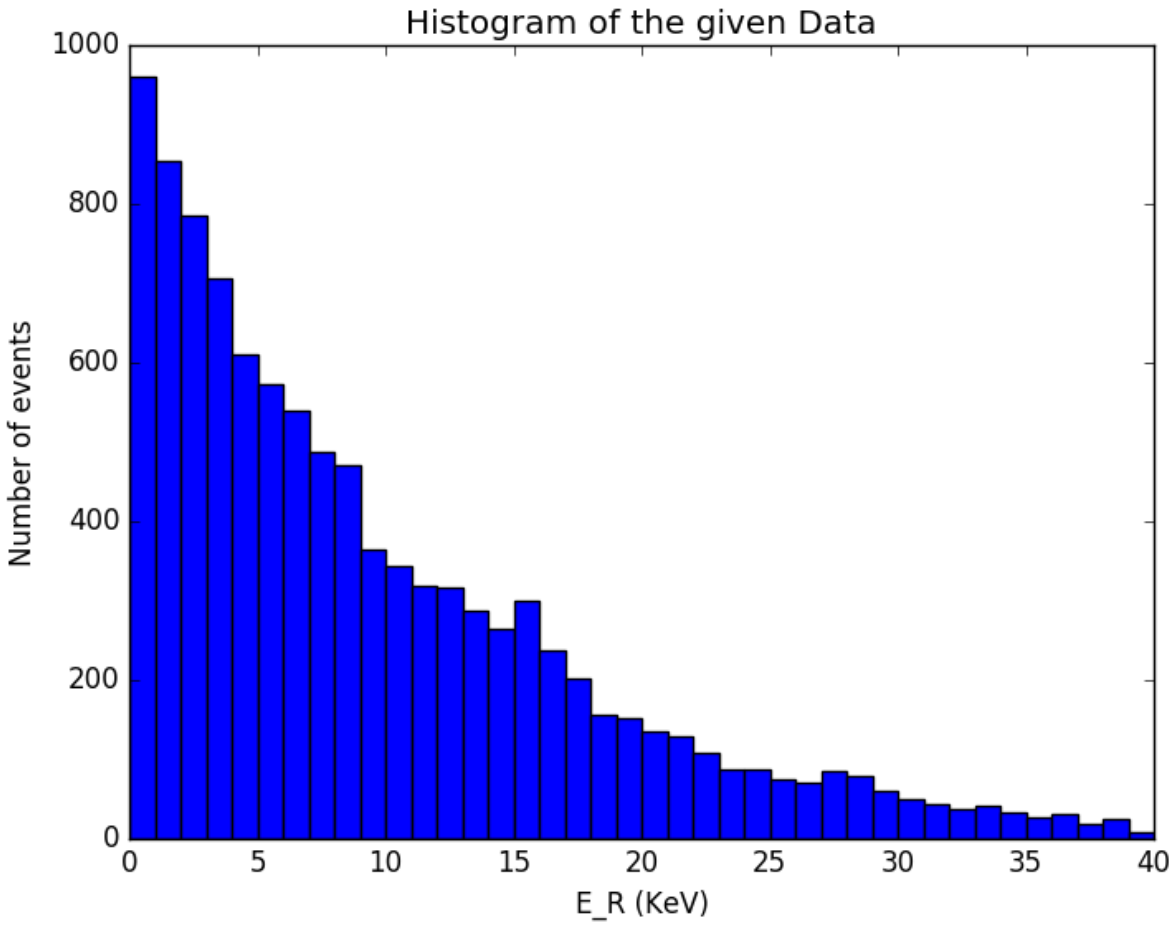
Code

```
# importing numpy library for sorting arrays and simple processes on array
import numpy as np
# importing pandas library for reading csv files
import pandas as pd
# importing pyplot from matplotlib to plot graphs
import matplotlib.pyplot as plt
# importing math library to perform math functions
import math

# defining function to find the integration (number of events) of single bin of
background histogram
def integrate_background(low_limit, up_limit):
        return (10000*(math.exp(-(low_limit/10.0)) - math.exp(-(up_limit/10.0))))

# defining function to find the integration (number of events) of single bin of
darkmatter data/histogram
def integrate_darkmatter(low_limit, up_limit, sigma):
        if (low_limit >= 5 and up_limit <= 15):
                return (sigma*20*(((up_limit**2) - (low_limit**2))/2.0 - 5.0))
        elif (low_limit >= 15 and up_limit <= 25):
                return (sigma*20*(25.0 - ((up_limit**2) - (low_limit**2))/2.0))

        else:
                return 0.0

# defining lists to store log_likelihood values and parameter values
log_likelihood = []
s_axis = []

# defining function to find log_likelihood of histograms (background + darkmatter
data)
def generate_likelihood():
        flag = 0
        i = 36.0
        while i <= 39:
                log_likelihood.append(0)
                s = 10**(i - 39)
                s_axis.append(i)
                for j in range(0, 40):
                        mean = integrate_darkmatter(j, j+1, s) +
integrate_background(j, j+1)
                        log_likelihood[flag] += (-1)*(mean) + data[' Number of
events'][j]*math.log(mean)
                i += 0.04
                flag += 1

# defining function to find the derivative of likelihood function
def derivative(array):
        array_prime = []
        for i in range(0, (len(array) - 1)):
                value = (array[i+1] - array[i])/(s_axis[i+1] - s_axis[i])
                array_prime.append(value)
        return array_prime

# reading data from csv files
data = pd.read_csv('recoilenergydata_EP219.csv')

# plotting histogram of the given data
plt.bar(data['E_R (KeV)'], data[' Number of events'], width = 1, align = 'center')
plt.title('Histogram of the given Data')
plt.xlabel('E_R (KeV)')
plt.ylabel('Number of events')
plt.show()

# defining list to store mean value for each bin of background data
background_mean = []
for i in range(0, 40):
        background_mean.append(integrate_background(i, i+1))
```

