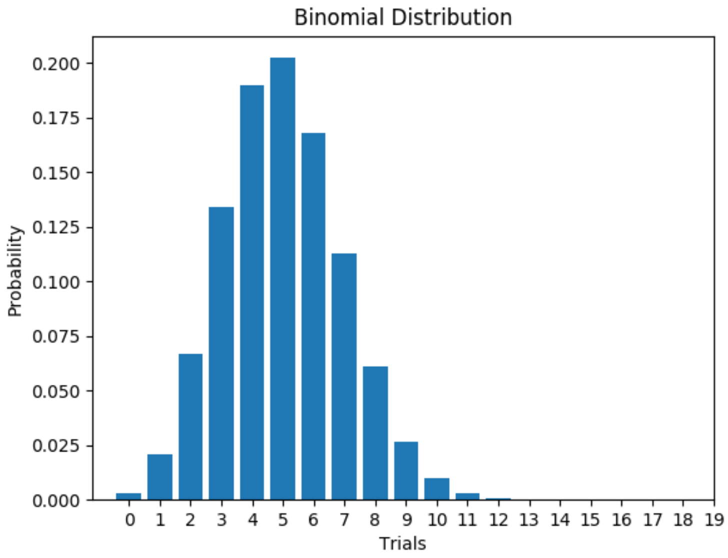
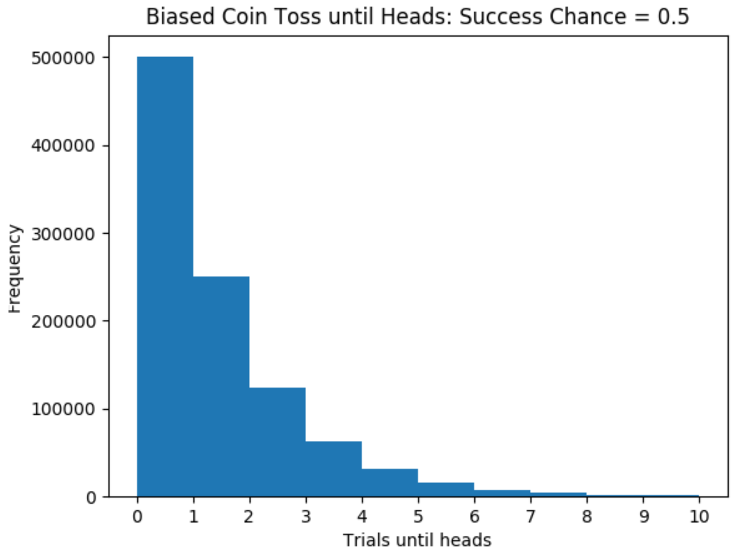
Conor Falvey
MATH 305
Nov. 8, 2018

