Project 4 - Central Limit Theorem

California State University Long Beach

EE 381

Probability and Statistics

Christopher Masferrer

T/TH 5:30 - 6:20

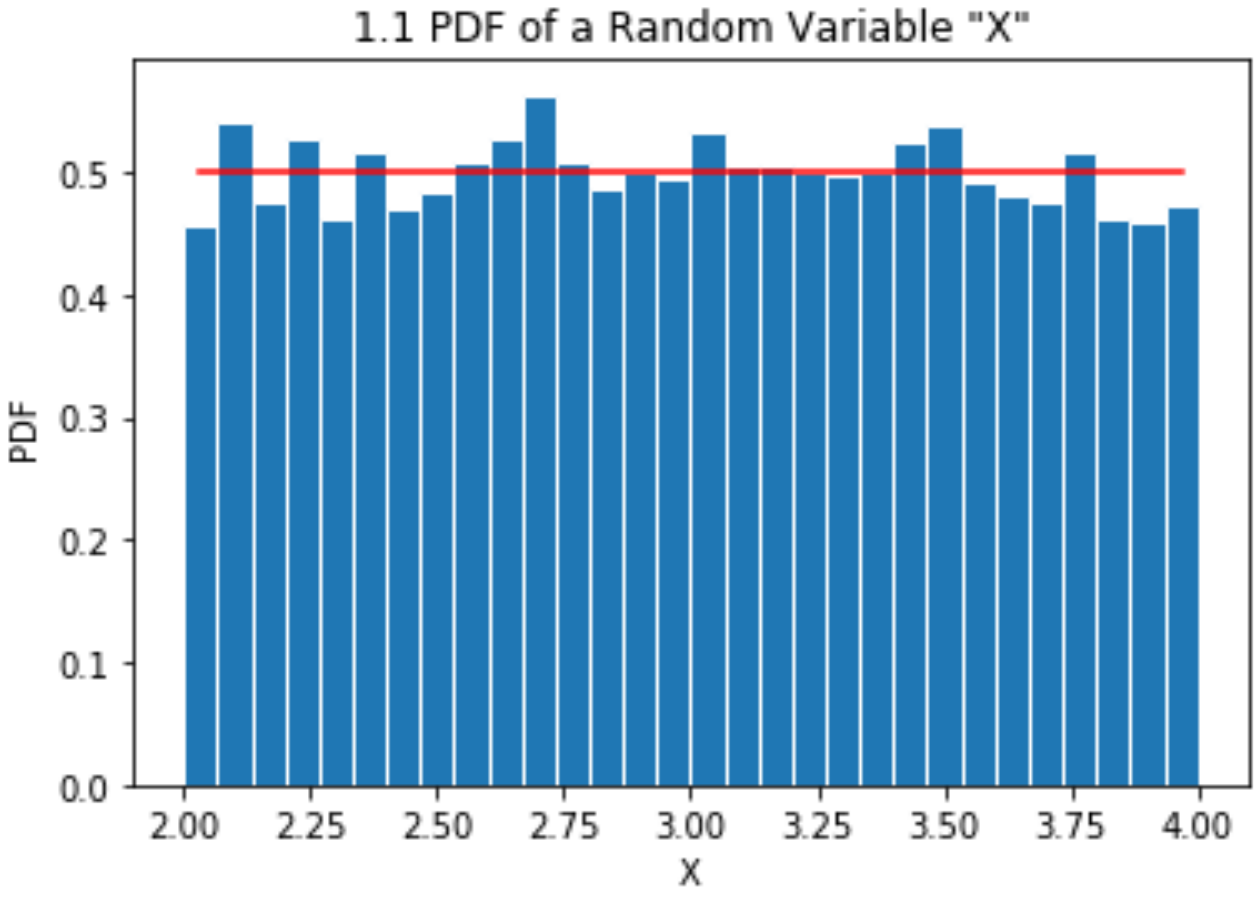
Experiment 1.1: Simulate a Uniform Random Variable

Intro: For this experiment I will be generating a set of random variables with uniform distribution using the function numpy.random.uniform(a, b, n). The results will be displayed in a chart and the mean and deviations will be calculated.

Methodology: I created a function that generates a set of random variables. Another function is used for calculating f that is the used for the plot. The mean and deviation calculations are done in the main uniformRV function.

