CS 237 Fall 2019 Homework Eight

Due date: PDF file due Thursday November 7th @ 11:59PM in GradeScope with 6-hour grace period ¶

Late deadline: If submitted up to 24 hours late, you will receive a 10% penalty (with same 6 hours grace period)

Note: This homework has only 8 problems, and is worth 40 points (5 points each, as usual)

General Instructions

Please complete this notebook by filling in solutions where indicated. Be sure to "Run All" from the Cell menu before submitting.

There are two sections to the homework: problems 1 - 6 are analytical problems about last week's material, and the remaining problems are coding problems.

In [1]:

```
# General useful imports
import numpy as np
from numpy import arange,linspace,mean, var, std
import matplotlib.pyplot as plt
from numpy.random import random, randint, uniform, choice, binomial, geometric,
poisson,seed
from math import exp,ceil
from collections import Counter
import pandas as pd
%matplotlib inline

def C(N,K):
    if(K < N/2):
        K = N-K
    X = [1]*(K+1)
    for row in range(1,N-K+1):
        X[row] *= 2
        for col in range(row+1,K+1):
            X[col] = X[col]+X[col-1]
    return X[K]


# Numpy basic stats functions

# https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.statistics.html


X = [1,2,3]

# mean of a list
mean(X)

# population variance
var(X)

# sample variance
var(X,ddof=1)

# population standard deviation
std(X)

# sample standard deviation
std(X,ddof=1)


# Scipy statistical functions

from scipy.stats import norm,expon

# https://docs.scipy.org/doc/scipy/reference/stats.html

# given random variable X (e.g., housing prices)
# normally distributed with  mu = 60, sigma = 40

#a. Find P(X<50)
norm.cdf(x=50,loc=60,scale=40) # 0.4012936743170763

#b. Find P(X>50)
norm.sf(x=50,loc=60,scale=40) # 0.5987063256829237
```

```
#c. Find P(60<X<80)
norm.cdf(x=80,loc=60,scale=40) - norm.cdf(x=60,loc=60,scale=40)

#d. how much top most 5% expensive house cost at least? or find x where P(X>x) =
0.05
norm.isf(q=0.05,loc=60,scale=40)

#e. how much top most 5% cheapest house cost at least? or find x where P(X<x) =
 0.05
norm.ppf(q=0.05,loc=60,scale=40)

#f give the endpoints of the range for the central alpha percent of the distribu
tion
norm.interval(alpha=0.3, loc=60, scale=140)



#g. generate random variates
norm.rvs(loc=60, scale=40, size=10)

# Exponential distribution
# The library lets you set where the distribution begins (so keep loc=0)
# and the scale (=std dev); since std dev = 1/lambda, then we need to
# invert lambda to input the scale:

lam = 2                                 # example rate parameter
expon.pdf(x=3,loc=0,scale=1/lam)        # library uses scale, so must invert lambda

expon.rvs(loc=0,scale=1/lam,size=10)


# Utility functions

# Round to 4 decimal places
def round4(x):
    return round(float(x)+0.00000000001,4)
```

Analytical Problems

For the following problems, use the statistical functions given at the top of this notebook.

Also consider using the Distributions Notebook posted online to visualize these distributions (but use scipy.stats for the calculations).

You are not required to do so, but a nice touch is to print out your answer in a code block, i.e., if you were asked "What is the probability in the standard normal that a value occurs in the interval between -0.94 and 1.2 standard deviations from 0," you could answer as follows:

```
In [2]:
```

```
mu = 0
sigma = 1
lo = -0.94
hi = 1.2

answer = norm.cdf(x=hi,loc=mu,scale=sigma) - norm.cdf(x=lo,loc=mu,scale=sigma)
print("Solution: " + str(round4(answer)))
```

```
Solution: 0.7113
```

Problem One (Verifying the CLT)

In this problem you will reproduce a result I showed in class, showing how the sample mean of samples from an exponential distribution produce a normal distribution following the prediction of the CLT.

