EP219: Data Analysis and Interpretation

Assignment Report 5

By Team: *Significantly Different*

October/28/2018 - November/09/2018

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..

We have to conclude whether or not the data favors the presence of a dark matter signal. Let us consider the null Hypothesis to be that dark matter does not exist (i.e. = 0).
We have to find the p-value of the data.
Can we rule out the null hypothesis with the data that we have conclude that we have discovered dark matter at the 95% confidence-level?

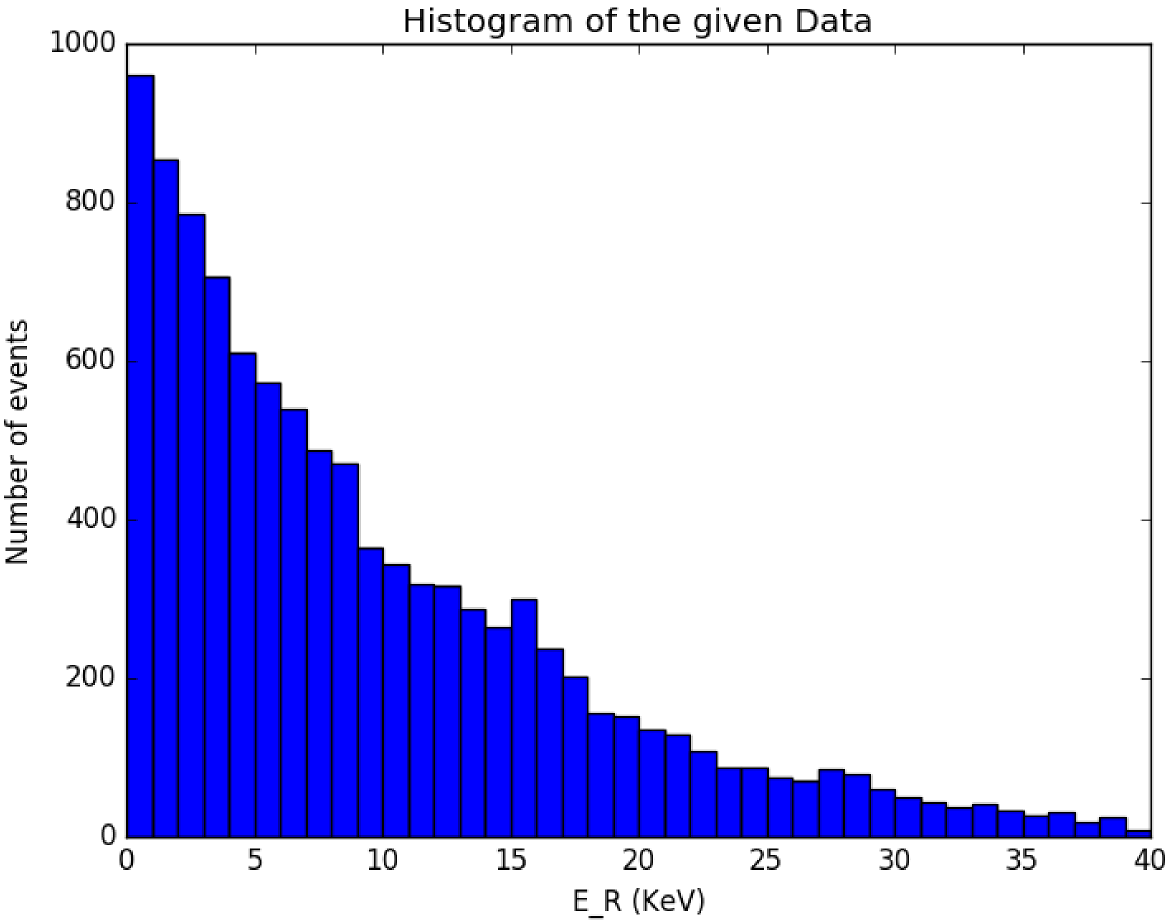
Code

```
# 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))))

# 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.savefig("original_data.png")
plt.show()

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

# plotting histogram of the background data
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.savefig("hist_background.png")
plt.show()

# generating pseudo data
test_data = [] # storing test statistic for different histograms (log likelihood)
count = 0
num_test_data = 10000
scale_factor = 100 # for scaling test data
while (count < num_test_data):
        log_likelihood = 0
        background_number_of_events = []
        for i in range(0, original_bins): # gererating poisson distributed pseudo
data from given data
                background_number_of_events.append(np.random.poisson(lam =
background_mean[i]))
        count += 1
        for j in range(0, original_bins): # finding log likelihood of pseudo data
                ti = background_mean[j]
                log_likelihood += (-1)*(ti) + background_number_of_events
[j]*math.log(ti)
        test_data.append(log_likelihood/scale_factor) # storing test statistic
after scaling it

heights, ts_values = [], []
min_stat, max_stat = min(test_data), max(test_data)
num_bins = 150
width = float(max_stat-min_stat)/num_bins
total_area = 0.0

# finding height and total area for normalising test statistic distribution
for i in range(num_bins):
        bin_min = min_stat + (i*width)
        bin_max = min_stat + ((i+1)*width)
```

