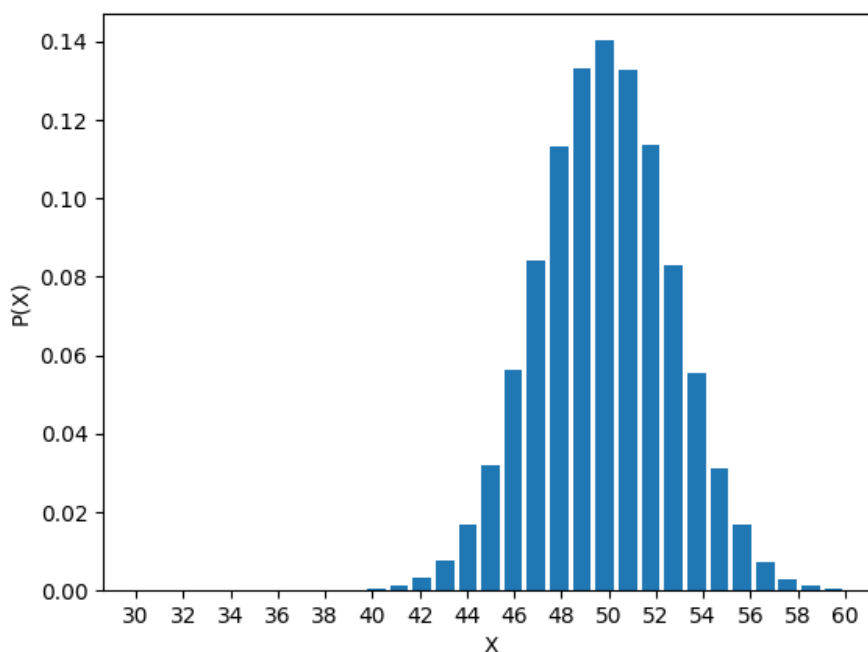


## Problem Set #5 Solutions

With problems by Mehran Sahami and Chris Piech

1. a. Given a prior  $Beta(a, b)$ , after  $n$  successes and  $m$  failures your posterior distribution is  $Beta(a + n, b + m)$ . Thus we have a posterior belief of  $Beta(2 + 7, 2 + 2) = \mathbf{Beta(9,4)}$ .
- b.  $P(X > 0.5) = 1 - F_X(0.5) = 0.927$
2. a. Bar graph of simulated probabilities:



- b. By the Central Limit Theorem, the sum of 100 IID variables can be approximated as a normal. Let  $X_i \sim \text{Uniform}(0, 1)$  be a single random variable and  $X = \sum_{i=1}^{100} X_i$ . Then

$$X \sim N(n \cdot E[X_i], n \cdot \text{Var}(X_i)) = N(100 \cdot 0.5, 100 \cdot \frac{1}{12}) = \mathbf{N(50, 8.33)}$$

You could have also gotten to this answer by calculating the  $E[X]$  and  $\text{Var}(X)$ .  $E[X]$  is the sum of the expectations of all the  $X_i$  values =  $100 \cdot 0.5 = 50$ . Since each of the  $X_i$  are independent  $\text{Var}(X)$  is the sum of the variances of all the  $X_i$  values =  $100 \cdot \text{Var}(X_i) = 100 \cdot 1/12 = 8.33$ . We want students to justify using a normal by citing the central limit theorem.

c.

$$\begin{aligned}
 P(47.5 < X < 48.5) &= P\left(\frac{47.5 - 50}{\sqrt{100/12}} < Z < \frac{48.5 - 50}{\sqrt{100/12}}\right) \\
 &= \phi(-0.520) - \phi(-0.867) \\
 &= (1 - \phi(0.520)) - (1 - \phi(0.867)) = \phi(0.867) - \phi(0.520) \\
 &\approx 0.807 - 0.6985 \approx 0.11
 \end{aligned}$$

Which is very similar to the results I got from sampling. Of note, 100,000 calculations means that these numbers should only agree (consistently) to two decimal places.