Results:

Table 1: Statistics for a Uniform Distribution			
Expectation		**Standard Deviation**	
Theoretical Calculation	Experimental Calculation	Theoretical Calculation	Experimental Calculation
3.0	2.9956588623354246	0.3333333333333333	0.5715200132738938



Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 23 15:52:59 2019

@author: christophermasferrer
"""

#Christopher Masferrer
#EE 381
#Lab 4
import numpy as np
import matplotlib.pyplot as plt

def UnifPDF(a, b, X):
    f = (1/abs(b-a))*np.ones(np.size(X))
    return f

def uniformRV(a, b, n):
    X = np.random.uniform(a, b, n)

    #Plotting
    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 graph
    plt.close('all')

    #Plot the bar graph
    fig1 = plt.figure(1)
    plt.bar(b1,h1,width = barwidth,edgecolor=edgecolor)

    #Plot the Uniform PDF
    f = UnifPDF(a,b,b1)
    plt.plot(b1,f,'r')
    plt.title('1.1 PDF of a Random Variable "X"')
```

```
    plt.xlabel('X')
    plt.ylabel('PDF')

    #Calculate mean and deviation
    mu = np.mean(X)
    sigma = np.std(X)
    e_mean = (a+b)/2
    e_std = ((b-a)**(2))/12
    print('Problem 1a:')
    print('Experimental Mean: ', mu)
    print('Theoretical Mean: ', e_mean)
    print('Experimental Standard Deviation: ', sigma)
    print('Theoretical Standard Deviation: ', e_std)

a = 2.0
b = 4.0
n = 10000
uniformRV(a, b, n)
```

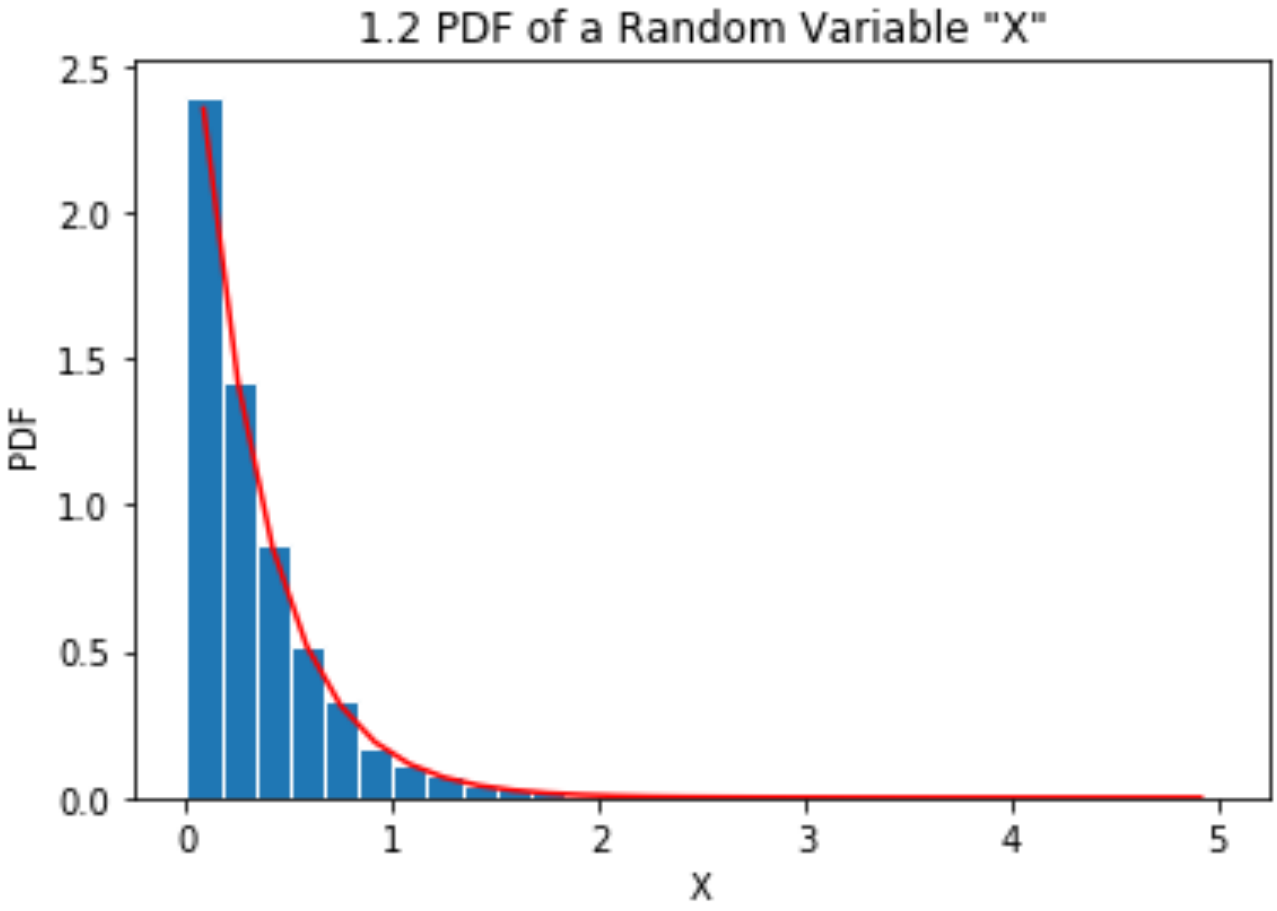
Experiment 1.2: Simulate a Exponential Random Variable

Intro: For this experiment I will be generating a set of random variables with exponential distribution using the function numpy.random.exponential(beta, n). The results will be displayed in a chart and the mean and deviations will be calculated.

Methodology: Similarly to the previous part, I created a function that generates a set of random variables. Another function is used for calculating f that is the used for the plot. The mean and deviation calculations are done in the main exponentialRV function.

Results:

Table 1: Statistics for a Uniform Distribution			
Expectation		**Standard Deviation**	
Theoretical Calculation	Experimental Calculation	Theoretical Calculation	Experimental Calculation
0.33	0.32417443964821613	0.33	0.31724078438433617



Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 24 09:59:16 2019

@author: christophermasferrer
"""

#Christopher Masferrer
#EE 381
#Lab 4
import numpy as np
import matplotlib.pyplot as plt

def ExpPDF(beta, X):
    f = ((1/beta)*np.exp(-(1/beta)*X))*np.ones(np.size(X))
    return f

def exponentialRV(beta, n):
    X = np.random.exponential(beta, n)

    #Plotting
    nbins = 30; #Number of bins
    edgecolor = 'w' #Color separating bars in the bargraph
    bins = [float(X) for X in np.linspace(0,5,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 graph
    plt.close('all')

    #Plot the bar graph
    fig1 = plt.figure(1)
    plt.bar(b1,h1,width = barwidth,edgecolor=edgecolor)

    #Plot the Exponential PDF
    f = ExpPDF(beta,b1)
    plt.plot(b1,f,'r')
    plt.title('1.2 PDF of a Random Variable "X"')
    plt.xlabel('X')
    plt.ylabel('PDF')
```

```
    #Calculate mean and deviation
    mu = np.mean(X)
    sigma = np.std(X)
    e_mean = beta
    e_std = beta
    print('Problem 1b:')
    print('Experimental Mean: ', mu)
    print('Theoretical Mean: ', e_mean)
    print('Experimental Standard Deviation: ', sigma)
    print('Theoretical Standard Deviation: ', e_std)

beta = 0.33
n = 10000
exponentialRV(beta, n)
```

