COSE471 hw1

GAOZHONGSONG

TOTAL POINTS

22 / 26

QUESTION 1

1 Q1 1 / 1

✓ + 1 pts **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 2

2 Q2a 1 / 1

✓ + 1 pts **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 3

3 Q2b 1 / 1

✓ + 1 pts **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 4

4 Q2c 1 / 1

✓ + 1 pts **most = "Sam"**

+ 0 pts Failed to pass all unit tests

QUESTION 5

5 Q2d 1 / 1

✓ + 1 pts **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 6

6 Q3a 3 / 4

✓ + 1 pts **$$\frac{\partial}{\partial x} f(x,y)=2x+4y$$**

✓ + 1 pts **$$\frac{\partial}{\partial y}(f(x,y))=4x + 6y^2 -3 e^{-3y} + \frac{2}{2y}$$**

+ 1 pts $$\nabla f(x,y)=[2x + 4y, 4x + 6y^2 -3 e^{-3y} + \frac{2}{2y}]^T$$

✓ + 1 pts **$$\nabla f(2,-1)=[0, 13 - 3 e^{3}]^T$$**

+ 0 pts WRONG

QUESTION 7

7 Q3b 2 / 2

✓ + 1 pts **$$\dfrac{\mathrm{d}}{\mathrm{d}x}\sum_{i=1}^{10} (i - x)^2=-2 \left(55 - 10x \right)$$**

✓ + 1 pts **$$x = 5.5$$**

+ 0 pts This rubric is removed

+ 0 pts WRONG

QUESTION 8

8 Q4a 0 / 2

+ 2 pts Correct

✓ + 0 pts **Not_correct**

+ 0 pts Click here to replace this description.

QUESTION 9

9 Q4b 2 / 2

✓ + 2 pts **Correct**

+ 0 pts WRONG

QUESTION 10

10 Q5 1 / 2

+ 1 pts $$ P(\text{Has Cancer} | \text{Tested Positive}) = \dfrac{P(\text{Tested Positive} | \text{Has Cancer}) \times P(\text{Has Cancer})}{P(\text{Tested Positive})} $$

✓ + 1 pts **$$ P(\text{Has Cancer} | \text{Tested Positive}) = \dfrac{0.80 \times 0.01}{0.80 \times 0.01 + 0.096 \times (1 - 0.01)} \approx 0.078$$**

+ 0 pts WRONG

QUESTION 11

11 Q6a 2 / 2

✓ + 2 pts **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 12

12 Q6b **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 13

13 Q6c **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 14

14 Q7a **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 15

15 Q7b **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 16

16 Q7c **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 17

17 Q7d **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 18

18 Q7e **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

In [7]:

```
def mean(population):
    """
    Compute the mean of population (mu).

    Args:
        population: a numpy array of numbers of shape [N,]
    Returns:
        the mean of population (mu).
    """
    # Calculate the mean of a population
    # BEGIN YOUR CODE
    # ------------------------
    mu = sum(population)/len(population)
    return mu
    # ------------------------
    # END YOUR CODE

def variance(population):
    """
    Compute the variance of population (sigma squared).

    Args:
        population: a numpy array of numbers of shape [N,]
    Returns:
        the variance of population
    """
    # Calculate the variance of a population
    # BEGIN YOUR CODE
    # ------------------------
    avg = sum(population)/len(population)
    var = sum((x-avg)**2 for x in population) / len(population)
    return var
    # ------------------------
    # END YOUR CODE
```

In [8]:

```
ok.grade("q1");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 4
    Failed: 0
[oooooooooo] 100.0% passed
```

1 Q1 1 / 1

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

Question 2 (4 pt)

Question 2a

Joey, Deb, and Sam are shopping for fruit at K-Bowl. K-Bowl, true to its name, only sells fruit bowls. A fruit bowl contains some fruit and the price of a fruit bowl is the total price of all of its individual fruit.

