Homework 3
Krish Patel
CS C121

Question 1:

a) Likelihood of gene:

For control: $X_{ij} \sim \text{Poisson}(S_i B_j)$

For experiment $X_{ij} \sim \text{Poisson}(S_i B_j \delta)$

Likelihood for control: $\prod_{i=1}^{3} \left(\frac{e^{-S_i B_j} (S_i B_j)^{X_{ij}}}{X_{ij}!} \right)$

Likelihood for experiment $= \prod_{i=4}^{6} \left(\frac{e^{-S_i B_j \delta} (S_i B_j \delta)^{X_{ij}}}{X_{ij}!} \right)$

Total likelihood $= \prod_{i=1}^{3} \left(\frac{e^{-S_i B_j} (S_i B_j)^{X_{ij}}}{X_{ij}!} \right) \prod_{i=4}^{6} \left(\frac{e^{-S_i B_j \delta} (S_i B_j \delta)^{X_{ij}}}{X_{ij}!} \right)$

b) Maximum log likelihood.

$$\log(\mathcal{L}_n) = \log(\mathcal{L}_n(\delta, S, B))$$

$$= \sum_{i=1}^{3} -S_i B_j + X_{ij} \log(S_i B_j) - \log(X_{ij}!)$$

$$+ \sum_{i=4}^{6} -S_i B_j \delta + X_{ij} \log(S_i B_j \delta) - \log(X_{ij}!)$$

$$\frac{\partial(\log \mathcal{L}_n)}{\partial \delta} = \sum_{i=4}^{6} -S_i B_j + \frac{X_{ij}}{\delta}$$

double derivative $= \sum_{i=4}^{6} -\frac{X_{ij}}{\delta^2}$ which is negative, thus local maxima.

Equating first derivative to zero.

$$\sum_{i=4}^{6} -S_i B_j = \frac{1}{\delta} \sum_{i=4}^{6} -X_{ij} \qquad \delta = \frac{\sum_{i=4}^{6} X_{ij}}{\sum_{i=4}^{6} S_i B_j}$$

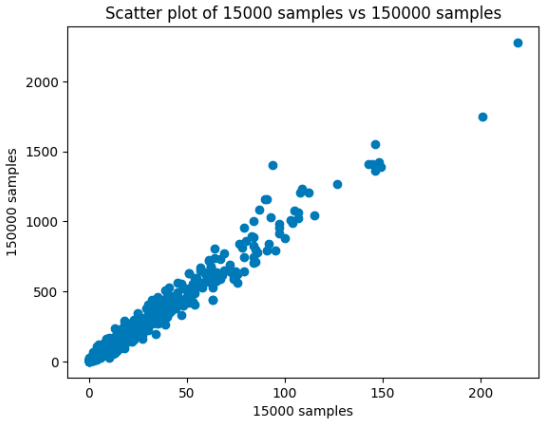
c The independence of distributions implies that the variables xij are sampled without being affected by δ. From this, it ensures that estimating δ using specific datasets (e.g., where i=4,5,6) remains unbiased by the distribution of xij. In the estimation process, each xij's contribution to δ is adjusted relative to βj, which also ensures appropriate weighting for accurate estimation. This method aligns δ with observed data following a Poisson distribution, which proves the reliability of estimates by aligning with the theoretical derivations of the MLE from above.

Question 2:

a) Code:

```
with open('p.tsv') as f:
    data_p = 
                (function) array: Any

data_p = np.array([list(map(float, line.split('\t'))) for line in data_p])
print(data_p.shape)
print(data_p[1])



datap = np.squeeze(data_p)
dataq = np.squeeze(data_q)
#creating three different samples using random sampling and replacement
sample1 = np.random.multinomial(15000, datap)
sample2 = np.random.multinomial(30000, datap)
sample3 = np.random.multinomial(150000, datap)

merged_samples = np.vstack((sample1, sample2, sample3))
# print(merged_samples.shape)

#plotting a scatter plot of 15000 samples vs 150000 samples
plt.scatter(sample1, sample3)
plt.xlabel('15000 samples')
plt.ylabel('150000 samples')
plt.title('Scatter plot of 15000 samples vs 150000 samples')
plt.show()
```