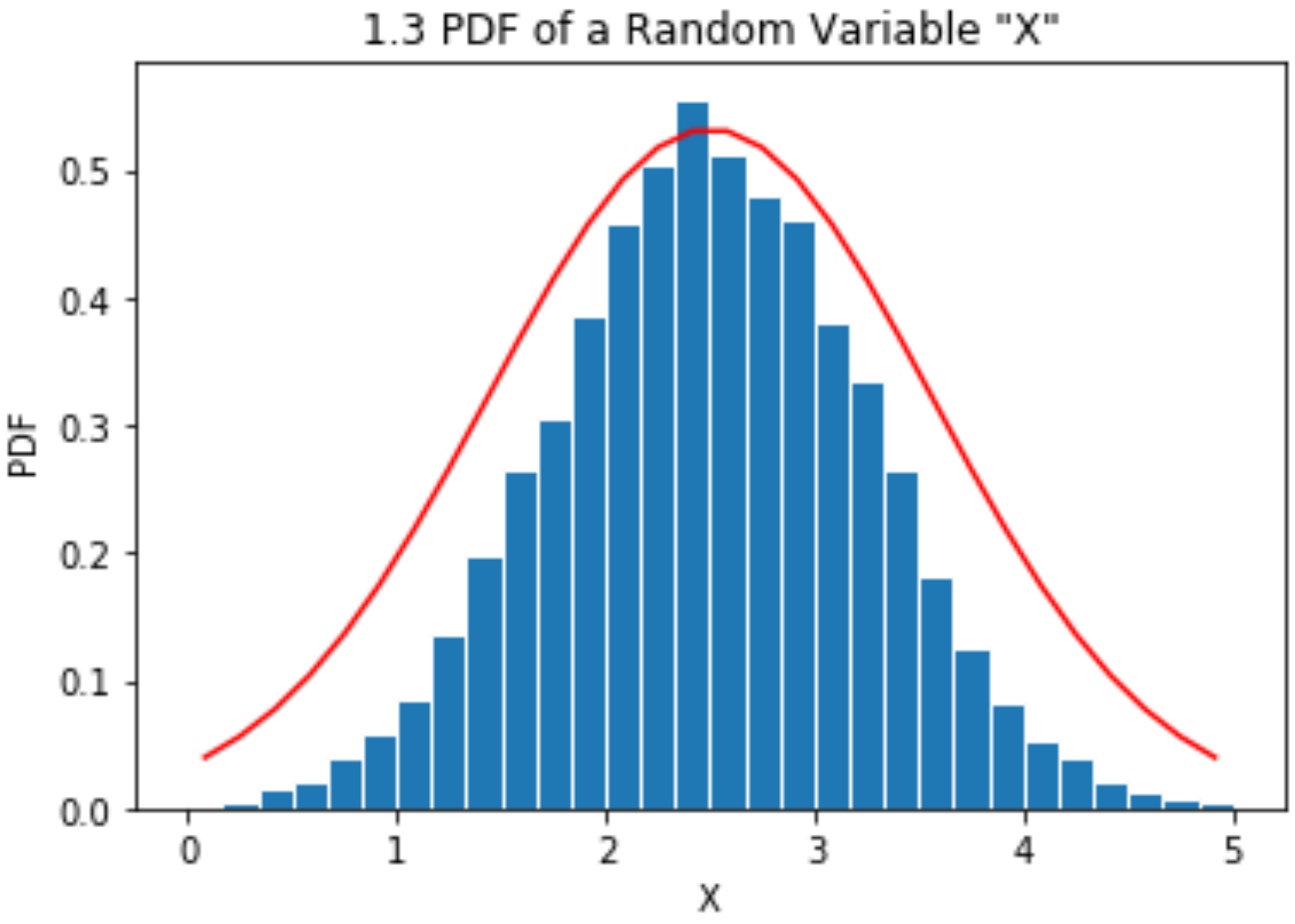
Experiment 1.3: Simulate a Gaussian Random Variable