Berkeley Bowl has apples for $2.00, bananas for \$1.00, and cantaloupes for $4.00 (expensive!). The price of each of these can be written in a vector:

$$v = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}$$

K-Bowl sells the following fruit bowls:

1. 2 of each fruit
2. 5 apples and 8 bananas
3. 2 bananas and 3 cantaloupes
4. 10 cantaloupes

Create a 2-dimensional numpy array encoding the matrix B such that the matrix-vector multiplication

$$Bv$$

evaluates to a length 4 column vector containing the price of each fruit bowl. The first entry of the result should be the cost of fruit bowl #1, the second entry the cost of fruit bowl #2, etc.

In [13]:

```
v = np.array([2,1,4])

# BEGIN YOUR CODE
# ------------------------
B = np.array([[2,2,2],
              [5,8,0],
              [0,2,3],
              [0,0,10]])
# ------------------------
# END YOUR CODE

# The notation B @ v means: compute the matrix multiplication Bv
B @ v
```

Out[13]:

```
array([14, 18, 14, 40])
```

In [14]:

```
ok.grade("q2a");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 2b

Joey, Deb, and Sam make the following purchases:

- Joey buys 2 fruit bowl #1s and 1 fruit bowl #2.
- Deb buys 1 of each fruit bowl.
- Sam buys 10 fruit bowl #4s (he really like cantaloupes).

Create a matrix A such that the matrix expression

$$ABv$$

evaluates to a length 3 column vector containing how much each of them spent. The first entry of the result should be the total amount spent by Joey, the second entry the amount sent by Deb, etc.

Note that the tests for this question do not tell you whether your answer is correct. That's up to you to determine.

2 Q2a 1 / 1

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

In [15]:

```
A = np.array([
    [2, 1, 0, 0],
    # Finish this!
    # BEGIN YOUR CODE
    # ------------------------
    [1,1,1,1],[0,0,0,10]
    # ------------------------
    # END YOUR CODE
])

A @ B @ v
```

Out[15]:

```
array([ 46,  86, 400])
```

In [16]:

```
ok.grade("q2b");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 2c

Who spent the most money? Assign `most` to a string containing the name of this person.

In [17]:

```
# BEGIN YOUR CODE
# ------------------------
most = 'Sam'
# ------------------------
# END YOUR CODE
```

In [18]:

```
ok.grade("q2c");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

3 Q2b 1 / 1

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

In [15]:

```
A = np.array([
    [2, 1, 0, 0],
    # Finish this!
    # BEGIN YOUR CODE
    # ------------------------
    [1,1,1,1],[0,0,0,10]
    # ------------------------
    # END YOUR CODE
])

A @ B @ v
```

Out[15]:

```
array([ 46,  86, 400])
```

In [16]:

```
ok.grade("q2b");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 2c

Who spent the most money? Assign `most` to a string containing the name of this person.

In [17]:

```
# BEGIN YOUR CODE
# ------------------------
most = 'Sam'
# ------------------------
# END YOUR CODE
```

In [18]:

```
ok.grade("q2c");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

4 Q2c 1 / 1

✓ **+ 1 pts** **most = "Sam"**

+ 0 pts Failed to pass all unit tests

Question 2d

Let's suppose K-Bowl changes their fruit prices, but you don't know what they changed their prices to. Joey, Deb, and Sam buy the same quantity of fruit baskets and the number of fruit in each basket is the same, but now they each spent these amounts:

$$x = \begin{bmatrix} 80 \\ 80 \\ 100 \end{bmatrix}$$

Use `np.linalg.inv` and the above final costs to compute the new prices for the individual fruits as a vector called `new_v`.

In [19]:

```
# BEGIN YOUR CODE
# ------------------------
x = np.array([80,80,100])
c = A@B
new_v = np.linalg.inv(c) @ np.array(x)
# ------------------------
# END YOUR CODE
new_v
```

Out[19]:

```
array([ 5.5       ,  2.20833333,  1.        ])
```

In [20]:

```
tmp = ok.grade("q2d");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed
```

5 Q2d 1 / 1

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

Question 3a (4 pt)

Suppose we have the following scalar-valued function:

$$f(x, y) = x^2 + 4xy + 2y^3 + e^{-3y} + \ln(2y)$$

Compute the partial derivative $\frac{\partial}{\partial x} f(x, y)$:

Answer:

$$2x + 4y$$

Now compute the partial derivative $\frac{\partial}{\partial y} f(x, y)$:

Answer:

$$4x + 6y^2 - 3e^{-3y} + 1/y$$

Finally, using your answers to the above two parts, compute $\nabla f(x, y)$. Also what is the gradient at the point (x, y) = (2, -1):

Note that ∇ represents the gradient.

Answer:

$$\nabla f(2, 1) = \begin{bmatrix} 0 \\ 13 - 3e^3 \end{bmatrix}$$

Question 3b (2 pt)

Find the value(s) of x which minimizes the expression below. Justify why it is the minimum.

$\sum_{i=1}^{10} (i - x)^2$

6 Q3a 3 / 4

✓ **+ 1 pts** **$$\frac{\partial}{\partial x} f(x,y)=2x+4y$$**

✓ **+ 1 pts** **$$\frac{\partial}{\partial y}(f(x,y))=4x + 6y^2 -3 e^{-3y} + \frac{2}{2y}$$**

+ 1 pts $$\nabla f(x,y)=[2x + 4y, 4x + 6y^2 -3 e^{-3y} + \frac{2}{2y}]^T$$

✓ **+ 1 pts** **$$\nabla f(2,-1)=[0, 13 - 3 e^{3}]^T$$**

+ 0 pts WRONG

Question 3a (4 pt)

Suppose we have the following scalar-valued function:

$$f(x, y) = x^2 + 4xy + 2y^3 + e^{-3y} + \ln(2y)$$

Compute the partial derivative $\frac{\partial}{\partial x} f(x, y)$:

Answer:

$$2x + 4y$$

Now compute the partial derivative $\frac{\partial}{\partial y} f(x, y)$:

Answer:

$$4x + 6y^2 - 3e^{-3y} + 1/y$$

Finally, using your answers to the above two parts, compute $\nabla f(x, y)$. Also what is the gradient at the point (x, y) = (2, -1):

Note that ∇ represents the gradient.

Answer:

$$\nabla f(2, 1) = \begin{bmatrix} 0 \\ 13 - 3e^3 \end{bmatrix}$$

Question 3b (2 pt)

Find the value(s) of x which minimizes the expression below. Justify why it is the minimum.

$\sum_{i=1}^{10} (i - x)^2$

Answer:
First:

$$let: f(x) = \sum_{i=1}^{10}(1-x)^2$$

Then

$$f'(x) = -2\sum_{i=1}^{10}(i-x) = -110 + 20x = 0$$

Finally solve the equation to get

$$10x = 55, x = 5.5$$

Question 4a (2 pt)

Let $\sigma(x) = \dfrac{1}{1+e^{-x}}$.

Show that $\sigma(-x) = 1 - \sigma(x)$.

Answer:

$$\sigma(-x) = 1/(1+e^x) = 1/e^x(1/e^x+1) = e^{-x}/e^{-x}+1 = 1-1/1+e^{-x} = 1-\sigma(x)$$

Question 4b (2 pt)

Show that the derivative can be written as:

$$\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$$

Answer:

$$\frac{\partial}{\partial x}\sigma(x) = 1/e^x(1+e^{-x})^2 = 1/e^x+1+e^{-x}+1 = 1/((e^x+1)(e^{-x}+1)) = \sigma(x)(1-\sigma(x))$$

7 Q3b 2 / 2

✓ + 1 pts $$\dfrac{\mathrm{d}}{\mathrm{d}x}\sum_{i=1}^{10} (i - x)^2=-2 \left(55 - 10x \right)$$

✓ + 1 pts $$x = 5.5$$

+ 0 pts This rubric is removed

+ 0 pts WRONG

Answer:
First:

$$let: f(x) = \sum_{i=1}^{10}(1-x)^2$$

Then

$$f'(x) = -2\sum_{i=1}^{10}(i-x) = -110 + 20x = 0$$

Finally solve the equation to get

$$10x = 55, x = 5.5$$

Question 4a (2 pt)

Let $\sigma(x) = \dfrac{1}{1+e^{-x}}$.

Show that $\sigma(-x) = 1 - \sigma(x)$.

Answer:

$$\sigma(-x) = 1/(1+e^x) = 1/e^x(1/e^x+1) = e^{-x}/e^{-x}+1 = 1-1/1+e^{-x} = 1-\sigma(x)$$

Question 4b (2 pt)