Scatterplot: According to this, the approximate line for this graph if we run a linear regression is approximately y = 10x, which makes sense as sample1 has 10x less samples than sample3



b) Code:

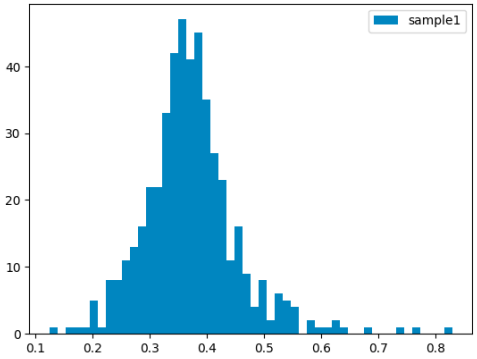
```
def DESeq_normalization(samples):
    num_samples, num_genes = samples.shape
    Sjg_hat = np.zeros((num_samples, num_genes))
    denominator = np.ones(num_genes)
    Sj_hat = np.ones(num_genes)
    for i in range(num_genes):
        for j in range(num_samples):
            denominator[i] *= samples[j][i]

    for i in range(num_genes):
        for j in range(num_samples):
            if(denominator[i] == 0):
                Sjg_hat[j][i] = 0
            else:
                Sjg_hat[j][i] = samples[j][i] / denominator[i]**(1/num_samples)
    Sj_hat = np.median(Sjg_hat, axis=1)
    return Sjg_hat, Sj_hat
```

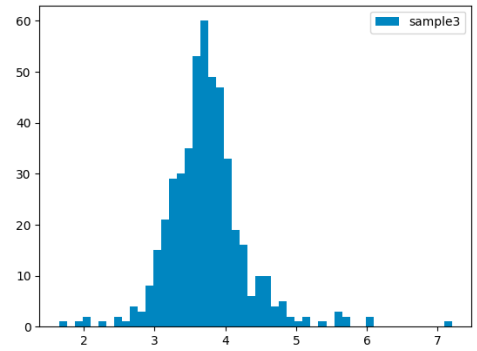
c) Histograms

Sj_hat= [0.3671611 0.73318019 3.69156583]

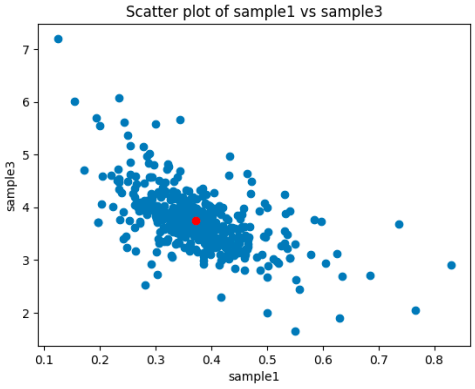
These values of sj_hat reflect the size of the samples. The ratio is approximately: 1:2:10

Sj_hat = median(sjg_hat) over an entire sample. Approximately, this median value depends on the sample size. By dividing sjg_hat by the respective sj_hat, we normalize it so that it centers around approximately 1, and we can make better comparison