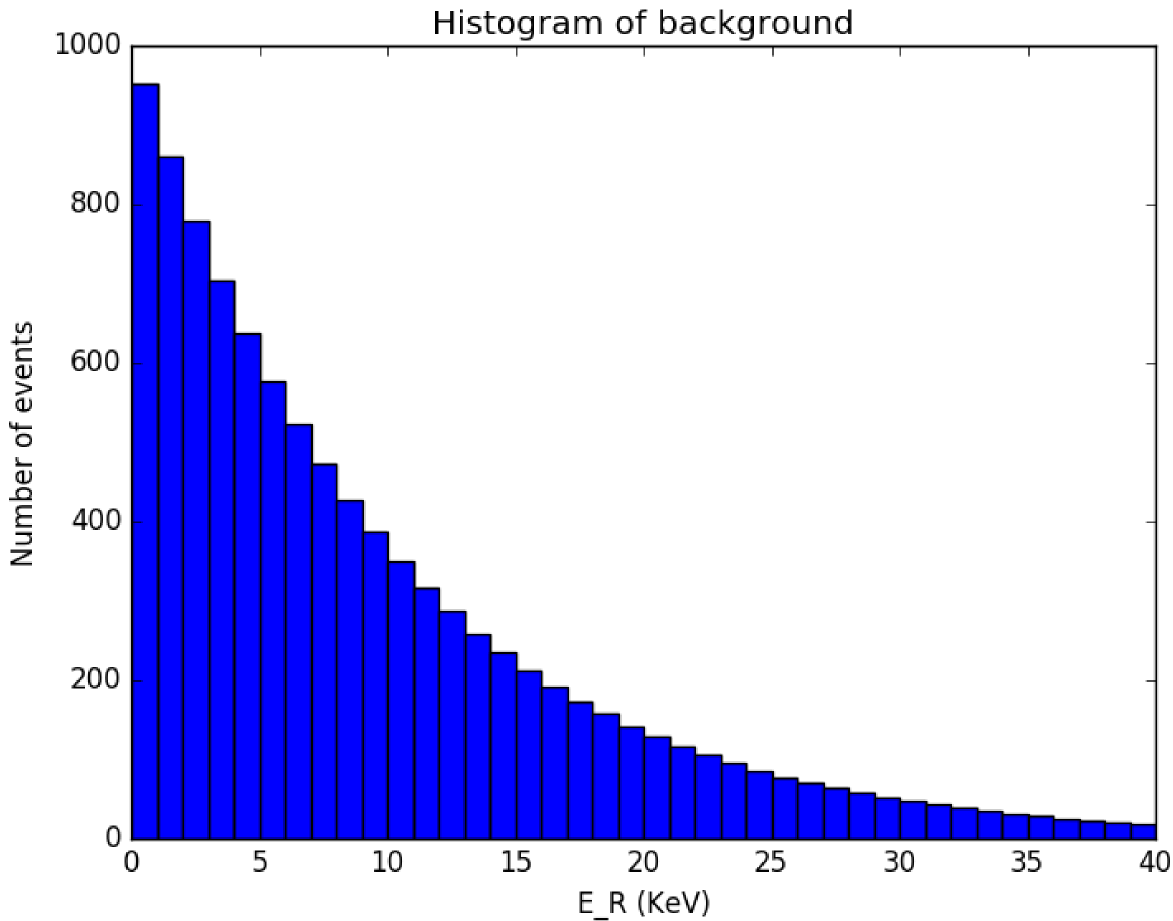
```
        num_obs = np.sum((bin_min<=np.array(test_data))*(np.array
(test_data)<bin_max))
        heights.append(num_obs) # num of observations in given interval
        total_area += heights[-1]*width # total area of histogram data
        ts_values.append((bin_min+bin_max)/2.0) # center of bin

# normalising test statistic data
bin_heights = np.array(heights)*width/total_area

# finding critical test statistic value
found_area = 0.0
now_bin = num_bins
while(found_area<0.05): # running loop while till we get area greater than 0.05
        now_bin -= 1
        found_area += bin_heights[now_bin]
crit_ts_value = now_bin
print(ts_values[crit_ts_value]) # Printing the critical value of test statistic

# plotting the normalized test statistic distribution
bins = np.linspace(min_stat, max_stat, num_bins+1)
plt.bar(ts_values, bin_heights, width = width, align = 'center')
plt.title('test statistic distribution')
plt.xlabel('test statistic data')
plt.ylabel('f(Ts|Ho)')
plt.savefig("prob_density.png")
plt.show()

# calculating the test statistic of the observed data
test_data_obs = 0
for i in range(0, original_bins):
        ti =  background_mean[i]
        test_data_obs += (-1)*(ti) + data[' Number of events'][i]*math.log(ti)

print(test_data_obs/scale_factor)
```

