Project #3 – Samples and statistics

EE 511: Fall 2019

Qiong Wang Id: 5906740674

Tool: PyCharm

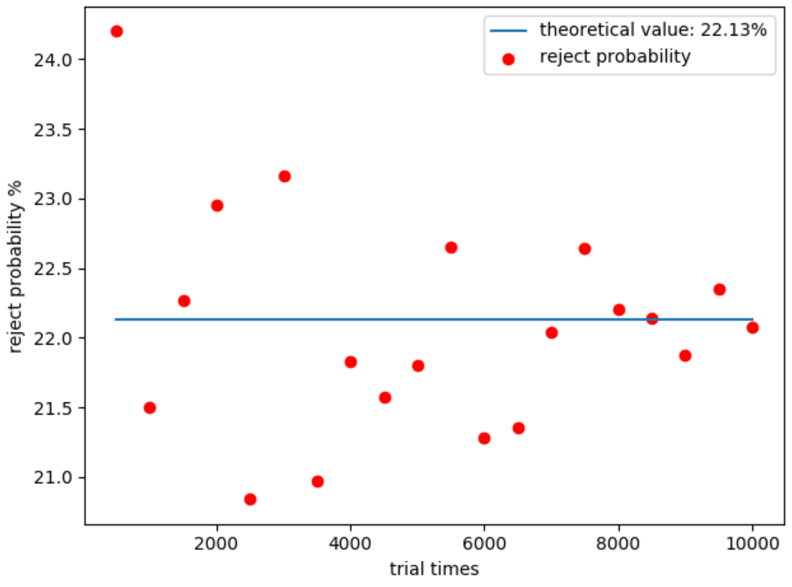
1. Question 1

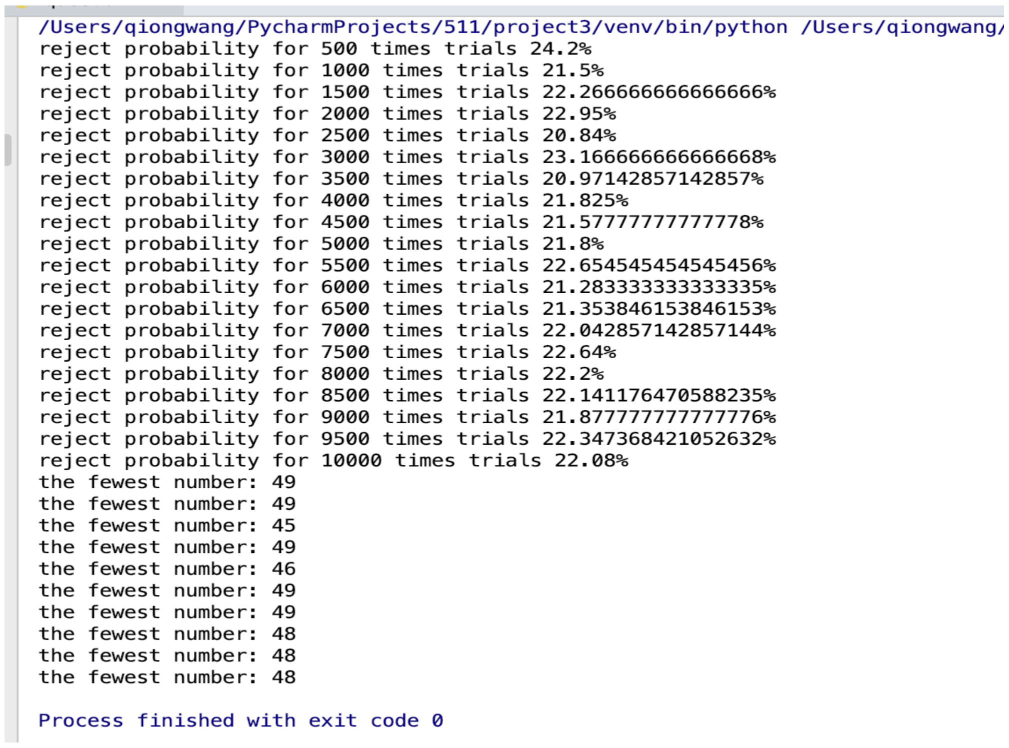
Analysis: There are two sub-questions in this part, the first one is to simulate the lot sampling and calculate the reject probability. There are 125 microchips, and 6 of them is defective, so the theoretical probability to reject is P = 1 – (119*118*117*116*115) / (125*124*123*122*121) = 22.13%. So, I simulate it 20 times and compare the simulated value with theoretical value, from previous experience, I think the simulated value would converge to theoretical value, when I increase the simulation times, and then I repeat the experiment 20 times and each time I simulate range(500, 10000, 500) times, and average the result, and I guess the result will be close to theoretical value. When I am coding, I allocate a list with integer from 1 to 125, 1 to 6 are stand for defective chips, others are good chips. For each check, I draw a number randomly from this list without replacement, if the number is less or equal to 6 find out a defective chip, stop the trial, and store the result. In the end I calculate the reject probability and plot it. For the second part, the experiment method is almost same as part 1, draw a number from the list without replacement. However, the maximum draw times is not a fix number, I use a loop to check all of the possible maximum draw times range (1, 126) and calculate the reject probability for each situation, until the reject probability is greater than 95%. In order to minimize the experiment error, for each maximum draw times, I simulate 1000 times, if the reject times is bigger than 950, the probability is greater than 95%. I run the code 10 time, to get 10 results, and compare them.

