Vinh Vu

Dr. Chassiakos

EE 381

March 27th, 2019

Project 4: Simulating Continuous Random Variables with Various Distributions

**Problem 1: Simulate continuous random variables with selected distributions**

* 1. **Simulate a Uniform Random Variable**

**Introduction:** The Python function numpy.random.uniform(a,b,n) generates n random numbers with uniform probability distribution in open interval [a,b). When generating a uniform probability distribution, the PDF is defined as:

f(x) = ; and P(X = F(x) =

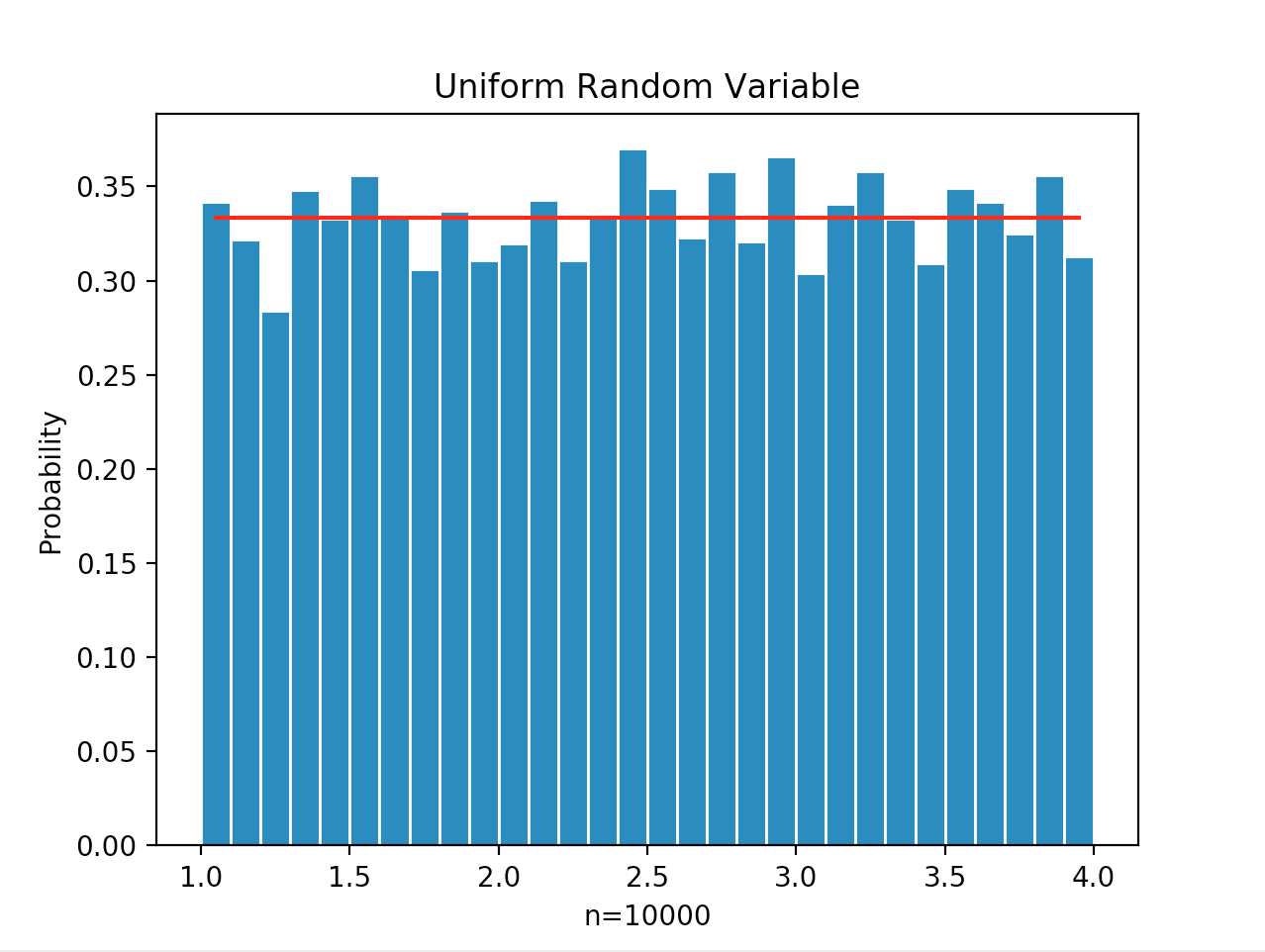
The mean and variance of a uniformly distributed random variable X are given by:

E(X) = μx = ; Var(X) = σ2x =

**Methodology:** We are to create a random variable X with a uniform distribution by using the Python function numpy.random.uniform(a,b,n) to generate n values of the R.V. X during open interval [a,b). For this problem, a value is 1.0 and b value is 4.0 and n value is 10,000 indicating 10,000 values. After generating n values, we use histogram function to plot a bargraph of 10,000 values. On the same graph, the probability density function of R.V. is graphed as well. Using the mean and variance formulas provided, theoretical values are calculated and then compared to experimental measurement from numpy.mean and numpy.std.