Since we did something very similar in HW 07, I reproduce some of that code; all you have to do is

(a) to generate the exponential variates (see the first code cell above!) and run the experiments and

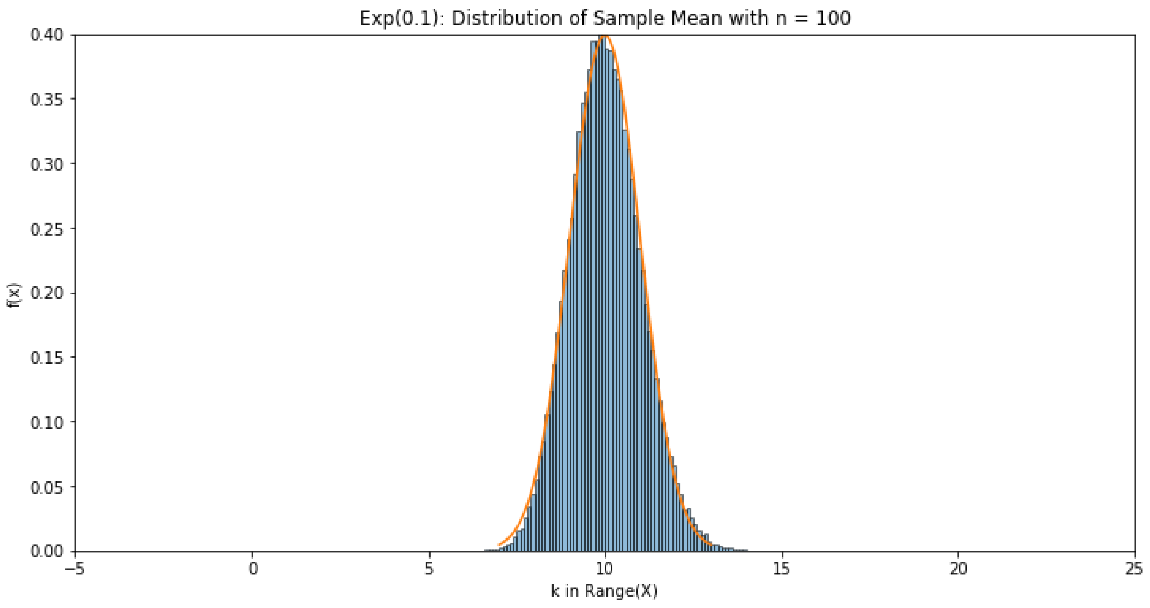
(b) think about what you are seeing and answer a simple question.

In [3]:

```
# Part (a)

# Use the numpy function expon.rvs(...) to generate random exponential variates
 with
# rate parameter lam; this function simply generates n of them and returns the m
ean of those n numbers

def sampleMeanExponential(lam,n):
    sum1 = 0
    value = expon.rvs(loc=0,scale=1/lam,size=n)
    for x in value:
         sum1 = sum1 + x
    average = sum1 / n
    return average

# Display the result  of generating num_trials values of the sample mean using
# the above function, and graph the result, adapting the code from Problem 1 and
using
# an appropriate bin_width to demonstrate the results most clearly

# Define the boundaries of bins with the specified width around the mean,
# to plus/minus at least 4 * sigma

# bin_width is in units of sigma, so bin_width = 0.1 means sigma/10

def makeBins(mu,sigma,bin_width):
    numBins = ceil(4/bin_width)
    bins = [mu+sigma*bin_width*x for x in range(-numBins,numBins+1)]
    return bins

def display_sample_mean_exponential(lam,n,num_trials,bin_widths=0.1):
    fig, ax = plt.subplots(1,1,figsize=(12,6))
    plt.title('Exp('+str(lam)+'): Distribution of Sample Mean with n = ' + str(n
))
    plt.ylabel("f(x)")
    plt.xlabel("k in Range(X)")
    ax.set_xlim(-5,25)
    ax.set_ylim(0,0.4)
    mu = 1/lam
    sigma = (1/lam) / (n**0.5)

    # use exponential to generate random samples
    X = [sampleMeanExponential(lam,n) for i in range(num_trials)]
    plt.hist(X,bins=makeBins(mu,sigma,bin_widths), normed=True,edgecolor='k',alp
ha=0.5) # bins are of width 1/10**decimals

    # Now generate the theoretical normal for sample mean with std dev sigma/sqr
t(n)

    X2 = np.linspace(mu-sigma*3,mu+sigma*3,100)
    Y = [norm.pdf(x,mu,sigma) for x in X2]
    plt.plot(X2,Y)
    plt.show()


lam = 0.1
num_trials = 10**5
```

```
n = 100
display_sample_mean_exponential(lam,n,num_trials,bin_widths=0.1)
```

```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:40: Ma
tplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be remo
ved in 3.1. Use 'density' instead.
```



Part (b)

Answer in a short sentence: At what value does it seem the sample means are normally distributed? Is 30 a reasonable number to suggest for when this becomes "normally distributed"?