Intro: For this experiment I will be generating a set of random variables with gaussian distribution using the function numpy.random.normal(mu, sigma, n). The results will be displayed in a chart and the mean and deviations will be calculated.

Methodology: Similarly to the previous parts, I created a function that generates a set of random variables. Another function is used for calculating f that is the used for the plot. The mean and deviation calculations are done in the main gaussianRV function.

Results:

Table 1: Statistics for a Uniform Distribution			
Expectation		**Standard Deviation**	
Theoretical Calculation	Experimental Calculation	Theoretical Calculation	Experimental Calculation
2.495793653682687	2.495793653682687	0.7529405520548513	0.7529405520548513



Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 24 10:30:26 2019

@author: christophermasferrer
"""

#Christopher Masferrer
#EE 381
#Lab 4
import numpy as np
import matplotlib.pyplot as plt

def GaussPDF(mu, sigma, X):
    f = (1/(sigma*np.sqrt(2 *np.math.pi)))*np.exp(-((X-mu)**2)/
(2*sigma)**2)*np.ones(np.size(X))
    return f

def gaussianRV(mu, sigma, n):
    X = np.random.normal(mu, sigma, n)

    #Plotting
    nbins = 30; #Number of bins
    edgecolor = 'w' #Color separating bars in the bargraph
    bins = [float(X) for X in np.linspace(0,5,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 graph
    plt.close('all')

    #Plot the bar graph
    fig1 = plt.figure(1)
    plt.bar(b1,h1,width = barwidth,edgecolor=edgecolor)

    #Plot the Exponential PDF
    f = GaussPDF(mu, sigma,b1)
    plt.plot(b1,f,'r')
    plt.title('1.3 PDF of a Random Variable "X"')
    plt.xlabel('X')
    plt.ylabel('PDF')
```

```
    #Calculate mean and deviation
    mu = np.mean(X)
    sigma = np.std(X)
    e_mean = mu
    e_std = sigma
    print('Problem 1c:')
    print('Experimental Mean: ', mu)
    print('Theoretical Mean: ', e_mean)
    print('Experimental Standard Deviation: ', sigma)
    print('Theoretical Standard Deviation: ', e_std)

