Alexander Beckwith

Math 855 - Prob w/ Applications

HW2

In [ ]:
```
# 3rd party imports
from matplotlib.pyplot import subplots as plt

# local imports
from samplespace import DiceSampleSpace, CoinSampleSpace
```

Problems from Chapter 1: (page 30): 46, 48, 49, 50, 52, 56, 59, 60, 74 and 77. Total points 30"=10*3".

1. Urn A has three red balls and two white balls, and urn B has two red balls and five white balls. A fair coin is tossed. If it lands heads up, a ball is drawn from urn A; otherwise, a ball is drawn from urn B.

a. What is the probability that a red ball is drawn?

b. If a red ball is drawn, what is the probability that the coin landed heads up?

In [ ]:
```
class Urn:
    # this function sums the red and white balls to form the urn's total_balls
    def _set_total_balls(self):
        self.total_balls = self.red_balls + self.white_balls

    # this function divides the white balls by the total balls
    # to get the prob of picking white from the urn
    def _set_white_prob(self):
        self.white_prob = self.white_balls / self.total_balls

    # this function divides the red balls by the total balls
    # to get the prob of picking red from the urn
    def _set_red_prob(self):
        self.red_prob = self.red_balls/self.total_balls

    # this function instantiates the urn object with the count of red and white
    # it then calls the set functions to perform the above calculations
    def __init__(self, white_balls, red_balls):
        self.white_balls = white_balls
        self.red_balls = red_balls
        self._set_total_balls()
        self._set_red_prob()
        self._set_white_prob()

    # this function produces the object's response when called by the print fu
    def __repr__(self):
        return f"I am an urn with {self.white_balls} white balls and {self.red_

print("We'll create two Urn objects to represent urn A and B.")
urn_a = Urn(2, 3)
```

```
urn_b = Urn(5, 2)

print("Urn A:", urn_a)
print("Urn B:", urn_b, "\n")

# an urn is chosen by coin flip
# therefore, either urn being chosen has a probability:
coin_prob = 0.5
print(f"Flipping a coin means each urn has a probability of being chosen of {co

print("This is conditional probability, so we need to show: P( A | B ) = P( A ∩
# we must consider the prob of a red ball being chosen
# along with the prob of each urn
prob_urn_a_red_ball = round(coin_prob * urn_a.red_prob, 4)
prob_urn_b_red_ball = round(coin_prob * urn_b.red_prob, 4)
print(f"The prob of urn A being chosen and a red ball being chosen from urn A i
print(f"The prob of urn B being chosen and a red ball being chosen from urn B i

prob_red_ball = prob_urn_a_red_ball + prob_urn_b_red_ball
print(f"46a | the total probability of choosing a red ball from either is the s

# a head is associated with pulling from urn a,
# so if a red ball was chosen,
prob_urn_a_given_red = prob_urn_a_red_ball / prob_red_ball
print("\tP( Head | Red ) = P ( Head ∩ Red ) / P( Red )")
print(f"46b | the probability of a head having been flipped when a red ball was
```

```
We'll create two Urn objects to represent urn A and B.
Urn A: I am an urn with 2 white balls and 3 red balls.
Urn B: I am an urn with 5 white balls and 2 red balls.

Flipping a coin means each urn has a probability of being chosen of 0.5.
This is conditional probability, so we need to show: P( A | B ) = P( A ∩ B) /
P( B )

The prob of urn A being chosen and a red ball being chosen from urn A is 0.3
The prob of urn B being chosen and a red ball being chosen from urn B is 0.142
9

46a | the total probability of choosing a red ball from either is the sum, 0.4
429

        P( Head | Red ) = P ( Head ∩ Red ) / P( Red )
46b | the probability of a head having been flipped when a red ball was chosen
is 0.6774
```

1. An urn contains three red and two white balls. A ball is drawn, and then it and another ball of the same color are placed back in the urn. Finally, a second ball is drawn.

a. What is the probability that the second ball drawn is white?

b. If the second ball drawn is white, what is the probability that the first ball drawn was red?