Histograms

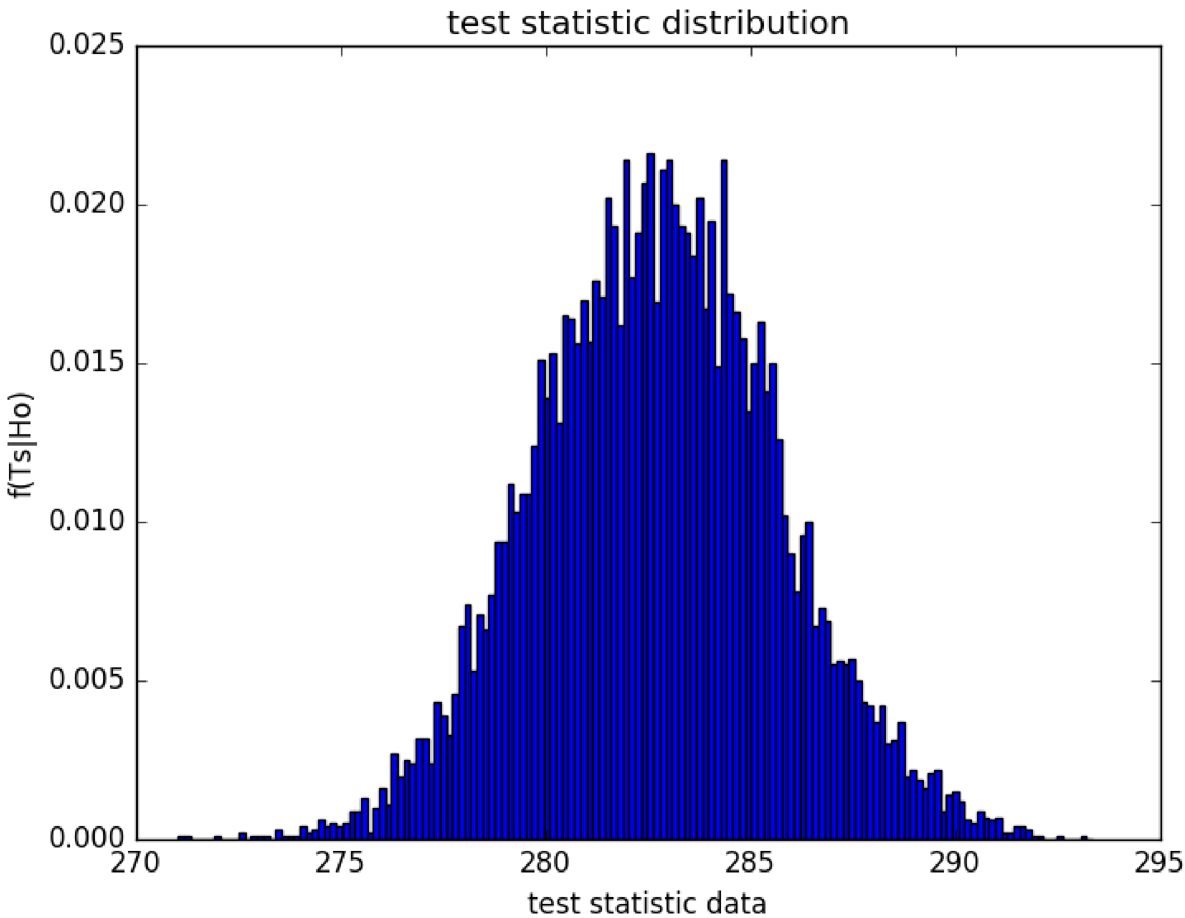
A) Histogram of the given data



B) Histogram for the background processes



C) Probability distribution of the test statistics (Test statistic is log(Likelihood))



Conclusion

By ploting this data we conclude that:

a) p-value of the given data is 0.05 (p-value is basically the area under the probability density function in the critical region)

b) Critical value of test statistic is 287.615.

c) Test statistic for given data is 507.528.

d) Test statistic of the given data is far beyond the critical value of test statistic (i.e it is in tne critical region) So, we can say that null hypothesis (Dark matter does not exist) is ruled out by 95% confidence level.

Team Contribution

a) **Vashishtha Kochar** - Team Leader 25%

b) **Nihal Barde** - Programmer 25%

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

d) **Ram** - Report writer 25%