CS 237 Fall 2019 Homework Eight Solution

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)

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 - 8 are analytical problems about last week's material, and the remaining problems are coding problems which will be discussed in lab next week.

In [4]:

```
# 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 distribution
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)

```

In [5]:
```
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
```

In [33]:
```
0.4989/(30**0.5)
```

Out[33]:
```
0.09108626131310912
```

In [32]:
```
norm.cdf(x=66.13,loc=67,scale=0.345)
```

Out[32]:
```
0.005838813686631015
```

Problem One (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 [6]:
```
print("\nSolution (a)\n")
RngX = [ 2, 3, 6, 8, 11 ]

# Using functions from numpy library

mu = mean(RngX)
sigma = std(RngX)

print("\tPopulation mean = " + str(mu) + "\n\tPopulation standard deviation = " + str(round4(
```

```
Solution (a)

        Population mean = 6.0
        Population standard deviation = 3.2863
```

In [7]:
```
print("\nSolution (b)\n")

Rng_X_2 = [ [RngX[i],RngX[j]] for i in range(len(RngX)) for j in range(len(RngX))]

sample_means = [mean(s) for s in Rng_X_2]

mean_of_sample_means = mean(sample_means)

print("\tMean of sample means = " + str(mean_of_sample_means))
```

```
Solution (b)

        Mean of sample means = 6.0
```

In [8]:
```
print("\nSolution (c)\n")

sample_means = [mean(s) for s in Rng_X_2]

sigma_of_sample_means = std(sample_means)

print("\tStandard deviation of sample means = " + str(round4(sigma_of_sample_means)))
```

```
Solution (c)

        Standard deviation of sample means = 2.3238
```

In [9]:
```
print("\nSolution (d)\n")

print("\tCLT would predict that standard deviation of sample means = standard deviation of po

print("\t"+str(round4(sigma_of_sample_means)) + ' = ' + str(round4(sigma / (2**0.5))))
```

```
Solution (d)

        CLT would predict that standard deviation of sample means = standard deviation of popul
ation / sqrt(2):

        2.3238 = 2.3238
```

Problem 2 (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.

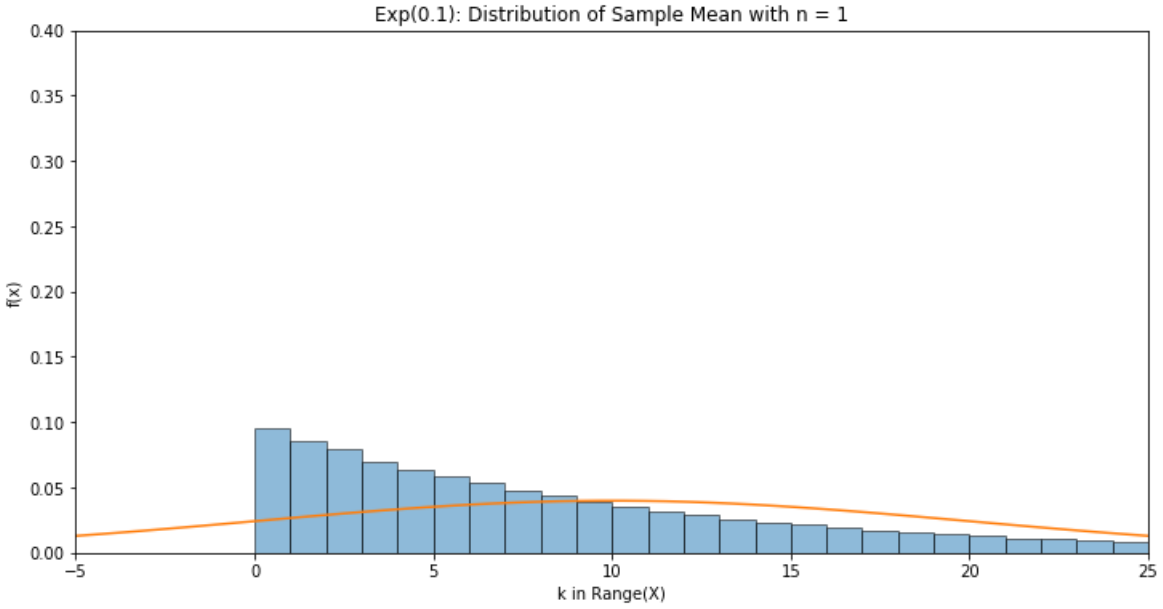
The result is shown below (we have left the charts produced by the solution so you can see what you are aiming for).