In [4]:

```
print("I believe the value of 100 makes the sample means normally distributed. 3
0 does not seam like a reasonable number for n because even though the graph loo
ks like it is normally distributed, it is slightly skewed to the right, making i
t not normal.")
```

```
I believe the value of 100 makes the sample means normally distribut
ed. 30 does not seam like a reasonable number for n because even tho
ugh the graph looks like it is normally distributed, it is slightly
skewed to the right, making it not normal.
```

Problem Two (CLT)

Consider a population R_X consisting of the five numbers $\{2, 3, 6, 8, 11\}$, where the random variable X selects a number equiprobably from this population and returns it.

(a) Find the mean and the (population) standard deviation of this population. (Hint: Use the functions shown in the first code cell.)

Now consider all 25 possible samples of size 2 that can be selected with replacements from this population, i.e.,

```
(2, 2)       (2, 3)   ....   (2, 11)
(3, 2)       (3, 2)   ....   (3, 11)

     ....

(11, 2)      (11, 3)  ....    (11, 11)
```

Let us denote by `XBar2` the random variable that selects one of these pairs (i.e., a random sample of size 2) and returns the mean of these two numbers; `XBar2` has a distribution, mean, and standard deviation, the same as any random variable, and the CLT tells us something about these statistics.

(b) Calculate the precise value of `E(XBar2)`, i.e., the mean of the sample distribution of means for samples of size 2 from this population. (Note that this should demonstrate to you that this value is the same as the value found in part (a).)

(c) What is the precise value of the standard deviation of `XBar2`, i.e., the standard deviation of the sample distribution of means for samples of size 2 from this population?

(d) Show that the value you calculated in (c) is the same as that which would be predicted using the CLT.

In [5]:

```
print("Part a")
X = [2,3,6,8,11]
print("Mean is",mean(X))
print("Population standard deviation is", round4(std(X)))

print()

print("Part b")
X = [2,5/2,8/2,10/2,13/2,5/2,6/2,9/2,11/2,14/2,8/2,9/2,12/2,14/2,17/2,10/2,11/2,
14/2,16/2,19/2,13/2,14/2,17/2,19/2,11]
print("Mean is",mean(X))

print()

print("Part c")
X = [2,5/2,8/2,10/2,13/2,5/2,6/2,9/2,11/2,14/2,8/2,9/2,12/2,14/2,17/2,10/2,11/2,
14/2,16/2,19/2,13/2,14/2,17/2,19/2,11]
print("Standard Deviation is",round4(std(X)))

print()

print("Part d")
print("The population mean is 6 and the population variance is 10.8. Since the s
ample mean is 0.5 * mean * 2 = 6 and the sample population variance is square ro
ot of 10.8 / square root of 2 = 2.3238, the value calculated in part c is same a
s what is predicted using CLT")
```

```
Part a
Mean is 6.0
Population standard deviation is 3.2863

Part b
Mean is 6.0

Part c
Standard Deviation is 2.3238

Part d
The population mean is 6 and the population variance is 10.8. Since
the sample mean is 0.5 * mean * 2 = 6 and the sample population vari
ance is square root of 10.8 / square root of 2 = 2.3238, the value c
alculated in part c is same as what is predicted using CLT
```

Problem Three (CLT and Sampling)

Suppose the heights of 3400 male students at a university are normally distributed with mean 68 inches and standard deviation 3 inches.

(a) Supposing samples of size 25 are taken from this population (with replacement). What would be the expected mean and standard deviation of the resulting sample distribution of means?

(b) Supposing we wanted to get more accuracy in our sampling procedure, so that we wanted the standard deviation of the sample distribution of means to be at most 0.25 inches. What is the smallest sample size we could use to insure this?

(c) Supposing you take 80 samples of size 25, in how many samples would you expect to find the mean between 66.8 and 68.3 inches?