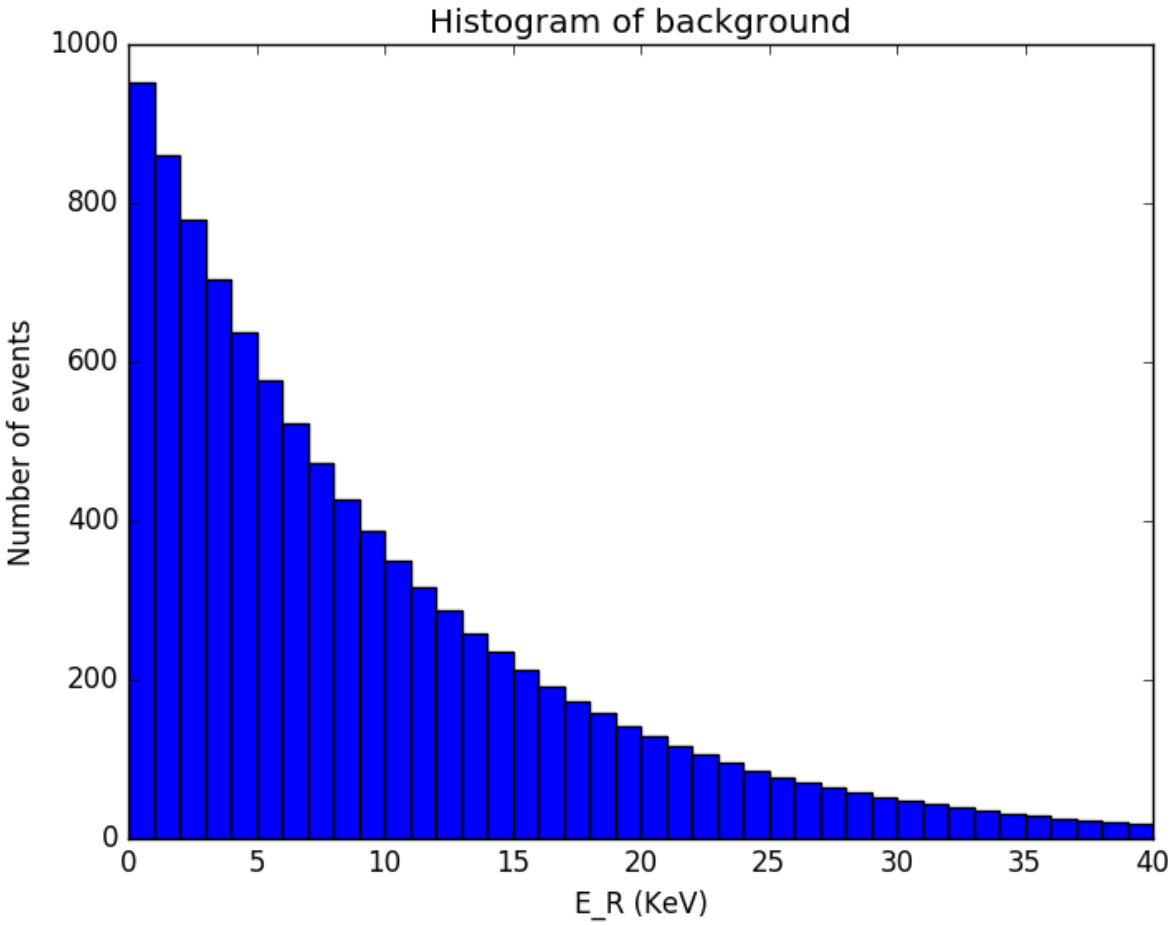
```
plt.bar(data['E_R (KeV)'], background_mean, width = 1, align = 'center')
plt.title('Histogram of background')
plt.xlabel('E_R (KeV)')
plt.ylabel('Number of events')
plt.show()

# defining list to store different values of sigma
number_of_events_mean_s = [1, 2, 3, 4, 5]

# plotting the histogram for background + signal
s = 0.01*(10**(-39))
for i in range(0, 5):
        number_of_events_mean_s[i] = []
        for j in range(0, 40):
                number_of_events_mean_s[i].append(integrate_darkmatter(j, j+1, s))

        for k in range(0, 40):
                number_of_events_mean_s[i][k] = background_mean[k] +
number_of_events_mean_s[i][k]
        plt.bar(data['E_R (KeV)'], number_of_events_mean_s[i], width = 1, align =
'center')
        plt.title('Histogram of the signal for sigma = ' + str(s))
        plt.xlabel('E_R (KeV)')
        plt.ylabel('Number of events')
        plt.show()
        s = s*10

# calling the generate_likelihood function
generate_likelihood()

# plotting log_likelihood function with respect to the parameter
sigma
plt.plot(s_axis, log_likelihood)
plt.title('plot of log likelihood')
plt.xlabel('log10(parameter), where parameter is in fb')
plt.ylabel('log likelihood')
plt.show()

# finding the first derivative of the log_likelihood function
log_like_prime = derivative(log_likelihood)

# finding the second derivative of the log_likelihood function
log_like_double_prime = derivative(log_like_prime)

# finding the error delta (spread in log_likelihood plot)
delta = -1/(log_like_double_prime[np.argmax(log_likelihood)])

# printing the value of sigma at which log_likelihood is maximum
print('value of sigma at which log likelihood is maximum is: ', 10**(s_axis
[np.argmax(log_likelihood)] - 39))

# printing the error/spread in log_likelihood plot
print('error/spread in likelihood function is:', math.sqrt(delta))
```