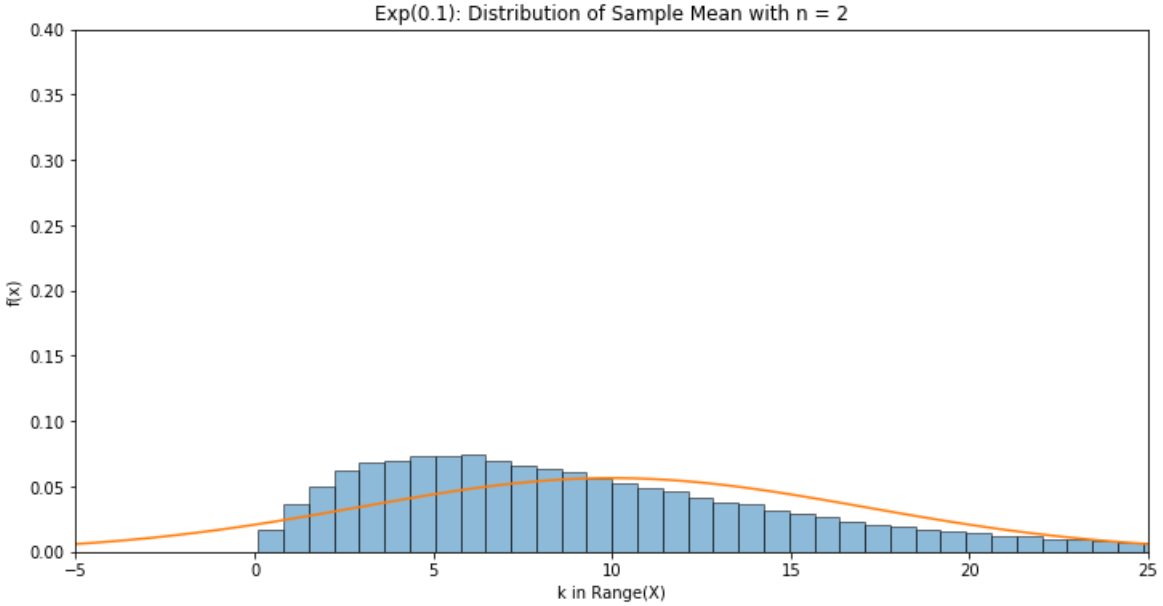
Follow the instructions given in the comments in the code cell below, using code and the bin_width you investigated in HW 07.

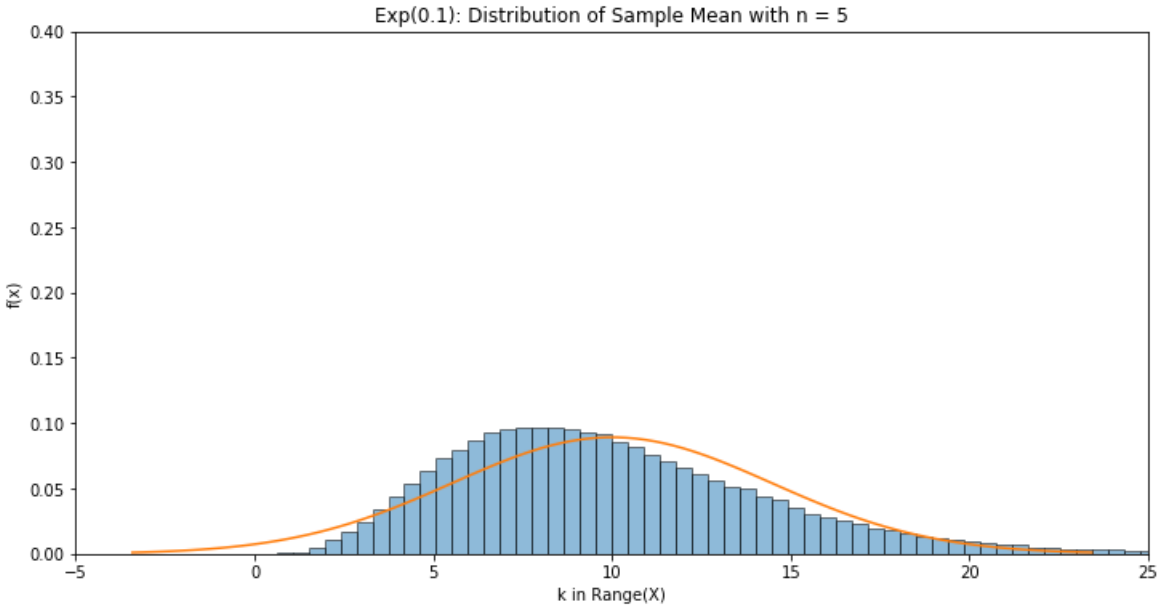
In [10]:

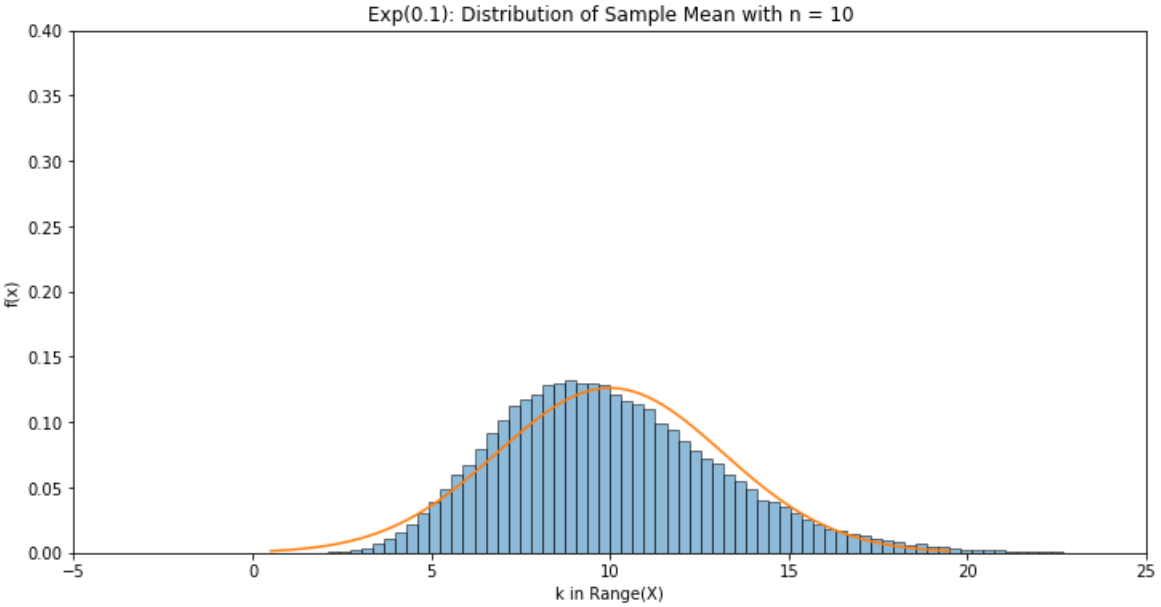
```
# Use the numpy function expon.rvs(...) to generate random exponential variates with
# rate parameter lam; this function simply generates n of them and returns the mean

def sampleMeanExponential(lam,n):
    return mean( expon.rvs(loc=0,scale=1/lam,size=n) )

# Now 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',alpha=0.5) # bin

    # Now generate the theoretical normal for sample mean with std dev sigma/sqrt(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
display_sample_mean_exponential(lam,1,num_trials)
display_sample_mean_exponential(lam,2,num_trials)
display_sample_mean_exponential(lam,5,num_trials)
display_sample_mean_exponential(lam,10,num_trials)
display_sample_mean_exponential(lam,30,num_trials)
display_sample_mean_exponential(lam,100,num_trials)
```