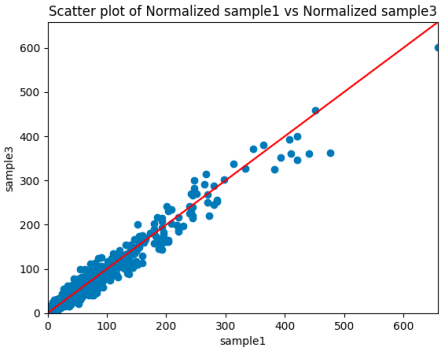




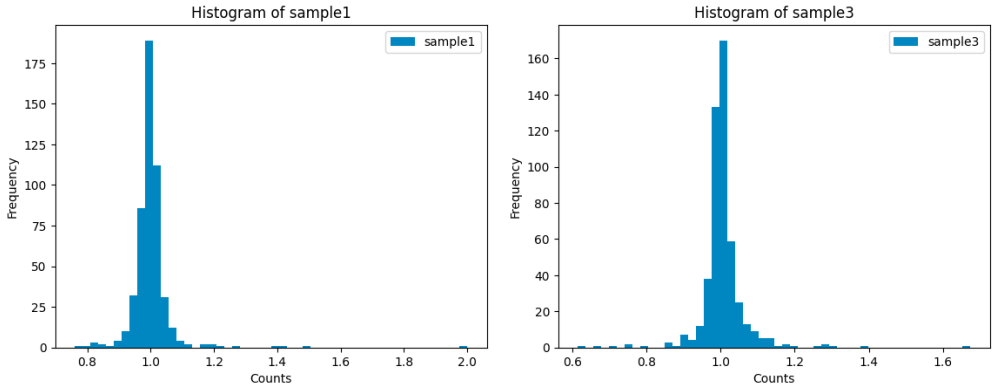

```
Coordinates of mean of the scatter plot are 0.37265874486537753
3.7416676801023065
According to the graph below, the center of sample1 and sample3 is the
coordinates above(derived using the mean of the values). According to this,
there is no describable relationship between sample1 and sample3, except for
their mean being 10 times the other. As they are different samples, we also
expect them to have different shapes.
```



d) Scatterplot: The line here is y = x
Post normalization, the relationship becomes much more clear, and the data centers around y = x. Thus it shows that accounts for the different in sample size.

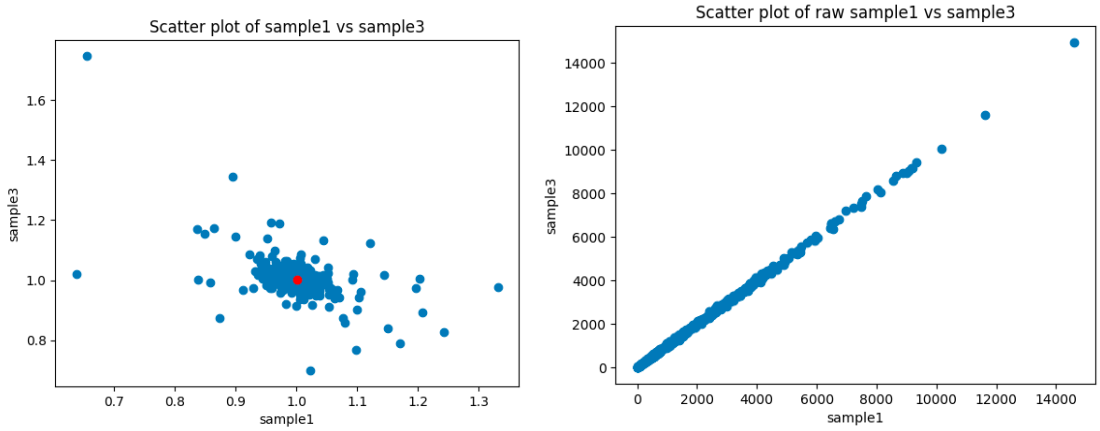


e) Now, we can see from the histograms below that the histograms are centered around 1.



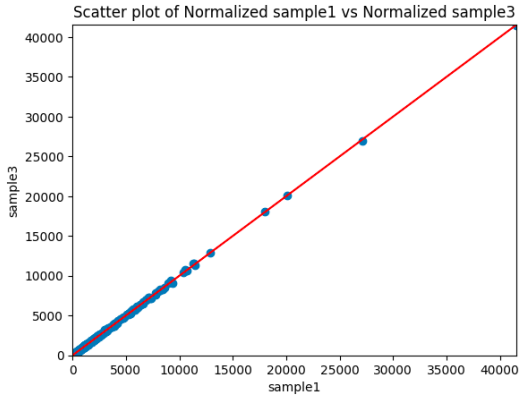
`Center coordinates are:  1.0013250427001033 1.0022076629294296`