Histograms

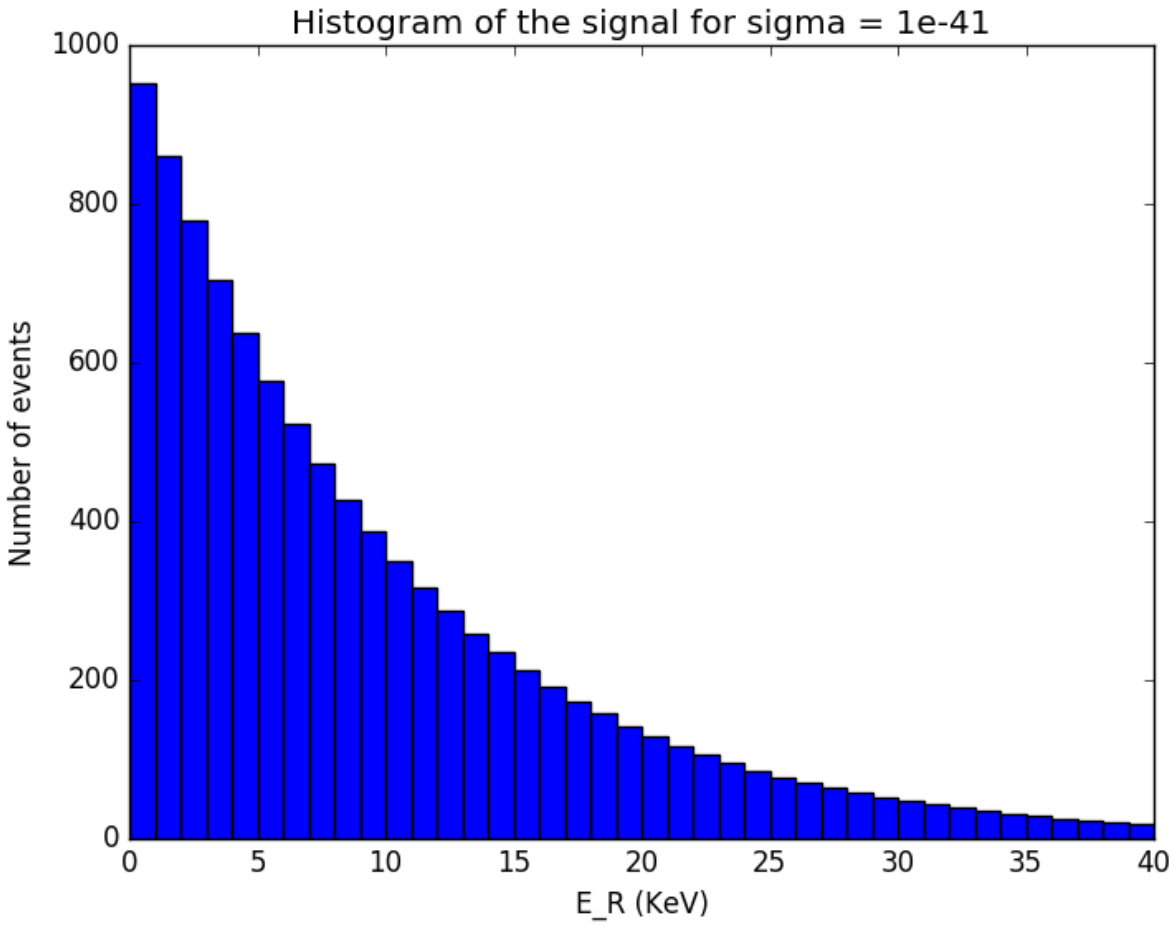
A) Histogram of the given data