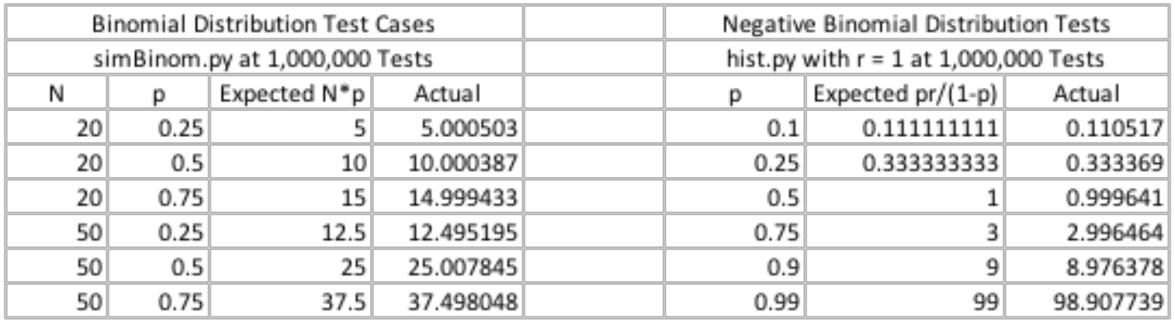
Homework #10

1. The sum of x_i results divided by the number of experiments should converge to N*p, or 5 in the sample case where N = 20 and p = .25 as numExperiments gets arbitrarily large. Expected behavior is observed as numExperiments increases and as N and p are changed.
2. See hist.py
3. We expect the sum of the x_i results divided by numExperiments to converge to the mean of a negative binomial distribution, or pr/(1-p). Expected behavior is observed as numExperiments increases. Convergence was also tested for varying p and r values.