mu = 2.5
sigma = 0.75
n = 10000
gaussianRV(mu, sigma, n)
```

Experiment 2: The Central Limit Theorem

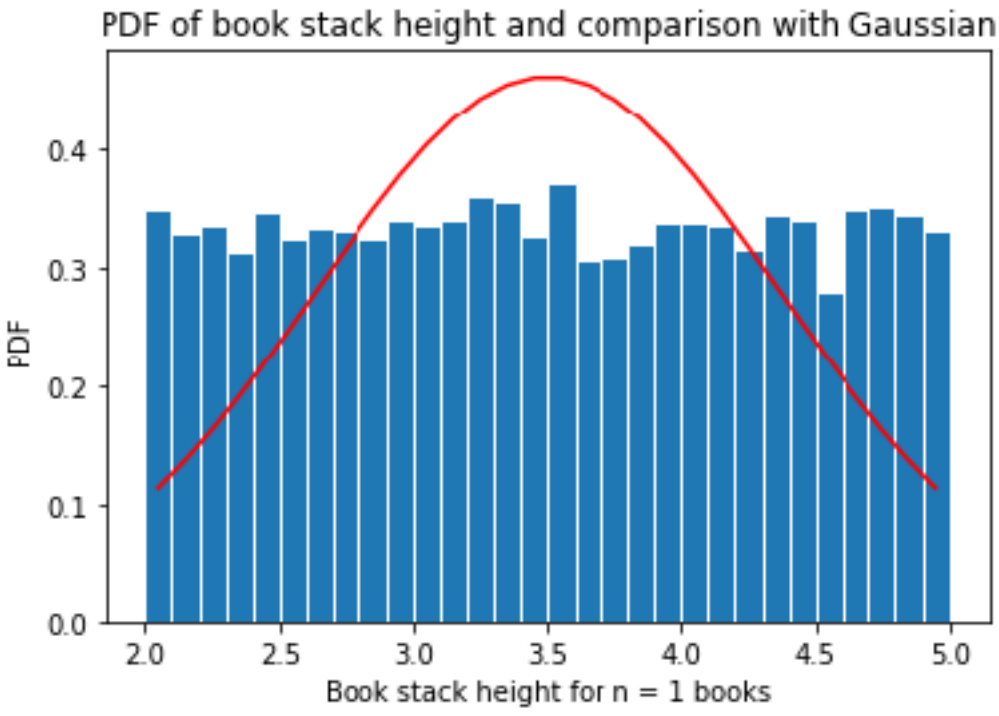
Intro: For this experiment I will be calculating the mean and standard deviation of a varied number of books. The results will be displayed in a chart and a graph of each of the stats will be displayed in their respective graph.

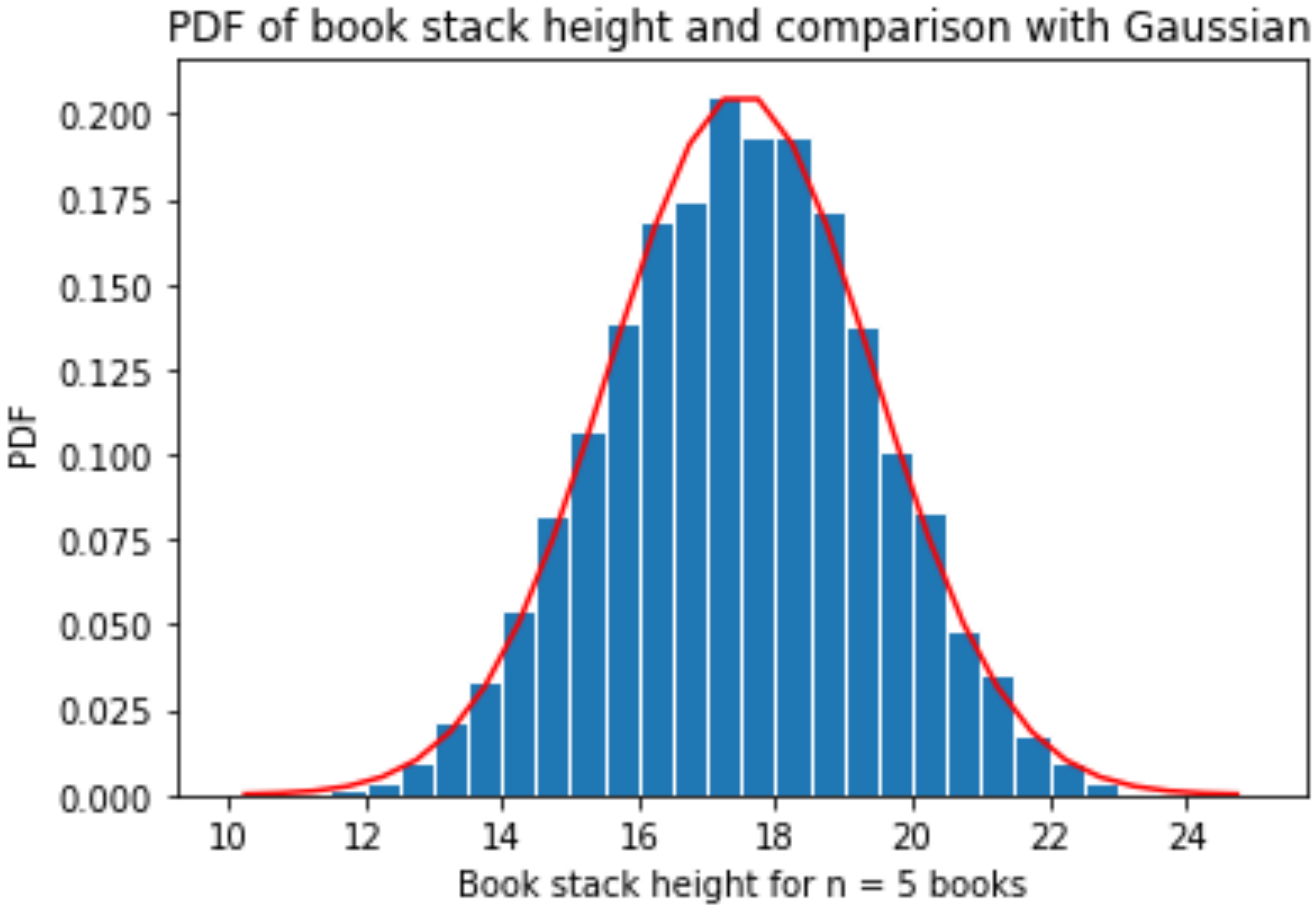
Methodology: I used a single method to calculate the mean, standard deviation, and generate the graph called generate. To calculate separate values of n, 1 and 5 and 15, generate is called 3 times with the value of nbooks altered to the appropriate value of n.

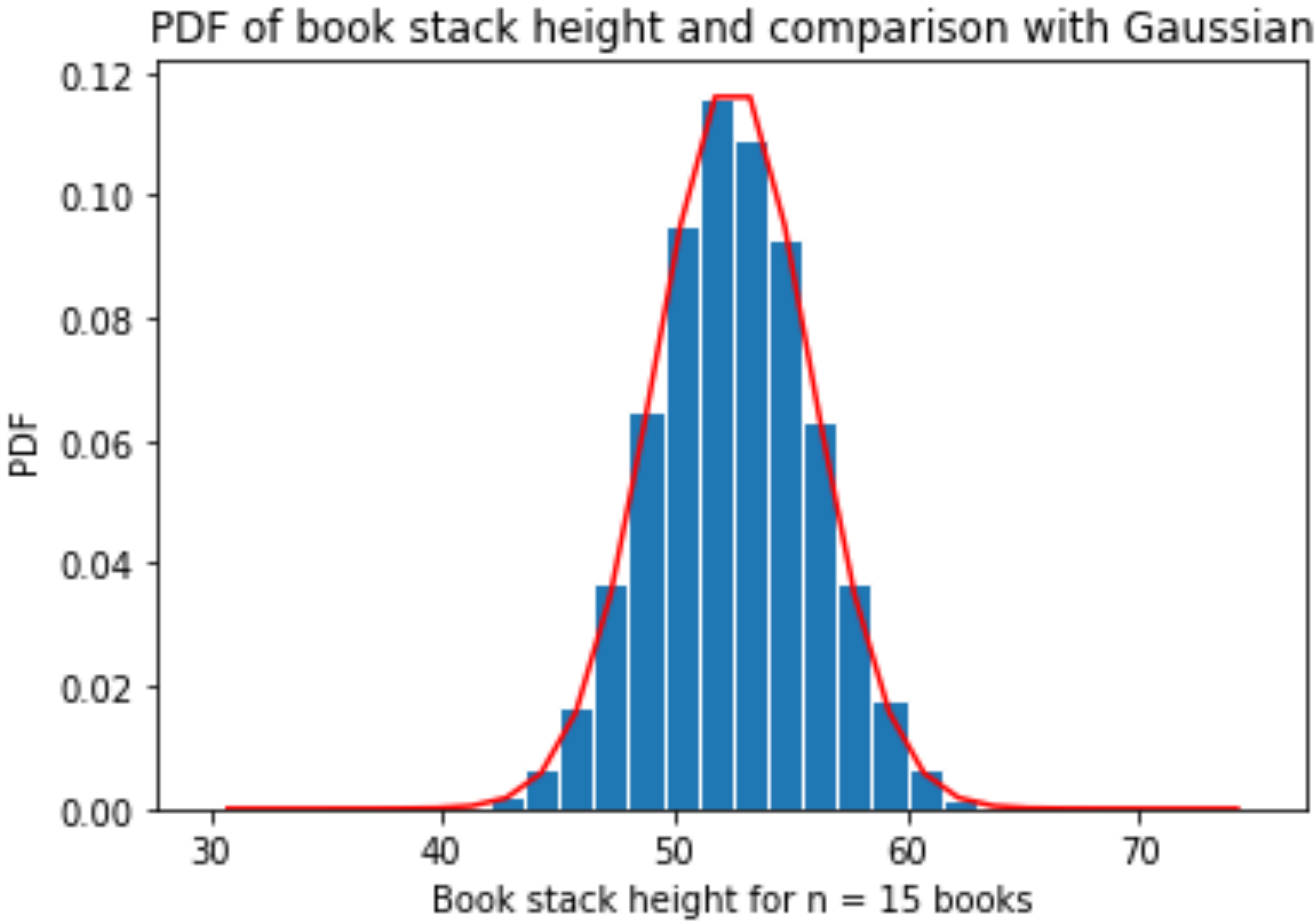
Results:

Mean thickness of a single book (cm)	Standard deviation of thickness (cm)
3.498273005109789	0.8670478025448525

Number of books n	Mean thickness of a stack of n books (cm)	Standard deviation of the thickness for n books
n = 1	3.498273005109789	0.8670478025448525
n = 5	17.48555488806188	1.9229147959880404
n = 15	52.49250998428247	3.392508017291685




PDF of book stack height and comparison with Gaussian
PDF
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
10
12
14
16
18
20
22
24
Book stack height for n = 5 books


PDF of book stack height and comparison with Gaussian
PDF
0.12
0.10
0.08
0.06
0.04
0.02
0.00
30
40
50
60
70
Book stack height for n = 15 books

Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 27 11:15:18 2019

@author: christophermasferrer
"""

#Christopher Masferrer
#EE 381
#Lab 4
import numpy as np
import matplotlib.pyplot as plt

#Variables
a = 2.0
b = 5.0
N = 10000
mu_x = (a + b) / 2
sig_x = np.sqrt((b - a) ** 2/12)
X = np.zeros((N, 1))

#Generate the values of the RV X
def generate(a, b, nbooks, N):
    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
    edgecolor = 'w'
    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]
    plt.close('all')

    #Plot the bar graph
    fig1 = plt.figure(1)
    plt.bar(b1, h1, width = barwidth, edgecolor = edgecolor)
```

```
    #Calculate Gaussian Function
    f = gaussian(mu_x*nbooks, sig_x*np.sqrt(nbooks), b1)
    plt.plot(b1, f, 'r')
    plt.title('PDF of book stack height and comparison with Gaussian')
    plt.xlabel('Book stack height for n = ' + str(nbooks) + ' books')
    plt.ylabel('PDF')
    plt.show()

    #Calculate mean
    mu_mean = np.mean(X)
    print('Mu mean: ', mu_mean)
    sig_mean = np.std(X)
    print('Sigma mean: ', sig_mean)

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

#n = 1, 5, 15
generate(a, b, 1, N)
generate(a, b, 5, N)
generate(a, b, 15, N)
```

