EE 381 Project 4

Central Limit Theorem:
Simulating Continuous Random Variables with Variable Distributions

by Alexandra McBride

November 2, 2019

California State University, Long Beach

Dr. Anastasios Chassiakos

Problem 1

Introduction

This problem involves continuous random variables, and we will be simulating variable distributions of those random variables. We will look at three distributions of the random variable, which are the uniform distribution, the exponential distribution, and the normal distribution. We will be plotting the bar graph and probability density function for each distribution to get a visual representation of these distributions. Lastly, we want to look at the expectation and standard deviation of these distributions. We will get an experimental result for the expectation and standard deviation from using our generated random variables, and we will them compare them with our results calculated using the theoretical formulas.

Methodology

For this problem, I will be using Python in the PyCharm IDE. The tools used will include methods from the `numpy,math,`and `mathplotlib.pyplot` libraries.

<u>Uniform Distribution</u>

For the uniform distribution, we will be creating a random variable X and generate it with a uniform distribution in the open interval $[a, b)$. The program will create 10,000 random variables in this interval using the `numpy.uniform()` method. The program will the generate a bar graph for the uniform distribution, which will show the values from a to b. It will also generate the probability distribution function (PDF), which will be represented by a red bar on the graph. The PDF is given by:

$$f(x) = \begin{cases} \dfrac{1}{(b-a)} & a \leq x \leq b \\ 0 \ \ otherwise & \end{cases}$$

We will then get the experimental mean μ_x and standard deviation σ_x values for the random variable X, using `numpy.mean()` and `numpy.std()` methods, then calculate the theoretical mean μ_x and standard deviation σ_x using formulas:

$$\mu_x = \frac{a+b}{2} \qquad \sigma_x = \sqrt{\frac{(b-a)^2}{12}}$$

We will then compare the results for the mean and standard deviation for the random variable X.

<u>Exponential Distribution</u>

For the exponential distribution, we will be creating a random variable T and generate it with an exponential distribution. The program will create 10,000 random variables using the `numpy.exponential()` method. The program will the generate a bar graph for the exponential distribution. It will also generate the PDF, which will be represented by a red bar on the graph. The PDF is given by:

$$f_t(t;\beta) = \begin{cases} \frac{1}{\beta}\exp\left(\frac{1}{\beta}t\right) & t \leq 0 \\ 0 & otherwise \end{cases}$$

We will then get the experimental mean μ_x and standard deviation σ_x values for the random variable X, using `numpy.mean()` and `numpy.std()` methods, then calculate the theoretical mean μ_x and standard deviation σ_x using:

$$\mu_x = \beta \qquad \sigma_x = \beta$$

We will then compare the results for the mean and standard deviation for the random variable X.

Normal Distribution

For the normal distribution, we will be creating a random variable T and generate it with an exponential distribution. The program will create 10,000 random variables using the `numpy.normal()` method. The program will the generate a bar graph for the normal distribution. It will also generate the PDF, which will be represented by a red bar on the graph. The PDF is given by:

$$f(x) = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}}\exp\{-\frac{(x-\mu)^2}{2\sigma^2}\} & t \leq 0 \\ 0 & otherwise \end{cases}$$

We will then get the experimental mean μ_x and standard deviation σ_x values for the random variable X, using `numpy.mean()` and `numpy.std()` methods, then calculate the theoretical mean μ_x and standard deviation σ_x using:

$$\mu_x = \mu \qquad \sigma_x = \sigma$$

We will then compare the results for the mean and standard deviation for the random variable X.

Results and Conclusion

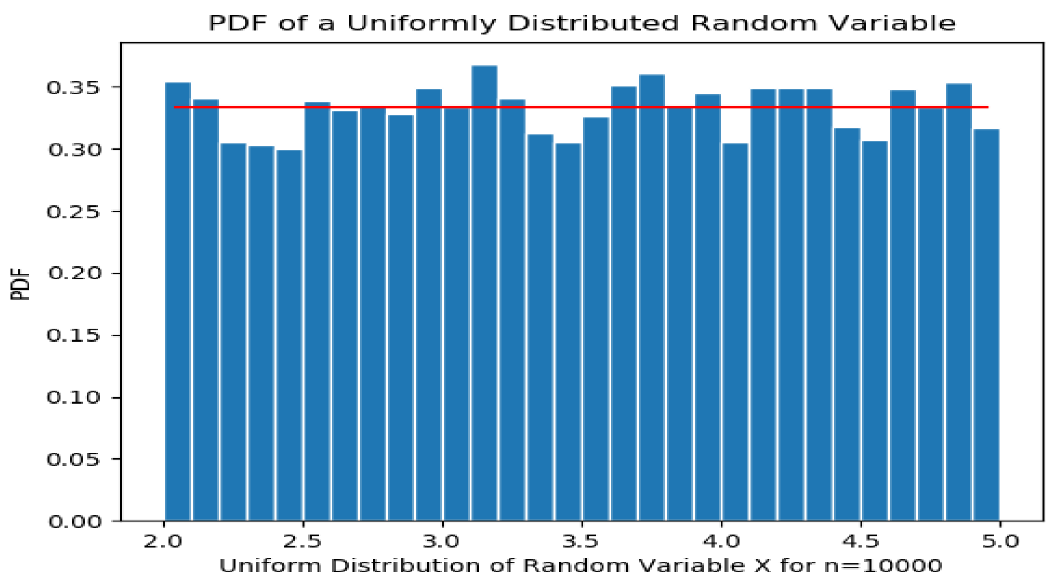
Uniform Distribution

The values for the uniform distribution to produce these results are as follows:

a= 2.0 b=5.0

Here are the results for the uniform distribution:

Bar Graph with Probability Density Function



Results Table

Table 1: Statistics for Uniform Distribution			
Expectation		**Standard Deviation**	
Experimental	**Theoretical**	**Experimental**	**Theoretical**
3.4989578436703743	3.5	0.8676570020256748	0.8660254037844386

From the graph, we see that the distribution of the random variable appears to be very even over the interval between 2.0 and 5.0, which is the result that is expected. From comparing the results for the expectation and the standard deviation, the experimental and theoretical results seem to be very close for both, so the methods used for the experimental results are accurate.

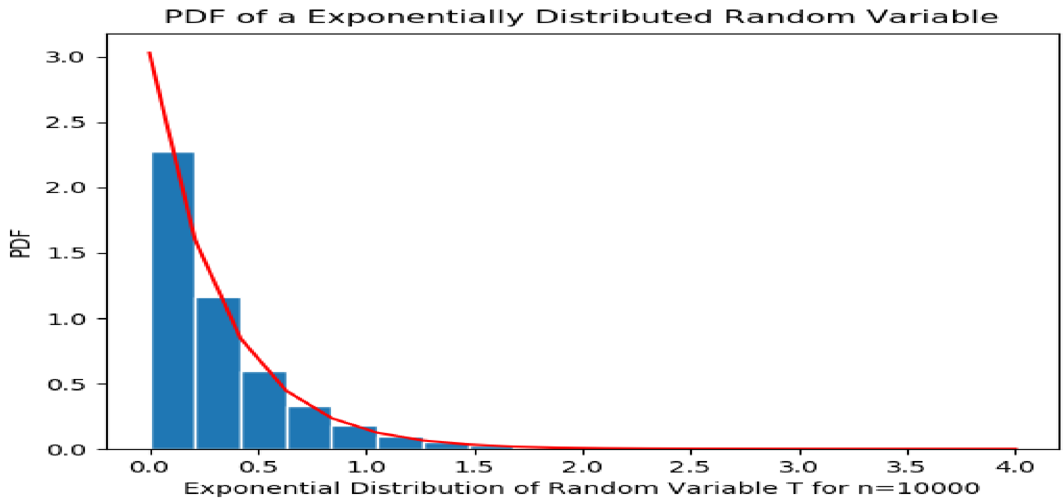
Exponential Distribution

The values for the uniform distribution to produce these results are as follows:

$$\beta = 0.33$$

Here are the results for the exponential distribution:

Bar Graph with Probability Density Function



Results Table

Table 2: Statistics for Exponential Distribution			
Expectation		**Standard Deviation**	
Experimental	**Theoretical**	**Experimental**	**Theoretical**
0.3332410282163558	0.33	0.3321761077973152	0.33

From the graph, we see that the distribution of the random variable is highest at 0.0 but begins to decrease as the random variable t increases, which is the result that is expected. From comparing the results for the expectation and the standard deviation, the experimental and theoretical results seem to be very close for both, so the methods used for the experimental results are accurate.

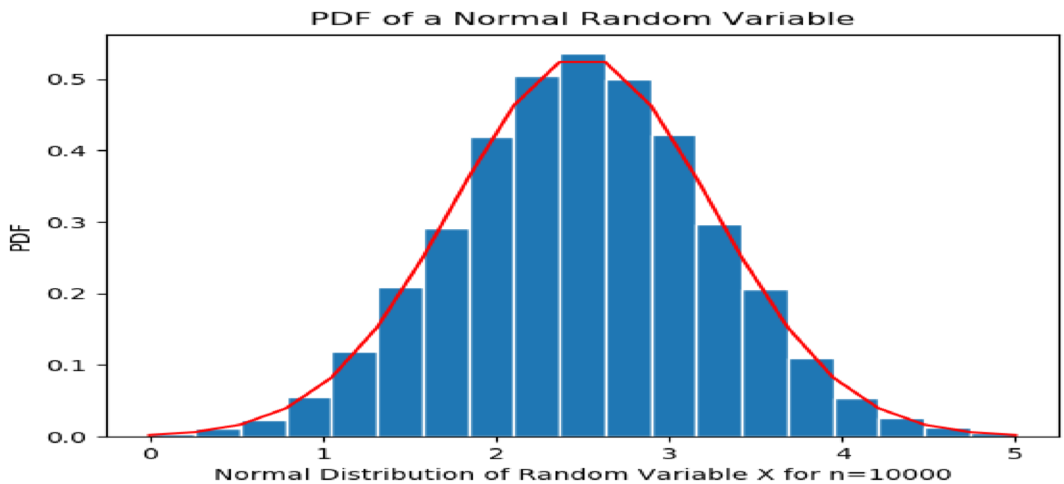
Normal Distribution

The values for the uniform distribution to produce these results are as follows:

$$\mu = 2.5 \quad \sigma = 0.75$$

Here are the results for the normal distribution:

Bar Graph with Probability Density Function



Results Table

Table 3: Statistics for Normal Distribution			
Expectation		**Standard Deviation**	
Experimental	**Theoretical**	**Experimental**	**Theoretical**
2.515754161602878	2.5	0.7513650542452536	0.75

From the graph, we see that the distribution of the random variable is highest at the mean, which is at about 2.5. This is the result that is expected for a normal distribution. From comparing the results for the expectation and the standard deviation, the experimental and theoretical results seem to be very close for both, so the methods used for the experimental results are accurate.

Appendix

File: uniformDistribution.py

```
# PROBLEM 1a  The parameter values are a=2.0  b=5.0

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

#PLOT THE UNIFORM PDF
def UnifPDF(a,b,x):
 f=(1/abs(b-a))*np.ones(np.size(x))
 return f

def uniform():
```

```
#Generate the values of the RV X
a=2.0; b=5.0; n=10000;
x=np.random.uniform(a,b,n)

#Create bins and histogram
nbins=30; # Number of bins
edgecolor='w'; # Color separating bars in the bargraph
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] # Width of bars in the bargraph
plt.close('all')


# PLOT THE BAR GRAPH
fig1=plt.figure(1)
plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)
f=UnifPDF(2.0,5.0,b1)
plt.plot(b1,f,'r') #red line
plt.title("PDF of a Uniform Random Variable")
plt.xlabel("Uniform Distribution of Random Variable X for n=10000")
plt.ylabel("PDF")

#Saving the file
filename = input("Enter a name and extension (.pdf) to save the file as :")
plt.savefig(filename)

#CALCULATE THE MEAN AND STANDARD DEVIATION

#Theoretical
Expct=(a+b) / 2
print(Expct)
SD = math.sqrt(((b-a)**2)/12)
print(SD)

#Experimental
mu_x=np.mean(x)
print(mu_x)
sig_x=np.std(x)
print(sig_x)
```

uniform()

File: exponentialDistribution.py

```
# PROBLEM 1b  The parameter value is beta =0.33

import numpy as np
import matplotlib.pyplot as plt

#PLOT THE UNIFORM PDF
def ExpPDF(beta):
  x=np.linspace(0, 4, 20)
  e=np.exp((-1/beta)*x)
  return(1/beta)*e

def exponential():

   #Generate the values of the RV X
   beta=0.33; n=10000;
   t=np.random.exponential(beta,n)
   print(t)

   #Create bins and histogram
   nbins=30; # Number of bins
   edgecolor='w'; # Color separating bars in the bargraph
   bins=[float(x) for x in np.linspace(0,4,20)]
   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] # Width of bars in the bargraph
   plt.close('all')


   # PLOT THE BAR GRAPH
   fig1=plt.figure(1)
   plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)
   x1=np.linspace(0,4,20)
   y=ExpPDF(beta)
   plt.plot(x1,y,'r') #red line
   plt.title("PDF of a Exponentially Distributed Random Variable")
   plt.xlabel("Exponential Distribution of Random Variable T for n=10000")
   plt.ylabel("PDF")
```

```
   #Saving the file
   filename=input("Enter a name and extension (.pdf) to save the file as :")
   plt.savefig(filename)


   #CALCULATE THE MEAN AND STANDARD DEVIATION

   #Experimental
   mu_x=np.mean(t)
   print(mu_x)
   sig_x=np.std(t)
   print(sig_x)

exponential()
```

File: normalDistribution.py

```
#PROBLEM 1c The parameter values are mu=2.5 , sigma= 0.75

import numpy as np
import matplotlib.pyplot as plt

#PLOT THE Normal PDF
def gaussian(mu,sig):
 x = np.linspace(0, 5, 20)
 f=np.exp(-(x-mu)**2/(2*sig**2))/(sig*np.sqrt(2*np.pi))
 return f

def normal():
 # Generate the values of the RV X
 mu=2.5 ; sig=0.75; n=10000;
 x=np.random.normal(mu,sig,n)

# Create bins and histogram
 nbins=30; # Number of bins
 edgecolor='w'; # Color separating bars in the bargraph
 bins=[float(x) for x in np.linspace(0,5,20)]
 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] # Width of bars in the bargraph
 plt.close('all')
```

```
# PLOT THE BAR GRAPH
fig1=plt.figure(1)
plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)
x1=np.linspace(0,5,20)
# PLOT THE GAUSSIAN FUNCTION
f=gaussian(mu,sig)
plt.plot(x1,f,'r')
plt.title("PDF of a Normal Random Variable")
plt.xlabel("Normal Distribution of Random Variable X for n=10000")
plt.ylabel("PDF")

#Saving the file
filename=input("Enter a name and extension (.pdf) to save the file as :")
plt.savefig(filename)

# CALCULATE THE MEAN AND STANDARD DEVIATION

# Experimental
mu_x = np.mean(x)
print(mu_x)
sig_x = np.std(x)
print(sig_x)
```