3. a. Let  $X_i$  be the amount of money that person  $i$  gives.  
 $E[X_i] = 0.2 * 1 + 0.35 * 5 + 0.3 * 10 + 0.05 * 20 = \mathbf{5.95}$
- b.  $E[X_i^2] = 0.2 * 1^2 + 0.35 * 5^2 + 0.3 * 10^2 + 0.05 * 20^2 = 58.95$   
 $Var(X_i) = E[X_i^2] - E[X_i]^2 = 58.95 - 5.95 * 5.95 = \mathbf{23.5475}$
- c. Let  $Y$  be the total amount of money that the band earns.  $Y = \text{sum of all the } X_i\text{'s}$ . By the central limit theorem, since the  $X_i$ 's are IID and the number of  $X_i$ 's is more than 20, we expect  $Y$  to be normally distributed.  
 $Y \sim N(n * E[X_i], n * Var(X_i))$   
 $Y \sim \mathbf{N(297.5, 1177.35)}$
- d.

$$\begin{aligned}
 P(Y > 350) &= 1 - P(Y < 350) \\
 &= 1 - P\left(Z < \frac{350 - 297.5}{\sqrt{1177.35}}\right) \\
 &= 1 - \phi(1.53) \approx \mathbf{0.06}
 \end{aligned}$$

(better with a continuity correction... but you get the same answer)

4.  $Cov(X, Y) = E[XY] - E[Y]E[X]$ .  
As we have seen from class,  $E[X] = 3.5$   
 $E[Y] = E[X^2] = \sum_x \frac{1}{6}x^2 = \frac{1}{6}(1 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2) = 15.1667$   
 $E[XY] = E[X^3] = \sum_x \frac{1}{6}x^3 = \frac{1}{6}(1 + 2^3 + 3^3 + 4^3 + 5^3 + 6^3) = 73.5$   
Finally,  $Cov(X, Y) = 73.5 - 3.5 * 15.1667 = \mathbf{20.4}$
5.  $2X \sim N(2 * 1, 2^2 * 2) = N(2, 8)$  by linear transformation of a normal RV. Using convolution, you get that  $2X + Y \sim N(2 + 1, 8 + 2) = \mathbf{N(3, 10)}$ .
6. Think of the dice roll as a multinomial with  $n = 6$  and  $p_1 = \dots = p_6 = \frac{1}{6}$ :

$$\frac{P(X_1 = 1, \dots, X_6 = 1)}{P(X_6 = 6)} = \frac{\binom{n}{1, \dots, 1} p_1 * \dots * p_6}{\binom{n}{6} p_6^6} = \frac{\binom{6}{1, \dots, 1} \left(\frac{1}{6}\right)^1 \dots \left(\frac{1}{6}\right)^1}{\binom{6}{6} \left(\frac{1}{6}\right)^6} = \frac{6! \left(\frac{1}{6}\right)^6}{1 \left(\frac{1}{6}\right)^6} = 6!$$

7. a.  $E[X] = E[X] = \sum_{i=1}^6 P(X=i)E[X] = 1/6(6+5+4+3)+1/6(\sum_{i=1}^2 i + E[X])$   
 Solving for  $E[X]$  yields:  $E[X] = 1/6(6+5+4+3+2+1) + 2/6E[X]$   
 $2/3E[X] = 1/6(6+5+4+3+2+1)$   
 $E[X] = 1/4(6+5+4+3+2+1) = 21/4 = \mathbf{5.25}$
- b. Note that  $Y \sim \text{Geo}(p = 4/6 = 2/3)$ , as we will repeatedly roll the die until we get a 3, 4, 5 or 6 (out of 6 possible values that could be rolled). Since the expectation of a Geometric random variable is  $1/p$ , we get  $E[Y] = 1/(2/3) = \mathbf{3/2}$ .