Show that the derivative can be written as:

$$\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$$

Answer:

$$\frac{\partial}{\partial x}\sigma(x) = 1/e^x(1+e^{-x})^2 = 1/e^x+1+e^{-x}+1 = 1/((e^x+1)(e^{-x}+1)) = \sigma(x)(1-\sigma(x))$$

8 Q4a **0 / 2**

+ **2 pts** Correct

✓ **+ 0 pts** **Not_correct**

+ **0 pts** Click here to replace this description.

Answer:
First:

$$let: f(x) = \sum_{i=1}^{10}(1-x)^2$$

Then

$$f'(x) = -2\sum_{i=1}^{10}(i-x) = -110 + 20x = 0$$

Finally solve the equation to get

$$10x = 55, x = 5.5$$

Question 4a (2 pt)

Let $\sigma(x) = \dfrac{1}{1+e^{-x}}$.

Show that $\sigma(-x) = 1 - \sigma(x)$.

Answer:

$$\sigma(-x) = 1/(1+e^x) = 1/e^x(1/e^x+1) = e^{-x}/e^{-x}+1 = 1-1/1+e^{-x} = 1-\sigma(x)$$

Question 4b (2 pt)

Show that the derivative can be written as:

$$\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$$

Answer:

$$\frac{\partial}{\partial x}\sigma(x) = 1/e^x(1+e^{-x})^2 = 1/e^x+1+e^{-x}+1 = 1/((e^x+1)(e^{-x}+1)) = \sigma(x)(1-\sigma(x))$$

9 Q4b **2 / 2**

✓ **+ 2 pts** **Correct**

+ 0 pts WRONG

Question 5 (2 pt)

Consider the following scenario:

Only 1% of 40-year-old women who participate in a routine mammography test have breast cancer. 80% of women who have breast cancer will test positive, but 9.6% of women who don't have breast cancer will also get positive tests.

Suppose we know that a woman of this age tested positive in a routine screening. What is the probability that she actually has breast cancer?

Hint: Use Bayes' rule

Answer:

$$p(cancer|positive) = (0.01 * 0.08)/((0.01 * 0.8) + (0.99 * 0.096)) = 0.07764$$

Question 6

Consider (once again) a sample of size n drawn at random with replacement from a population in which a proportion p of the individuals are called successes.

Let S be the random variable that denotes the number of successes in our sample. (As stated above, S follows the binomial distribution.) Then, the probability that the number of successes in our sample is **at most** s (where $0 \leq s \leq n$) is

$$P(S \leq s) = P(S = 0) + P(S = 1) + \ldots + P(S = s) = \sum_{k=0}^{s} \binom{n}{k} p^k (1-p)^{n-k}$$

We obtain this by summing the probability that the number of successes is exactly k, for each value of $k = 0, 1, 2, \ldots, s$.

10 Q5 **1 / 2**

+ **1 pts** $$ P(\text{Has Cancer} | \text{Tested Positive}) = \dfrac{P(\text{Tested Positive} | \text{Has Cancer}) \times P(\text{Has Cancer})}{P(\text{Tested Positive})} $$

✓ **+ 1 pts** **$$ P(\text{Has Cancer} | \text{Tested Positive}) = \dfrac{0.80 \times 0.01}{0.80 \times 0.01 + 0.096 \times (1 - 0.01)} \approx 0.078$$**

+ **0 pts** WRONG

Question 6a (2pt)

Please fill in the function `prob_at_most` which takes n, p, and s and returns $P(S \leq s)$ as defined above. If the inputs are invalid: for instance, if p > 1 or s > n then return 0."

Hint: One way to compute the binomial coefficients is to use SciPy module, which is a collection of Python-based software for math, probability, statistics, science, and enginnering. Feel free to use scipy.special.comb (https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.comb.html#scipy.special.comb)**

In [21]:

```
from scipy import special

def prob_at_most(n, p, s):
    """
    returns the probability of S <= s
    Input n: sample size; p : proportion; s: number of successes at most
    """
    # BEGIN YOUR CODE
    # ------------------------
    if ((p > 1) or s > n):
        return 0
    else:
        result = 0
        for i in range(s+1):
            result += special.comb(n, i) * (p**i) * ((1-p)**(n-i))
        return result
    # ------------------------
    # END YOUR CODE
```

In [22]:

```
ok.grade("q6a");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 3
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 6b (1pt)

In an election, supporters of Candidate C are in a minority. Only 45% of the voters in the population favor the candidate.

Suppose a survey organization takes a sample of 200 voters at random with replacement from this population. Use `prob_at_most` to write an expression that evaluates to the chance that a majority (more than half) of the sampled voters favor Candidate C.

11 Q6a **2 / 2**

✓ **+ 2 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

Question 6a (2pt)

Please fill in the function `prob_at_most` which takes n, p, and s and returns $P(S \leq s)$ as defined above. If the inputs are invalid: for instance, if p > 1 or s > n then return 0."

Hint: One way to compute the binomial coefficients is to use SciPy module, which is a collection of Python-based software for math, probability, statistics, science, and enginnering. Feel free to use scipy.special.comb (https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.comb.html#scipy.special.comb)**

In [21]:

```
from scipy import special

def prob_at_most(n, p, s):
    """
    returns the probability of S <= s
    Input n: sample size; p : proportion; s: number of successes at most
    """
    # BEGIN YOUR CODE
    # ------------------------
    if ((p > 1) or s > n):
        return 0
    else:
        result = 0
        for i in range(s+1):
            result += special.comb(n, i) * (p**i) * ((1-p)**(n-i))
        return result
    # ------------------------
    # END YOUR CODE
```

In [22]:

```
ok.grade("q6a");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 3
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 6b (1pt)

In an election, supporters of Candidate C are in a minority. Only 45% of the voters in the population favor the candidate.

Suppose a survey organization takes a sample of 200 voters at random with replacement from this population. Use `prob_at_most` to write an expression that evaluates to the chance that a majority (more than half) of the sampled voters favor Candidate C.

In [23]:

```
# BEGIN YOUR CODE
# ------------------------
p_majority = prob_at_most(200, 0.45, 101)
# ------------------------
# END YOUR CODE
```

In [24]:

```
ok.grade("q6b");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 6c (1pt)

Suppose each of five survey organizations takes a sample of voters at random with replacement from the population of voters in Part **b**, independently of the samples drawn by the other organizations.

- Three of the organizations use a sample size of 200
- One organization uses a sample size of 300
- One organization uses a sample size of 400

Write an expression that evaluates to the chance that in at least one of the five samples the majority of voters favor Candidate C. You can use any quantity or function defined earlier in this exercise.

In [25]:

```
# BEGIN YOUR CODE
# ------------------------
prob_6c = 1 - [x > 0.5 for x in [prob_at_most(i, 0.45, int(i/2)+1) for i in [200, 200, 200, 300
, 400]]].count(True)/5
# ------------------------
# END YOUR CODE
```

In [26]:

```
ok.grade("q6c");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

12 Q6b **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

In [23]:

```
# BEGIN YOUR CODE
# ------------------------
p_majority = prob_at_most(200, 0.45, 101)
# ------------------------
# END YOUR CODE
```

In [24]:

```
ok.grade("q6b");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 6c (1pt)

Suppose each of five survey organizations takes a sample of voters at random with replacement from the population of voters in Part **b**, independently of the samples drawn by the other organizations.

- Three of the organizations use a sample size of 200
- One organization uses a sample size of 300
- One organization uses a sample size of 400

Write an expression that evaluates to the chance that in at least one of the five samples the majority of voters favor Candidate C. You can use any quantity or function defined earlier in this exercise.