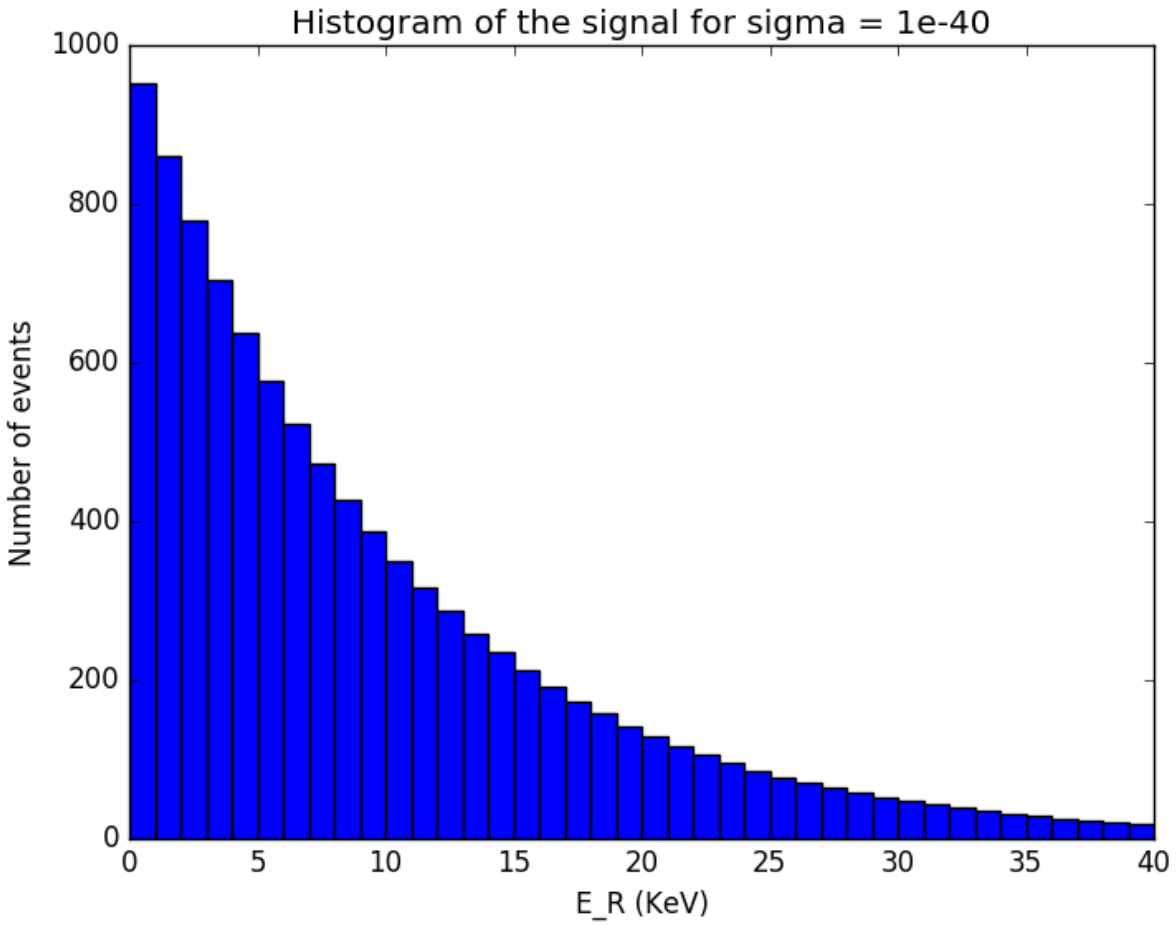
B) Histogram for the background processes