Code:

```python
import random
import matplotlib.pyplot as plt


def main():
    reject_prob = []
    for i in range(500, 10500, 500):
        reject_prob.append(100 * count_defective(i, 5) / i)
        print('reject probability for '+str(i)+' times trials',
str(reject_prob[-1]) + '%')

    plt.figure()
    plt.scatter([x for x in range(500, 10500, 500)], reject_prob,
label='reject probability', color='red')
    plt.plot([x for x in range(500, 10500, 500)], [22.13]*20,
label='theoretical value: 22.13%', linestyle='-')
    plt.xlabel('trial times')
    plt.ylabel('reject probability %')
    plt.legend()
    plt.show()

    for i in range(10):
        for index in range(126):
            if count_defective(1000, index) >= 950:
                print('the fewest number:', index)
                break


def count_defective(check_times: int, check_threshold) -> int:
    result = 0
    for times in range(check_times):
        lot = [x for x in range(1, 126)]
        flag = True
        for i in range(check_threshold):
            test = random.sample(lot, 1)[0]
            if test < 7:
                flag = False
                break
            lot.remove(test)

        if not flag:
            result += 1

    return result


if __name__ == '__main__':
    main()
```

Result: From the result, I know that, the reject probability is close to 22.13%, the maximum experiment error is 9.5%, when times = 500, p = 24.2%, and from the result, when I increase the experiment times, the probability become close to the theoretical value, and the graph become more plate. For the second part, the fewest check times is between 45 and 49, and most of the result is 48 or 49, which is close to theoretical value.




```
/Users/qiongwang/PycharmProjects/511/project3/venv/bin/python /Users/qiongwang/
reject probability for 500 times trials 24.2%
reject probability for 1000 times trials 21.5%
reject probability for 1500 times trials 22.266666666666666%
reject probability for 2000 times trials 22.95%
reject probability for 2500 times trials 20.84%
reject probability for 3000 times trials 23.166666666666668%
reject probability for 3500 times trials 20.97142857142857%
reject probability for 4000 times trials 21.825%
reject probability for 4500 times trials 21.57777777777778%
reject probability for 5000 times trials 21.8%
reject probability for 5500 times trials 22.654545454545456%
reject probability for 6000 times trials 21.283333333333335%
reject probability for 6500 times trials 21.353846153846153%
reject probability for 7000 times trials 22.042857142857144%
reject probability for 7500 times trials 22.64%
reject probability for 8000 times trials 22.2%
reject probability for 8500 times trials 22.141176470588235%
reject probability for 9000 times trials 21.877777777777776%
reject probability for 9500 times trials 22.347368421052632%
reject probability for 10000 times trials 22.08%
the fewest number: 49
the fewest number: 49
the fewest number: 45
the fewest number: 49
the fewest number: 46
the fewest number: 49
the fewest number: 49
the fewest number: 48
the fewest number: 48
the fewest number: 48

Process finished with exit code 0
```

2. Question 2

Analysis: In this question, I use two method to estimate the number of arrived cars per hour. The first method is Bernoulli trial, I subdivide the hour into 10000 intervals, and each interval is 12ms. For each interval, the probability that arrive a car is about 1.2e-3, and for an hour the number of arrive cars is same as 10000 times Bernoulli trials. For second method I directly use inverse transform method to generate a Poisson distribution histogram. As we all know the Bernoulli trial will converge to Poisson when p is small and n is large, and np is a constant positive number. So, Poisson method would generate a better result. In order to compare which method is better, I use the PMF of passion distribution to plot a histogram which is the theoretical value for Poisson distribution.