**Results and Conclusion:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 1: Statistic for a Uniform Distribution** | | | |
| **Expectation** | | **Standard Deviation** | |
| Theoretical  Calculation | Experimental Measurement | Theoretical  Calculation | Experimental Measurement |
| 2.5 | 2.5072336784760916 | 0.866025 | 0.8629342986881777 |



**Appendix:**

# Vinh Vu, 015347252

# Lab 4, Problem 1.1

# Uniform R.V

import numpy as np

import random

import string

import matplotlib

import matplotlib.pyplot as plt

a = 1.0

b = 4.0

n = 10000

x = np.random.uniform(a,b,n)

# Create bins and histograms

nbins = 30

edgecolor = 'w'

bins = [float(x) for x in np.linspace(a,b,nbins+1)]

h1, bin\_edges = np.histogram(x,bins, density=True)

# Define points on the horizontal axis

be1 = bin\_edges[0:np.size(bin\_edges)-1]

be2 = bin\_edges[1:np.size(bin\_edges)]

b1 = (be1+be2)/2

barwidth = b1[1]-b1[0]

plt.close('all')

# Plot the bar graph

fig1 = plt.figure("Uniform Random Variable")

plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)

# Plot the uniform pdf

def UnifPDF(a,b,x):

f = (1/abs(b-a))\*np.ones(np.size(x))

return f

f=UnifPDF(a,b,b1)

plt.plot(b1,f,'r')

plt.title("Uniform Random Variable")

plt.xlabel('n=10000')

plt.ylabel('Probability')

plt.show()

#calculate the mean and std

mu\_x = np.mean(x)

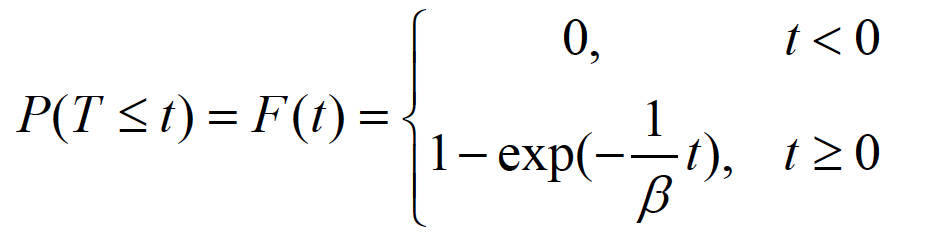
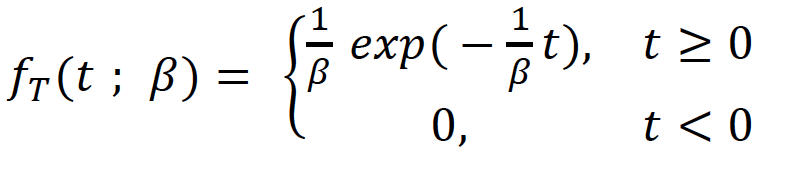
sig\_x = np.std(x)

print("mu\_x =",mu\_x)

print("sig\_x =",sig\_x)

* 1. **Simulating an Exponentially distributed Random Variable**

**Introduction:** The Python function numpy.random.exponential (beta,n) generates n random numbers with exponential probability function. When generating the exponential probability function, the PDF of a random variable is defined as:



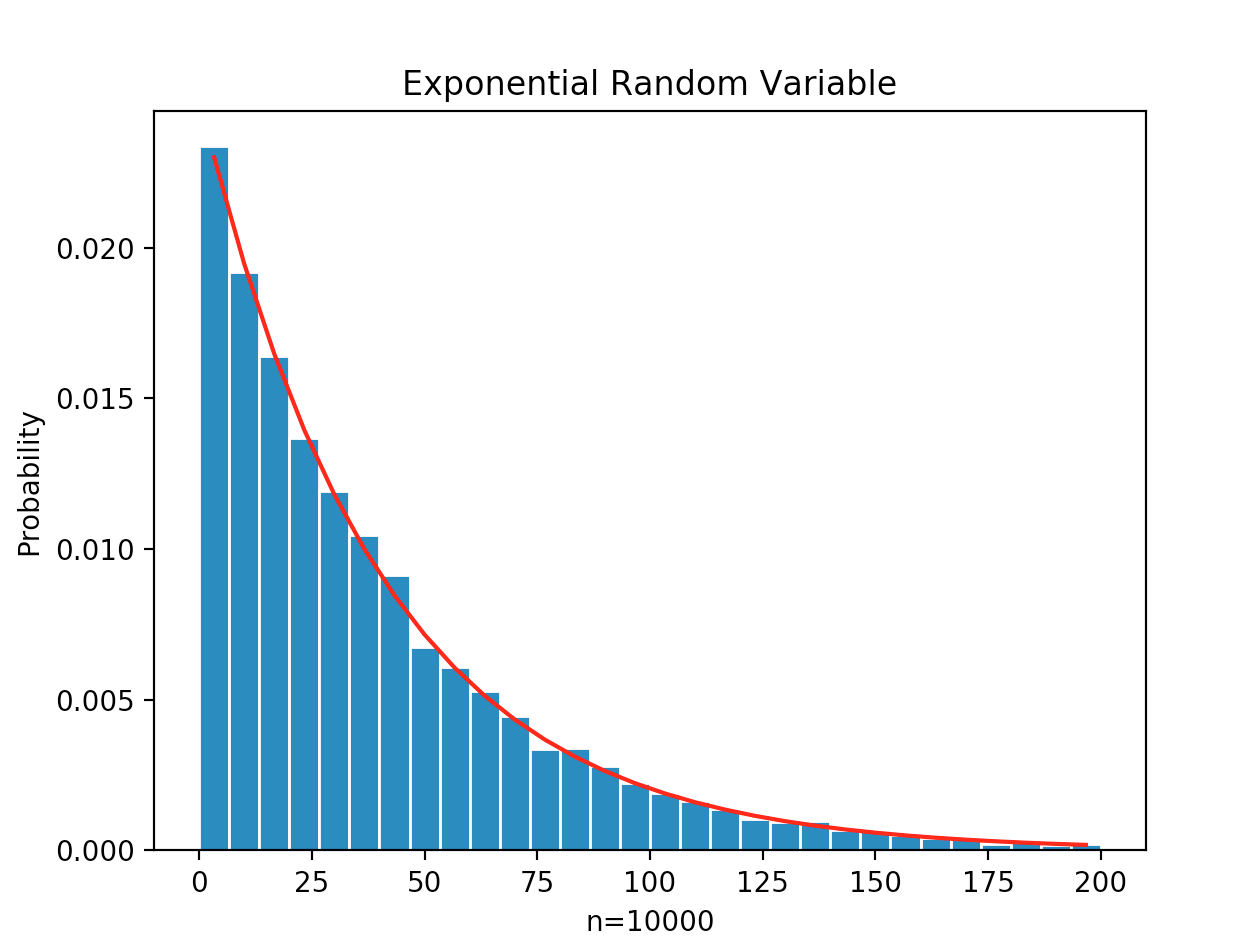
For exponential probability distribution random variables, the mean and standard deviation are given by:

;

**Methodology:** We are to create a random variable T with an exponential distribution by using the Python function numpy.random.exponential (beta,n) to generate n values of R.V. T. n value remains the same, 10,000, and beta is given as 40. After generating 10,000 values for random variable T, we use histogram to plot a bargraph of the experimental values of T. On the same graph, the probability desnsity function of R.V. is graphed as well. Using the mean and variance formulas provided, theoretical values are calculated and then compared to experimental measurement from numpy.mean and numpy.std.

**Results and Conclusion:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 1: Statistic for a Exponential Distribution** | | | |
| **Expectation** | | **Standard Deviation** | |
| Theoretical  Calculation | Experimental Measurement | Theoretical  Calculation | Experimental Measurement |
| 40 | 40.282819791880925 | 40 | 40.356735144258685 |



**Appendix:**

# Vinh Vu, 015347252

# Lab 4, Problem 1.2

# Exponential R.V

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

beta = 40

n = 10000

t = np.random.exponential(beta,n)

# Create bins and histograms

nbins = 30

edgecolor = 'w'

bins = [float(t) for t in np.linspace(0,200,nbins+1)]

h1, bin\_edges = np.histogram(t,bins, density=True)

# Define points on the horizontal axis

be1 = bin\_edges[0:np.size(bin\_edges)-1]

be2 = bin\_edges[1:np.size(bin\_edges)]

b1 = (be1+be2)/2

barwidth = b1[1]-b1[0]

plt.close('all')

# Plot the bar graph

fig1 = plt.figure(1)

plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)

# Plot the uniform pdf

def ExpPDF(beta,t):

f=np.exp((-1/beta)\*t)\*(1/beta)

return f

f=ExpPDF(beta,b1)

plt.plot(b1,f,'r')

plt.title("Exponential Random Variable")

plt.xlabel("n=10000")

plt.ylabel("Probability")

plt.show()

#calculate the mean and std

mu\_x = np.mean(t)

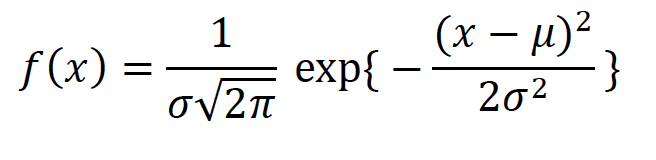
sig\_x = np.std(t)

print("mu\_x=",mu\_x)

print("sig\_x=",sig\_x)

* 1. **Simulating a Normal Random Variable**

**Introduction:** The Python function numpy.random.normal(mu, sigma, n) generates n random numbers from a Gaussian probability distribution with mean and standard deviation . The PDF of a normal random variable X with mean and standard deviation is defined as:



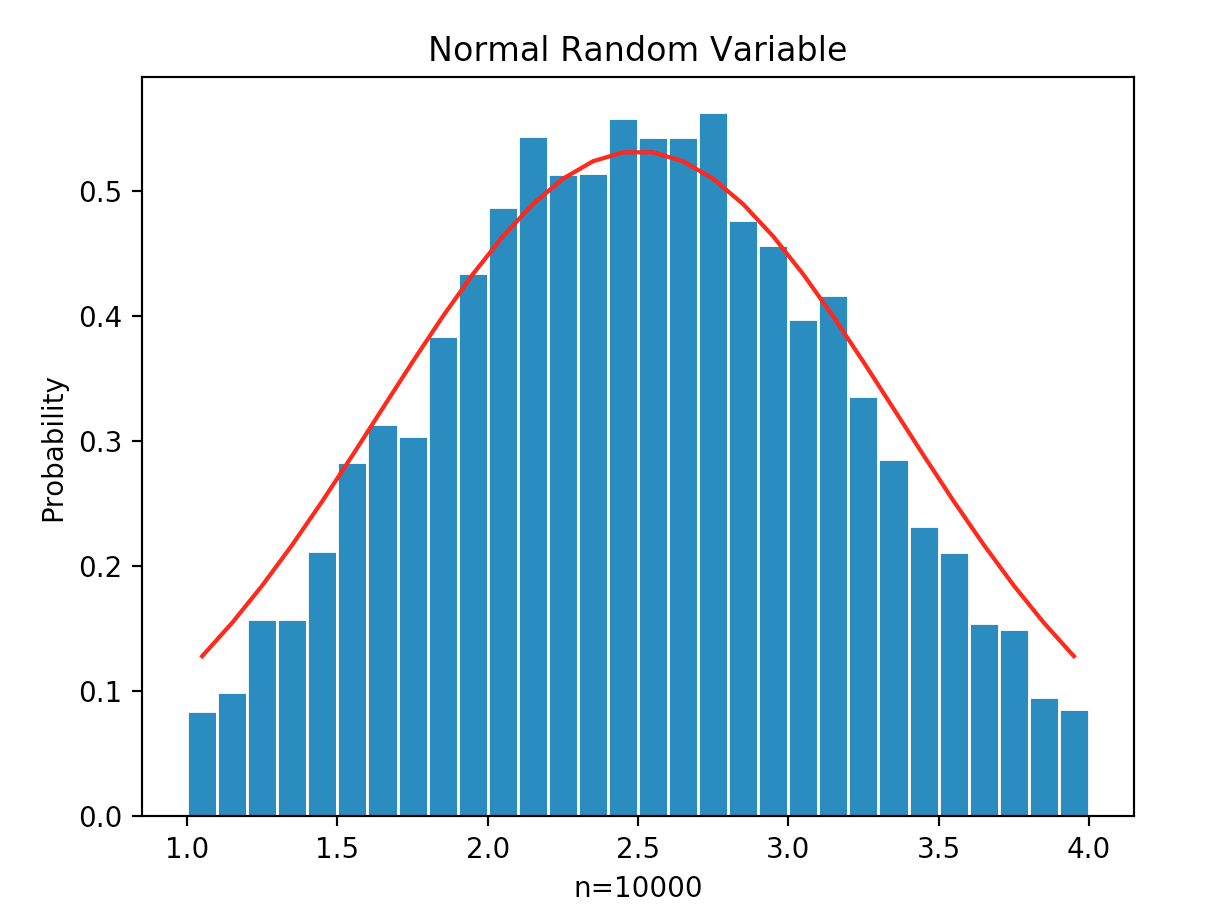
The mean and variance of the normally distributed random variable X are defined as:

E(X) = ; Var(X) =

**Methodology:** We are to create a random variable X with a normal distribution using Python function numpy.random.normal(mu, sigma, n) to generate n values of the R.V. X. Given mu value is 2.5, sigma value is 0.75, and n remains 10,000. After generating 10,000 values for random variable X, we use histogram to plot a bargraph of the experimental values of X. On the same graph, the probability desnsity function of R.V. is graphed as well. Using the mean and variance formulas provided, theoretical values are calculated and then compared to experimental measurement from numpy.mean and numpy.std.

**Results and Conclusion:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 1: Statistic for a Normal Distribution** | | | |
| **Expectation** | | **Standard Deviation** | |
| Theoretical  Calculation | Experimental Measurement | Theoretical  Calculation | Experimental Measurement |
| 2.5 | 2.487389748269103 | 0.75 | 0.7518657075787537 |



**Appendix:**

# Vinh Vu, 015347252

# Lab 4, Problem 1.3

# Normal R.V

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

mu=2.5

sigma=0.75

n=10000

x = np.random.normal(mu,sigma,n)

# Create bins and histograms

nbins = 30

edgecolor = 'w'

bins = [float(t) for t in np.linspace(1,4,nbins+1)]

h1, bin\_edges = np.histogram(x,bins, density=True)

# Define points on the horizontal axis

be1 = bin\_edges[0:np.size(bin\_edges)-1]

be2 = bin\_edges[1:np.size(bin\_edges)]

b1 = (be1+be2)/2

barwidth = b1[1]-b1[0]

plt.close('all')

# Plot the bar graph

fig1 = plt.figure(1)

plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)

# Plot the uniform pdf

def NormPDF(mu,sigma,z):

f=np.exp(-(z-mu)\*\*2/(2\*\*sigma\*\*2))\*(1/(sigma\*np.sqrt(2\*np.pi)))

return f

f=NormPDF(mu,sigma,b1)

plt.plot(b1,f,'r')

plt.title("Normal Random Variable")

plt.xlabel("n=10000")

plt.ylabel("Probability")

plt.show()

# Calculate the mean and std

mu\_x = np.mean(x)

sig\_x = np.std(x)

print("mu\_x=",mu\_x)

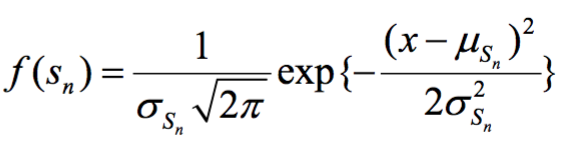
print("sig\_x=",sig\_x)

**Problem 2: The Central Limit Theorem**

**Introduction:** For Central Limit Theorem, there are n independent random variables, X1 to Xn with the same probability distribution. The sum Sn = X1+X2+…+Xn with mean and . This sum Sn is a random variable with mean and standard deviation are defined as:

;

The Central Limit Theorem states that as the probability distribution of the R.V. Sn approaches a normal distribution. The PDF of the normally distributed R.V. Sn is defined as:



**Methodology:** For theoretical calculations, W, the thickness of a collection of books, is a random variable uniformly distributed between interval [a,b). We are to calculate the mean and standard deviation between a and b being 1.0 and 4.0 cm. Results is record in Table 1. The collection of books is piled in a stack of n = 1, 5, 10, or 15 books. The width Sn of a stack of n books is a random variable and it has a mean and standard deviation of . We are to calculate the mean and standard deviation of the stacked books, for n=1, 5, and 15. Calculations are recorded on Table 2. After theoretical calculations, we plot the results using histogram. n value for books are {1, 5, 15} and we run the experiments N = 10,000 to simulate random variable S = Wn. On the same figure, we plot the normal distribution probability function and compare it with the plot of f(x) defined as above.