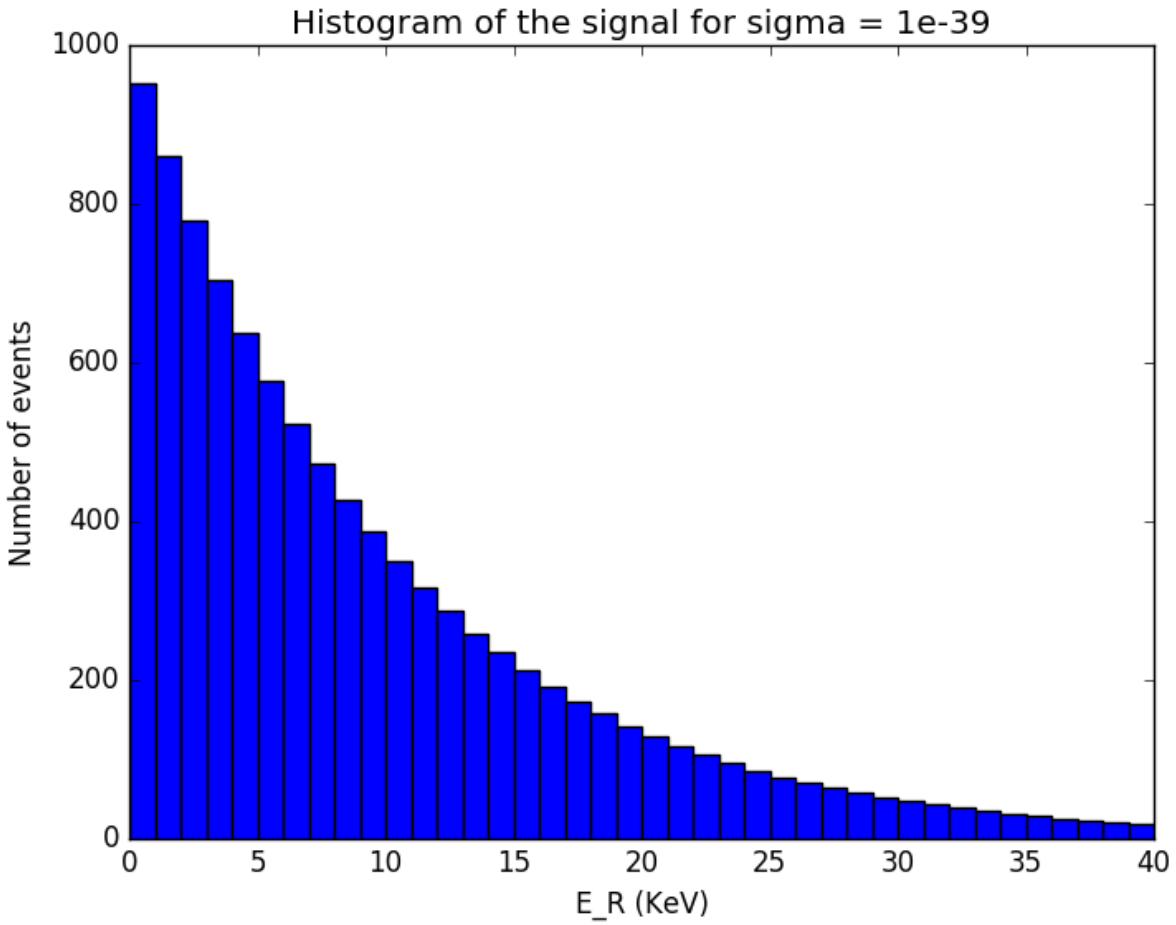
C) For sigma = 0.01 fb, calculated the mean number of events in each bin considering both background and signal and plotted the histogram



