Data Analysis and Visualization Lab 4

Vaibhav Amit Patel (201401222)

Tanmay Patel (201401409)

Group-23

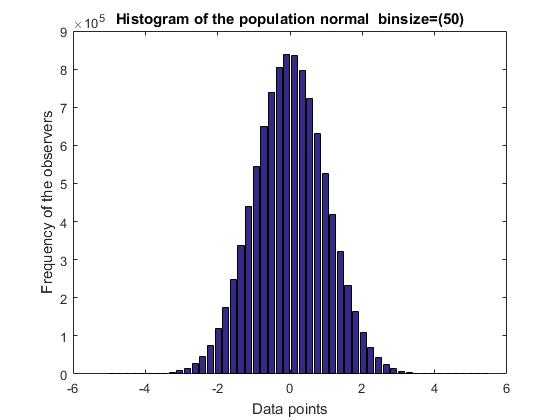
# Experiment 1:

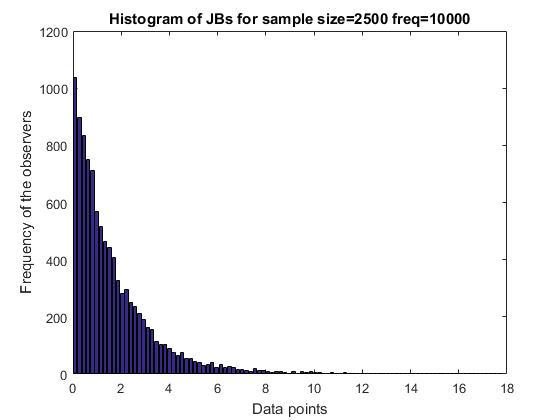
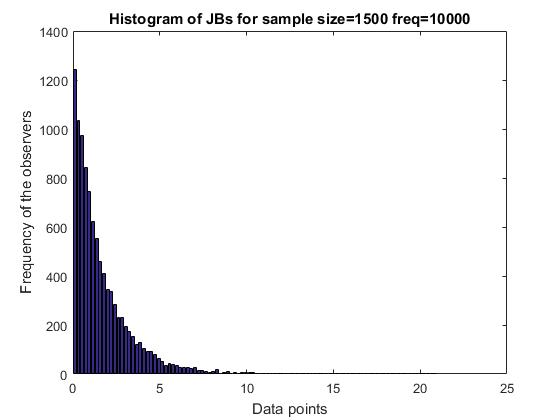
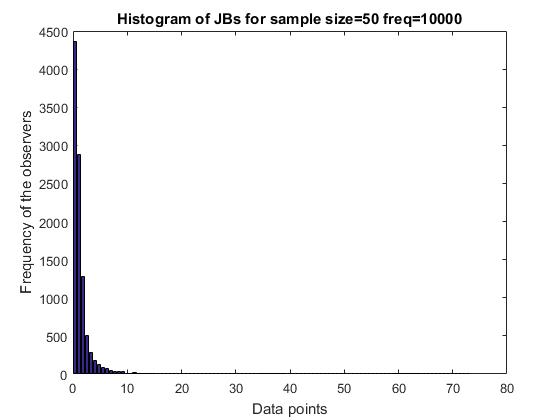
Load data\_lab4.mat. ‘population\_normal’ is a collection of 10 million observations drawn from a standard normal distribution. Assume this to be the population of interest. The Jarque-Bera (JB) test is a type of hypothesis testing which assumes that the JB test statistic follows a chi-squared distribution, if samples are drawn from a normal population. The aim of this experiment is to verify this assumption. Your solution should consider 3 different sample sizes: 50, 1500, and 2500. In which case is the assumption of chi-squared distribution more accurate? Based on the answer, use the corresponding sampling distribution and α = 0.05, to ascertain the normality of the 5 samples provided. (note: for this experiment do not use the actual pdf but the experimental sampling distribution)

Here is the plot for histogram of sample size of 1500 samples

Here is the plot for histogram for sample size 2000

**Histogram of the data for population normal:**





The mean square error for above three sample size are as follows

**50 Samples : 0.05270954982723089**

**1500 Samples :0.011570768400108308**

**2500 Samples :0.010808983371253258**

Here as sample size is increasing the mean square error is decreasing and it’s absolute value is also too low. This verifies our assumption that the JBstatistics asymptotically follows chi squared distribution.

