

Probability & Statistics Project

Hana Ali Rashid, hr05940
Tasmiya Malik, Student ID
Ifrah Ilyas, Student ID

April 25, 2021

Q1: Random Walk

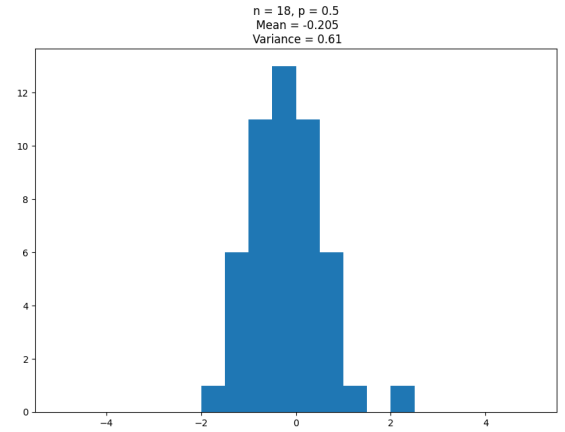
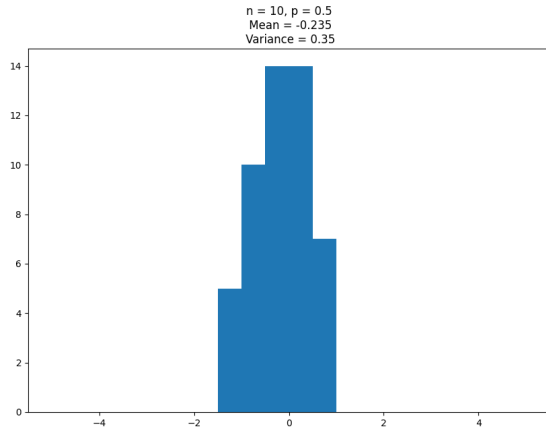
1.1

Function implementation in Python:

```
1 def get_updated_position(n,p):
2     pos = 0 #position
3     for _ in range(n):
4         rand = random.randint(1,100) #generating a random number in the range 1 to 100
5         if rand < p*100:
6             pos += 1 #move one step right
7         else:
8             pos -= 1 #move one step left
9     return pos #return final position
```

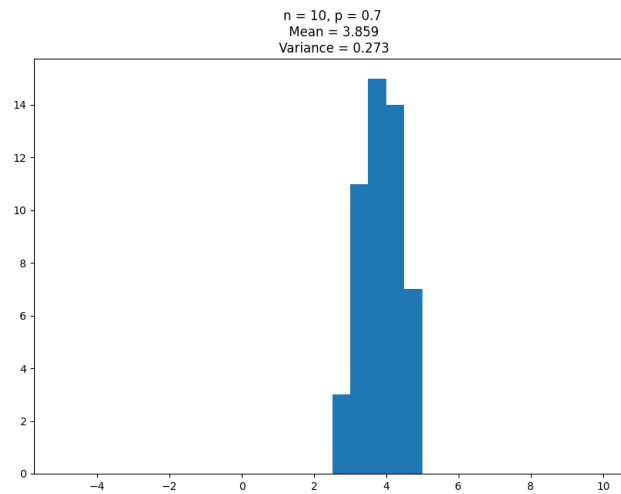
Calling the function for several iterations to get multiple expected values:

```
1 def main_11():
2     '''Calculating and plotting expected outcomes for various combinations of n and p'''
3     p = 0.7
4     n = 10
5     expected = []
6     for j in range(50): #expected values for each (n,p) for 50 iterations
7         outcomes = []
8         for i in range(25):
9             outcomes.append(get_updated_position(n,p))
10            expected.append(sum(outcomes)/25) #appending the expected (average) value for each(n,p)
11        #plotting and showing a histogram of calculated expected values
12        fig, ax = plt.subplots(figsize=(10, 7))
13        ax.hist(expected, bins = range(-5,10))
14        plt.title('n = '+str(n)+', p = '+str(p)+'\n Mean = '+str(round(statistics.mean(expected),3))+'\n Variance = '+str(round(statistics.variance(expected),3)))
15        plt.savefig("Q1_histograms/q1"+'_n = '+str(n)+'_ p = '+str(p)+'.png')
16        plt.show()
```



(a) No. of steps taken = 10 with probability of moving a step right = 0.5.

(b) No. of steps taken = 18 with probability of moving a step right = 0.5.



(c) No. of steps taken = 10 with probability of moving a step right = 0.7.

Histograms produced by the above code for expected final positions of objects against frequency, where n is the number of steps taken and p is the probability of the object moving one step to the right.

Histograms (a) and (b) have the same value of p and varying number of steps n .

Both histograms appear to follow a normal distribution, with (a) having a mean of -0.235 and a variance of 0.35 and (b) having a mean of -0.205 and a variance of 0.61 . Increasing the number of steps has not had a significant effect on the mean but did increase the variance.

Histograms (a) and (c) have the same number of steps n and varying values of p .

Histogram (c) also appears to follow a normal distribution, having a mean of 3.859 and a variance of 0.273 . Increasing p appears to have affected the mean of the distribution but not the variance as such.

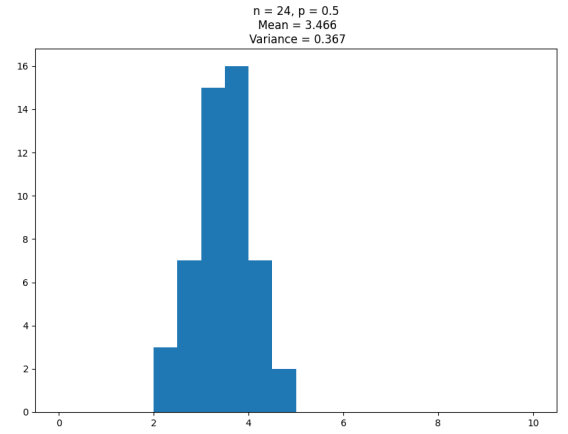
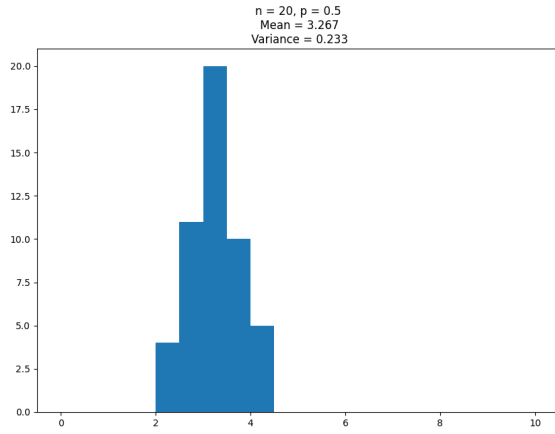
1.2

Function implementation in Python:

```
1 def get_updated_position_restricted(n,p):
2     pos = 0 #position
3     for _ in range(n):
4         rand = random.randint(1,100) #generating a random number in the range 1 to 100
5         if rand < p*100 or pos <= 0: #move one step right if pos == 0
6             pos += 1
7         else:
8             pos -= 1 #move one step left
9     return pos #return final position
```

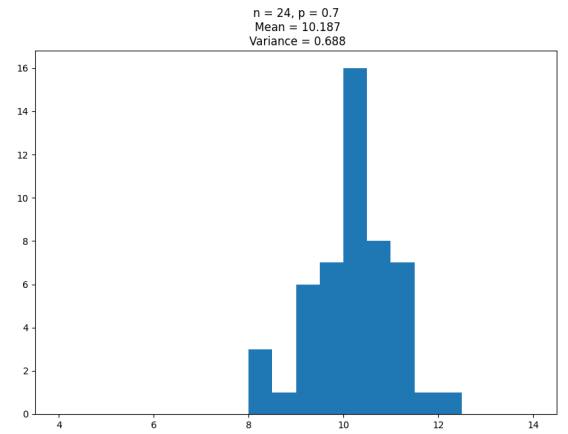
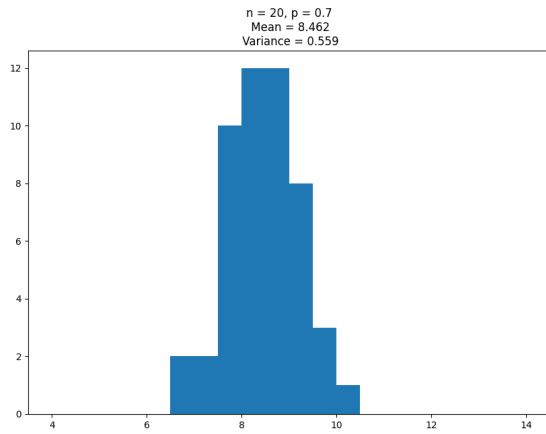
Calling the function for several iterations to get multiple expected values:

```
1 def main_12():
2     '''Calculating and plotting expected outcomes for various combinations of n and p'''
3     p = 0.7
4     n = 24
5     expected = []
6     for j in range(50): #expected value for each (n,p) for 25 iterations
7         outcomes = []
8         for i in range(25):
9             outcomes.append(get_updated_position_restricted(n,p))
10        expected.append(sum(outcomes)/25) #appending the expected (average) value for each(n,p)
11    #plotting and showing a histogram of calculated expected values
12    fig, ax = plt.subplots(figsize =(10, 7))
13    ax.hist(expected, bins =
14    [4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9,9.5,10,10.5,11,11.5,12,12.5,13,13.5,14])
15    plt.title('n = '+str(n)+', p = '+str(p)+'\n Mean = '+str(round(statistics.mean(expected)
16    ,3))+'\n Variance = '+str(round(statistics.variance(expected),3)))
17    plt.savefig("Q1_histograms/Q1.22 "+"_n = '+str(n)+'_p = '+str(p)+'.png')
18    plt.show()
```



(d) No. of steps taken = 20 with probability of moving a step right = 0.5.

(e) No. of steps taken = 24 with probability of moving a step right = 0.5.



(f) No. of steps taken = 20 with probability of moving a step right = 0.7.

(g) No. of steps taken = 24 with probability of moving a step right = 0.7.

Histograms produced by the above code for expected final positions of objects against frequency, where n is the number of steps taken and p is the probability of the object moving one step to the right.

Histograms (d) and (e) have the same value of p and varying number of steps n . Both histograms appear to follow a normal distribution, with (d) having a mean of 3.267 and a variance of 0.233 and (e) having a mean of 3.466 and a variance of 0.367. Again, increasing the number of steps has not had a significant effect on the mean but did increase the variance.

Histograms (f) and (g) have the same number of steps n respectively as (d) and (e) and varying values of p . They also appear to follow a normal distribution, with (f) having a mean of 8.462 and a variance of 0.559 and (g) having a mean of 10.187 and a variance of 0.668. Increasing p appears to have increased the mean and variance, but the mean has increased more significantly.

Additionally, due to the added constraint in this part, we see that no expected value for the final position is negative.

1.3

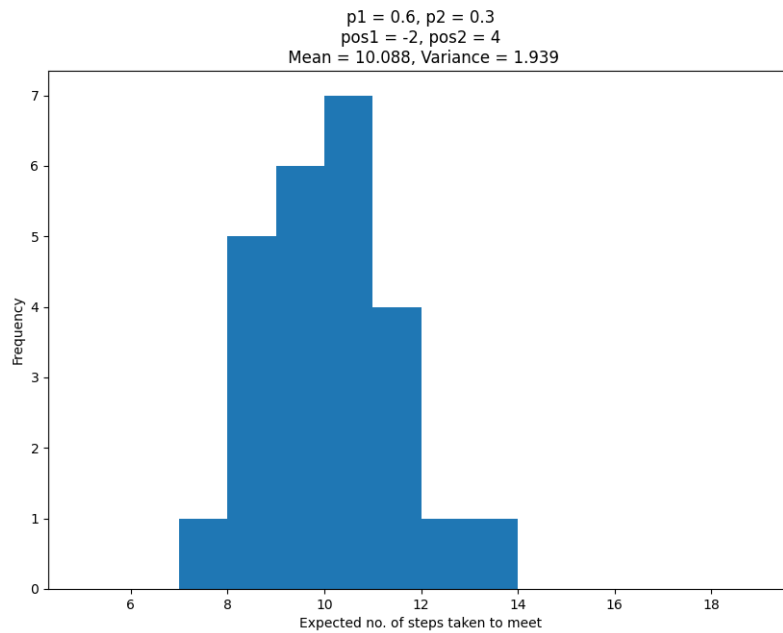
Function implementation in Python:

```
1 def stepsToMeet(pos1,pos2,p1,p2):
2     count = 0 #keeps count of number of steps taken for objects to meet
3     while pos1 != pos2:
4         rand = random.randint(1,100) #generating a random number in the range 1 to 100 to
         determine outcome
5         if rand < p1*100:
6             pos1 += 1 #move one step right
7         else:
8             pos1 -= 1 #move one step left
9         rand = random.randint(1,100) #generating a random number in the range 1 to 100
10        if rand < p2*100:
11            pos2 += 1 #move one step right
12        else:
13            pos2 -= 1 #move one step left
14        count += 1
15    return count
```

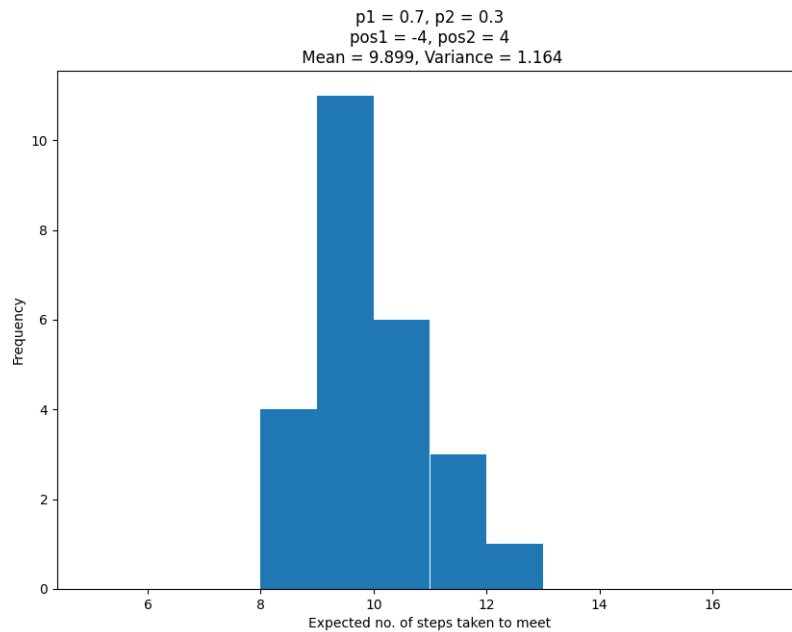
Calling the function for several iterations to get multiple expected values:

```
1 def main_13():
2     '''Calculating and plotting expected outcomes for various combinations of n & p'''
3     expected = []
4     p1 = 0.7
5     p2 = 0.3
6     pos1 = -4
7     pos2 = 4
8     for i in range(25): #calculating the expected value for each (n,p) for 25 iterations
9         outcomes = []
10        for j in range(25):
11            outcomes.append(stepsToMeet(pos1,pos2,p1,p2))
12        #calculating the average expected value for each(n,p)
13        expected.append(sum(outcomes)/25) #appending the expected (average) value for each(n,p)
14    #plotting a histogram of calculated expected values
15    fig, ax = plt.subplots(figsize =(10, 7))
16    ax.set_xlabel('Expected no. of steps taken to meet')
17    ax.set_ylabel('Frequency')
18    ax.hist(expected, bins = range(5,18))
19    plt.title('p1 = '+str(p1)+' , p2 = '+str(p2)+'\npos1 = '+str(pos1)+' , pos2 = '+str(pos2)+'\n
n Mean = '+str(round(statistics.mean(expected),3))+' , Variance = '+str(round(statistics.
variance(expected),3))')
20    plt.savefig("Q1_histograms/Q1.3 "+'_p1 = '+str(p1)+'_p2 = '+str(p2)+'_pos1 = '+str(pos1)+'
_pos2 = '+str(pos2)+'(6).png')
21    plt.show()
```

Histograms produced by the above code for expected number of steps taken for two objects walking randomly to meet, against frequency. p_1 and p_2 are the respective probabilities of object 1 and object 2 moving one step to the right, and $pos1$ and $pos2$ denote the starting positions of object 1 and object 2 respectively.



The above histogram shows that the expected number of steps to meet for two objects follows a normal distribution of mean 10.09 and variance 1.94 when they begin 6 steps apart with the given probabilities.



The above histogram shows that the expected number of steps to meet for two objects follows a normal distribution of mean 9.90 and variance 1.16 when they begin 8 steps apart with the given probabilities.