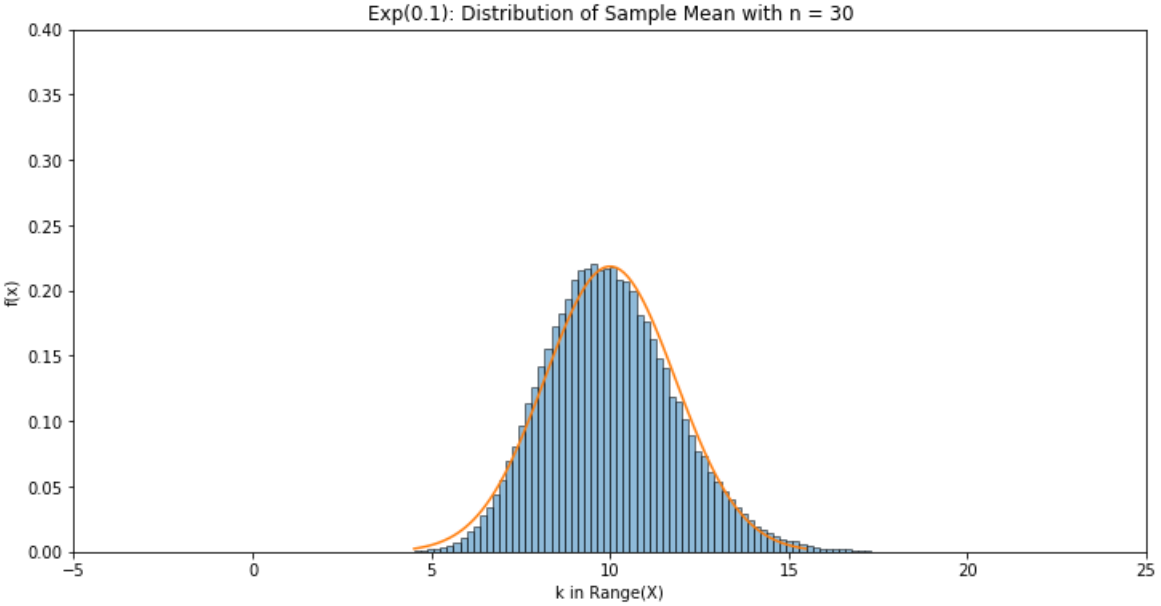
```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:33: MatplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' i
nstead.
```

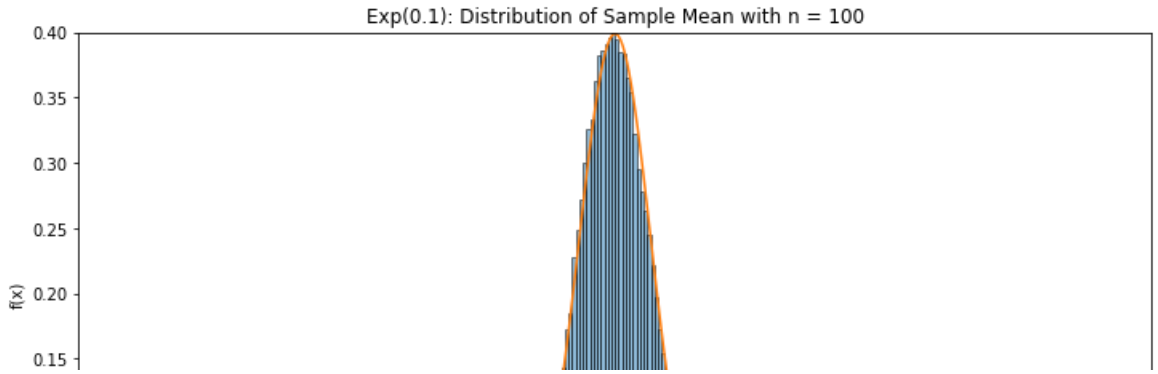

Exp(0.1): Distribution of Sample Mean with n = 1
f(x)
k in Range(X)


Exp(0.1): Distribution of Sample Mean with n = 2
f(x)
k in Range(X)


Exp(0.1): Distribution of Sample Mean with n = 5
f(x)
k in Range(X)


Exp(0.1): Distribution of Sample Mean with n = 10
f(x)
k in Range(X)
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−5
0
5
10
15
20
25