# Experiment 2:

1. In this experiment we compare the means for all the four vectors and compare which option is better for each of the bit rate. The means for the different options and bit rates are as shown:

**The mean for full HD 3MB is: 2.19230769231**

**The mean for full HD 9MB is: 6.34615384615**

**The mean for Upscaled 3MB is: 2.76923076923**

**The mean for Upscaled 9MB is: 6.19230769231**

The analysis done here is solely based on MOS

It is evident from the means of the observed data that the mean for **full HD 9MB** is higher than that of the mean for **Upscaled 9MB**. This favours the option 1 for 9 MB/s data rate. However, for **full HD 3MB** the mean is lower than that of the **Upscaled 3MB**. This favours the option2 for 3 MB/s data rate.

Now, as the analysis given above just takes into account, the mean opinion scores, we don’t take into account the spread and skewness of the data. This is a major flaw in this way of recommendation.

2. Now, we do the paired t test for the comparison of upscaled data and the full HD data for both the transmission rates. The value of *t* and the corresponding *p-value* for both the transmission rates is shown below:

**For 9MB transmission rate: t = 3.36 and p value=0.002**

**For 3MB transmission rate: t = 6.91 p value = 2.98 \* 10-7**

The p value shows our confidence in our Hypothesis that the mean does not change. Note that for 3 MB transmission rate, the confidence interval is much smaller than that of 9 MB. However, the t value for 9 MB is higher than that of 3 MB. This indicates that the change in mean for 3 MB is higher. For the problem at hand, we should consider transmitting by downscaling only if the change in mean is much less as this implies that the observers don’t observe a considerable difference. Thus, we should consider transmitting by downscaling for 9 MB data rate, but not for 3 MB data rate.

3.We see that the conclusion using the paired t test contradicts with the previous analysis using MOS. The reason behind this conclusion is that the MOS does not take into consideration the spread of the data, while paired t test is much more comprehensive test.