In [ ]:
```
print("We'll create three urn objects, one to represent the first choice and th
urn1 = Urn(2, 3)
urn1r = Urn(2, 4)
urn1w = Urn(3, 3)
print("\tUrn Start:", urn1)
```

```
print("\tUrn if first red:", urn1r)
print("\tUrn if first white:", urn1w, "\n")

print("The prob of the second ball drawn being white is the sum of the prob of
choice_ww = round(urn1.white_prob * urn1w.white_prob, 4)
choice_rw = round(urn1.red_prob * urn1r.white_prob, 4)

print(f"\tThe prob of picking white after first picking white is {choice_ww}")
print(f"\tThe prob of picking white after first picking red is {choice_rw}")

prob_2w = round(choice_rw + choice_ww, 4)
print(f"48a | the prob of the second ball being white is the sum, {prob_2w}\n")

# the prob of the first being red when second is white
# is the prob of red given white out of all cases when white is second

prob_1r2w = round(choice_rw / prob_2w, 4)
print("\tP( Red1 | White2 ) = P ( Red1 ∩ White2 ) / P( White2 )")
print(f"48b | the prob of first being red when 2nd is white is {prob_1r2w}")
```

```
We'll create three urn objects, one to represent the first choice and the othe
rs to represent the options of the second round.
        Urn Start: I am an urn with 2 white balls and 3 red balls.
        Urn if first red: I am an urn with 2 white balls and 4 red balls.
        Urn if first white: I am an urn with 3 white balls and 3 red balls.

The prob of the second ball drawn being white is the sum of the prob of those
cases.
        The prob of picking white after first picking white is 0.2
        The prob of picking white after first picking red is 0.2
48a | the prob of the second ball being white is the sum, 0.4

        P( Red1 | White2 ) = P ( Red1 ∩ White2 ) / P( White2 )
48b | the prob of first being red when 2nd is white is 0.5
```

1. A fair coin is tossed three times.

a. What is the probability of two or more heads given that there was at least one head?

b. What is the probability given that there was at least one tail?

In [ ]:

```
css = CoinSampleSpace(3)
print(f"A fair coin tossed three times gives the sample space: {css.ss}, which
print("To find P( Heads >= 2 | Heads >= 1), we need to find P( Heads >= 2 ∩ Hea
print("To find P( Heads >= 2 | Tails >= 1), we need to find P( Heads >= 2 ∩ Tai

options1h = [x for x in css.ss if "h" in x]
options1t = [x for x in css.ss if "t" in x]
print(f"There are {len(options1h)} options with >= 1 head: {options1h}")
print(f"There are {len(options1t)} options with >= 1 tail: {options1t}\n")

options2h_in1h = [x for x in options1h if x.count("h") > 1]
options2h_in1t = [x for x in options1t if x.count("h") > 1]
print(f"There are {len(options2h_in1h)} options with > 1 head and >= 1 head: {o
print(f"There are {len(options2h_in1t)} options with > 1 head and >= 1 tail: {o

n_1h = len(options1h)
n_1t = len(options1t)
```

```
n_2h_in1h = len(options2h_in1h)
n_2h_in1t = len(options2h_in1t)

prob_2h_given_1h = n_2h_in1h / n_1h
prob_2h_given_1t = n_2h_in1t / n_1t

print(f"49a | the prob of 2 heads given there was >= 1 head is {n_2h_in1h} / {r
print(f"49b | the prob of 2 heads given there was >= 1 tail is {n_2h_in1t} / {r
```

```
A fair coin tossed three times gives the sample space: ['ttt', 'tth', 'tht',
'thh', 'htt', 'hth', 'hht', 'hhh'], which has 8 options.
To find P( Heads >= 2 | Heads >= 1), we need to find P( Heads >= 2 ∩ Heads >=
1) and P( Heads >= 1)
To find P( Heads >= 2 | Tails >= 1), we need to find P( Heads >= 2 ∩ Tails >=
1) and P( Tails >= 1)

There are 7 options with >= 1 head: ['tth', 'tht', 'thh', 'htt', 'hth', 'hht',
'hhh']
There are 7 options with >= 1 tail: ['ttt', 'tth', 'tht', 'thh', 'htt', 'hth',
'hht']

There are 4 options with > 1 head and >= 1 head: ['tth', 'tht', 'thh', 'htt',
'hth', 'hht', 'hhh']
There are 3 options with > 1 head and >= 1 tail: ['ttt', 'tth', 'tht', 'thh',
'htt', 'hth', 'hht']

49a | the prob of 2 heads given there was >= 1 head is 4 / 7, or 0.5714
49b | the prob of 2 heads given there was >= 1 tail is 3 / 7, or 0.4286
```

1. Two dice are rolled, and the sum of the face values is six. What is the probability that at least one of the dice came up a three?