Exp(0.1): Distribution of Sample Mean with n = 30
f(x)
k in Range(X)
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−5
0
5
10
15
20
25



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 [11]:
```
print("\nSolution (a)\n")

mu = 68
sigma = 3
n =25

print("\tMean of sampling distribution of means = " + str(mu))
print("\n\tStandard deviation of sampling distribution of means = " + str(round4(sigma/(n**0.
```

```
Solution (a)

        Mean of sampling distribution of means = 68

        Standard deviation of sampling distribution of means = 0.6
```

In [12]:
```
print("\nSolution (b)\n")

print("\tWe seek n such that 3/sqrt(n) = 0.25, so sqrt(n) = 3/0.25 = 12, so n = 144")

```

```
Solution (b)

        We seek n such that 3/sqrt(n) = 0.25, so sqrt(n) = 3/0.25 = 12, so n = 144
```

In [13]:
```
print("\nSolution (c)\n")

p = norm.cdf(x=68.3,loc=mu,scale=3/5) - norm.cdf(x=66.8,loc=mu,scale=3/5)

print("\tXbar25 ~ N(68,9/25) with sigma = 3/5 so P(66.8 < X < 68.3) = " + str(round4(p)))

print("\n\tSolution = 80 * " + str(round4(p)) + " = " + str(round4(80*p)) + " or could round
```

```
Solution (c)

        Xbar25 ~ N(68,9/25) with sigma = 3/5 so P(66.8 < X < 68.3) = 0.6687

        Solution = 80 * 0.6687 = 53.497 or could round to 53
```

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 [14]:
```
print("\nNote: This is same as last problem but Bernoulli with p = 0.548.")

print("\nSolution (a)\n")

mu = 0.548
n = 30
sigma = (mu*(1-mu))**0.5
sigma_sample = sigma/(n**0.5)

print("\tMean of sample distribution of means = " + str(mu))
print("\n\tStandard deviation of sample distribution of means = " + str(round4(sigma_sample))
```

```
Note: This is same as last problem but Bernoulli with p = 0.548.

Solution (a)

        Mean of sample distribution of means = 0.548

        Standard deviation of sample distribution of means = 0.0909
```

In [15]:
```
print("\nSolution (b)")

print("\n\tStandard deviation of population = " + str(round4(sigma)))

print("\n\tWe seek the smallest n such that 0.4977/sqrt(n) <= 0.05, or the smallest")
print("\tinteger n such that sqrt(n) >= 0.4977/0.05 = 9.954, so n = ceil(9.954**2)")

print("\n\tSolution = 100     Check:  " + str(sigma) + "/sqrt(100) = " + str(sigma/(100**0.5)

```

```
Solution (b)

        Standard deviation of population = 0.4977

        We seek the smallest n such that 0.4977/sqrt(n) <= 0.05, or the smallest
        integer n such that sqrt(n) >= 0.4977/0.05 = 9.954, so n = ceil(9.954**2)

        Solution = 100     Check:  0.4976906669810074/sqrt(100) = 0.04976906669810074
```

In [16]:
```
print("\nSolution (c)")

p = norm.cdf(x=0.558,loc=mu,scale=sigma_sample) - norm.cdf(x=0.538,loc=mu,scale=sigma_sample)

print('\n\tXbar30 ~ N(0.548,'+str(round4(sigma_sample)) + '**2) with sigma = '+str(round4(sig

print("\n\tSolution = 100 * " + str(round4(p)) + " = " + str(round4(100*p)) + " or could roun

```

```
Solution (c)

        Xbar30 ~ N(0.548,0.0909**2) with sigma = 0.0909 so P(0.538 < X < 0.558) = 0.0876

        Solution = 100 * 0.0876 = 8.7633 or could round to 9
```

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 [17]:
```
print("\nSolution (a)")

print("\n\tThis is X ~ B(120,0.5) and we seek P(X >= 75)")

from scipy.stats import binom

sum = 0
for k in range(75,121):
    sum += binom.pmf(k,120,0.5)

print("\n\tSolution = " + str(round4(sum)))