# Codes

## **Experiment 1:**

**'''**

**Created on Sep 19, 2016**

**@author: KANDARP**

**'''**

**import numpy as np**

**import scipy.stats as stats**

**from scipy.io import loadmat**

**import matplotlib.pyplot as plt**

**import matplotlib.mlab as mlab**

**import math**

**import pandas**

**from scipy.stats.kde import gaussian\_kde**

**from scipy.stats import norm**

**def obseredProbability(data,value):**

**count = 0**

**for i in data:**

**if(i<value):**

**count+=1**

**return (count/(float)(len(data)))**

**def normalize(data):**

**mean = np.mean(data)**

**var = np.var(data)**

**dev = np.std(data)**

**tmp = (data-mean)/float(dev)**

**return tmp**

**def MSE(data,data2,values):**

**mse = 0.**

**for i in values:**

**op = obseredProbability(data,i)**

**cp = obseredProbability(data2, i);**

**mse+=(op-cp)\*\*2**

**return math.sqrt(mse/((float)(len(values))))**

**def normplot(data,key):**

**mean = np.mean(data)**

**var = np.var(data)**

**dev = np.std(data)**

**tmp = (data-mean)/float(dev)**

**plt.figure()**

**plt.hist(tmp,normed=True,bins=50,align='mid')**

**x = np.linspace(-3, 3, 100)**

**plt.plot(x,mlab.normpdf(x, 0, 1))**

**plt.title('Histogram for '+key)**

**plt.legend(['Standard Normal','Histogram from the observed data'])**

**#plt.savefig('Normalised Histogram for '+key)**

**plt.show()**

**def skewness(data):**

**mean = np.mean(data)**

**std = np.std(data)**

**tmp = 0**

**for i in data:**

**tmp += (float(i-mean)/float(std))\*\*3**

**return float(tmp)/float(len(data))**

**def kurtosis(data):**

**mean = np.mean(data)**

**std = np.std(data)**

**tmp = 0**

**for i in data:**

**tmp += (float(i-mean)/float(std))\*\*4**

**return float(tmp)/float(len(data))**

**def rational\_approximation(t):**

**c = [2.515517, 0.802853, 0.010328]**

**d = [1.432788, 0.189269, 0.001308]**

**numerator = (c[2]\*t + c[1])\*t + c[0]**

**denominator = ((d[2]\*t + d[1])\*t + d[0])\*t + 1.0**

**return t - numerator / denominator**

**def normal\_CDF\_inverse(p):**

**assert p > 0.0 and p < 1**

**if p < 0.5:**

**return -rational\_approximation( math.sqrt(-2.0\*math.log(p)))**

**else:**

**return rational\_approximation(math.sqrt(-2.0\*math.log(1.0-p)))**

**def getZScore(data,key):**

**tmp = normalize(data)**

**tmp2 = sorted(tmp)[:len(tmp)-1]**

**theoretical=[]**

**size = len(data)**

**for i in range(1,size):**

**tmp1 = normal\_CDF\_inverse(float(i)/float(size))**

**theoretical.append(tmp1)**

**plt.figure()**

**plt.plot(sorted(data[:len(data)-1]),theoretical,'o')**

**plt.plot(sorted(data[:len(data)-1]),tmp2,'+')**

**plt.title('Normal Probability plot for '+key)**

**plt.legend(['Theoretical Z scores','Observed Z'])**

**plt.xlabel('Data in sorted manner')**

**plt.ylabel('Z Score')**

**plt.show()**

**def getJBScore(data):**

**ans=0**

**size=len(data)**

**S=skewness(data)**

**C=kurtosis(data)**

**return (size\*1.0/6) \*(S\*S + 1.0/4 \*(C-3)\*\*2 );**

**data=loadmat('data\_lab5.mat')**

**n=[50,1500,2500]**

**values=[0,1.2,3,4,5,6,7,8,9];**

**for i in n:**

**data=[ ]**

**for j in range(1,1000) :**

**x=np.random.standard\_normal(i);**

**data=np.append(data,getJBScore(x))**

**data2=np.random.chisquare(2,1000)**

**print(MSE(data, data2, values));**

**plt.figure()**

**plt.hist(data,bins=100,normed=True)**

**plt.hist(data2,bins=100,normed=True)**

**plt.title(str("Histogram for "+str(i)+" samples"))**

**plt.legend(['JB','Chisquare'])**

**plt.xlabel('values');**

**plt.ylabel('normalized frequency');**

**plt.show();**

## Experiment 2

import numpy as np

import scipy.stats as stats

from scipy.io import loadmat

import matplotlib.pyplot as plt

import matplotlib.mlab as mlab

import math

import pandas

from scipy.stats.kde import gaussian\_kde

from scipy.stats import norm

def printMean(data,key):

m = np.mean(data)

print('The mean for '+key+' is: '+str(m))

def plotHistogram(data,key):

plt.figure()

plt.hist(data,normed=False,bins=50,align='mid')

plt.title('Histogram for '+key)

def diffSum(data1,data2):

temp = data1 - data2

ans = np.cumsum(temp)

return ans

def diffSquareSum(data1,data2):

temp = data1 - data2

print('original '+str(temp))

temp = np.square(temp)

print('squared '+str(temp))

ans = np.cumsum(temp)

return ans

def getT(data1,data2):

n = len(data1)

sum = diffSum(data1,data2)

sums = diffSquareSum(data1,data2)

t = sum/(math.sqrt((n\*sums) - sum\*sum)/n-1)

return t

def printt(data1,data2,key):

t = getT(data1,data2)

print('The t for '+key+' is: '+t)

data = loadmat('data\_lab5.mat')

f3 = np.transpose(data['src1\_fullHD\_3Mb'])

f9 = np.transpose(data['src1\_fullHD\_9Mb'])

u3 = np.transpose(data['src1\_720p\_upscaled\_3Mb'])

u9 = np.transpose(data['src1\_720p\_upscaled\_9Mb'])

printMean(f3,'for full HD 3MB')

printMean(f9,'for full HD 9MB')

printMean(u3,'for Upscaled 3MB')

printMean(u9,'for Upscaled 9MB')

#

# plotHistogram(f3,'for full HD 3MB')

# plotHistogram(f9,'for full HD 9MB')

# plotHistogram(u3,'for Upscaled 3MB')

# plotHistogram(u9,'for Upscaled 9MB')

print(stats.ttest\_rel(f9,u9))

print(stats.ttest\_rel(f3,u3))