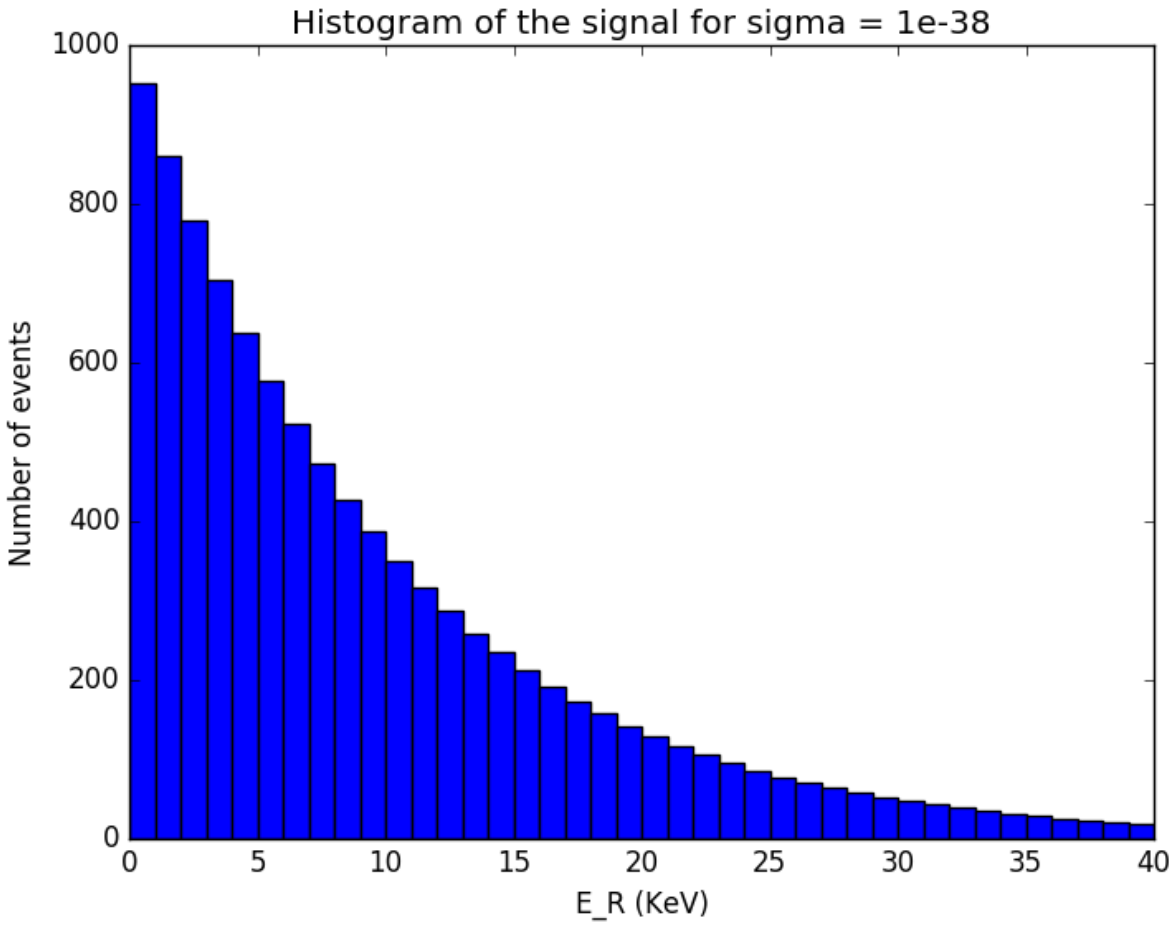
D) For sigma = 0.1 fb, calculated the mean number of events in each bin considering both background and signal and plotted the histogram