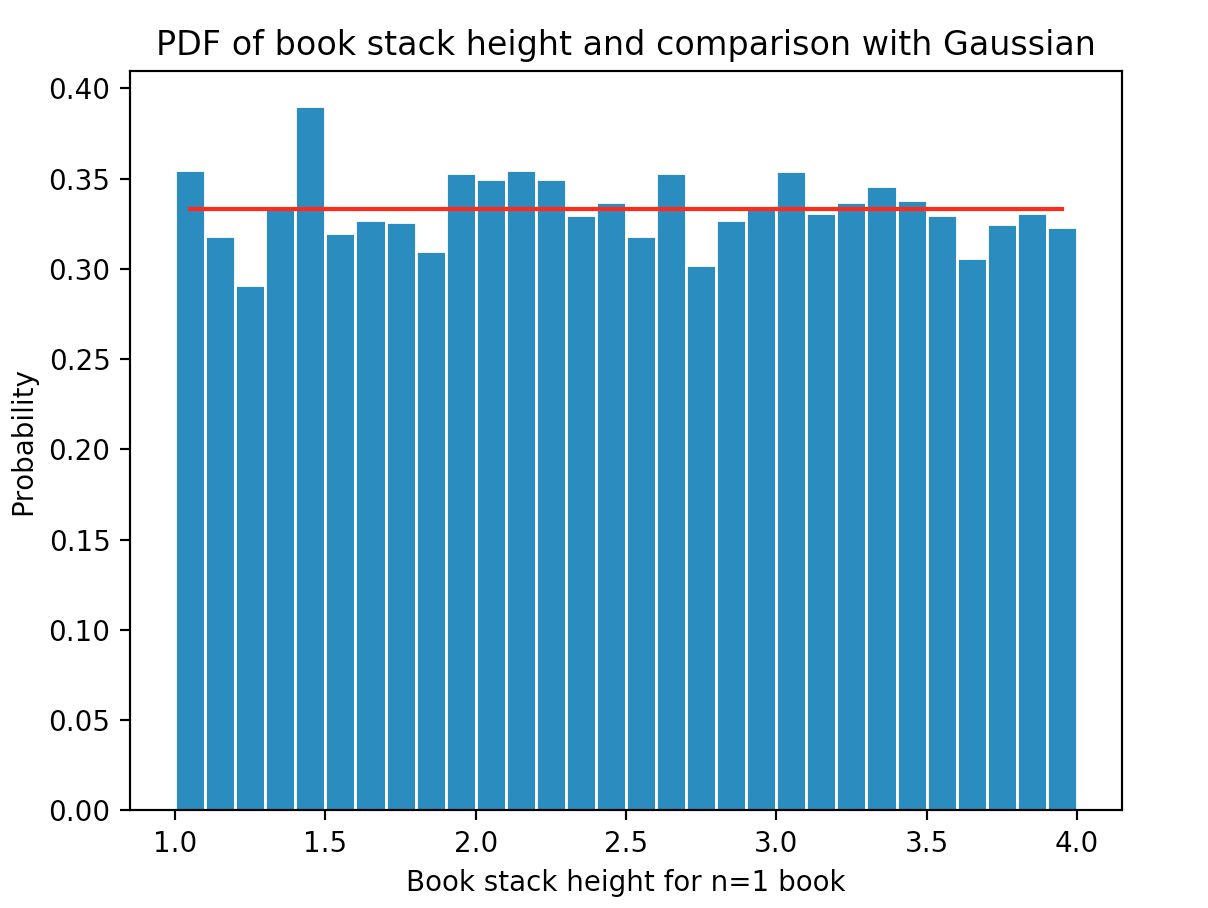
**Results and Conclusion:**

**Table 1:**

|  |  |
| --- | --- |
| Mean thickness of a single book (cm) | Standard Deviation of thickness (cm) |
| 2.5 | 0.75 |

