Homework 3

Krish Patel

CS C121

Question 1:

A close-up of a paper

Description automatically generated

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.

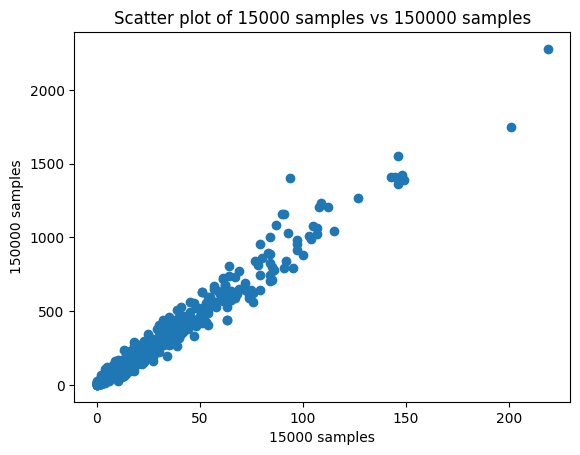
Question 2:

1. Code:

A screen shot of a computer program

Description automatically generated

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



1. Code:

A computer screen shot of a computer program

Description automatically generated

1. 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

A graph of a tower

Description automatically generated with medium confidence A graph of a tower

Description automatically generated

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.

A diagram of a scatter plot

Description automatically generated

1. 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.

A diagram of a scatter plot

Description automatically generated

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

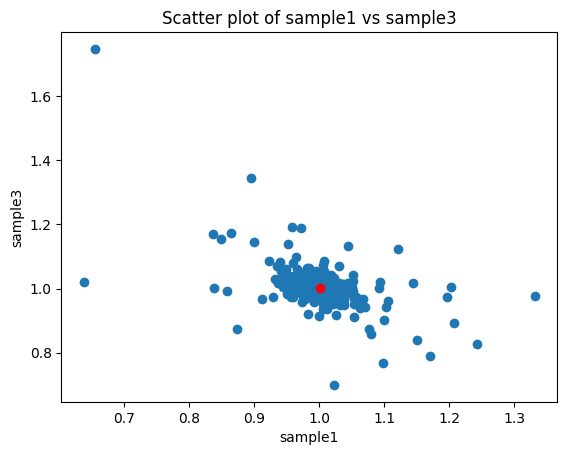
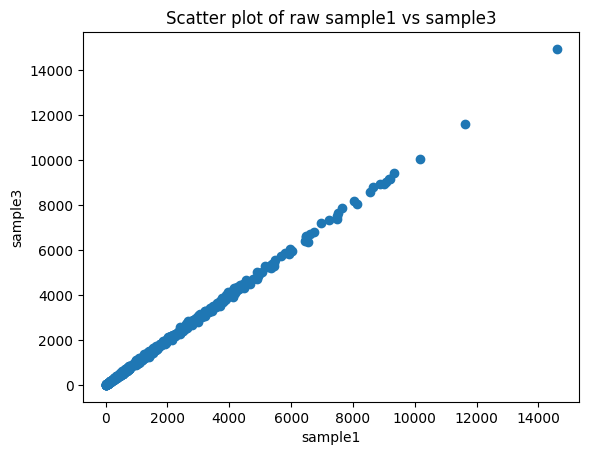
A graph of a graph

Description automatically generated A graph of a number of blue squares

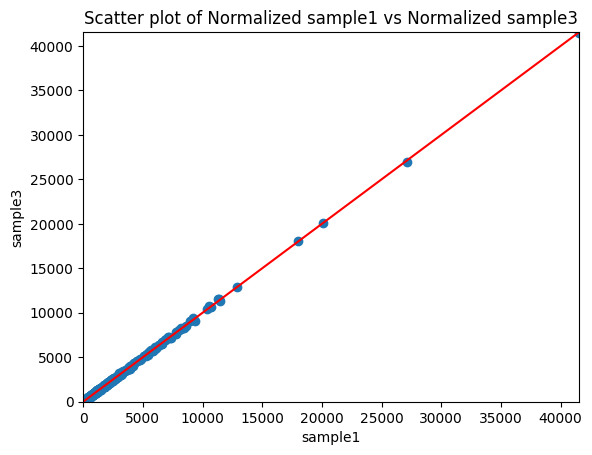
Description automatically generated with medium confidence

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.



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.

1. Plots:

A graph of blue dots

Description automatically generated A diagram of a scatter plot

Description automatically generated

A comparison of a graph

Description automatically generated

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.