8. Using the definition of variance:

$$\text{Var}(Y) = E[Y^2] - E[Y]^2$$

We already know from class  $E[Y] = 15$ . To calculate  $E[Y^2]$ , we can use the law of total expectation:

$$\begin{aligned} E[Y^2] &= \sum_{i \in \{1,2,3\}} E[Y^2 | X=i]p(X=i) \\ &= \frac{1}{3} \left( 3^2 + E[(5+Y)^2] + E[(7+Y)^2] \right) \end{aligned}$$

A few algebra steps gives the following equation:

$$E[Y^2] = \frac{1}{3}(443 + 2E[Y^2]) \implies E[Y^2] = 443$$

Now we can plug this back into the original equation to get the variance:

$$\text{Var}(Y) = 443 - 15^2 = 218$$

9. (a) What is the approximate probability that Program A completes in less than 950 seconds? Define a new random variable  $X_A = \sum_{i=1}^{20} x_i$  where  $x_i$  is a random variable representing the time for the  $i$ th algorithm to finish running. Since they're IID, we can apply the Central Limit Theorem to approximate the  $X_A$  distribution:

$$X_A \sim \mathcal{N}(20 * 50, 20 * 100) \implies X_A \sim \mathcal{N}(1000, 2000)$$

Now we can calculate  $P(X_A < 950)$ :

$$P(X_A < 950) = \Phi\left(\frac{950 - 1000}{\sqrt{2000}}\right) = \Phi(-1.12) = 1 - \Phi(1.12) \approx 0.1314$$

- (b) What is the approximate probability that Program B completes in less than 950 seconds? Following the same steps outlined in part (a) above give our approximate  $X_B$  distribution:

$$X_B \sim \mathcal{N}(1040, 4000)$$

Now we can calculate  $P(X_B < 950)$ :

$$P(X_B < 950) = \Phi\left(\frac{950 - 1040}{\sqrt{4000}}\right) \approx 0.0778$$

- (c) What is the approximate probability that Program A completes in less time than B?  
Define a new random variable  $X_{A-B} = X_A - X_B$ . Since the sum/difference of random variables is also normal,

$$X_{A-B} \sim \mathcal{N}(1000 - 1040, 2000 + 4000)$$

$$X_{A-B} \sim \mathcal{N}(-40, 6000)$$

If program A finishes in less time than program B, then  $X_{A-B} < 0$ . We can calculate the probability of this:

$$P(X_{A-B} < 0) = \Phi\left(\frac{0 - (-40)}{\sqrt{6000}}\right) = \Phi(0.516) \approx 0.697$$

10.
  - a. A few reasons: Some students might do activity1 and activity2 and their knowledge of the first activity might affect their performance in the second activity, the populations in activity1 and activity 2 could be entirely different, there could be a confounding variable of time (e.g. if activity2 occurs during Christmas time and activity1 doesn't), etc. Since the sample of students who get the two treatments are different, you will not get a legitimate comparison.
  - b. A few reasons: education systems might be different, access to resources might be different, time zones are different, etc. Since the sample of students who get the two treatments are different, you will not get a legitimate comparison.
11. A/B testing. Over a two-week period, Coursera randomly assigns each student to either be given activity1 (group A), or activity2 (group B). The activity that is shown to each student and the student's measured learning outcomes can be found in the file: learningOutcomes.csv
  - a. What is the difference in sample means of learning outcomes between students who were given activity1 and students who were given activity2?

$$\bar{X}_1 - \bar{X}_2 = -8.2$$

- b. Calculate a p-value for the observed difference in means reported in part (a). In other words: assuming the learning outcomes for students who had been given activity1 and activity2 were identically distributed, what is the probability that you could have sampled two groups of students such that you could have observed a difference of means as extreme, or more extreme, than the one calculated from your data? Describe any code you used to calculate your answer.

The p-value can be calculated using bootstrapping. Here is our code:

```
def nullHypothesis(setA, setB, observedDiff):
    universalSet = copy.deepcopy(setA)
```

```
universalSet.extend(setB)
count = 0
for i in range(10000):
    sampleA = sample(universalSet, len(setA))
    sampleB = sample(universalSet, len(setB))
    diff = abs(np.mean(sampleB) - np.mean(sampleA))
    if diff >= observedDiff: count += 1
return count / 10000.0
```

#### The resulting p-value is 0.04

- c. File background.csv stores the background of each user. Student backgrounds fall under three categories: more experience, average experience, less experience. For each of the three backgrounds calculate a difference in means in learning outcome between activity1 and activity2, and the p-value of that difference. Describe your methodology.