**Table 2:**

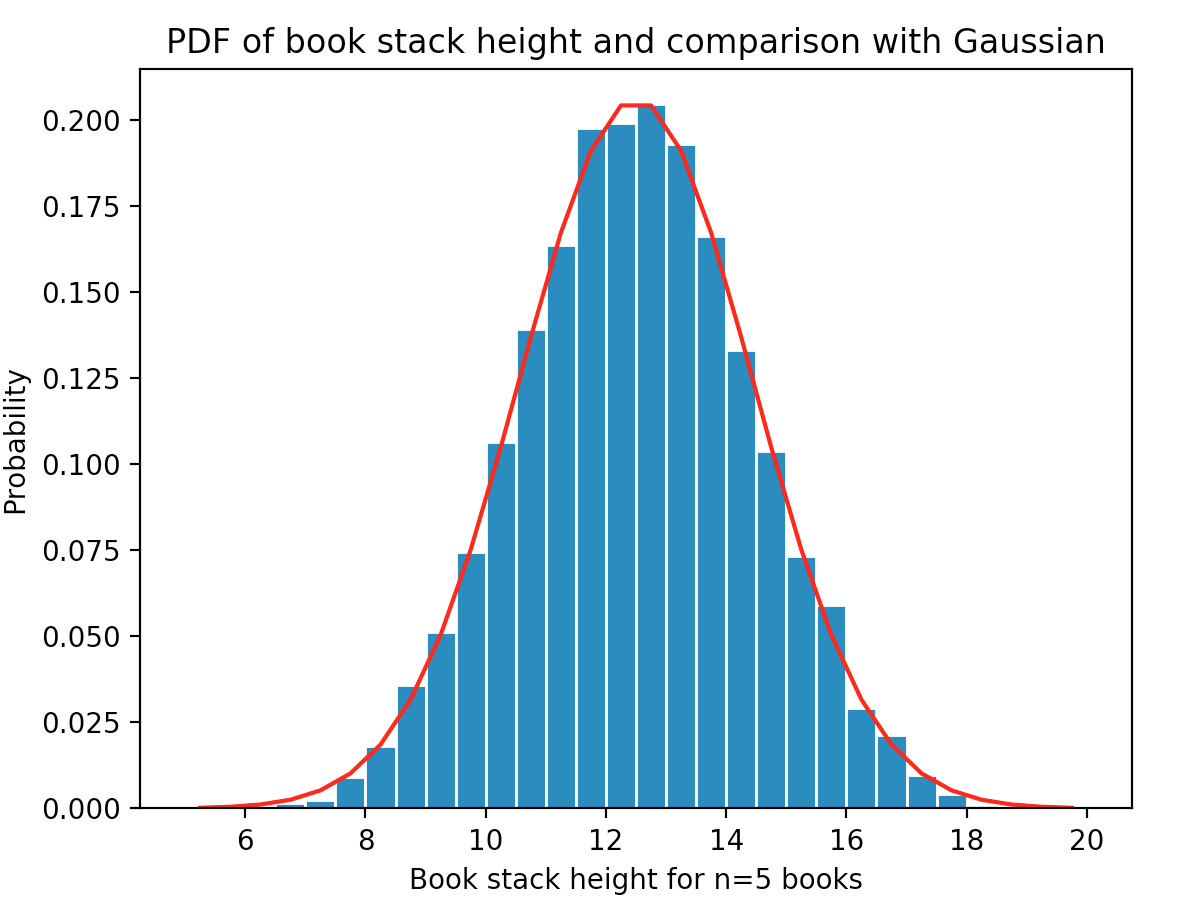
|  |  |  |
| --- | --- | --- |
| Number of books n | Mean thickness of a stack of n books (cm) | Standard deviation of the thickness for n books |
| n = 1 | = 2.5 | = 0.75 |
| n = 5 | = 12.5 | = 3.75 |
| n = 15 | = 37.5 | = 11.25 |



nbooks= 1

mu\_x= 1.9371276728643023

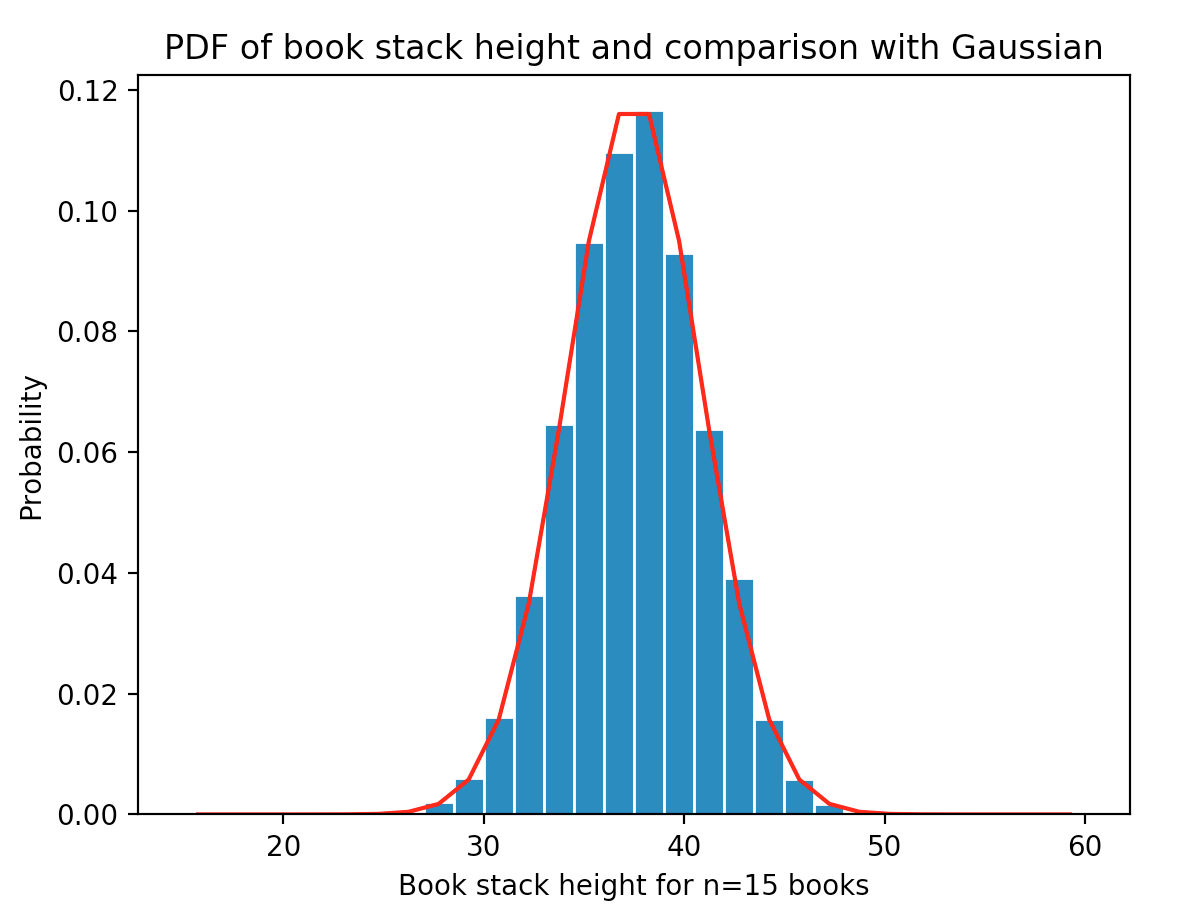
sig\_x= 0.0



nbooks= 5

mu\_x= 10.858311209293973

sig\_x= 1.7540057347236415



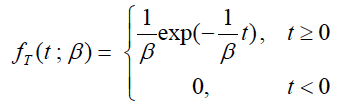
nbooks= 15

mu\_x= 34.068064895565406

sig\_x= 3.312551004160586

**Problem 3: Distribution of the Sum of Exponential RV’s**

**Introduction:** The problem involves a battery-operated medical monitor. The lifetime (T) is a random variable with an exponentially distributed lifetime. A battery lasts an average of days. The PDF of the battery life time is given as:



Therefore, the mean and standard deviation of the random variable T are:

;

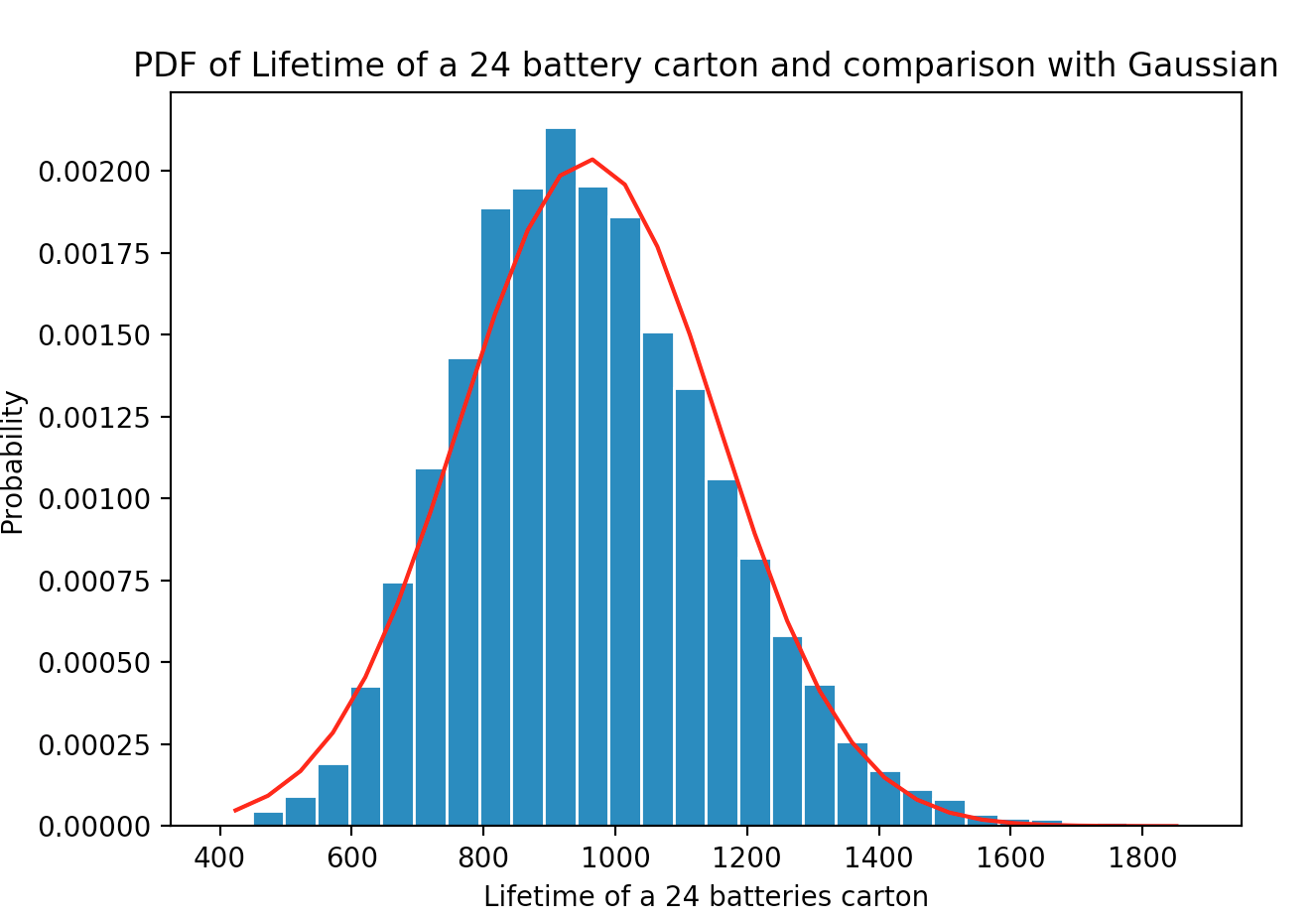
When a battery fails, it is replaced by a new one. Batteries are purchased in a carton of 24. The goal is to simulate the R.V. representing the lifetime of a carton of 24 batteries, and create a histogram. The PDF of one carton of 24 batteries can be approximated by a normal distribution with mean and standard deviation given as:

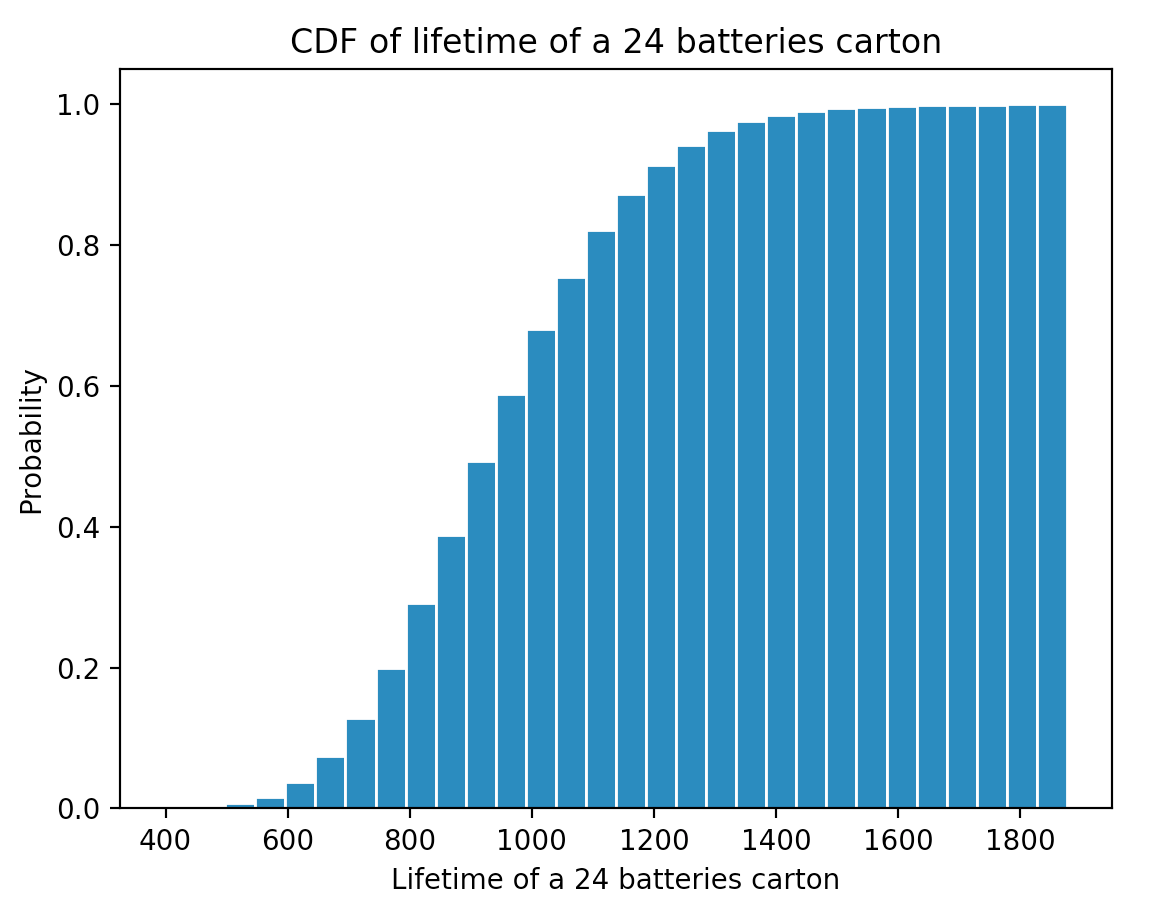
;

**Methodology:** We are to create a vector of 24 elements that represents a carton. Each one of the elements is an exponentially distributed random variable (T), with mean lifetime equal to . The sum of the elements is defined as: . The experiment is repeated N=10000 times to create the histogram for PDF of the lifetime of a carton. In order to create the CDF, the Python function numpy.cumsum is used. Since the CDF is the integral of the PDF, the sum is multiplied by the barwidth to calculate the areas. The correct CDF should be an increasing graph, starting at 0.0 and end at 1.0.

**Results and Conclusion:**

|  |  |
| --- | --- |
| **QUESTION** | **ANS.** |
| 1. Probability that the carton will last longer than three years | **0.23** |
| 2. Probability that the carton will last between 2.0 and 2.5 years | **0.37** |





mu\_c= 959.7514293280067

sig\_c= 195.70314549232413

**Appendix:**

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

# Generate the values of the RV X

beta = 40

nbat = 24

n=10000

C=np.zeros((n,1))

for i in range (0,n):

t = np.random.exponential(beta,nbat)

w=np.sum(t)

C[i]=w

# Create bins and histogram

nbins=30

edgecolor='w'

bins=[float(t) for t in np.linspace(min(C),max(C),nbins+1)]

h1, bin\_edges = np.histogram(C,bins,density=True)

# Define points on the horizontal axis

be1=bin\_edges[0:np.size(bin\_edges)-1]

be2=bin\_edges[1:np.size(bin\_edges)]

b1=(be1+be2)/2

barwidth=b1[1]-b1[0]

plt.close('all')

# Plot the bar graph

fig1=plt.figure(1)

plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)

# Plot the uniform PDF

def gaussian(mu, sig, z):

f = np.exp(-(z - mu)\*\*2/(2\*sig\*\*2))/(sig\*np.sqrt(2\*np.pi))

return f

mu = nbat \* beta

sig = beta \* (np.sqrt(nbat))

f = gaussian(mu,sig,b1)

plt.plot(b1,f,'r')

plt.title('PDF of Lifetime of a 24 battery carton and comparison with Gaussian')

plt.xlabel('Lifetime of a 24 batteries carton')

plt.ylabel('Probability')

plt.show()

fig2 = plt.figure(1)

h2 = np.cumsum(h1)\*barwidth

plt.bar(b1,h2, width=barwidth, edgecolor = edgecolor)

plt.title('CDF of lifetime of a 24 batteries carton')

plt.xlabel('Lifetime of a 24 batteries carton')

plt.ylabel('Probability')

plt.show()

# Calculate the mean and std

mu\_C=np.mean(C)

print(mu\_C)

sig\_C=np.std(C)

print(sig\_C)