In [ ]:
```
dss = DiceSampleSpace(2)
print("Two six-sided dice rolled produces the sample space of size", len(dss.ss

sum_six = [x for x in dss.ss if int(x[0]) + int(x[-1]) == 6]
print(f"There are {len(sum_six)} options in that space which sum to six: {sum_s

sum_six_has_three = [y for y in sum_six if "3" in y]
print(f"And only {len(sum_six_has_three)} option that sums to six and the roll

prob_six_has_three = len(sum_six_has_three) / len(sum_six)

print(f"50 | the prob of the roll having a three when the sum is six is {len(su
```

```
Two six-sided dice rolled produces the sample space of size 36
There are 5 options in that space which sum to six: ['15', '24', '33', '42',
'51']
And only 1 option that sums to six and the roll contains a three: ['33']

50 | the prob of the roll having a three when the sum is six is 1 / 5, or 0.2
```

1. Suppose that 5 cards are dealt from a 52-card deck and the first one is a king. What is the probability of at least one more king?

In [ ]:
```
print("To find the probability that at least one more king will be drawn,")
print("we'll want to find the prob of that not happening, aka the complement.\r
```

```
n_deck = deck_post1 = 51
n_kings = kings_post1 = 3
print(f"After the first king is drawn, there are {deck_post1} cards in the deck
draws_left = 4

print("Drawing cards...")
total_prob_not_king = 1
while draws_left > 0:
    n_not_kings = n_deck - n_kings
    p_not_king = n_not_kings / n_deck
    total_prob_not_king *= p_not_king
    draws_left -= 1
    n_deck -= 1
    print(f"\tWith {n_not_kings} non-kings left, the prob of this draw not beir
    print(f"\tThere are {draws_left} draws left and the cumulative product so f
print()

prob_one_plus_king = round(1 - total_prob_not_king, 4)
print(f"52 | The prob of at least one king in the next four draws is the comple
```

```
To find the probability that at least one more king will be drawn,
we'll want to find the prob of that not happening, aka the complement.

After the first king is drawn, there are 51 cards in the deck and 3 of them ar
e kings.

Drawing cards...
        With 48 non-kings left, the prob of this draw not being a king is 48 /
50 = 0.9412
        There are 3 draws left and the cumulative product so far is 0.9412
        With 47 non-kings left, the prob of this draw not being a king is 47 /
49 = 0.94
        There are 2 draws left and the cumulative product so far is 0.8847
        With 46 non-kings left, the prob of this draw not being a king is 46 /
48 = 0.9388
        There are 1 draws left and the cumulative product so far is 0.8305
        With 45 non-kings left, the prob of this draw not being a king is 45 /
47 = 0.9375
        There are 0 draws left and the cumulative product so far is 0.7786

52 | The prob of at least one king in the next four draws is the complement of
the prob there are no kings drawn, 0.2214
```

1. A couple has two children. What is the probability that both are girls given that the oldest is a girl? What is the probability that both are girls given that one of them is a girl?

In [ ]:
```
print("Let's assume that the prob of either a boy or girl is 0.5")
sex_ss = CoinSampleSpace(2, response_set="bg")
print("The sample space of possible children outcomes is", sex_ss.ss, "\n")
print("All outcomes have a probability of 0.5^2 = 0.25")

one_g = [x for x in sex_ss.ss if "g" in x]
print(f"There are {len(one_g)} cases where at least one child is a girl: {one_g

both_g_when_one_g = [y for y in one_g if y.count("g") == 2]
print(f"There is {len(both_g_when_one_g)} case where both child are girls: {bot
```

```
print("56a | The sex of existing children doesn't affect the prob of subsequent
print(f"56b | P( gg | >= 1g ) =  P( gg ∩ >= 1g ) / P( >= 1g ) = {0.5 ** 2} / {3
```

```
Let's assume that the prob of either a boy or girl is 0.5
The sample space of possible children outcomes is ['bb', 'bg', 'gb', 'gg']

All outcomes have a probability of 0.5^2 = 0.25
There are 3 cases where at least one child is a girl: ['bg', 'gb', 'gg'], P( >
= 1g ) = 0.75
There is 1 case where both child are girls: ['gg']

56a | The sex of existing children doesn't affect the prob of subsequent child
ren's sex, therefore P( youngest = g ) = 0.5
56b | P( gg | >= 1g ) =  P( gg ∩ >= 1g ) / P( >= 1g ) = 0.25 / 0.75 = 0.3333
```

1. A box has three coins. One has two heads, one has two tails, and the other is a fair coin with one head and one tail. A coin is chosen at random, is flipped, and comes up heads.

a. What is the probability that the coin chosen is the two-headed coin?

b. What is the probability that if it is thrown another time it will come up heads?

c. Answer part (a) again, supposing that the coin is thrown a second time and comes up heads again.