For students with **less** experience  
Activity1 is better than Activity2  
Difference in means = 26.0  
P-Value < 0.001  
Based on the 311 students with less experience

For students with **average** experience  
Activity2 is better than Activity1  
Difference in means = 25.0  
P-Value < 0.01  
Based on the 229 students with average experience

For students with **more** experience  
Activity2 is better than Activity1  
Difference in means = 28.4  
P-Value < 0.0001  
Based on the 511 students with average experience

All of these values were calculated using bootstrapping, but only for the population of students with the given background.

Our actual Python implementation of this question is included below for reference:

```

1 '''
2 Implementation of coding portion of Question 11 of
3 HW #5 for CS 109: Probability for Computer Scientists
4
5 Author: Sam Schwager
6 Last edited: 12/5/2018
7 '''
8
9 import numpy as np
10 import pandas as pd
11
12
13 def calculate_p_value(sample_1, sample_2, num_iters, verbose=True):
14     # State which activity is preferred based on our observations
15     if verbose: print("Activity 1 preferred") if np.mean(sample_1) > np.mean(sample_2) else print("Activity 2 preferred")
16
17     # We'll compare the absolute means from our bootstrapped
18     # samples to this observed difference in means
19     observed_mean_diff = abs(np.mean(sample_1) - np.mean(sample_2))
20
21     if verbose: print ("Absolute observed mean difference is: " + str(observed_mean_diff))
22
23     # We'll draw bootstrap samples from universal_set
24     universal_set = np.concatenate((sample_1, sample_2))
25     count = 0
26
27     for i in range(num_iters):
28         resample_1 = np.random.choice(universal_set, sample_1.shape[0])
29         resample_2 = np.random.choice(universal_set, sample_2.shape[0])
30
31         if abs(np.mean(resample_1) - np.mean(resample_2)) > observed_mean_diff: count += 1
32
33     return float(count) / num_iters
34
35
36 def main():
37     # Read in learning outcomes and replace the strings corresponding
38     # to each activity with an appropriate integer value
39     learning_outcomes = pd.read_csv("learningOutcomes.csv", header=None)\
40         .replace(["activity1", "activity2"], [1, 2])
41
42     # Get numpy array with first column corresponding to activity
43     # and second column corresponding to learning outcome
44     learning_outcomes_arr = learning_outcomes.values[:, 1:3]
45
46     # Select rows where first column corresponds to desired activity
47     activity_1_outcomes = learning_outcomes_arr[learning_outcomes_arr[:,0] == 1]
48     activity_2_outcomes = learning_outcomes_arr[learning_outcomes_arr[:,0] == 2]
49
50     # Use numpy array slicing to calculate absolute mean difference
51     mean_diff = abs(np.mean(activity_1_outcomes[:, 1]) - np.mean(activity_2_outcomes[:, 1]))
52     print ("Part a: The absolute difference in means is " + str(mean_diff))
53
54     mean_diff_p_val = calculate_p_value(activity_1_outcomes[:, 1], activity_2_outcomes[:, 1], 10000, verbose=False)
55     print ("Part b: The p-value for the observed difference in means from part a is " + str(mean_diff_p_val))
56
57     # Read in background data for part c
58     backgrounds = pd.read_csv("background.csv", header=None)
59
60     # Create sets of less, average, and more experienced IDs
61     less_experience_ids = set(backgrounds.loc[backgrounds[1] == "less"].loc[:, 0])
62     avg_experience_ids = set(backgrounds.loc[backgrounds[1] == "average"].loc[:, 0])
63     more_experience_ids = set(backgrounds.loc[backgrounds[1] == "more"].loc[:, 0])
64
65     # Select rows of learning outcomes meeting given experience conditions
66     less_experience_outcomes = learning_outcomes[learning_outcomes[0].isin(less_experience_ids)]
67     avg_experience_outcomes = learning_outcomes[learning_outcomes[0].isin(avg_experience_ids)]
68     more_experience_outcomes = learning_outcomes[learning_outcomes[0].isin(more_experience_ids)]
69
70     # Get numpy arrays with first column corresponding to activity
71     # and second column corresponding to learning outcome
72     less_arr = less_experience_outcomes.values[:, 1:3]
73     avg_arr = avg_experience_outcomes.values[:, 1:3]
74     more_arr = more_experience_outcomes.values[:, 1:3]
75
76     print ("There are " + str(less_arr.shape[0]) + " students with less experience")
77     print ("There are " + str(avg_arr.shape[0]) + " students with average experience")
78     print ("There are " + str(more_arr.shape[0]) + " students with more experience")
79
80     less_pval = calculate_p_value(less_arr[less_arr[:,0] == 1][:, 1], less_arr[less_arr[:,0] == 2][:, 1], 10000)
81     avg_pval = calculate_p_value(avg_arr[avg_arr[:,0] == 1][:, 1], avg_arr[avg_arr[:,0] == 2][:, 1], 10000)
82     more_pval = calculate_p_value(more_arr[more_arr[:,0] == 1][:, 1], more_arr[more_arr[:,0] == 2][:, 1], 10000)
83
84     print ("Part c (i): The p-value for the observed difference in means for less-experienced users is " + str(less_pval))
85     print ("Part c (ii): The p-value for the observed difference in means for average-experienced users is " + str(avg_pval))
86     print ("Part c (iii): The p-value for the observed difference in means for more-experienced users is " + str(more_pval))
87
88
89 if __name__ == "__main__":
90     main()