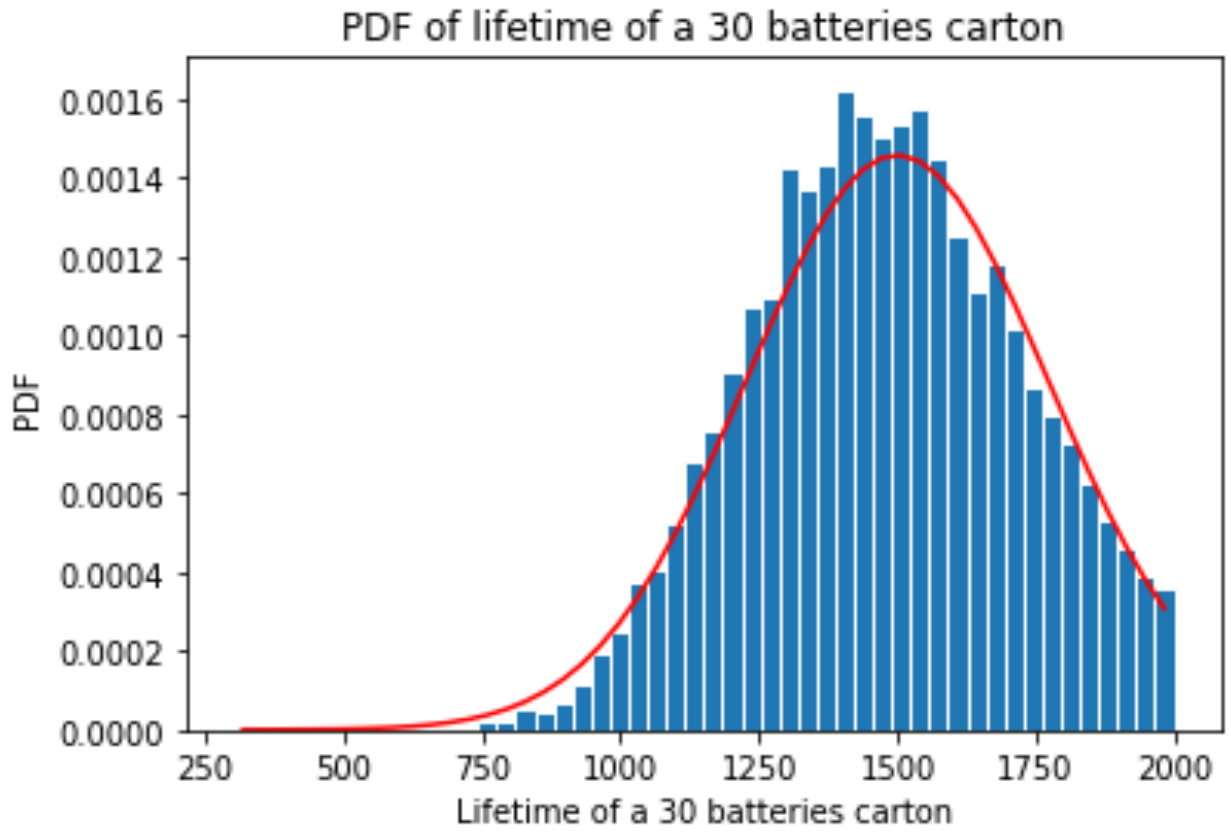
Experiment 3: Distribution of the sum of Exponential RVs