Compared to the scatterplot in c), we can see that the data is now centered around 1, thus making it seem that the sample size is the same. Along with that, there is less variation with this data as there are more samples in consideration. This can mathematically be proven as the probabilities are more centered to the actual value



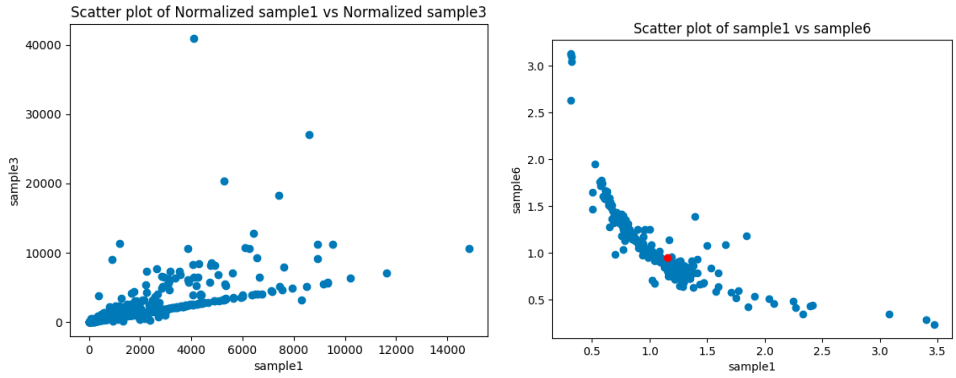
Considering the scatter plot of sample1 and sample3, which are for the raw counts, it is evident that the best fit of the line is y = x, which is plotted as well. The samples are the same size, thus we don't need to normalize to get the y=x approximation above.

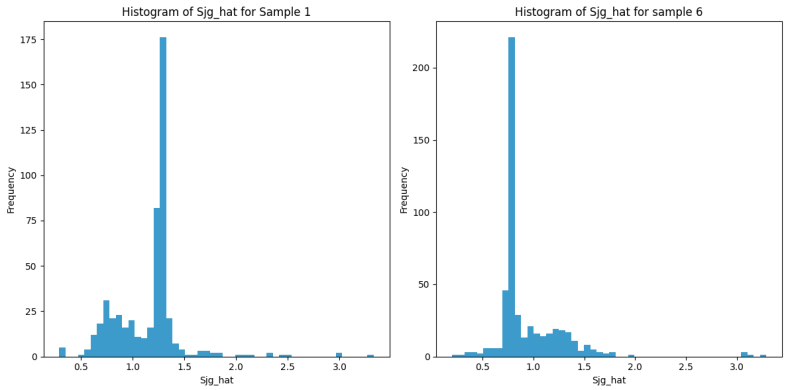
f)



According to the figure above, the difference from the plots in c) and e) is that the data is much more centered around y=x. Along with that, there are outliers in the data, which are not visible in the previous plots.

g) Plots:





From the plots above

When comparing the sample1 and sample6 sj_hat values, a hyperbola type curve can be observed with the mean values around 0.8 for the sample 6 and 1.25 for sample y. the distributions are also away from the mean/median , this is mostly because p and q represent different distributions. Also looking at the normalized plots, we can see that there is a large deviation from the y=x line, which is due to sample3(which is actually sample 6) deviating from the sample1 , which is the control group.