In [6]:

```
print("Part a")
print("Mean is 68")
print("Sample standard deviation is", 3/(25**.5))

print()

print("Part b")
print("With the standard deviation being 0.25, the equation is 0.25 = (3 / sqaur
e root of n)")
print("Therefore, n equals (3/0.25) ** 2")
print("The smallest sample size we could use to insure this is", (3/0.25)**2)

print()

print("Part c")
print("The p value is P(66.3 < X < 68.8)")
print("which is equal to P((66.3 - 68)/(3/sqrt(25)) < z < (66.8 - 68)/(3/sqrt(2
5)))")
mu = 0
sigma = 1
lo = (66.8 - 68)/(3/5)
hi = (68.3 - 68)/(3/5)
answer = norm.cdf(x=hi,loc=mu,scale=sigma) - norm.cdf(x=lo,loc=mu,scale=sigma)
print(round4(80* answer), "is the number of samples expected")
```

```
Part a
Mean is 68
Sample standard deviation is 0.6

Part b
With the standard deviation being 0.25, the equation is 0.25 = (3 /
sqaure root of n)
Therefore, n equals (3/0.25) ** 2
The smallest sample size we could use to insure this is 144.0

Part c
The p value is P(66.3 < X < 68.8)
which is equal to P((66.3 - 68)/(3/sqrt(25)) < z < (66.8 - 68)/(3/sq
rt(25)))
53.497 is the number of samples expected
```

Problem Four (CLT and Sampling)

Suppose you know that when there was a vote for the major of Cambridge, MA, 54.8% of the people voted for candidate A.

(a) Supposing samples of size 30 are taken, what would you expect to be the mean and standard deviation of the sampling distribution of proportions? Give your result in terms of percentages.

(b) Supposing we wanted to get more accuracy in our sampling procedure, so that we wanted the standard deviation of the sample distribution of proportions to be at most 5%. What is the smallest sample size we could use to insure this?

(c) Supposing you take 100 samples of size 30, in how many samples would you expect to find the proportion accurate to 1%, i.e., between 53.8% and 55.8% ? (Don't worry about using the continuity correction here.)

Hint: This is the same as the last problem, but where the population represents the outcomes of a Bernoulli experiment with p = 0.548. In such cases, we do not need to be given the population standard deviation, because it is determined by a formula involving p; (get out your "cheatsheet" from the midterm!). This is a special case called "sampling with proportions."

In [7]:

```
print("Part a")
print("Mean is 54.8%")
print("Population standard deviation is", round4(((0.548)*(1-0.548)/30)**0.5 * 1
00), "%")

print()

print("Part b")
print("The standard deviation has to be less than 5%")
print("sqrt(p*(1-p)/n) < 0.05")
print("p*(1-p)/n < 0.0025")
print("n > (0.548)*(1-0.548)/0.0025")
print("n > 99.0784")
print("The smallest sample size we could use to insure this is", 99+1,"since we
 need to always round up the value")

print()

print("Part c")
print("The margin of error is +/- 0.01")
print("Margin of error = Standard deviation * Z Value")
print("+/- 0.01 = 0.090865 * Z")
print("Z = (+/- 0.01)/0.090865")
print("Z = +/- 0.1101")
print("P(Z < 0.11) = 0.5438")
print("P(Z < -0.11) = 0.4562")
print("45.6% confidence interval is between 53.8% and 55.8%")
print("Number of samples needed is", 45.6)
```

```
Part a
Mean is 54.8%
Population standard deviation is 9.0865 %

Part b
The standard deviation has to be less than 5%
sqrt(p*(1-p)/n) < 0.05
p*(1-p)/n < 0.0025
n > (0.548)*(1-0.548)/0.0025
n > 99.0784
The smallest sample size we could use to insure this is 100 since we
need to always round up the value

Part c
The margin of error is +/- 0.01
Margin of error = Standard deviation * Z Value
+/- 0.01 = 0.090865 * Z
Z = (+/- 0.01)/0.090865
Z = +/- 0.1101
P(Z < 0.11) = 0.5438
P(Z < -0.11) = 0.4562
45.6% confidence interval is between 53.8% and 55.8%
Number of samples needed is 45.6
```

Problem Five (Sampling Theory)

This problem considers three different ways of answering a question about samples from an infinite population. Suppose you flip a fair coin 120 times. What is the probability that 5/8's or more of the flips will be heads?

(a) First solve this problem precisely using the binomial.

(b) Next, solve the problem by using the normal approximation to the binomial (using the continuity correction).

(c) Finally, solve it as a problem in sampling: note that this is a question about proportions (so use the mean and standard deviation derived from the Bernoulli) and the sample size is 120. The sample distribution of means now gives us an estimate of the proportion of the population which is heads (which we know to be 0.5), the expected standard deviation, and we want to know the probability that the proportion (i.e., the mean) of the sample is 5/8's or more. Show all work.

Note that you should properly use the continuity correction by subtracting 1/2n.