```

```
Solution (a)

        This is X ~ B(120,0.5) and we seek P(X >= 75)

        Solution = 0.0039
```

In [18]:
```
print("\nSolution (b)")

print("\n\tY ~ N(60,30) with sigma = 5.4772 and we seek P(X >= 74.5)")

print("\n\tSolution = " + str(round4(norm.sf(x=74.5,loc=60,scale=5.4772))))
```

```
Solution (b)

        Y ~ N(60,30) with sigma = 5.4772 and we seek P(X >= 74.5)

        Solution = 0.0041
```

In [19]:
```
print("\nSolution (c)")

mu = 0.5
sigma = (mu*(1-mu))**0.5
n = 120
sigma_samplemean = sigma/(n**0.5)

print("Std dev of sample mean: ", sigma_samplemean)

print("\n\tY ~ N(0.5,0.0456^2) with sigma = 0.0456 and we seek P(X >= 0.625)")

print("\n\tSolution = " + str(round4(norm.sf(x=5/8,loc=0.5,scale=sigma_samplemean))))

```

```
Solution (c)
Std dev of sample mean:  0.04564354645876384

        Y ~ N(0.5,0.0456^2) with sigma = 0.0456 and we seek P(X >= 0.625)

        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 [20]:
```
print("\nSolution (a)")

n = 100

mu = 67.45
sigma = 2.93
sigma_sample = sigma / (n**0.5)

print("\n\tSolution: " + str(mu) + " +/- " + str(round4(2 * sigma_sample)))
```

```
Solution (a)

	Solution: 67.45 +/- 0.586
```

In [21]:
```
print("\nSolution (b)")

print("\n\tSolution: " + str(mu) + " +/- " + str(round4(3 * sigma_sample)))
```

```
Solution (b)

	Solution: 67.45 +/- 0.879
```

In [22]:
```
print("\nSolution (c)")

n = 100

mu = 67.45
sigma = 2.93
sigma_sample = sigma / (n**0.5)

print("\n\tSolution: " + str(mu) + " +/- " + str(round4(1.96 * sigma_sample)))
```

```
Solution (c)

	Solution: 67.45 +/- 0.5743
```

In [23]:
```
print("\nSolution (d)")

print("\n\tSolution: " + str(mu) + " +/- " + str(round4(2.5758 * sigma_sample)))
```

```
Solution (d)

	Solution: 67.45 +/- 0.7547
```

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

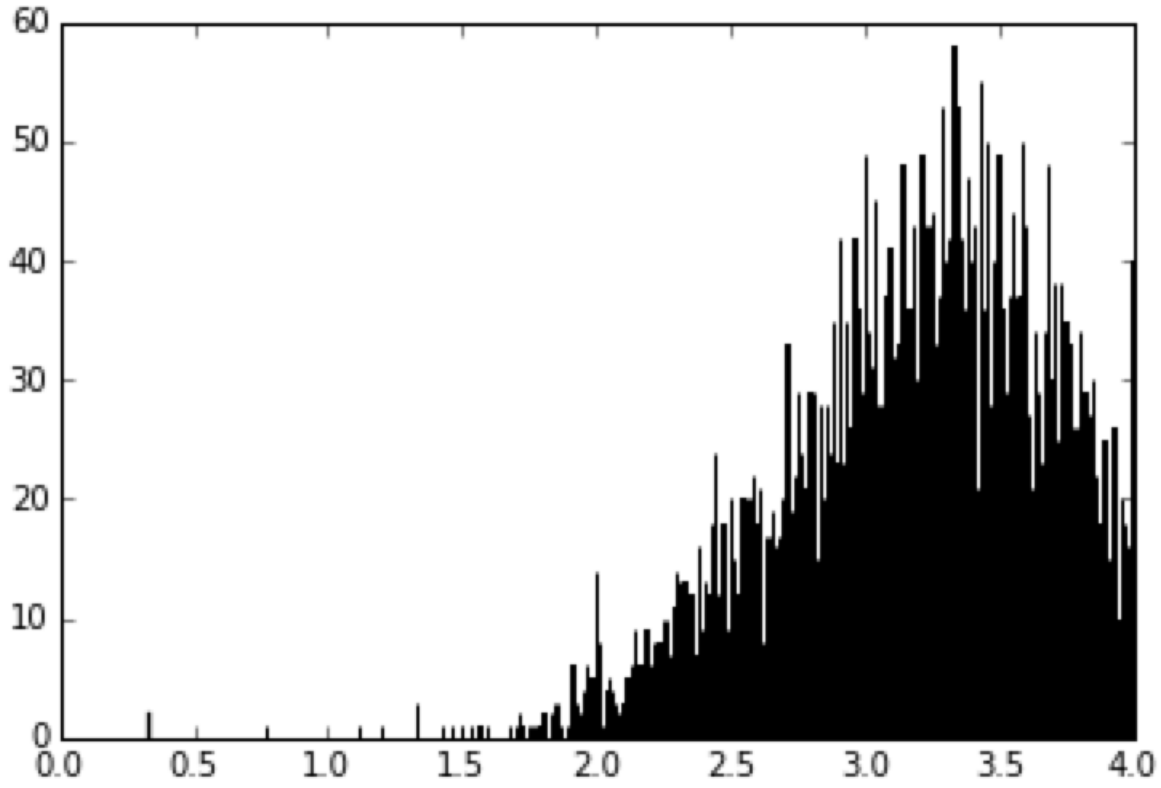
The GPAs for 4897 individuals from an institution of higher education in the northeastern United States will be read into a list using the following Pandas code:

In [24]:
```
studs = pd.read_csv('http://www.cs.bu.edu/fac/snyder/cs237/Homeworks,%20Labs,%20and%20Code/St
gpaList = studs['GPA'].tolist()
studs[0:10]
```

Out[24]:

	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 [25]:
```
# Part (a)