Code:

```
import numpy as np
import matplotlib.pyplot as plt


with open('q.tsv') as f:
    data_q = f.read().splitlines()
data_q = np.array([list(map(float, line.split('\t'))) for line in data_q])
print(data_q.shape)
print(data_q[1])

with open('p.tsv') as f:
    data_p = f.read().splitlines()
```

```
data_p = np.array([list(map(float, line.split('\t'))) for line in data_p])
print(data_p.shape)
print(data_p[1])



datap = np.squeeze(data_p)
dataq = np.squeeze(data_q)
#creating three different samples using random sampling and replacement
sample1 = np.random.multinomial(15000, datap)
sample2 = np.random.multinomial(30000, datap)
sample3 = np.random.multinomial(150000, datap)

merged_samples = np.vstack((sample1, sample2, sample3))
# print(merged_samples.shape)

#plotting a scatter plot of 15000 samples vs 150000 samples
plt.scatter(sample1, sample3)
plt.xlabel('15000 samples')
plt.ylabel('150000 samples')
plt.title('Scatter plot of 15000 samples vs 150000 samples')


plt.show()

#Defining the DESeq normalization function
def DESeq_normalization(samples):
    num_samples, num_genes = samples.shape
    Sjg_hat = np.zeros((num_samples, num_genes))
    denominator = np.ones(num_genes)
    Sj_hat = np.ones(num_genes)
    for i in range(num_genes):
        for j in range(num_samples):
            denominator[i] *= samples[j][i]

    for i in range(num_genes):
        for j in range(num_samples):
            if(denominator[i] == 0):
                Sjg_hat[j][i] = 0
            else:
                Sjg_hat[j][i] = samples[j][i] / denominator[i]**(1/num_samples)
    Sj_hat = np.median(Sjg_hat, axis=1)
    return Sjg_hat, Sj_hat
```

```
Sjg_hat, Sj_hat= DESeq_normalization(merged_samples)
#removing all zero values
Sjg_hat = Sjg_hat[:, np.all(Sjg_hat != 0, axis=0)]



#histogram for sample 1 and sample 3
plt.figure(1)
plt.hist(Sjg_hat[0], bins=50, alpha=0.9, label='sample1')
plt.legend(loc='upper right')
plt.show()


plt.figure(2)
plt.hist(Sjg_hat[2], bins=50, alpha=0.9, label='sample3')
plt.legend(loc='upper right')
plt.show()


#scatterplot for sample 1 and sample 3
# Calculate the center coordinates
center_x = np.mean(Sjg_hat[0])
center_y = np.mean(Sjg_hat[2])

plt.figure(3)
plt.scatter(Sjg_hat[0], Sjg_hat[2])
plt.scatter(center_x, center_y, color='red')  # Plot the center as a red dot
plt.xlabel('sample1')
plt.ylabel('sample3')
plt.title('Scatter plot of sample1 vs sample3')

# Show the plot
plt.show()
print("Coordinates of the scatter plot are", center_x, center_y)

#Normalization
Normalized1 = Sjg_hat[0]/Sj_hat[0]
Normalized2 = Sjg_hat[1]/Sj_hat[1]
Normalized3 = Sjg_hat[2]/Sj_hat[2]
Normalized = np.vstack((Normalized1, Normalized2, Normalized3))
print(Sj_hat)

#scatterplot for sample 1 and sample 3 with red y = x line


plt.figure(4)
plt.scatter(sample1/Sj_hat[0], sample3/Sj_hat[2])
plt.xlabel('sample1')
```

```
plt.ylabel('sample3')
plt.title('Scatter plot of Normalized sample1 vs Normalized sample3')
max_val = max(max(sample1/Sj_hat[0]), max(sample3/Sj_hat[2]))
plt.xlim(0, max_val)
plt.ylim(0, max_val)
plt.plot([0, max_val], [0, max_val], color='red')

plt.show()
# Generate samples
#setting seed
np.random.seed(0)
sample1 = np.random.multinomial(1000000, datap)
