Shamma Kabir

Tony DePalatis

Data Science Lab 1

* 1. **Code:**

import numpy

import matplotlib.pyplot as plt

a = numpy.random.normal(-10, 5, 1000)

b = numpy.random.normal(10, 5, 1000)

c = a + b

# the histogram of the data

plt.hist(c, bins=30) plt.xlabel('Points')

plt.title(r'Histogram of New Data Set')

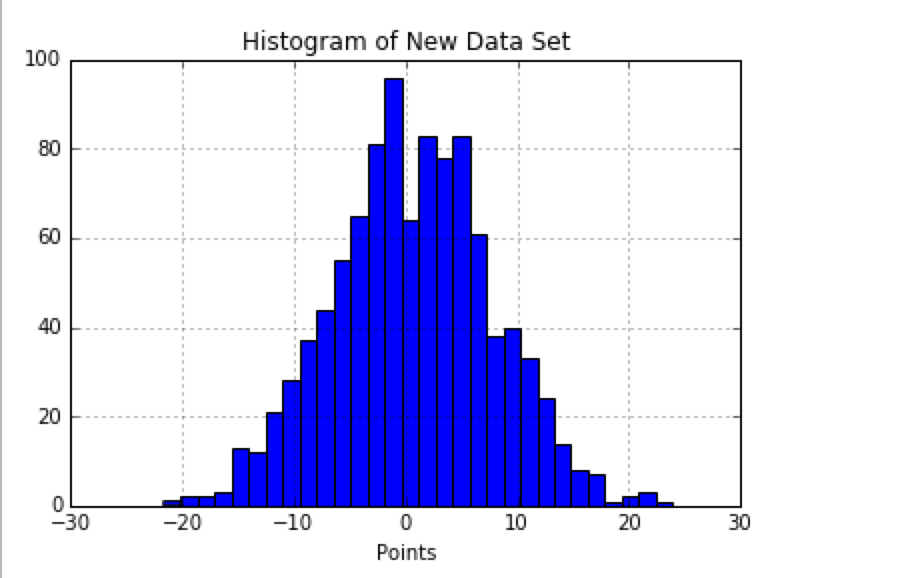
plt.grid(True)

plt.show()

variance = numpy.var(c)

mean = numpy.mean(c)

* 1. **Histogram of the Sum of two Gaussian:**



* 1. **Code:**

import numpy as np

import matplotlib.pyplot as plt

# n = 5

sampleMeans = []

for x in range(1000):

samples = np.random.binomial(1, 0.5, 5) sampleMeans.append(float(sum(samples)) / 5)

plt.hist(sampleMeans, bins=30)

plt.show()

# n = 300

sampleMeans = []

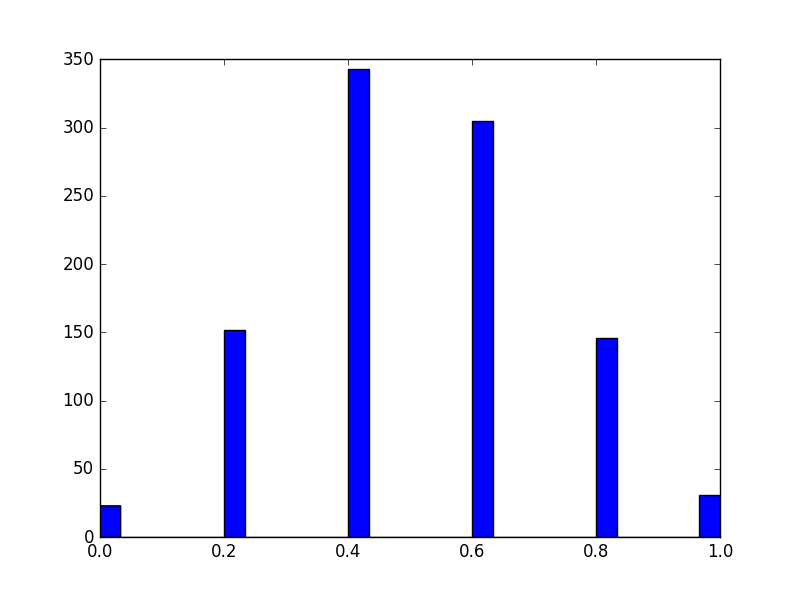
for x in range(1000):