In [ ]:
```
coins = ["(hh)", "(ht)", "(tt)"]
n_coins = len(coins)
coins_w_heads = [x for x in coins if "h" in x]
n_coins_w_heads = len(coins_w_heads)

print(f"There are {n_coins} coins which will be represented this way: {coins}")
print(f"\tThe prob of part a can be represented as P( (hh) | Head ) = P( (hh) ∩

prob_each = 1 / n_coins
prob_each_head = [1, 0.5, 0]
prob_each_prob_head = [round(prob_each * x, 4) for x in prob_each_head]
dict_each_head_prob = dict(zip(coins, prob_each_head))
dict_each_prob_each_head = dict(zip(coins, prob_each_prob_head))
print(f"The prob of any (coin) being chosen is {round(prob_each, 4)}")
print(f"The prob for each flipping a head is {dict_each_head_prob}")
print(f"The prob for picking each and also flipping a head is the product, {dic

total_head_prob = sum(list(dict_each_prob_each_head.values()))
print(f"The total probability of a head being flipped is the sum, P( Head ) = {
p_pick_hh_h = dict_each_prob_each_head["(hh)"]
prob_a = round(p_pick_hh_h / total_head_prob,4)
print(f"59a | P( (hh) | Head ) = P( (hh) ∩ Head ) / P( Head ) = {p_pick_hh_h} /

h_flip2 = [round(prob_each_head[i] * prob_each_prob_head[i], 4) for i in range(
dict_h_flip2 = dict(zip(coins, h_flip2))
p_hh = round(sum(h_flip2), 4)
print("To find the prob of flipping a second head, we mult the prob of each by
print(f"59b | P( HeadHead ) = {p_hh}\n")

p_hh_hh = dict_h_flip2["(hh)"]
prob_c = round(p_hh_hh / p_hh, 4)
print(f"59c | P( (hh) | HeadHead ) = P( (hh) ∩ HeadHead ) / P( HeadHead ) = {p_
```

```
There are 3 coins which will be represented this way: ['(hh)', '(ht)', '(tt)']
        The prob of part a can be represented as P( (hh) | Head ) = P( (hh) ∩
Head ) / P( Head )

The prob of any (coin) being chosen is 0.3333
The prob for each flipping a head is {'(hh)': 1, '(ht)': 0.5, '(tt)': 0}
The prob for picking each and also flipping a head is the product, {'(hh)': 0.
3333, '(ht)': 0.1667, '(tt)': 0.0}
The total probability of a head being flipped is the sum, P( Head ) = 0.5

59a | P( (hh) | Head ) = P( (hh) ∩ Head ) / P( Head ) = 0.3333 / 0.5 = 0.6666

To find the prob of flipping a second head, we mult the prob of each by its pr
ob of head again: {'(hh)': 0.3333, '(ht)': 0.0833, '(tt)': 0.0}
59b | P( HeadHead ) = 0.4166

59c | P( (hh) | HeadHead ) = P( (hh) ∩ HeadHead ) / P( HeadHead ) = 0.3333 /
0.4166 = 0.8
```

1. A factory runs three shifts. In a given day, 1% of the items produced by the first shift are defective, 2% of the second shift's items are defective, and 5% of the third shift's items are defective. If the shifts all have the same productivity, what percentage of the items produced in a day are defective? If an item is defective, what is the probability that it was produced by the third shift?

In [ ]:
```python
def_ratios = {
"shift_1": 0.01,
"shift_2": 0.02,
"shift_3": 0.05,
}
sum_ratios = sum(list(def_ratios.values()))
print(f"The ratio of defective items produced by each shift is {def_ratios}\n")

total_def_perc = round((sum_ratios / 3) * 100, 4)
print("60a | The percentage of defective items produced is the evenly weighted
print(f"\t({sum_ratios} / 3) * 100 = {total_def_perc}%\n")

print("Since there is no difference in productivity between shifts,")
print(f"the total productivity can be approximated as the sum of the ratios, {s

s3_def = def_ratios["shift_3"]
p_s3_def = round(s3_def / sum_ratios, 4)
print(f"60b | P( Shift3 | Defective) = P( Shift3 ∩ Defective) / P( Defective )
```

```
The ratio of defective items produced by each shift is {'shift_1': 0.01, 'shif
t_2': 0.02, 'shift_3': 0.05}

60a | The percentage of defective items produced is the evenly weighted averag
e of defective ratios per shift:
        (0.08 / 3) * 100 = 2.6667%

Since there is no difference in productivity between shifts,
the total productivity can be approximated as the sum of the ratios, 0.08

60b | P( Shift3 | Defective) = P( Shift3 ∩ Defective) / P( Defective ) = 0.05
/ 0.08 = 0.625
```

1. What is the probability that the following system works if each unit fails independently with probability p (see Figure 1.5)?

In [ ]:
```
print("The system works if one of the paths passes.")
print("Therefore, we need to find the probability that none of the paths work.'
print("\tProb of failure for paths w/ two units is the prob that both succeed,
p_1path_f = "p"
p_1path_s = f"(1 - {p_1path_f})"
p_2path_s = "(1 - p) * (1 - p)"
p_2path_f = f"[1 - {p_2path_s}]"
print(f"\tProb of both not failing is {p_2path_s}, so the prob of either failir

p_all_f = f"{p_2path_f} * {p_1path_f} * {p_2path_f}"
print("The probability that all fail is", p_all_f)
print(f"Therefore, the prob that at least one succeeds is [1 - {p_all_f}]")
```

```
The system works if one of the paths passes.
Therefore, we need to find the probability that none of the paths work.
        Prob of failure for paths w/ two units is the prob that both succeed,
or that both don't fail
        Prob of both not failing is (1 - p) * (1 - p), so the prob of either f
ailing is [1 - (1 - p) * (1 - p)]

The probability that all fail is [1 - (1 - p) * (1 - p)] * p * [1 - (1 - p) *
(1 - p)]
Therefore, the prob that at least one succeeds is [1 - [1 - (1 - p) * (1 - p)]
* p * [1 - (1 - p) * (1 - p)]]
```

1. A player throws darts at a target. On each trial, independently of the other trials, he hits the bull's-eye with probability .05. How many times should he throw so that his probability of hitting the bull's-eye at least once is .5?

In [ ]:
```
prob = "p"
comp = "(1 - p)"
print("The prob of at least one bullseye is the complement of the prob of there
p_no_bull = f"{comp}^n_throws"
print(f"The prob of no bullseyes is {p_no_bull}")
print(f"so the prob of at least one bullseye is 1 - {p_no_bull}\n")

print("Here's the output of a Python function that performs this calculation:")

# Here's the function
def prob_bullseye(n_throws: int, p_one_bullseye: float = 0.05):
    p_not_bullseye = 1 - p_one_bullseye
    p_no_bullseyes = p_not_bullseye ** n_throws
    p_one_plus_bullseye = 1 - p_no_bullseyes
    return p_one_plus_bullseye

# Starting lists to capture the function's output for viz
x = []
y = []

for i in range(1, 21):
    p = round(prob_bullseye(i), 4)
    x.append(i)                    # this is where the "x" and "y" values are
    y.append(p)
    print(f"\tthe prob of hitting at least one bullseye in {i} throws is {p}")
```

```
    if p >= 0.5:
        break
print()
print(f"Our pal needs to throw {i} darts to have a > 0.5 chance of having hit >
fig, ax = plt(1, 1)
ax.set_xlabel("n_throws")
ax.set_ylabel("prob of >= one bullseye")
ax.plot(x, y)
```

```
The prob of at least one bullseye is the complement of the prob of there being
no bullseyes thrown.
The prob of no bullseyes is (1 - p)^n_throws
so the prob of at least one bullseye is 1 - (1 - p)^n_throws

Here's the output of a Python function that performs this calculation:
        the prob of hitting at least one bullseye in 1 throws is 0.05
        the prob of hitting at least one bullseye in 2 throws is 0.0975
        the prob of hitting at least one bullseye in 3 throws is 0.1426
        the prob of hitting at least one bullseye in 4 throws is 0.1855
        the prob of hitting at least one bullseye in 5 throws is 0.2262
        the prob of hitting at least one bullseye in 6 throws is 0.2649
        the prob of hitting at least one bullseye in 7 throws is 0.3017
        the prob of hitting at least one bullseye in 8 throws is 0.3366
        the prob of hitting at least one bullseye in 9 throws is 0.3698
        the prob of hitting at least one bullseye in 10 throws is 0.4013
        the prob of hitting at least one bullseye in 11 throws is 0.4312
        the prob of hitting at least one bullseye in 12 throws is 0.4596
        the prob of hitting at least one bullseye in 13 throws is 0.4867
        the prob of hitting at least one bullseye in 14 throws is 0.5123

Our pal needs to throw 14 darts to have a > 0.5 chance of having hit >= one bu
llseye
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc64b59acb0>]
```