sample2 = np.random.multinomial(1000000, datap)
sample3 = np.random.multinomial(1000000, datap)

# Merge samples
merged_control = np.vstack((sample1, sample2, sample3))

# Perform DESeq normalization
Sjg_hat, Sj_hat = DESeq_normalization(merged_control)

#plotting Sjhats for sample 1
plt.figure(1)
plt.hist(Sjg_hat[0], bins=50, alpha=0.9, label='sample1')
plt.legend(loc='upper right')
plt.xlabel('Counts')
plt.ylabel('Frequency')
plt.title('Histogram of sample1')
plt.show()


# Remove NaN and inf values
Sjg_hat = np.nan_to_num(Sjg_hat)

# Histogram for sample 1 and sample 3
plt.figure(1)
plt.hist(Sjg_hat[0], bins=50, alpha=0.9, label='sample1')
plt.legend(loc='upper right')
plt.xlabel('Counts')
plt.ylabel('Frequency')
plt.title('Histogram of sample1')
plt.show()

plt.figure(2)
plt.hist(Sjg_hat[2], bins=50, alpha=0.9, label='sample3')
plt.legend(loc='upper right')
plt.xlabel('Counts')
plt.ylabel('Frequency')
```

```
plt.title('Histogram of sample3')
plt.show()

# Scatterplot for sample 1 and sample 3
plt.figure(3)
Sjg_hat_nonzero = Sjg_hat[:, (Sjg_hat[0] != 0) & (Sjg_hat[2] != 0)]
Sjg_hat = Sjg_hat[:, (Sjg_hat[0] != 0) & (Sjg_hat[2] != 0)]

# Calculate the center coordinates
center_x = np.mean(Sjg_hat_nonzero[0])
center_y = np.mean(Sjg_hat_nonzero[2])

plt.scatter(Sjg_hat_nonzero[0], Sjg_hat_nonzero[2])
plt.scatter(center_x, center_y, color='red')  # Plot the center as a red dot
plt.xlabel('sample1')
plt.ylabel('sample3')
plt.title('Scatter plot of sample1 vs sample3')
plt.show()

print("Center coordinates are: ", center_x, center_y)

# Normalization
Normalized1 = Sjg_hat[0] / Sj_hat[0]
Normalized3 = Sjg_hat[2] / Sj_hat[2]

# Scatterplot for normalized sample 1 and sample 3
plt.figure(4)
plt.scatter(Normalized1, Normalized3)
plt.xlabel('Normalized sample1')
plt.ylabel('Normalized sample3')
plt.title('Scatter plot of Normalized sample1 vs Normalized sample3')
max_val = max(max(Normalized1), max(Normalized3))
plt.xlim(0, max_val)
plt.ylim(0, max_val)
plt.plot([0, max_val], [0, max_val], color='red')
plt.show()

plt.figure(5)
plt.scatter(sample1, sample3)
plt.xlabel('sample1')
plt.ylabel('sample3')
plt.title('Scatter plot of sample1 vs sample3')
#line of best fit plotted red
m, b = np.polyfit(sample1, sample3, 1)
plt.plot(sample1, m*sample1 + b, color='red')


plt.show()
```

```
sample4 = np.random.multinomial(1000000, datap)
sample5 = np.random.multinomial(1000000, datap)
sample6= np.random.multinomial(1000000, datap)


merged_experiment = np.vstack((sample4, sample5, sample6))

def DESeq_normalization(samples):
    num_samples, num_genes = samples.shape
    Sjg_hat = np.zeros((num_samples, num_genes))
    denominator = np.ones(num_genes)
    Sj_hat = np.ones(num_genes)
    for i in range(num_genes):
        for j in range(num_samples):
            denominator[i] *= samples[j][i]

    for i in range(num_genes):
        for j in range(num_samples):
            if(denominator[i] == 0):
                Sjg_hat[j][i] = 0
            else:
                Sjg_hat[j][i] = samples[j][i] / denominator[i]**(1/num_samples)
                if(j == 2):
                    Sj_hat = np.median(Sjg_hat, axis=1)
    return Sjg_hat, Sj_hat




Sjg_hat2, Sj_hat2= DESeq_normalization(merged_experiment)
#removing NaN and inf values