In [25]:

```
# BEGIN YOUR CODE
# ------------------------
prob_6c = 1 - [x > 0.5 for x in [prob_at_most(i, 0.45, int(i/2)+1) for i in [200, 200, 200, 300
, 400]]].count(True)/5
# ------------------------
# END YOUR CODE
```

In [26]:

```
ok.grade("q6c");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

13 Q6c **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

Question 7a (1pt)

For your convenience, the results of the vote in the four pivotal states is repeated below:

State	Trump	Clinton	Total Voters
Florida	49.02	47.82	9,419,886
Michigan	47.50	47.27	4,799,284
Pennsylvania	48.18	47.46	6,165,478
Wisconsin	47.22	46.45	2,976,150

Using the table above, write a function `draw_state_sample(N, state)` that returns a sample with replacement of N voters from the given state. Your result should be returned as a list, where the first element is the number of Trump votes, the second element is the number of Clinton votes, and the third is the number of Other votes. For example, `draw_state_sample(1500, "florida")` could return `[727, 692, 81]`. You may assume that the state name is given in all lower case.

You might find `np.random.multinomial` useful.

In [40]:

```
def draw_state_sample(N, state):
    # BEGIN YOUR CODE
    # ------------------------
    if state == 'florida':
        a = 0.4902
        b = 0.4782
    elif state == 'michigan':
        a = 0.4705
        b= 0.4727
    elif state == 'pennsylvania':
        a = 0.4818
        b = 0.4746
    elif state == 'wisconsin':
        a = 0.4722
        b = 0.4645
    c = 1-a-b
    return np.random.multinomial(n=N, pvals=[a,b,c])
    # ------------------------
    # END YOUR CODE
```

In [41]:

```
ok.grade("q7a");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed
```

14 Q7a **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ **0 pts** Failed to pass all unit tests

Question 7b (1pt)

Now, create a function `trump_advantage` that takes in a sample of votes (like the one returned by `draw_state_sample`) and returns the difference in the proportion of votes between Trump and Clinton. For example `trump_advantage([100, 60, 40])` would return `0.2`, since Trump had 50% of the votes in this sample and Clinton had 30%.

In [29]:

```
def trump_advantage(voter_sample):
    # BEGIN YOUR CODE
    # ------------------------
    difference = (voter_sample[0] - voter_sample[1]) / sum(voter_sample)
    return difference
    # ------------------------
    # END YOUR CODE
```

In [30]:

```
ok.grade("q7b");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 7c (1pt)

Simulate Trump's advantage across 100,000 samples of 1500 voters for the state of Pennsylvania and store the results of each simulation in a list called `simulations`.

That is, `simulations[i]` should be Trump's proportion advantage for the `i+1`th simple random sample.

In [48]:

```
# BEGIN YOUR CODE
# ------------------------
simulations = [trump_advantage(draw_state_sample(1500, 'pennsylvania'))
               for i in range(100000)]
# ------------------------
# END YOUR CODE
```

15 Q7b **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

Question 7b (1pt)

Now, create a function `trump_advantage` that takes in a sample of votes (like the one returned by `draw_state_sample`) and returns the difference in the proportion of votes between Trump and Clinton. For example `trump_advantage([100, 60, 40])` would return `0.2`, since Trump had 50% of the votes in this sample and Clinton had 30%.

In [29]:

```
def trump_advantage(voter_sample):
    # BEGIN YOUR CODE
    # ------------------------
    difference = (voter_sample[0] - voter_sample[1]) / sum(voter_sample)
    return difference
    # ------------------------
    # END YOUR CODE
```

In [30]:

```
ok.grade("q7b");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 7c (1pt)

Simulate Trump's advantage across 100,000 samples of 1500 voters for the state of Pennsylvania and store the results of each simulation in a list called `simulations`.

That is, `simulations[i]` should be Trump's proportion advantage for the `i+1`th simple random sample.

In [48]:

```
# BEGIN YOUR CODE
# ------------------------
simulations = [trump_advantage(draw_state_sample(1500, 'pennsylvania'))
               for i in range(100000)]
# ------------------------
# END YOUR CODE
```

In [49]:

```
ok.grade("q7c");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 7d (1pt)

Now write a function `trump_wins(N)` that creates a sample of N voters for each of the four crucial states (Florida, Michigan, Pennsylvania, and Wisconsin) and returns 1 if Trump is predicted to win based on these samples and 0 if Trump is predicted to lose.

Recall that for Trump to win the election, he must either:

- Win the state of Florida and 1 or more other states
- Win Michigan, Pennsylvania, and Wisconsin

In [50]:

```
def trump_wins(N):
    # BEGIN YOUR CODE
    # ------------------------
    florida = trump_advantage(draw_state_sample(N, 'florida'))
    michigan = trump_advantage(draw_state_sample(N, 'michigan'))
    pennsylvania = trump_advantage(draw_state_sample(N, 'pennsylvania'))
    wisconsin = trump_advantage(draw_state_sample(N, 'wisconsin'))

    if (florida>0 and (michigan>0 or pennsylvania>0 or wisconsin>0)):
        return 1
    elif (michigan>0 and pennsylvania>0 and wisconsin>0):
        return 1
    else:
        return 0
    # ------------------------
    # END YOUR CODE
```

In [51]:

```
ok.grade("q7d");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

16 Q7c **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ **0 pts** Failed to pass all unit tests

In [49]:

```
ok.grade("q7c");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 7d (1pt)

Now write a function `trump_wins(N)` that creates a sample of N voters for each of the four crucial states (Florida, Michigan, Pennsylvania, and Wisconsin) and returns 1 if Trump is predicted to win based on these samples and 0 if Trump is predicted to lose.

Recall that for Trump to win the election, he must either:

- Win the state of Florida and 1 or more other states
- Win Michigan, Pennsylvania, and Wisconsin

In [50]:

```
def trump_wins(N):
    # BEGIN YOUR CODE
    # ------------------------
    florida = trump_advantage(draw_state_sample(N, 'florida'))
    michigan = trump_advantage(draw_state_sample(N, 'michigan'))
    pennsylvania = trump_advantage(draw_state_sample(N, 'pennsylvania'))
    wisconsin = trump_advantage(draw_state_sample(N, 'wisconsin'))

    if (florida>0 and (michigan>0 or pennsylvania>0 or wisconsin>0)):
        return 1
    elif (michigan>0 and pennsylvania>0 and wisconsin>0):
        return 1
    else:
        return 0
    # ------------------------
    # END YOUR CODE
```

In [51]:

```
ok.grade("q7d");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

17 Q7d **1 / 1**

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests

Question 7e

If we repeat 100,000 simulations of the election, i.e. we call `trump_wins(1500)` 100,000 times, what proportion of these simulations predict a Trump victory? Give your answer as `proportion_trump`.

This number represents the percent chance that a given sample will correctly predict Trump's victory *even if the sample was collected with absolutely no bias*.

In [52]:

```
# BEGIN YOUR CODE
# ------------------------
proportion_trump = sum([trump_wins(1500) for i in range(100000)])/100000
# ------------------------
# END YOUR CODE
proportion_trump
```

Out[52]:

```
0.67878
```

In [53]:

```
ok.grade("q7e");
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

_____________________________________________________________________
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed
```

Congratulations! You have completed HW1.

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output.,

Please save before submitting!

For your convenience, you can run this cell to run all the tests at once!

In [37]:

```
import os
from IPython.utils import io

print('{:5}|Wt{:6}|Wt{:6}'.format('Q', 'Passed', 'Failed'))
print('-----------------------')
for q in sorted(os.listdir("tests")):
    if q.startswith('q') and len(q) <= 10:
        with io.capture_output() as captured:
            score = ok.grade(q[:-3]);
        print('{:5}|Wt{:6}|Wt{:6}'.format(q[:-3], score['passed'], score['failed']))
```

```
Q    |  Passed| Failed
----------------------
q1   |       4|      0
q2a  |       2|      0
q2b  |       2|      0
q2c  |       2|      0
q2d  |       2|      0
q6a  |       3|      0
q6b  |       1|      0
q6c  |       1|      0
q7a  |       2|      0
q7b  |       1|      0
q7c  |       1|      0
q7d  |       1|      0
q7e  |       2|      0
```

Please generate pdf as follows and submit it to Gradescope.

File > Print Preview > Print > Save as pdf

18 Q7e 1 / 1

✓ **+ 1 pts** **Passed all unit tests**

+ 0 pts Failed to pass all unit tests