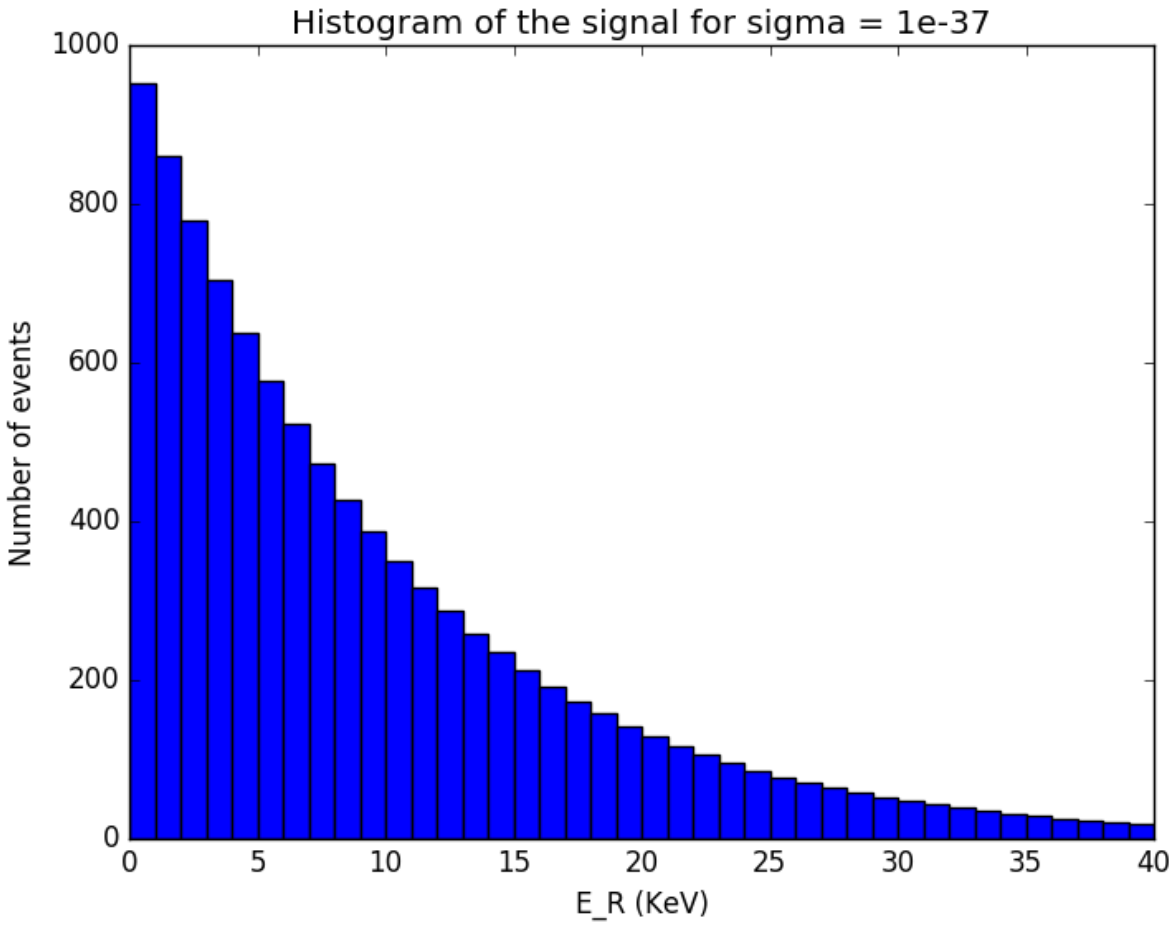
E) For sigma = 1 fb, calculated the mean number of events in each bin considering both background and signal and plotted the histogram



F) For sigma = 10 fb, calculated the mean number of events in each bin considering both background and signal and plotted the histogram