In [8]:

```
print("Part a")
print("Using Binomial")
print("120*5/8 = 75")
print("P(X >= 75) = C(120,75)*((1/2)**120) + C(120,76)*((1/2)**120) + ... + C(12
0,119)*((1/2)**120) + C(120,120)*((1/2)**120)")
print(0.003923)

print()

print("Part b")
expected = 120*0.5
sqrt = (120*.5*.5)**.5
mu = expected
sigma = sqrt
lo = 74.5
hi = 120
answer = norm.cdf(x=hi,loc=mu,scale=sigma) - norm.cdf(x=lo,loc=mu,scale=sigma)
print("Solution: " + str(round4(answer)))

print()

print("Part c")
proportion = 5/8
pheads = .5
sd = (0.5*0.5/120)**0.5
z_value = (proportion-pheads)/sd
mu = 0
sigma = 1
lo = z_value
hi = 9999999
answer = norm.cdf(x=hi,loc=mu,scale=sigma) - norm.cdf(x=lo,loc=mu,scale=sigma)
print("Solution: " + str(round4(answer)))
```

```
Part a
Using Binomial
120*5/8 = 75
P(X >= 75) = C(120,75)*((1/2)**120) + C(120,76)*((1/2)**120) + ... +
C(120,119)*((1/2)**120) + C(120,120)*((1/2)**120)
0.003923

Part b
Solution: 0.0041

Part c
Solution: 0.0031
```

Problem Six (Confidence Intervals)

Suppose an experiment is conducted where 100 students at BU are measured and their average height is found to be 67.45 inches, and the (sample) standard deviation to be 2.93 inches. Since 100 is a large sample, we use the sample standard deviation as an estimate of the population standard deviation. We may assume that heights are normally distributed.

(a) Suppose that you want to report the 95.45...% confidence interval (i.e., exactly 2 standard deviations). Give the results of this experiment.

(b) Repeat (a) but for the 99.73... % (exactly 3 standard deviations) confidence interval.

(c) Now suppose you want to report the precisely 95.0% confidence interval (which will be slightly less than 2 standard deviations -- find out the exact figure) Repeat (a) using this confidence interval.

(d) Repeat (c) but for the precisely 99.0% confidence interval.

In [9]:

```
print("Part a")
print("67.45 +/- 2*(2.93)/sqrt(100)")
value1 = 67.45 + 2*2.93/10
value2 = 67.45 - 2*2.93/10
print("Solution: (" + str(round4(value2)) + "," + str(round4(value1)) + ")")

print()

print("Part b")
print("67.45 +/- 3*(2.93)/sqrt(100)")
value1 = 67.45 + 3*2.93/10
value2 = 67.45 - 3*2.93/10
print("Solution: (" + str(round4(value2)) + "," + str(round4(value1)) + ")")

print()

print("Part c")
x = norm.ppf(q=0.05/2,loc=0,scale=1)
print("67.45 +/- x*(2.93)/sqrt(100)")
value1 = 67.45 + x*2.93/10
value2 = 67.45 - x*2.93/10
print("Solution: (" + str(round4(value1)) + "," + str(round4(value2)) + ")")

print()

print("Part d")
x = norm.ppf(q=0.01/2,loc=0,scale=1)
print("67.45 +/- x*(2.93)/sqrt(100)")
value1 = 67.45 + x*2.93/10
value2 = 67.45 - x*2.93/10
print("Solution: (" + str(round4(value1)) + "," + str(round4(value2)) + ")")
```

```
Part a
67.45 +/- 2*(2.93)/sqrt(100)
Solution: (66.864,68.036)

Part b
67.45 +/- 3*(2.93)/sqrt(100)
Solution: (66.571,68.329)

Part c
67.45 +/- x*(2.93)/sqrt(100)
Solution: (66.8757,68.0243)

Part d
67.45 +/- x*(2.93)/sqrt(100)
Solution: (66.6953,68.2047)
```

Lab/Implementation Problems

You will need to have Pandas installed in order to do these problems. [Optional: Learn more about Pandas by doing the Pandas Lab. (http://www.cs.bu.edu/fac/snyder/cs237/Homeworks,%20Labs,%20and%20Code/PandasLab.ipynb)]

Problem Seven

The GPAs for 4897 individuals from an institution of higher education in the northeastern United States will be read into a list using the following Pandas code:

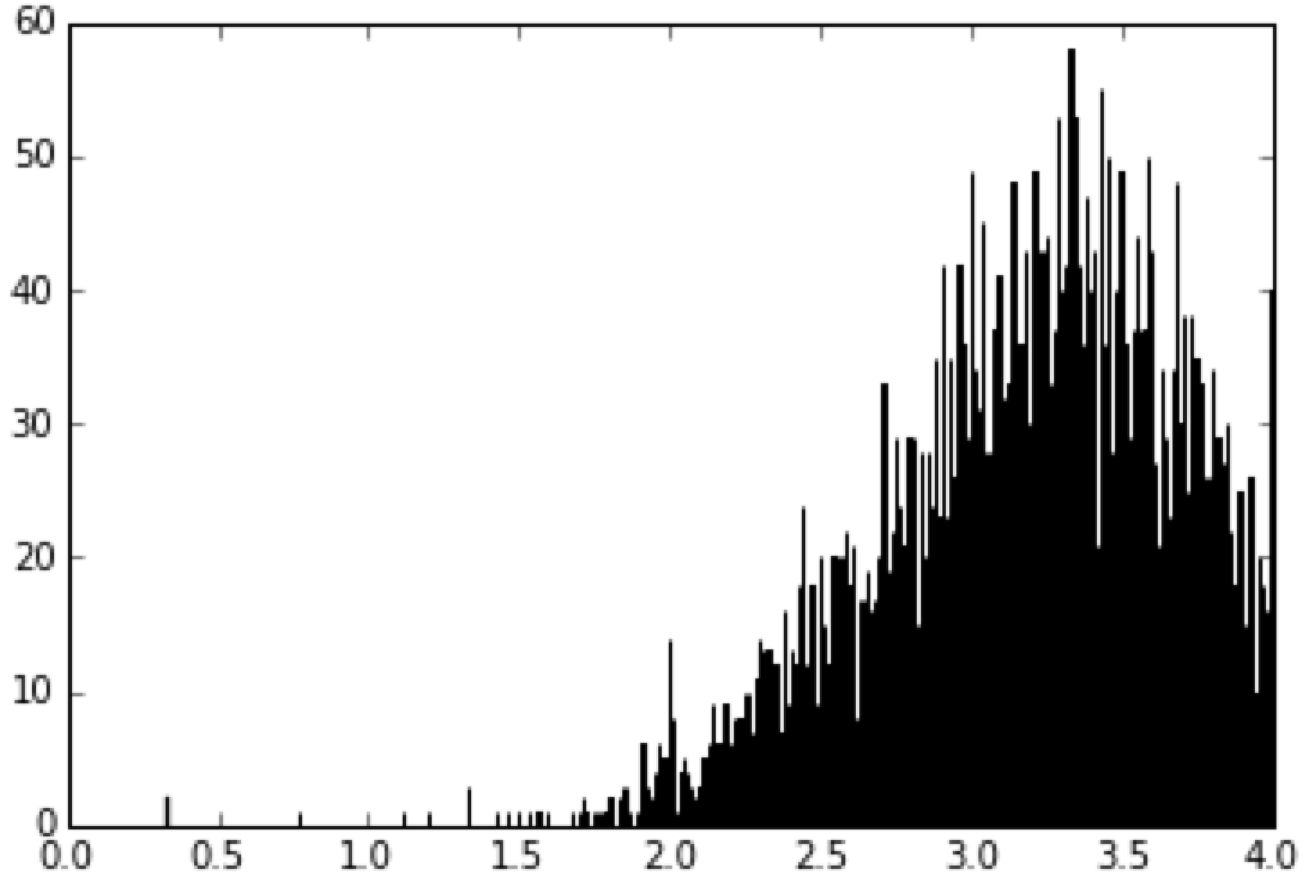
In [10]:

```
studs = pd.read_csv('http://www.cs.bu.edu/fac/snyder/cs237/Homeworks,%20Labs,%20
and%20Code/StudentData.csv')
gpaList = studs['GPA'].tolist()
studs[0:10]
```

Out[10]:

	Gender	ClassYear	GPA	CreditsEarned	TransferCredits	APCredits
0	0	2	4.0	33.0	0.0	12
1	1	3	4.0	16.0	52.0	16
2	0	2	4.0	32.0	0.0	0
3	1	3	4.0	68.0	14.0	0
4	1	2	4.0	36.0	0.0	56
5	1	4	4.0	100.0	0.0	32
6	1	4	4.0	36.0	0.0	60
7	0	3	4.0	32.0	0.0	28
8	1	2	4.0	36.0	0.0	44
9	1	3	4.0	72.0	0.0	36

For your information, the histogram shows that it is approximately normal, except for the limit at 4.0:



We will use this data (just the list of GPAs) for exploring the various ideas presented in lecture about sampling theory and confidence intervals.