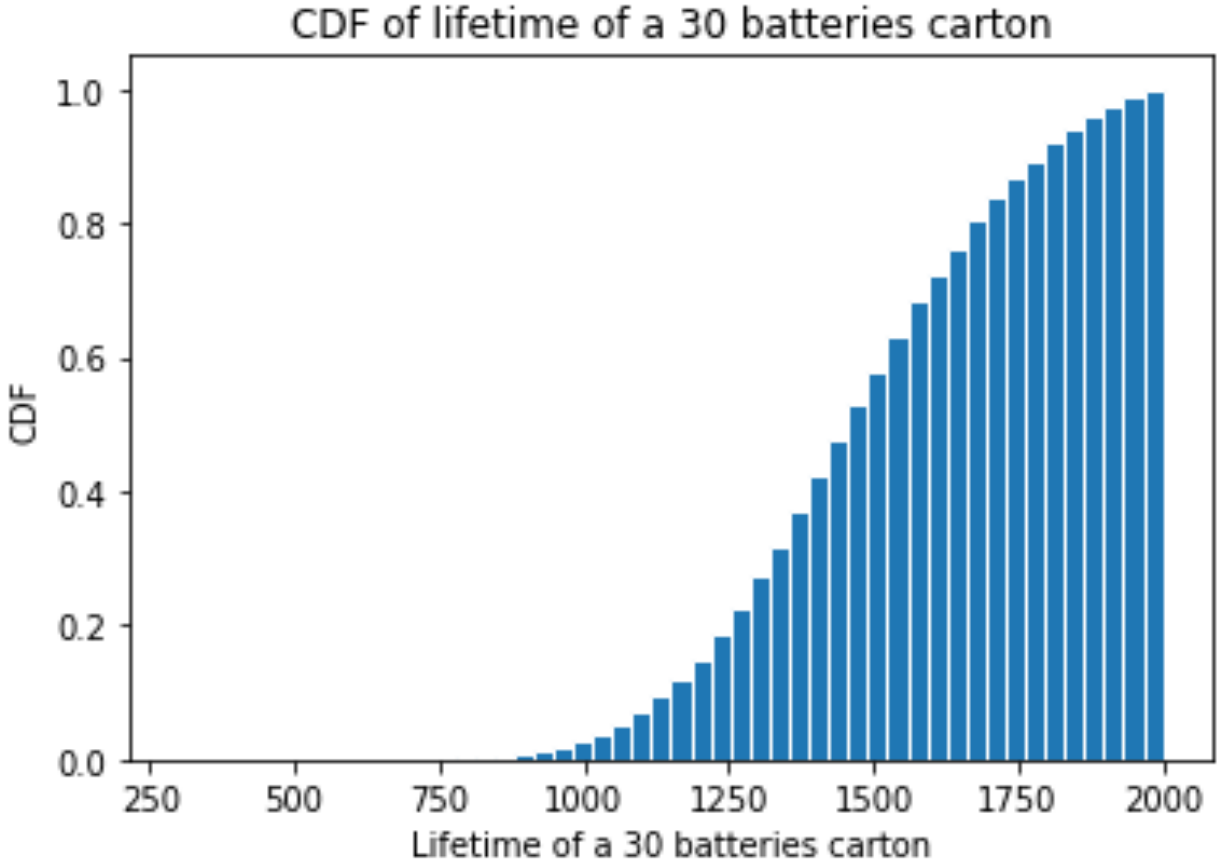
Intro: For this experiment I will be calculating the lifetime of a carton of 30 batteries. This experiment will be repeated 10,000 times and the results will be displayed in both a PDF chart and a CDF chart.