Problem 2

Introduction

This problem involves a collection of books, which each have continuous random variable W to represent the thickness of the book that is uniformly distributed between a minimum of a and a maximum of b cm centimeters. For the random variable W, we simply want to see the mean and standard deviation of the thickness of a given book. Following this, the books are piled into stacks, which have random variable S_n, which is the sum of the widths of n books. For the random variable W, we want to find mean and standard deviation of the thickness of the books for a stack of 1 book , a stack of 5 books, and a stack of 15 books, along with bar graphs for each of them.

Methodology

For this problem, I will be using Python in the PyCharm IDE. The tools used will include methods from the `numpy`,`math`,and `mathplotlib.pyplot` libraries.

Thickness of a single book

The thickness of a single book will be a uniform random variable W. The program will be creating the random variable W using `numpy.uniform()` and generate it with a uniform distribution in the open interval $[a, b)$. To calculate the mean thickness of a book μ_w and the standard deviation of the thickness of a book σ_w, the following formulas will be used:

$$\mu_w = \frac{a+b}{2} \qquad \sigma_w = \sqrt{\frac{(b-a)^2}{12}}$$

We will calculate these in the program and gather the results.

Thickness of a stack of books

The thickness of a stack of books will be a random variable S_n, where n is the number of books in the stack. The program will create the random variable S_n by generating n random variables W and getting the sum of them. The program will create 10,000 random variables S_n and generate a bar graph for the exponential distribution, which will show the values from 0 to 4. It will also generate the PDF, which will be represented by a red bar on the graph. The PDF is given by:

$$f(x) = \begin{cases} \frac{1}{\sigma_s\sqrt{2\pi}} \exp\{-\frac{(x-\mu_s)^2}{2{\sigma_s}^2}\} & t \leq 0 \\ 0 & otherwise \end{cases}$$

After this, we want to see the mean and standard deviation for the thickness of a stack of n books. To calculate the mean thickness of a stack of n book μ_{S_n} and the standard deviation σ_{S_n}, the following formulas will be used:

$$\mu_{S_n} = \mathrm{n}\mu_w \qquad \sigma_{S_n} = \sqrt{n}\sigma_w$$

We will then look at the results for a stack of books with the values n=1,5, and 15.

Results and Conclusion

Thickness of a single book

The values for the uniform distribution to produce these results are as follows:

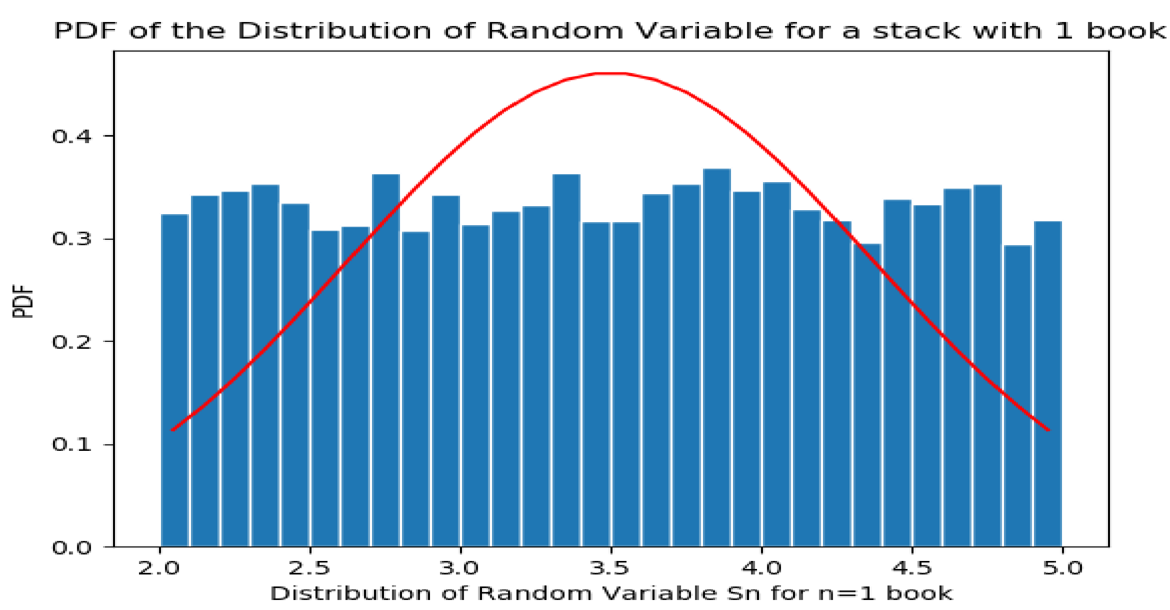
a= 2.0 b=5.0

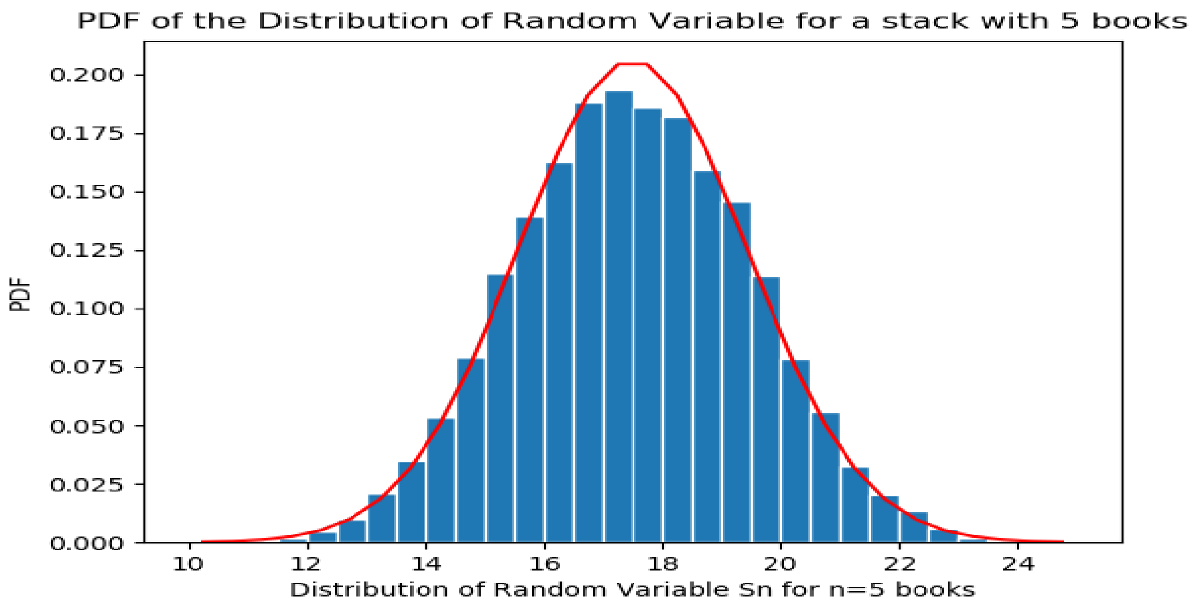
Here are the results for the uniform distribution:

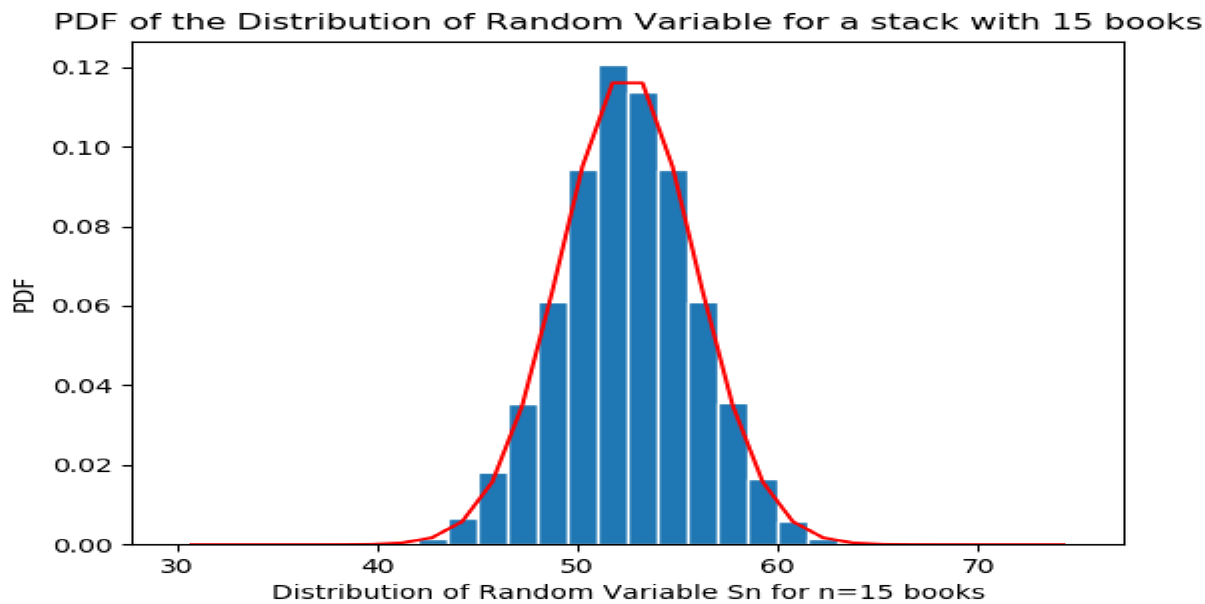
Mean thickness of a single book (cm)	Standard Deviation of thickness(cm)
$\mu_w = 3.5$	$\sigma_w = 0.8660254037844386$