```

12. Soln 12

13. a.  $\theta_1 \sim \text{Beta}(a = 6, b = 3)$   
 b. here is our pseudocode:

```
a1, b1 = 1, 1
a2, b2 = 1, 1
for i in range(100):
    v1 = sampleBeta(a1, b1)
    v2 = sampleBeta(a2, b2)
    chosen = argmax([v1, v2])
    worked = giveDrug(chosen)
    if chosen == 1 and worked: a1 += 1
    if chosen == 1 and not worked: b1 += 1
    if chosen == 2 and worked: a2 += 1
    if chosen == 2 and not worked: b2 += 1
```

- c. If  $X$  is a beta,  $E[X] = \frac{a}{a+b}$ :  
 $E[\theta_1] = \frac{11}{22} = 1/2$   
 $E[\theta_2] = \frac{76}{82} \approx 0.93$

14. (a) Here is our pseudocode to calculate probFlu() using **Joint Sampling**:

```
def bernoulli(p):
    if random() < p return 1 else return 0

def generateOneSample():
    sample = {}
    sample["stress"] = bernoulli(0.5)
    sample["exposure"] = bernoulli(0.1)
    sample["cold"] = bernoulli(probCold(sample["stress"],
                                         sample["exposure"]))
    sample["flu"] = bernoulli(probFlu(sample["stress"],
                                       sample["exposure"]))

    for i in range(1, 11):
        symptom = "X" + str(i)
        probSymptom = probSymptom(i, sample["cold"],
                                   sample["flu"])
        sample[symptom] = bernoulli(probSymptom)
    return sample

def probFlu():
    numExposureAndX2 = 0
    numFluExposureAndX2 = 0
```





```

def calculateDenominator():
    totalSum = 0.0
    possibleVals = [0,1]
    for f in possibleVals:
        for s in possibleVals:
            for c in possibleVals:
                for x1 in possibleVals:
                    for x3 in possibleVals:
                        ...
                            for x10 in possibleVals:
                                x = [x1,1,x3,...,x10]
                                totalSum += jointProb(s,1,c,f,x)
def probFlu():
    return calculateNumerator() / calculateDenominator()

```