Methodology: I used a single method to calculate the mean, standard deviation, and generate the two graphs. I used the gaussian function separately to calculate f.

Results:

Question	Answer
1. Probability that the carton will last longer than Y1 year.	0.999862
2. Probability that the carton will last between Y2 and Y3 years.	0.003246




CDF of lifetime of a 30 batteries carton
CDF
1.0
0.8
0.6
0.4
0.2
0.0
250
500
750
1000
1250
1500
1750
2000
Lifetime of a 30 batteries carton

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 28 16:34:20 2019

@author: christophermasferrer
"""

#Christopher Masferrer
#EE 381
#Lab 4
import numpy as np
import matplotlib.pyplot as plt

y1 = 3
y2 = 2
y3 = 4

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

def battery(n):
    beta = 50
    a = 300
    b = 2000
    N = 10000
    X = np.zeros((N,1))
    for x in range(0, N):
        s = np.random.exponential(beta,n)
        X[x] = np.sum(s)


    mu_c = n * beta
    mu_t = beta
    sig_t = beta
    sig_c = beta * np.sqrt(n)
    nbins = 50
    edgecolor = 'w'
    bins = [float (x) for x in np.linspace(a,b,nbins+1)]
    h1, bin_edges = np.histogram(X, bins, density = True)

    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')

    fig1 = plt.figure(1)
    plt.bar(b1,h1, width=barwidth, edgecolor = edgecolor)

    plt.title('PDF of lifetime of a 30 batteries carton')
    plt.xlabel('Lifetime of a 30 batteries carton')
    plt.ylabel('PDF')
    f=gaussian(mu_c,sig_c,b1)
    plt.plot(b1,f,'r')

    fig2 = plt.figure(2)
    h2 = numpy.cumsum(h1)*barwidth
    plt.bar(b1,h2, width=barwidth, edgecolor = edgecolor)
    plt.title('CDF of lifetime of a 30 batteries carton')
    plt.xlabel('Lifetime of a 30 batteries carton')
    plt.ylabel('CDF')


battery(30)
```