#histogram for sample 1 and sample 3
plt.figure(1)
plt.title("Scatter plot for sample 1 Sjg_hat for p.tsv")
plt.hist(Sjg_hat2[0], bins=50, alpha=0.9, label='sample1')
plt.legend(loc='upper right')
plt.show()


plt.figure(2)
plt.title("Scatter plot for sample 2 Sjg_hat for p.tsv")
plt.hist(Sjg_hat2[2], bins=50, alpha=0.9, label='sample3')
plt.legend(loc='upper right')
```

```
plt.show()


#scatterplot for sample 1 and sample 3
center_x = np.mean(Sjg_hat[0])
center_y = np.mean(Sjg_hat[2])
plt.figure(3)
plt.scatter(Sjg_hat2[0], Sjg_hat2[2])
plt.xlabel('sample1')
plt.ylabel('sample3')
plt.title('Scatter plot of sample1 vs sample3')
plt.show()

#Normalization
Normalized1_2 = Sjg_hat2[0]/Sj_hat2[0]
Normalized2_2 = Sjg_hat2[1]/Sj_hat2[1]
Normalized3_2 = Sjg_hat2[2]/Sj_hat2[2]
Normalized_2 = np.vstack((Normalized1_2, Normalized2_2, Normalized3_2))
print(Sj_hat2)

#scatterplot for sample 1 and sample 3

plt.figure(4)
plt.scatter(  sample1/Sj_hat[0], sample6/Sj_hat2[2])
plt.xlabel('sample1')
plt.ylabel('sample6')
plt.title('Scatter plot of Normalized sample1 vs Normalized sample3')
plt.show()

#printing the histograms for the samples above
plt.figure(5)
plt.hist(Sjg_hat2[0], bins=50, alpha=0.9, label='sample1')
plt.title("Histogram for sample 1 Sjg_hat for p.tsv")
plt.legend(loc='upper right')
plt.show()

plt.figure(6)
plt.hist(Sjg_hat2[2], bins=50, alpha=0.9, label='sample3')
plt.title("Histogram for sample 3 Sjg_hat for p.tsv")
plt.legend(loc='upper right')
plt.show()




# all 6 samples
sample1 = np.random.multinomial(1000000, datap)
sample2 = np.random.multinomial(1000000, datap)
```

```
sample3 = np.random.multinomial(1000000, datap)
sample4 = np.random.multinomial(1000000, dataq)
sample5 = np.random.multinomial(1000000, dataq)
sample6 = np.random.multinomial(1000000, dataq)
merged_samples = np.vstack((sample1, sample2, sample3))
merged_samples2 = np.vstack((sample4, sample5, sample6))

merged_all = np.vstack((merged_samples, merged_samples2))
Sjg_hat_all, Sj_hat_all = DESeq_normalization(merged_all)

# Remove NaN and inf values
Sjg_hat_all = np.nan_to_num(Sjg_hat_all)

# Histogram for sample 1 and sample 6
plt.figure(1)
plt.hist(Sjg_hat_all[0], bins=50, alpha=0.9, label='sample1')
plt.legend(loc='upper right')
plt.xlabel('Counts')
plt.ylabel('Frequency')

plt.title('Histogram of sample1')
plt.show()

plt.figure(2)
plt.hist(Sjg_hat_all[5], bins=50, alpha=0.9, label='sample6')
plt.legend(loc='upper right')

plt.xlabel('Counts')
plt.ylabel('Frequency')
plt.title('Histogram of sample6')
plt.show()


# Scatterplot for sample 1 and sample 6
plt.figure(3)
Sjg_hat_nonzero = Sjg_hat_all[:, (Sjg_hat_all[0] != 0) & (Sjg_hat_all[5] != 0)]
Sjg_hat_all = Sjg_hat_all[:, (Sjg_hat_all[0] != 0) & (Sjg_hat_all[5] != 0)]

# Calculate the center coordinates
center_x = np.mean(Sjg_hat_nonzero[0])
center_y = np.mean(Sjg_hat_nonzero[5])

plt.scatter(Sjg_hat_nonzero[0], Sjg_hat_nonzero[5])
plt.scatter(center_x, center_y, color='red')  # Plot the center as a red dot

plt.xlabel('sample1')
plt.ylabel('sample6')
plt.title('Scatter plot of sample1 vs sample6')
```

```
plt.show()
```

The above code doesn’t represent all the graphs, as modifications were made. However, most of the data is represented in the graphs.