You should use the scipy statistics functions (mean, var, std) shown in the code cell at the top of this document. Note carefully how population and sample standard deviations are calculated there.

(a) Calculate the mean and (population) standard deviation of this population and print them out to 4 decimal places. These are your benchmark values for the actual parameters of the population.

(b) Write a function `getSample(n)` to generate ONE random sample of n samples from `gpaList` , and return the sample mean and the sample standard deviation. Using the techniques developed in lecture, report on your estimation of the population mean using a confidence interval for 95% (NOT 95.14....%) confidence. Simply print out the confidence interval result for one random sample of size n = 30, in the same format dfas you did for Problem 6. (Do NOT use the data from (a) for this, remember, you are taking the sample because you supposedly can't get the whole population.)

Hint: Use the `numpy.random` function `choice(L,n)` , which takes a list L and returns n random values, chosen equiprobably and with replacement (you can modify this behavior with parameters), docs are here (https://docs.scipy.org/doc/numpy-1.14.0/reference/generated/numpy.random.choice.html).

(c) Write a function `getInterval(CL,n)` which generates a random sample of size n and returns the bounds for the confidence interval for the mean with confidence level (percentage) CL. (Hint: You can use the function `norm.interval(...)` to calculate the multiplier k in the confidence interval -- see the lecture slides!)

If you use this function to generate a confidence interval, then it has successfully predicted the mean if the actual population mean found in (a) is inside this interval. If everything works properly, then for a large number of trials, this function should succeed about CL % of the time!

Now run an experiment, with at least 10^5 trials/samples, and using CL = 95%, to determine what percentage of the trials successfully predicted the location of the actual mean.

This may vary but you should usually get a probability close to 95%. Running more trials will improve the accuracy.....

(d) Now repeat c, but using a 90% confidence interval. Again, your experiment should confirm this figure.

In [ ]:

In [ ]:

In [11]:

```
print("Part a")
print("Mean is " + str(round4(mean(gpaList))))
print("Variance is " + str(round4(var(gpaList))))
print("Standard Deviation is " + str(round4(std(gpaList))))

print()

print("Part b")
def getSample(n):
    lst = []
    for x in range(n):
        lst.append(gpaList[randint(0,len(gpaList)-1)])
    return [round4(mean(lst)), round4(std(lst))]

n = 30
lst = getSample(n)
x = norm.ppf(q=0.05/2,loc=0,scale=1)
value1 = lst[0] + x*lst[1]/(n**0.5)
value2 = lst[0] - x*lst[1]/(n**0.5)
print("Solution: (" + str(round4(value1)) + "," + str(round4(value2)) + ")")


print()

print("Part c")
mean = round4(mean(gpaList))
std = round4(std(gpaList))
def getInterval(CL,n):
    lst = []
    for x in range(n):
        lst.append(gpaList[randint(0,len(gpaList)-1)])
    sum1 = 0
    for x in lst:
        sum1 = sum1 + x
    average = sum1/n
    strr = 0
    for x in lst:
        strr = strr + (x-average)**2
    strr = strr/n
    strr = strr ** 0.5
    x = norm.interval(CL,average,strr)
    return x


num_trials = 10**1
proportion = 0

CL = 0.95
for x in range(num_trials):
    string = getInterval(CL,n)
    val1 = string[0]
    val2 = string[1]
    if val1 <= mean <= val2:
        proportion = proportion + 1
proportion = proportion / num_trials
print(proportion)
print()

print("Part d")

proportion = 0
CL = 0.9
for x in range(num_trials):
    string = getInterval(CL,n)
    val1 = string[0]
    val2 = string[1]
    if val1 <= mean <= val2:
        proportion = proportion + 1
proportion = proportion / num_trials
print(proportion)
```

```
Part a
Mean is 3.1729
Variance is 0.2434
Standard Deviation is 0.4934

Part b
Solution: (2.9939,3.3747)

Part c
1.0

Part d
1.0
```

Problem Eight

In this problem we will test "Bessel's Correction" for the sample variance (using n-1 rather then n in the denominator), using our GPA data from problem 7 above. This concept will be discussed on Monday 11/4 in lecture.

Write a functions `getSampVar(n)` which will generate a random sample of size n from the gpa data in Problem 7, and return the sample variance (which uses the value n-1 in the denominator) of the sample as calculated by the function `var(...)` given at the top of this notebook; then write a function `getPopVar(n)` which does the same thing, but returns the population variance of the randomly-chosen sample.