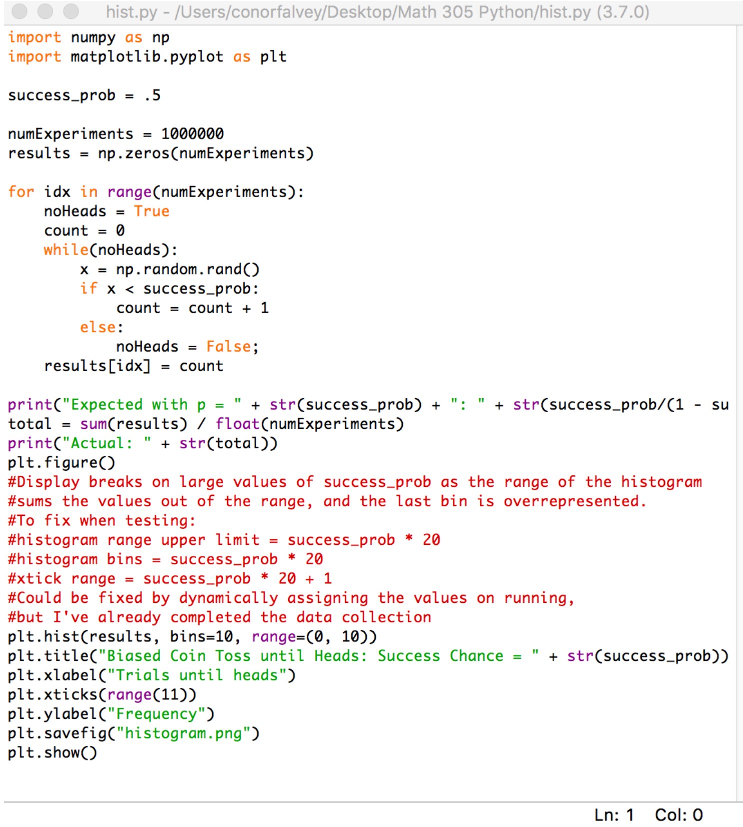
Resulting Graphs and Data Collection







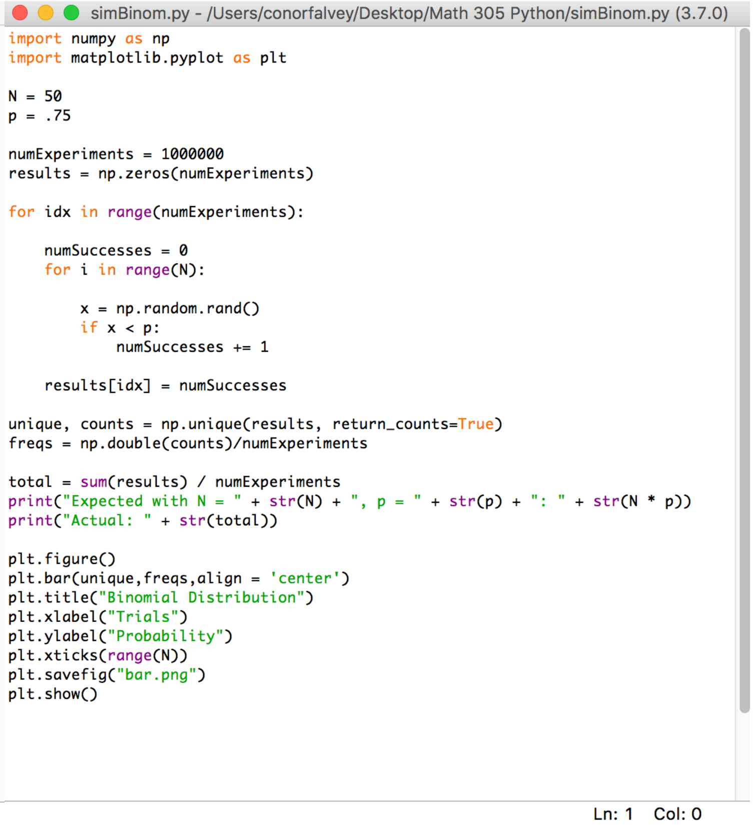
Binomial Distribution Test Cases					Negative Binomial Distribution Tests		
simBinom.py at 1,000,000 Tests					hist.py with r = 1 at 1,000,000 Tests		
N	p	Expected N*p	Actual		p	Expected pr/(1-p)	Actual
20	0.25	5	5.000503		0.1	0.111111111	0.110517
20	0.5	10	10.000387		0.25	0.333333333	0.333369
20	0.75	15	14.999433		0.5	1	0.999641
50	0.25	12.5	12.495195		0.75	3	2.996464
50	0.5	25	25.007845		0.9	9	8.976378
50	0.75	37.5	37.498048		0.99	99	98.907739



hist.py - /Users/conorfalvey/Desktop/Math 305 Python/hist.py (3.7.0)

```
import numpy as np
import matplotlib.pyplot as plt

success_prob = .5

numExperiments = 1000000
results = np.zeros(numExperiments)

for idx in range(numExperiments):
    noHeads = True
    count = 0
    while(noHeads):
        x = np.random.rand()
        if x < success_prob:
            count = count + 1
        else:
            noHeads = False;
    results[idx] = count

print("Expected with p = " + str(success_prob) + ": " + str(success_prob/(1 - su
total = sum(results) / float(numExperiments)
print("Actual: " + str(total))
plt.figure()
#Display breaks on large values of success_prob as the range of the histogram
#sums the values out of the range, and the last bin is overrepresented.
#To fix when testing:
#histogram range upper limit = success_prob * 20
#histogram bins = success_prob * 20
#xtick range = success_prob * 20 + 1
#Could be fixed by dynamically assigning the values on running,
#but I've already completed the data collection
plt.hist(results, bins=10, range=(0, 10))
plt.title("Biased Coin Toss until Heads: Success Chance = " + str(success_prob))
plt.xlabel("Trials until heads")
plt.xticks(range(11))
plt.ylabel("Frequency")
plt.savefig("histogram.png")
plt.show()
```

Ln: 1   Col: 0



simBinom.py - /Users/conorfalvey/Desktop/Math 305 Python/simBinom.py (3.7.0)

```
import numpy as np
import matplotlib.pyplot as plt

N = 50
p = .75

numExperiments = 1000000
results = np.zeros(numExperiments)

for idx in range(numExperiments):

    numSuccesses = 0
    for i in range(N):

        x = np.random.rand()
        if x < p:
            numSuccesses += 1

    results[idx] = numSuccesses

unique, counts = np.unique(results, return_counts=True)
freqs = np.double(counts)/numExperiments

total = sum(results) / numExperiments
print("Expected with N = " + str(N) + ", p = " + str(p) + ": " + str(N * p))
print("Actual: " + str(total))

plt.figure()
plt.bar(unique,freqs,align = 'center')
plt.title("Binomial Distribution")
plt.xlabel("Trials")
plt.ylabel("Probability")
plt.xticks(range(N))
plt.savefig("bar.png")
plt.show()
```

Ln: 1   Col: 0