Thickness of a stack of books

Here are the bar graphs with the probability distribution function for n=1 book ,5 books, and 15 books:







Here are the results for the mean and standard deviation for n=1,5, and 15:

Number of books n	Mean thickness of a stack of n books (cm)	Standard Deviation of the thickness of a stack of books (cm)
$n = 1$	μ_{S_n} =3.5	σ_{S_n} =0.8660254037844386
$n = 5$	μ_{S_n} =17.5	σ_{S_n} =1.9364916731037085
$n = 15$	μ_{S_n} =52.5	σ_{S_n} =3.3541019662496843

From the graphs, we see that with a stack of one book, the random variable appears to be uniformly distributed. But as the number of books increases, the random variable appears to be more normally distributed.

Appendix

File: centralLimitTheorem.py

```
# PROBLEM 2. The parameter values are:   a=2.0   cm;  b=5.0 cm
import numpy as np
import matplotlib.pyplot as plt

def gaussian(mu,sig,z):
  f=np.exp(-(z-mu)**2/(2*sig**2))/(sig*np.sqrt(2*np.pi))
  return f

def centralLimitTheorem():

 #Setting values f
 N=10000; nbooks=15; a=2.0; b=5.0;
 mu_w=(a+b)/2 ; sig_w=np.sqrt((b-a)**2/12)

 # Generate the values of the RV X
 X=np.zeros((N,1))
 for k in range(0,N):
  x=np.random.uniform(a,b,nbooks)
  w=np.sum(x)
  X[k]=w

# Create bins and histogram
 nbins=30; # Number of bins
 edgecolor='w'; # Color separating bars in the bargraph
 bins=[float(x) for x in np.linspace(nbooks*a, nbooks*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] # Width of bars in the bargraph
 plt.close('all')

# PLOT THE BAR GRAPH
 fig1=plt.figure(1)
 plt.bar(b1,h1, width=barwidth, edgecolor=edgecolor)
```

```
#PLOT THE PDF
 f=gaussian(mu_w*nbooks,sig_w*np.sqrt(nbooks),b1)
 plt.plot(b1,f,'r')
 plt.title("PDF of the Distribution of Random Variable for a stack with 15 books")
 plt.xlabel("Distribution of Random Variable Sn for n=15 books")
 plt.ylabel("PDF")

 #Saving the file
 filename=input("Enter a name and extension (.pdf) to save the file as :")
 plt.savefig(filename)
 mu_sn=mu_w*nbooks
 print(mu_sn)
 sig_sn=sig_w*np.sqrt(nbooks)
 print(sig_sn)

centralLimitTheorem()
```

Problem 3

Introduction

This problem involves a battery-operated critical medical monitor whose lifetime is represented by random variable T. The random variable T has an exponential-distributed lifetime and lasts and average of n days. The battery is purchased in a carton of n batteries, and the random variable C will represent the sum of the life of each battery in the carton. We want to find the mean and standard deviation of C, as well as create a plot of the probability distribution function and the cumulative distribution function in order to answer questions about the probabilities of how long the carton will last.

Methodology

For this problem, I will be using Python in the PyCharm IDE. The tools used will include methods from the `numpy`, `math`, and `mathplotlib.pyplot` libraries.

The carton of n batteries is represented by the random variable C .The program will create the random variable C by generating n random variables T and getting the sum of them. The program will create 10,000 random variables C and generate a bar graph for the exponential distribution, which will show the values from 0 to 4. It will also generate the PDF, which will be represented by a red bar on the graph. The PDF is given by:

$$f(x) = \begin{cases} \frac{1}{\sigma_C\sqrt{2\pi}} \exp\{-\frac{(x-\mu_C)^2}{2{\sigma_C}^2}\} & t \leq 0 \\ 0 \quad otherwise \end{cases}$$

After this, we want to see the mean and standard deviation for the lifetime of a carton of batteries μ_C and the standard deviation σ_C, the following formulas will be used:

$$\mu_C = \mathrm{n}\mu_T \qquad \sigma_C = \sqrt{n}\sigma_T$$

We want to compare the experimental PDF with the theoretical, so we will plot both the results using the formulas and the results from using `numpy.mean()` and `numpy.std()` methods. The experimental PDF will be plotted as a red line and the theoretical PDF will be plotted as a green line.