Then write a function `testPopVar(M,n)` which will generate M sample variances using `getPopVar(n)` , and return of the mean of these M values. Run this for M = 1000 and n = 2, 3, ..., 50 and store it in a list PV.

Do the same as the previous paragraph, but with a function `testSampVar(M,n)` which uses the sample variance and store it in a list SV.

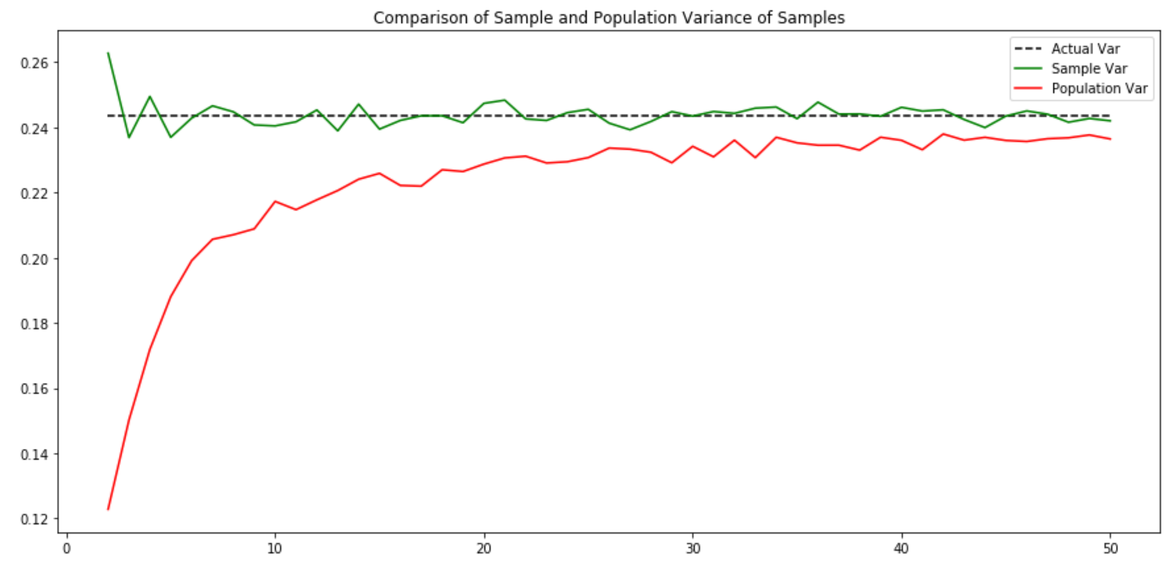
Graph these results with the actual variance calculated directly from the entire data set (i.e., the x axis is the values 2, 3, ..., 50, and the three curves are the (constant) value of the true population variance (a list [v, v, ..., v] where v is the actual population variance calculated in problem 7 (a)), and the y values in PV and SV).

The result of this problem is simply the graph comparing these two methods for estimating the variance for samples of size 2 .. 50.

Notice: How do these two estimators compare as n approaches 50?

In [12]:

```
def getSampVar(n):
    lst = []
    lst.append(choice(gpaList,n))
    return var(lst,ddof=1)

def getPopVar(n):
    lst = []
    for x in range(n):
        lst.append(gpaList[randint(0,len(gpaList)-1)])
    return var(lst)



seed(0)


def testPopVar(M,n):
    sum = 0
    for x in range(M):
        sum = sum + getPopVar(n)
    average = sum/M
    return average

M = 1000
lo = 2
hi = 50

PV = []
popMu = var(gpaList)
for x in range(lo,hi+1):
    PV.append(testPopVar(M,x))

def testSampVar(M,n):
    sum = 0
    for x in range(M):
        sum = sum + getSampVar(n)
    average = sum/M
    return average

SV = []

for x in range(lo,hi+1):
    SV.append(testSampVar(M,x))

plt.figure(figsize=(15,7))
plt.title("Comparison of Sample and Population Variance of Samples")
plt.plot([lo,hi],[popMu,popMu],color='k',linestyle='--',label="Actual Var")
plt.plot(range(lo,hi+1),SV,color='g',label="Sample Var")
plt.plot(range(lo,hi+1),PV,color='r',label="Population Var")
plt.legend()
plt.show()
```


Comparison of Sample and Population Variance of Samples
Actual Var
Sample Var
Population Var
0.26
0.24
0.22
0.20
0.18
0.16
0.14
0.12
0
10
20
30
40
50