# Use appropriate functions from scipy library given in first cell.

popMu = mean(gpaList)

popSigma = std(gpaList)

print("\n(a) mean = " + str(round4(popMu)) + "\t" + "sigma = " + str(round4(popSigma)) )


```

```
(a) mean = 3.1729       sigma = 0.4934
```

In [26]:
```
# Part (b)

n = 100

def getSample(n):
    samp = choice(gpaList,n)
    return (mean(samp),std(samp,ddof=1))

seed(0)

(sm,ssd) = getSample(n)
print(ssd)
sigma_sample = ssd / (n**0.5)

print("\n(b)  ", round4(sm),"+/-",round4(1.96 * sigma_sample))

```

```
0.444641678458309

(b)   3.2022 +/- 0.0871
```

In [27]:
```
# Part (c)

n = 100
CL = 0.95
num_trials = 10**5

def getCL(p):
    return norm.interval(alpha=p, loc=0, scale=1)[1]

def getInterval(c,n):
    (sm,ssd) = getSample(n)
    delta = c * ssd / (n**0.5)
    return (sm-delta,sm,sm+delta)

seed(0)

count = 0
c = getCL(CL)
for k in range(num_trials):
    (lo,_,hi) = getInterval(c,n)
    if (popMu >= lo and popMu <= hi):
        count += 1
print("(c)  ",round4(count/num_trials))
```

```
(c)   0.9461
```

In [28]:
```
# Part (d)

n = 50
CL = 0.90
num_trials = 10**5

seed(0)

count = 0
c = getCL(CL)
for k in range(num_trials):
    (lo,_,hi) = getInterval(c,n)
    if (popMu >= lo and popMu <= hi):
        count += 1
print("(d)  ",round4(count/num_trials))
```

```
(d)   0.8919
```

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 [29]:
```
def getSampVar(n):
    return var(choice(gpaList,n),ddof=1)

def getPopVar(n):
    return var(choice(gpaList,n))

# Solution for (a)

seed(0)

actualVar = var(gpaList)

lo = 2
hi = 51

def testPopVar(M,n):
    s = 0
    for k in range(M):
        s += getPopVar(n)
    return s / M

M = 1000

PV = [testPopVar(M,n) for n in range(lo,hi)]

def testSampVar(M,n):
    s = 0
    for k in range(M):
        s += getSampVar(n)
    return s / M

SV = [testSampVar(M,n) for n in range(lo,hi)]


print("(c)")

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

```
(c)
```