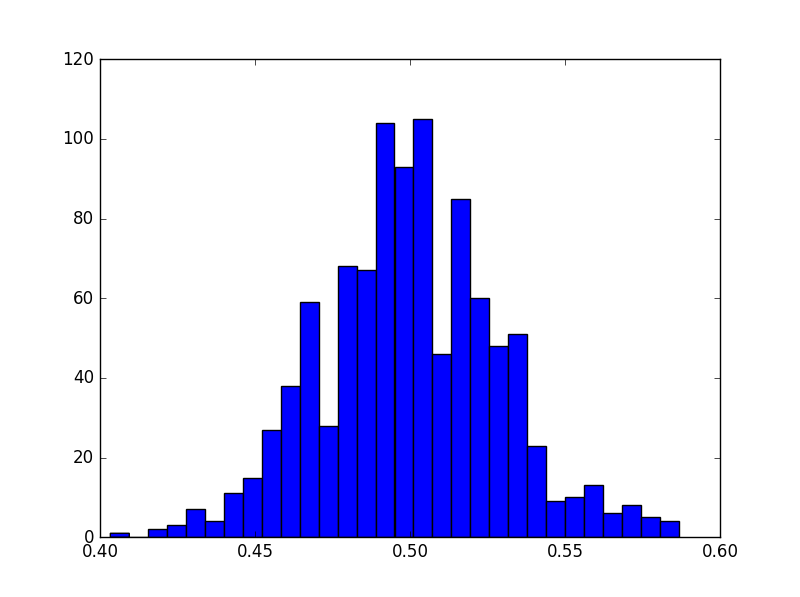
samples = np.random.binomial(1, 0.5, 300) sampleMeans.append(float(sum(samples)) / 300)

plt.hist(sampleMeans, bins=30)

plt.show()

* 1. **Small Sample Size, n = 5**



* 1. **Large Sample Size, n = 300**

1. **Code:**

import numpy as np

a = np.random.normal(0, 5, 25000)

#finding mean

total = 0

for i in range (0,24999):

total = total + a[i]

mean = total/250000

#finding standard deviation

numerator = 0

for i in range (0,24999):

first = a[i]- mean

sq = first\*first

numerator = numerator + sq

fraction = numerator/25000

std = np.sqrt(fraction)

1. 1. **Mean matrix** = [-4.98473101  5.09675713]
   2. **Covariance matrix** = [[ 20.00334121   0.80206716] [  0.80206716  30.16727405]]
   3. **Code:**

import numpy as np

import math

import matplotlib.pyplot as plt

mean = [-5, 5]

cov = [[20, 0.8], [0.8, 30]]

samples = np.random.multivariate\_normal(mean,cov, 10000)

sums = np.sum(samples, axis=0)

sampleMean = sums / len(samples)

#print np.mean(samples, axis=0)

varX = 0

varY = 0

sampleCov = 0

for x in range(len(samples)):

sampleCov += (samples[x][0] -sampleMean[0]) \* (samples[x][1] - sampleMean[1]) #calculate cov(x,y)

varX += math.pow((samples[x][0] - sampleMean[0]), 2) # calculate var(x)

varY += math.pow((samples[x][1] - sampleMean[1]), 2) # calculate var(y)

sampleCov /= len(samples)

varX /= len(samples)

varY /= len(samples)

covMatrix = np.array([[varX, sampleCov], [sampleCov, varY]])

#print np.cov(samples.T)

# Print results

print "Sample Mean Matrix: " #+ str(sampleMean)

print sampleMean

print "Sample Covariance Matrix: " #+ str(covMatrix)

print covMatrix

1. Patient Data Set
   1. There are 452 patients with 280 features.
   2. The meaning of the first four features, by inspection, could be age, gender, systolic blood pressure, and diastolic blood pressure, respectively.
   3. Code to change ‘?’ to column averages:

import pandas as pd

import numpy as np

data = pd.read\_csv("/Users/shammakabir/Downloads/PatientData.csv")

data\_replace = data.replace('?',np.nan)

data\_replace.fillna(data\_replace.mean())

data\_replace.to\_csv('out1.csv', sep=',')

* 1. You could test to see which features strongly influence the data set and which ones do not by deleting certain features and testing the accuracy of the new condition based on the known condition by using k-fold cross validation. Whenever the accuracy of the newly predicted condition is low after removing a certain feature, you will know that feature strongly influences the data set. Also, whenever the accuracy of the newly predicted condition is high (very similar or the same to known condition), then you will know that feature does not strongly influence the data set.
  2. Using the .corr() function in pandas, we think that feature 6, 87, and 89 are the most important features because those columns have the highest correlation coefficients in relation to the condition column. However, it’s also important to note that not all conditions will necessarily be heavily influenced by the same features. Therefore, for a different set of conditions, these features may or may not be as important.

1. Written Question:

