Caitlyn Jones

Prof. Yangyang Wang

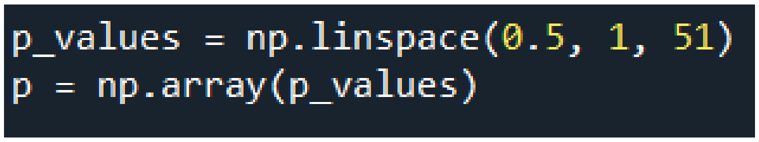
MATH 40A: Intro to Applied Math

25 September 2024

Homework 1

1.

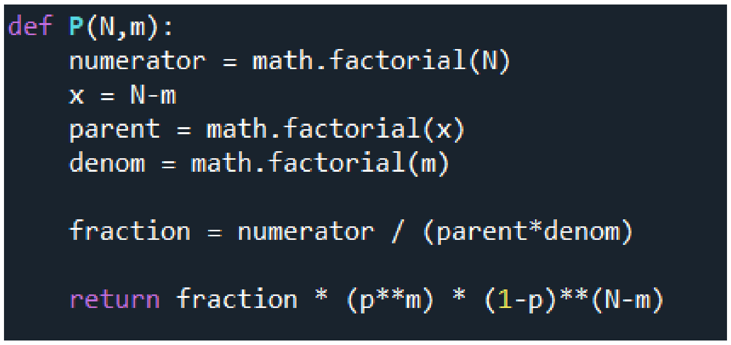
a. To create array `p`, I first created a `variable p_values` that will hold the 51 evenly spaced values of p from 0.5 to 1. Then, I created array `p` using `p_values`.



```
p_values = np.linspace(0.5, 1, 51)
p = np.array(p_values)
```

b. To create arrays `S1` and `S3`, I created a function `P(N,m)` that will create the formula for binomial distribution, which is

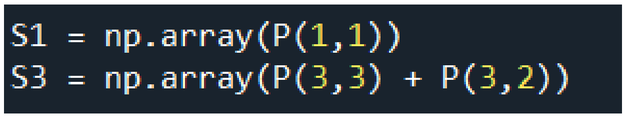
$P(N, m) = \frac{N!}{(N-m)!m!} p^m (1-p)^{N-m}$.



```
def P(N,m):
    numerator = math.factorial(N)
    x = N-m
    parent = math.factorial(x)
    denom = math.factorial(m)

    fraction = numerator / (parent*denom)

    return fraction * (p**m) * (1-p)**(N-m)
```

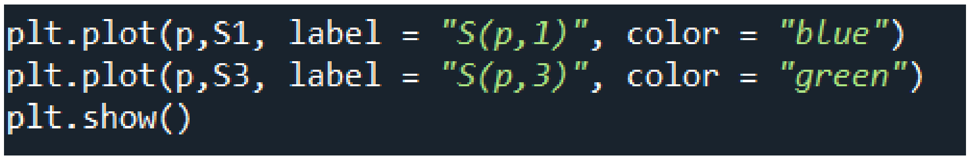
Then, I created arrays S1 and S3 based on the formulas

$S(p, 1) = P(1,1)$ and $S(p, 3) = P(3,3) + P(3,2)$

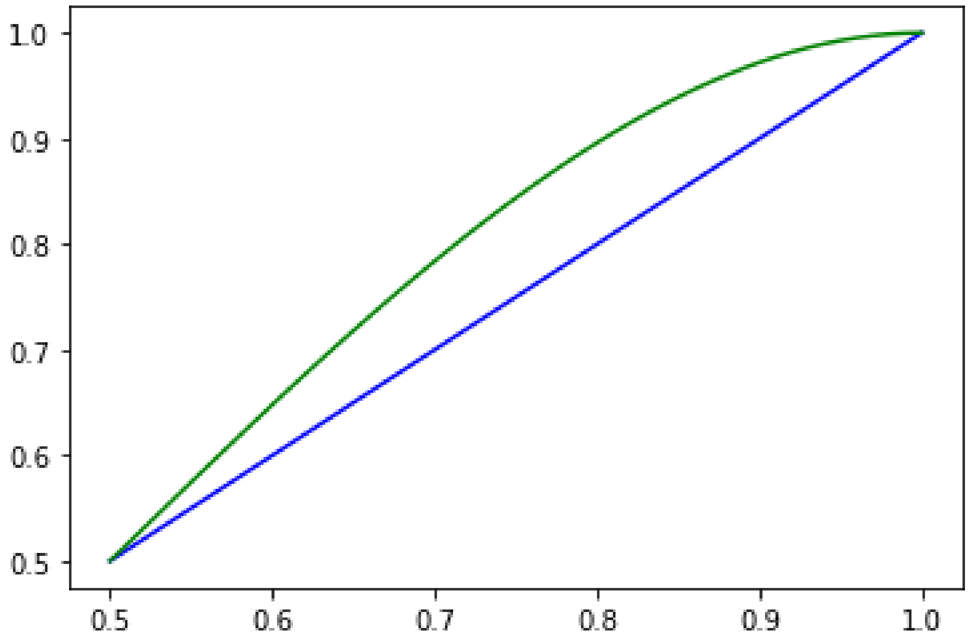


```
S1 = np.array(P(1,1))
S3 = np.array(P(3,3) + P(3,2))
```

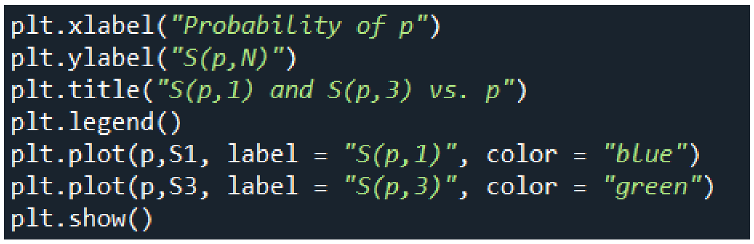
c. To graph these arrays, I created a graph to plot S1 vs p and S3 vs p.



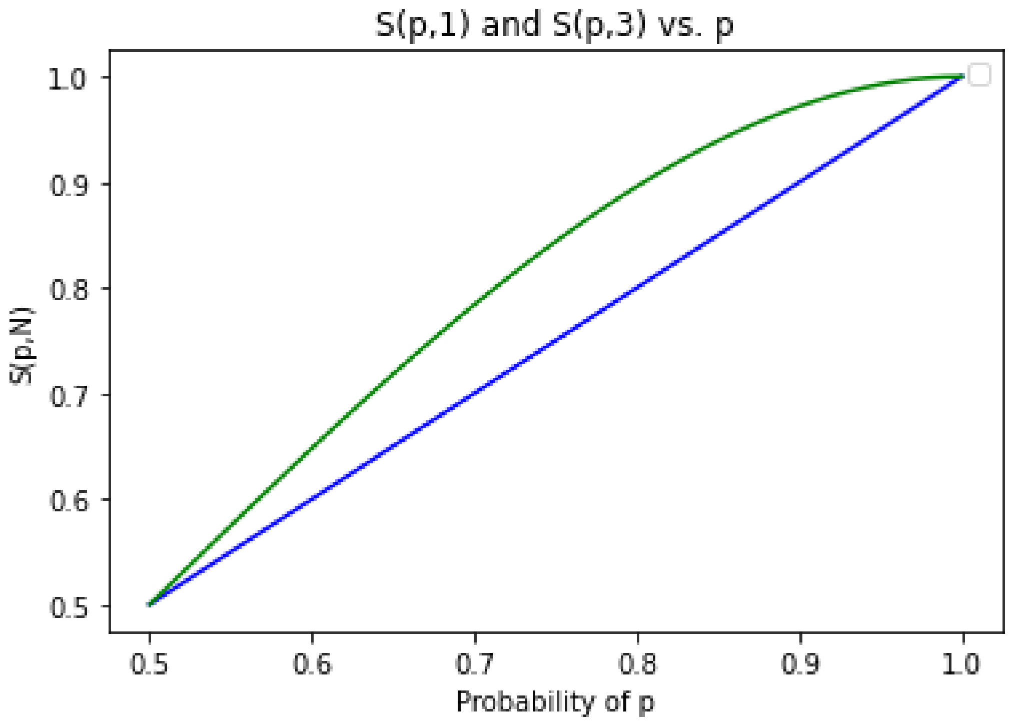
```
plt.plot(p,S1, label = "S(p,1)", color = "blue")
plt.plot(p,S3, label = "S(p,3)", color = "green")
plt.show()
```



d. Lastly, I added a legend so that anyone viewing the graph can easily identify what is happening.

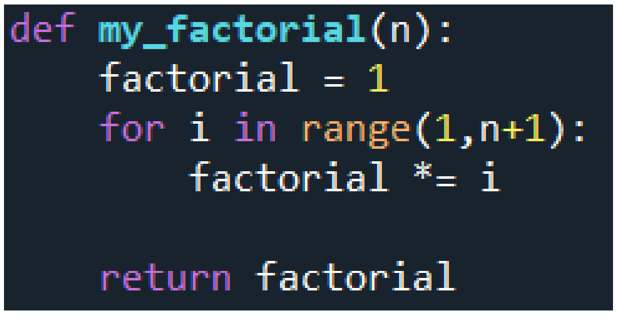


```
plt.xlabel("Probability of p")
plt.ylabel("S(p,N)")
plt.title("S(p,1) and S(p,3) vs. p")
plt.legend()
plt.plot(p,S1, label = "S(p,1)", color = "blue")
plt.plot(p,S3, label = "S(p,3)", color = "green")
plt.show()
```



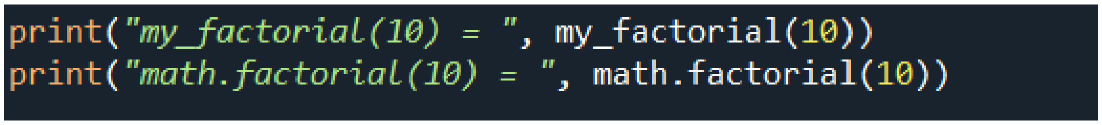
2.

a. To successfully compute the factorial of any number using function `my_factorial(n)`, I created variable `factorial = 1` inside the function, since every factorial ends with multiplying the rest of the factorial by 1. Then, I used a for loop to iterate through each number from 1 to n+1 (since the range is not inclusive of n, the range must be 1 to n+1) and multiply that number by the variable factorial.

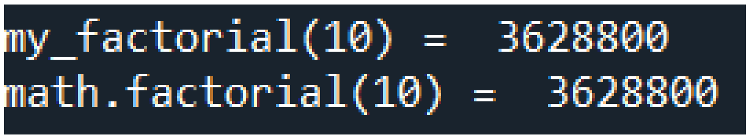


```python
def my_factorial(n):
    factorial = 1
    for i in range(1,n+1):
        factorial *= i

    return factorial
```

Next, I used `math.factorial(n)` to make sure that `my_factorial(n)` produces the same answer.

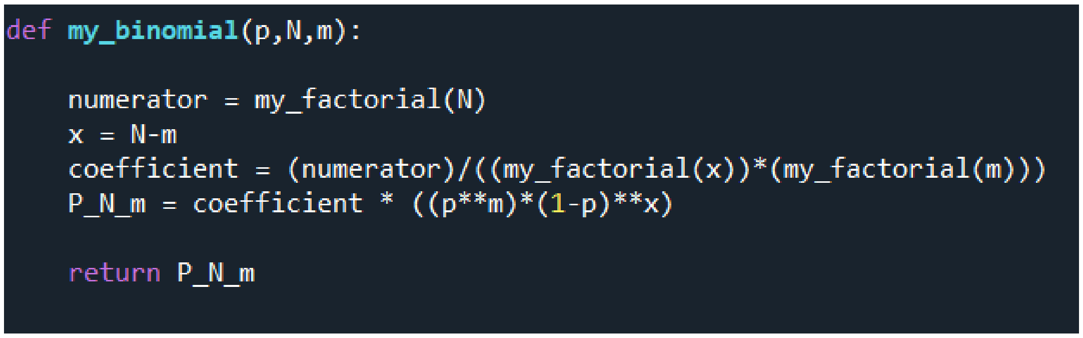


```
print("my_factorial(10) = ", my_factorial(10))
print("math.factorial(10) = ", math.factorial(10))
```



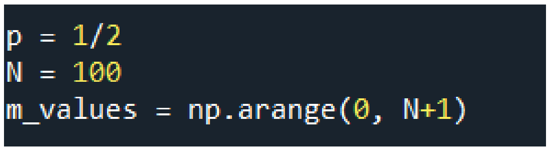
```
my_factorial(10) =  3628800
math.factorial(10) =  3628800
```

b. To create the function `my_binomial(p,N,m)`, I broke down the binomial $P(N,m) = \frac{N!}{(N-m)!m!}p^m(1-p)^{N-m}$ into 3 different variables: `numerator`, `x`, and `coefficient`. With these three variables, I then computed the formula for `P(N,m)`.



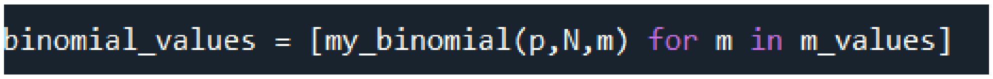
```
def my_binomial(p,N,m):

    numerator = my_factorial(N)
    x = N-m
    coefficient = (numerator)/((my_factorial(x))*(my_factorial(m)))
    P_N_m = coefficient * ((p**m)*(1-p)**x)

    return P_N_m
```

c. To plot `P(N,m)`, I created three variables: `p = ½`, `N = 100`, and `m_values` which contains values for m from 0 up to N (again, using N+1 because the range is exclusive of the last digit entry in the range).



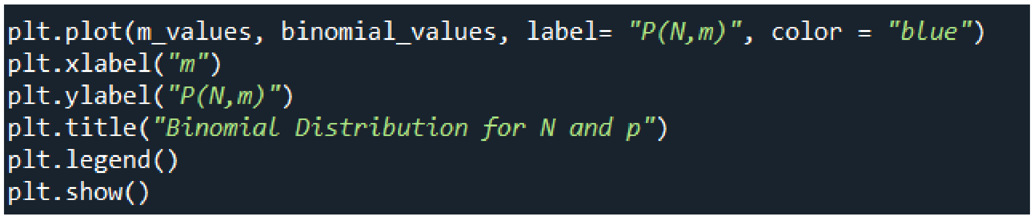
```
p = 1/2
N = 100
m_values = np.arange(0, N+1)
```

Then, I created an array `binomial_values` that will contain the computation of `my_binomial(p,N,m)` for every m value from `m_values`.

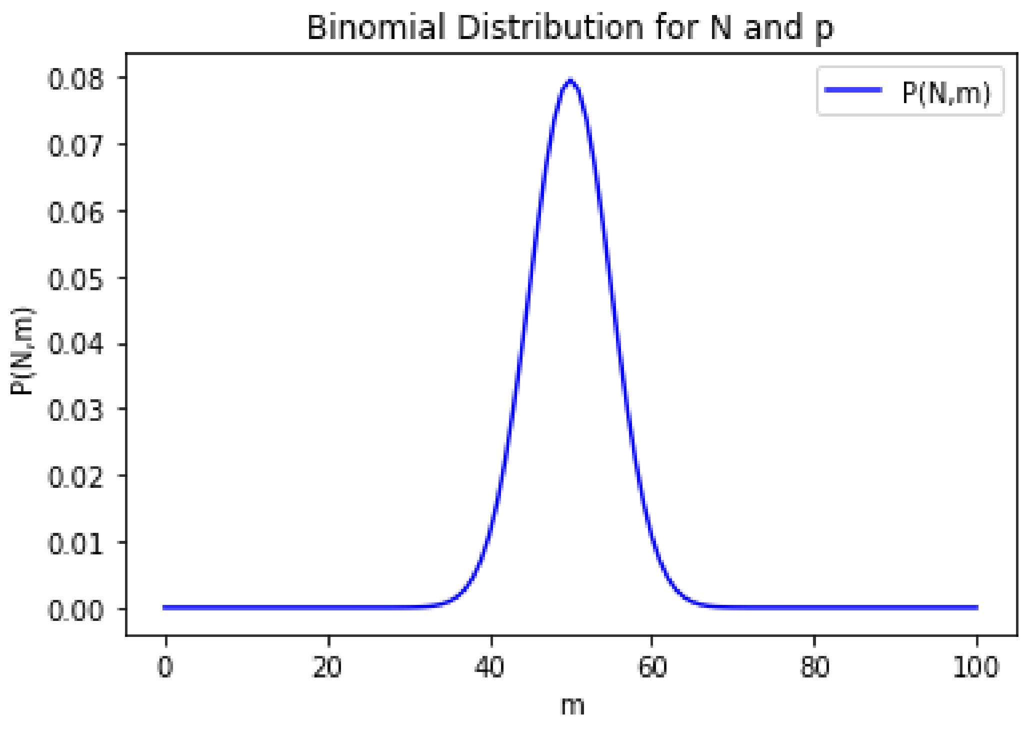


```
binomial_values = [my_binomial(p,N,m) for m in m_values]
```

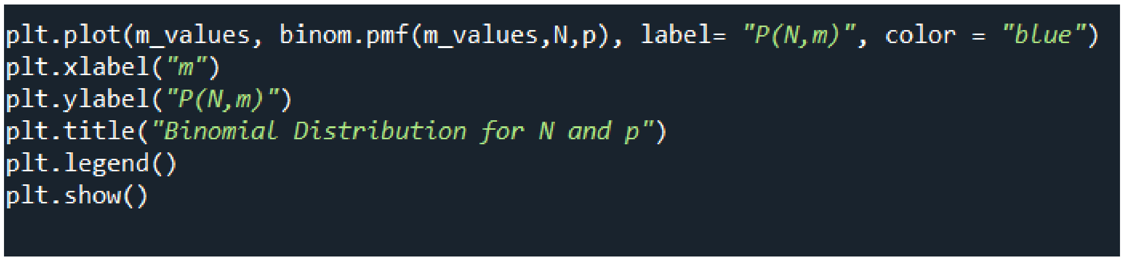
Lastly, for my graph, `m_values` is the x-axis and `binomial_values` is the y-axis.



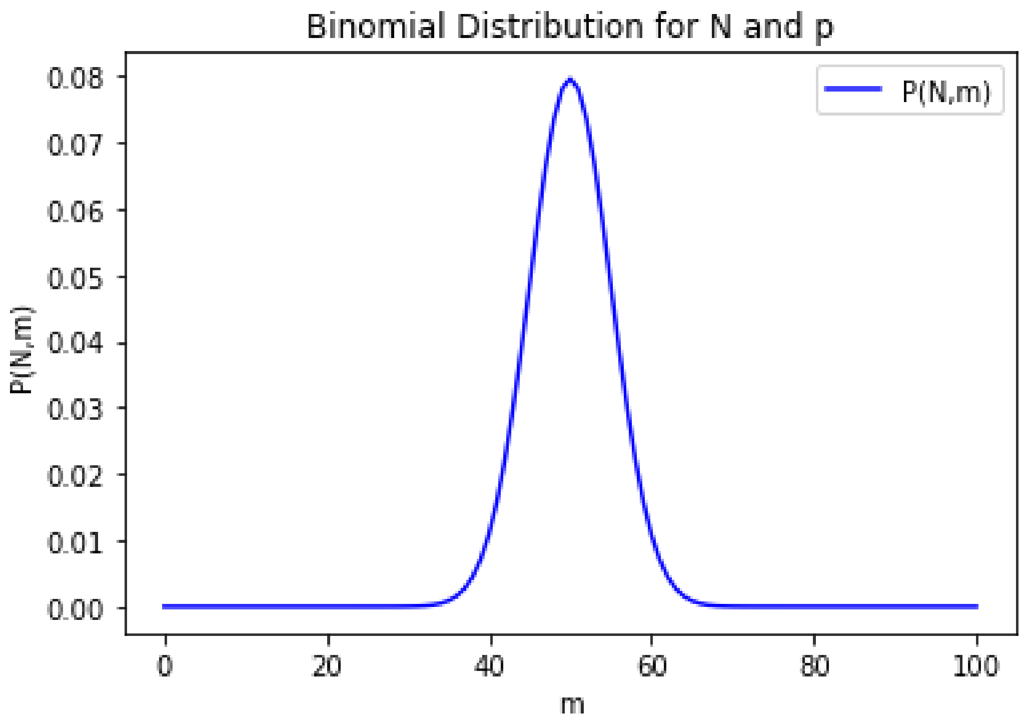
```
plt.plot(m_values, binomial_values, label= "P(N,m)", color = "blue")
plt.xlabel("m")
plt.ylabel("P(N,m)")
plt.title("Binomial Distribution for N and p")
plt.legend()
plt.show()
```



d. To check that `my_bionomial(p,N,m)` outputs the same graph as the `Scipy` module, I graphed `m_values` as the x-axis and binomial distribution function in `Spicy` as the y-axis. The results are the same!



```
plt.plot(m_values, binom.pmf(m_values,N,p), label= "P(N,m)", color = "blue")
plt.xlabel("m")
plt.ylabel("P(N,m)")
plt.title("Binomial Distribution for N and p")
plt.legend()
plt.show()
```



3.

Winner	Loser	Frequency	Theoretical Proportion
4	0	9	p^4+q^4
4	1	13	$4p^4q + 4pq^4$
4	2	11	$10p^4q^2+10p^2q^4$
4	3	11	$20p^4q^3+20p^3q^4$
		Total 44	1

Given the above table, imagine each line is labeled 1-4. Following are the equations of theoretical proportions using $P(N,m) = \frac{N!}{(N-m)!m!}p^m(1-p)^{N-m}$ with $q = (1-p)$.

Line 1:

$$P(4,4) + P(4,0)$$

$$\frac{4!}{(4-4)!\,4!}p^4q^{4-4} + \frac{4!}{(4-0)!\,0!}p^0q^{4-0}$$

$$p^4 + q^4$$

Line 2:

$P(5,4) + P(5,1)$

$$\frac{5!}{(5-4)!\,4!}p^4q^{5-4} + \frac{5!}{(5-1)!\,1!}p^1q^{5-1}$$

$5p^4q + 5pq^4$

However, because there are only 4 instances where the winning team wins 4 and losing team wins 1, so we can subtract one instance from $P(5,4)$ and $P(5,1)$.

So, the final equation will be $4p^4q + 4pq^4$.

Line 3:

$P(6,4) + P(6,2)$

$$\frac{6!}{(6-4)!\,4!}p^4q^{6-4} + \frac{6!}{(6-2)!\,2!}p^2q^{6-2}$$

$15p^4q^2 + 15p^2q^4$

Following the same logic for line 2, we can subtract 5 cases from each instance to get

$10p^4q^2 + 10p^2q^4$

Line 4:

$P(7,4) + P(7,3)$

$$\frac{7!}{(7-4)!\,4!}p^4q^{7-4} + \frac{7!}{(7-3)!\,3!}p^3q^{7-3}$$

$35p^4q^3 + 35p^3q^4$

But with subtracting 15 cases, we get

$20p^4q^3 + 20p^3q^4$

4.

a. Given $S(p,7)$, we get the following formula

$S(p,7) = P(4,4) + P(5,4) + P(6,4) + P(7,4)$

$$\mathrm{S}(\mathrm{p},7) = \mathrm{p}^4 + 5\mathrm{p}^4\mathrm{q} + 15\mathrm{p}^4\mathrm{q}^2 + 35\mathrm{p}^4\mathrm{q}^3$$

We can plug (1-p) back into q to get

$$\mathrm{S}(\mathrm{p},7) = \mathrm{p}^4 + 4\mathrm{p}^4(1-\mathrm{p}) + 10\mathrm{p}^4(1-\mathrm{p})^2 + 20\mathrm{p}^4(1-\mathrm{p})^3$$

b. With p = 0.65, we get the following

$$\mathrm{S}(0.65{,}7) = (0.65)^4 + 4(0.65)^4(1-0.65) + 10(0.65)^4(1-0.65)^2 + 20(0.65)^4(1-0.65)^3$$

$$\mathrm{S}(0.65{,}7) = .800$$

Therefore, the winning team wins 80% of the time.