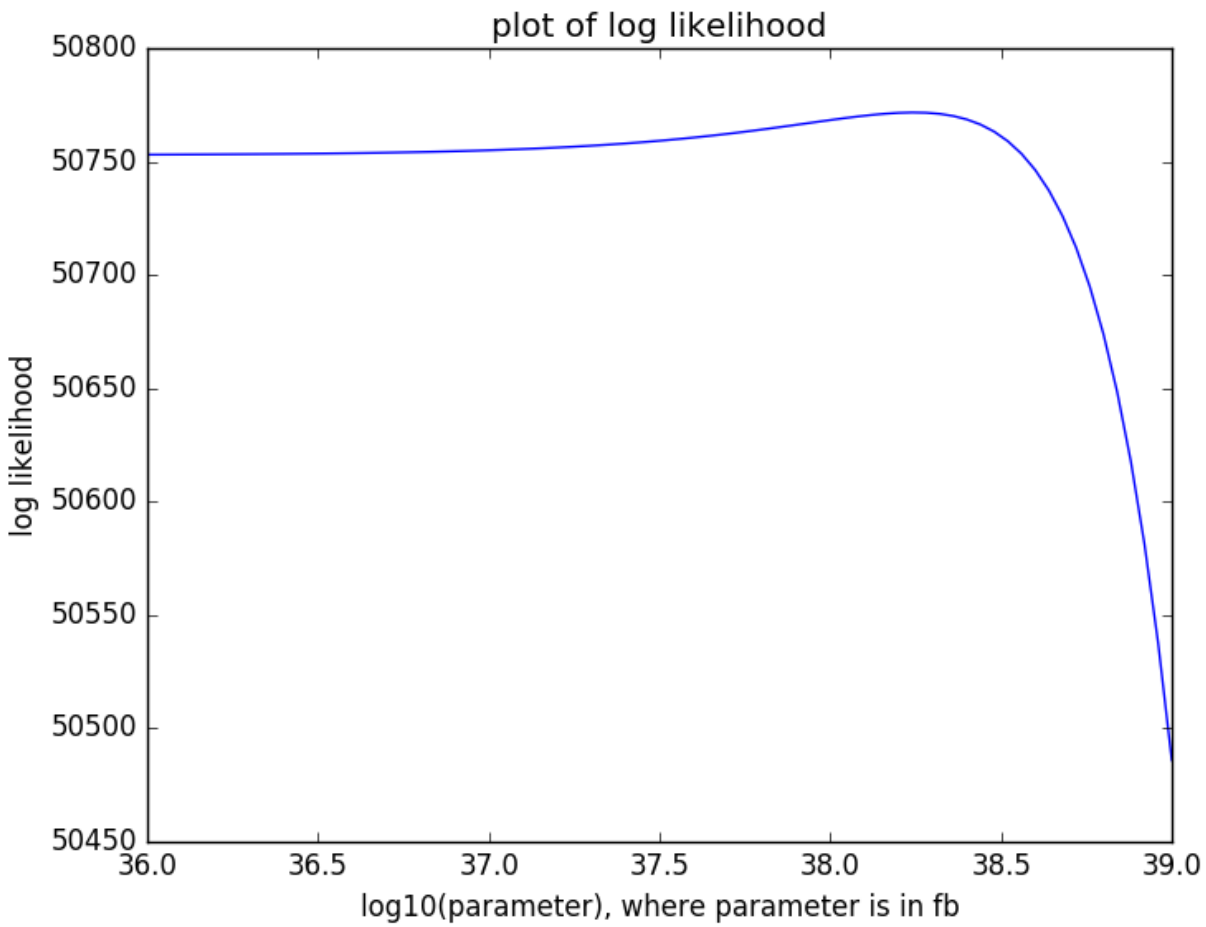


G) For sigma = 100 fb, calculated the mean number of events in each bin considering both background and signal and plotted the histogram



Log Likelihood

Plotted the log likelihood with respect to parameter sigma



Justification for the likelihood function

Since, we are given the histogram as data with large number of bins and small bin-width so we assumed the probability dirtribution to be poisson distribution. Because we have large data and probability of landing in one bin is small. Now, the expectation t_i of the number of observations in bin i is

$$t_i(\sigma) = Np_i(\sigma)$$

Where, N is total number of events, p_i is probability of landing into a bin and sigma is some parameter.

After assuming poisson distribution, Likelihood $L_i(\sigma)$ is given by

$$L_i(\sigma) = \frac{\exp(-t_i)t_i^{d_i}}{d_i!}$$
$$ln(L_i(\sigma)) = -t_i + d_i ln(t_i) - ln(d_i!)$$

Therefore,

$$ln(L(\sigma)) = \sum_{i=1}^{B}(-t_i + d_i ln(t_i))$$

we dropped $ln(d_i)$ since it does not depends on σ

But for histograms with background, we write $t_i + b_i$ instead of t_i t_i from signal and b_i from background.

Therefore,

$$ln(L) = \sum_{i=1}^{B}(-(t_i + b_i) + d_i ln(t_i + b_i))$$

Conclusion

By ploting this data we conclude that:

a) We cannot tell by eye whether or not we have found the dark matter signal

b) The value of sigma at which log likelihood function is maximum is $10^{0.76} cm^2$

c) Error or spread in likelihood function is $\delta = 0.066 cm^2$

Team Contribution

a) **Vashishtha Kochar** - Programmer 25%

b) **Nihal Barde** - Report writer 25%

c) **Adeem Jassani** - Web developer 25%

d) **Ram** - Team Leader 25%