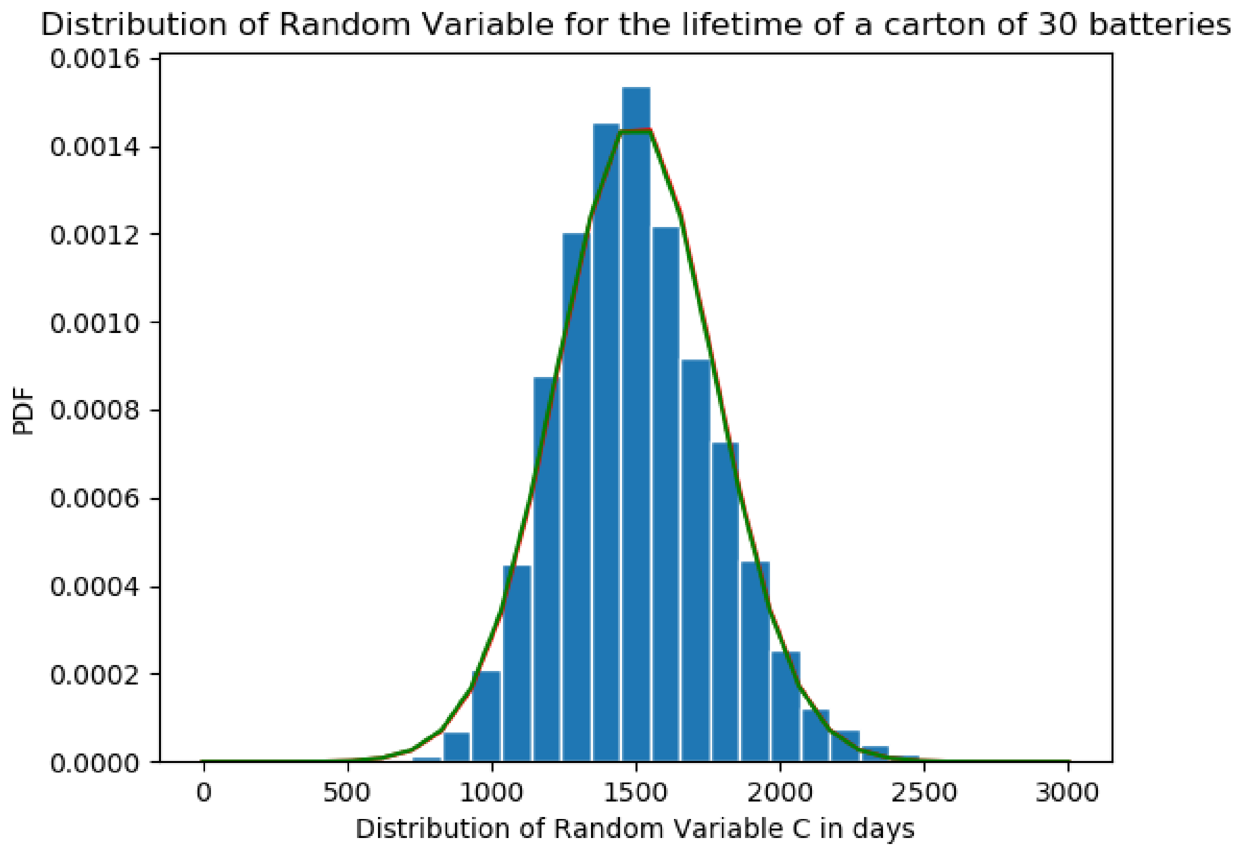
Lastly, we will create a cumulative distribution function, which will allow us to find the probabilities for the lifetime of a carton of batteries and answer questions regarding those probabilities.

Results and Conclusion

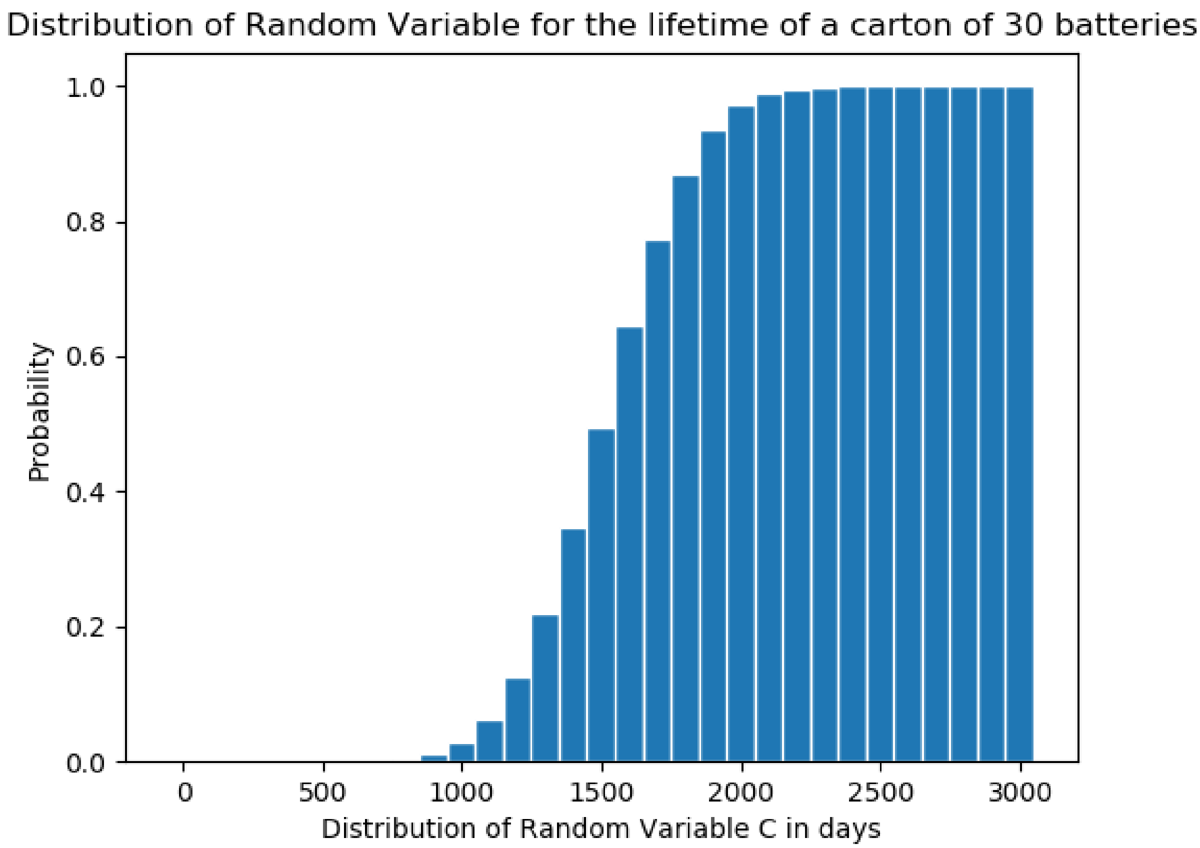
The values for the distribution to produce these results are as follows:

$$\beta = 50\ days \quad n = 30\ batteries$$

Here is the bar graph with the experimental and theoretical PDF:



Here is the CDF using the experimental PDF:



From these results using the above values, we see that the mean lifetime of a carton of batteries is 1500 days, or about 4 years. Using these results, we can now answer the following questions:

Question	**Answer**
1. Probability that the carton will last longer that Y1=3 years	0.92
2. Probability that the carton will last between Y2=2 and Y3=4 years	0.51

Appendix

File: batteryRV.py

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

 # PROBLEM 3. The parameter values are: beta =50 days; n=30 batteries; Y1=3 yrs; Y2=2 yrs; Y3=4 yrs;

def batteryRV():

```
# Setting Values
beta = 50; nBatteries = 30; N = 10000
mu_c = nBatteries * beta;
sig_c = np.sqrt(nBatteries) * beta

#Generating C
C=np.zeros((N,1))
for i in range(0,N):
  t = np.random.exponential(beta,nBatteries);
  c=np.sum(t)
  C[i] = c

# Create bins and histogram
edgecolor = 'w';  # Color separating bars in the bargraph
bins = [float(x) for x in np.linspace(0,3000,30)]
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]  # Width of bars in the bargraph
plt.close('all')


# PLOT THE BAR GRAPH
plt.bar(b1, h1,width=barwidth, edgecolor=edgecolor)

# PLOT THE PDF
#Experimental
mu_e=np.mean(C)
sig_e=np.std(C)
x1=np.linspace(0, 3000, 30)
f = gaussian(mu_e, sig_e, x1)
plt.plot(x1, f, 'r')

# Theoretical
g = gaussian(mu_c, sig_c, x1)
plt.plot(x1, g, 'g')
plt.title("Distribution of Random Variable for the lifetime of a carton of 30 batteries")
plt.xlabel("Distribution of Random Variable C in days")
plt.ylabel("PDF")

# Saving the file
filename = input("Enter a name and extension (.pdf) to save the file as :")
plt.savefig(filename)
```

```
    # mu_sn = mu_w * nbooks
    print(mu_c)
    # sig_sn = sig_w * np.sqrt(nbooks)
    print(sig_c)
    plt.close('all')

    #Plotting the CDF
    sum=np.cumsum(f)
    sum_0=np.append(0, sum)
    x1 = np.linspace(0,3000,len(sum_0))
    values=barwidth*sum_0
    plt.bar(x1,values,width=barwidth,edgecolor=edgecolor)
    plt.title("Distribution of Random Variable for the lifetime of a carton of 30 batteries")
    plt.xlabel("Distribution of Random Variable C in days")
    plt.ylabel("Probability")
    filename2 = input("Enter a name and extension (.pdf) to save the file as :")
    plt.savefig(filename2)


def gaussian(mu, sig, z):
    f = np.exp(-(z - mu) ** 2 / (2 * sig ** 2)) / (sig * np.sqrt(2 * np.pi))
    return f

batteryRV()
```