The limit of Bernoulli:

$$\lim_{n\to\infty}\binom{n}{k}p^k(1-p)^{n-k} = \lim_{n\to\infty}\binom{n}{k}\left(\frac{\mu}{n}\right)^k(1-\frac{\mu}{n})^{n-k}$$

$$\begin{aligned}\lim_{n\to\infty}\binom{n}{k}\left(\frac{\mu}{n}\right)^k(1-\frac{\mu}{n})^{n-k} &= \lim_{n\to\infty}\frac{n(n-1)(n-2)\cdots(n-k+1)}{k!}\frac{\mu^k}{n^k}\left(1-\frac{\mu}{n}\right)^{n-k}\\ &= \lim_{n\to\infty}\frac{\mu^k}{k!}\frac{n}{n}\cdot\frac{n-1}{n}\cdots\frac{n-k+1}{n}\left(1-\frac{\mu}{n}\right)^{-k}\left(1-\frac{\mu}{n}\right)^{n}\end{aligned}$$

Because:

$$\lim_{n\to\infty}\frac{n}{n}\cdot\frac{n-1}{n}\cdots\frac{n-k+1}{n}\left(1-\frac{\mu}{n}\right)^{-k} = 1$$

$$\lim_{n\to\infty}\left(1-\frac{\mu}{n}\right)^{n} = e^{-\mu}$$

We have:

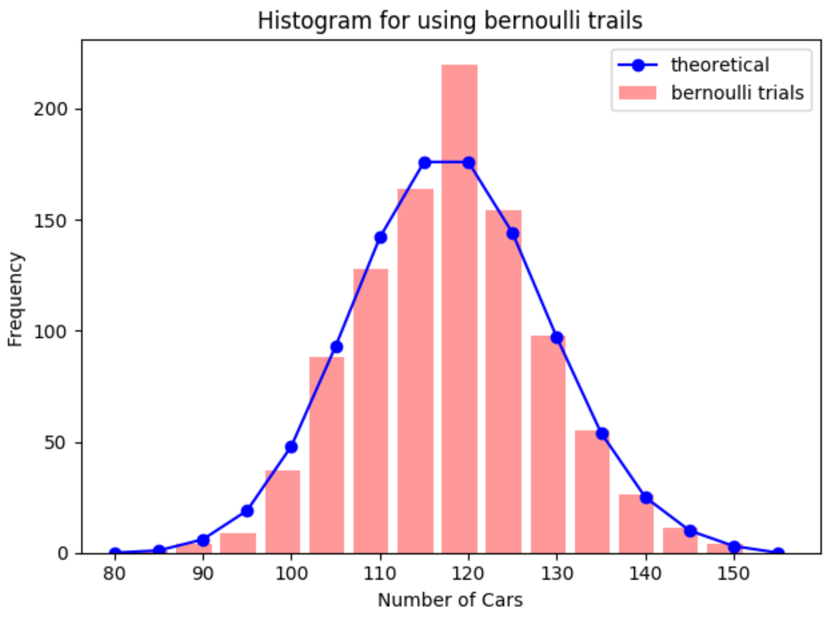
$$\lim_{n\to\infty}\binom{n}{k}\left(\frac{\mu}{n}\right)^k(1-\frac{\mu}{n})^{n-k} = \frac{\mu^k}{k!}e^{-\mu}$$

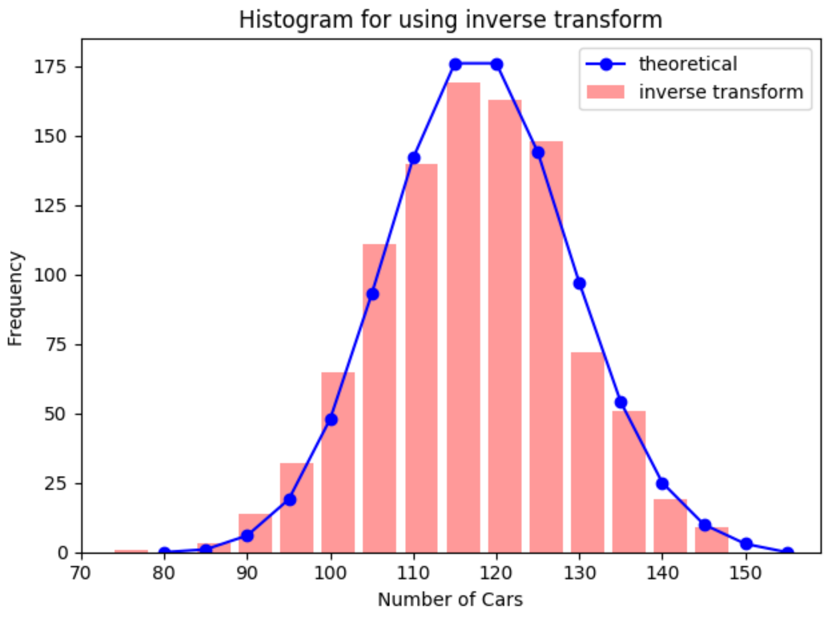
Code:

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


average = 120
N = 10000
p = average/N

poisson = []
prob = (average ** 79)*np.exp(-average)/np.math.factorial(79)
for i in range(16, 32):
    number = 0
    for j in range(5):
        x = 5*i+j
        prob *= (average/x)
        number += 1000 * prob
    poisson.append(int(number))


bernoulli_result = []
for i in range(1000):
    counter = 0
    for j in range(N):
        flip = np.random.uniform(0, 1)
        if flip <= p:
            counter += 1
    bernoulli_result.append(counter)

plt.figure(1)
plt.hist(bernoulli_result, bins=range(min(bernoulli_result) - 1,
max(bernoulli_result) + 2, 5), rwidth=0.8, alpha=0.4, color='r',
label='bernoulli trials',  align='left')
plt.plot(range(80, 160, 5), poisson, marker='o', color='b',
label='theoretical')
plt.xlabel('Number of Cars')
plt.ylabel('Frequency')
plt.title('Histogram for using bernoulli trails')
plt.legend()
```

```
inv_transform_result = []
for times in range(1000):
    U = np.random.uniform(0, 1)
    i = 0
    P = np.exp(-average)
    F = P
    while 1:
        if U < F:
            inv_transform_result.append(i)
            break
        else:
            P = (average*P)/(i+1)
            F += P
            i += 1

plt.figure(2)
plt.hist(inv_transform_result, bins=range(min(inv_transform_result)
- 1, max(inv_transform_result) + 2, 5), rwidth=0.8, alpha=0.4,
color='r', label='inverse transform',  align='left')
plt.plot(range(80, 160, 5), poisson, marker='o', color='b',
label='theoretical')
plt.xlabel('Number of Cars')
plt.ylabel('Frequency')
plt.title('Histogram for using inverse transform')
plt.legend()
plt.show()
```

Result: In the result, the blue point is the theoretical value for Poisson distribution, the histogram is Bernoulli trial and inverse transform method. From these two figures, I find that, both methods could have a good result when simulate Poisson question, but inverse method generate a better result.





3. Question 3

Analysis: This question I use a while loop to generate a uniform RV from 0 to 1, sum all the random variable until the sum is larger than four, store the number of RV. I repeat the experiment 100, 1000, 10000 times, plot the histogram, and calculate the mean of N. For each x ~ U (0, 1), all of the x is i.i.d. and mean(x) = 0.5. So, if I want N ~ $\{\sum_{i=1}^{N} x > 4\}$, E(N) >= 8.5

Code:

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


def main():
    sample_size = [100, 1000, 10000]
    for size in sample_size:
        samples = generate(size)
        plot_hist(samples, size)
        print('when using ' + str(size) + 'samples, E[N] =', 
np.mean(samples))
    plt.show()


def generate(size: int) -> list:
    result = []
    for i in range(size):
        N = 0
        counter = 0
        while N <= 4:
            N += np.random.uniform(0, 1)
            counter += 1
        result.append(counter)
    return result


def plot_hist(samples: list, size: int):
    plt.figure(size)
    plt.hist(samples, bins=range(min(samples) - 1, max(samples) + 2), 
rwidth=0.8, edgecolor='black', align='left')
```

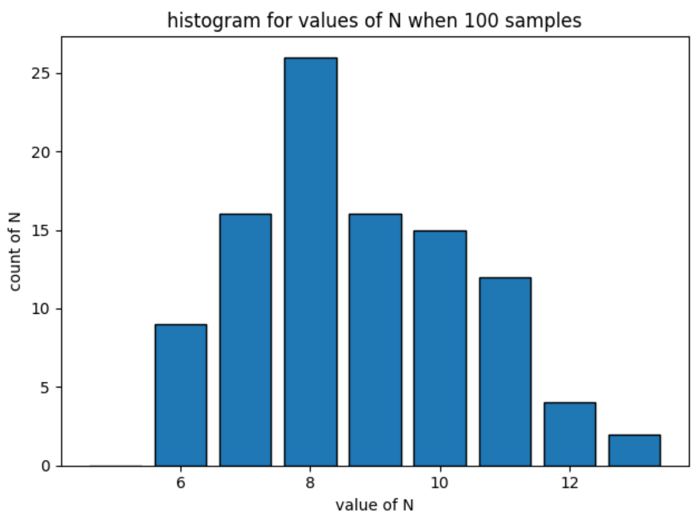
```
    plt.xlabel('value of N')
    plt.ylabel('count of N')
    plt.title('histogram for values of N when ' + str(size) + '
samples')


if __name__ == '__main__':
    main()
```

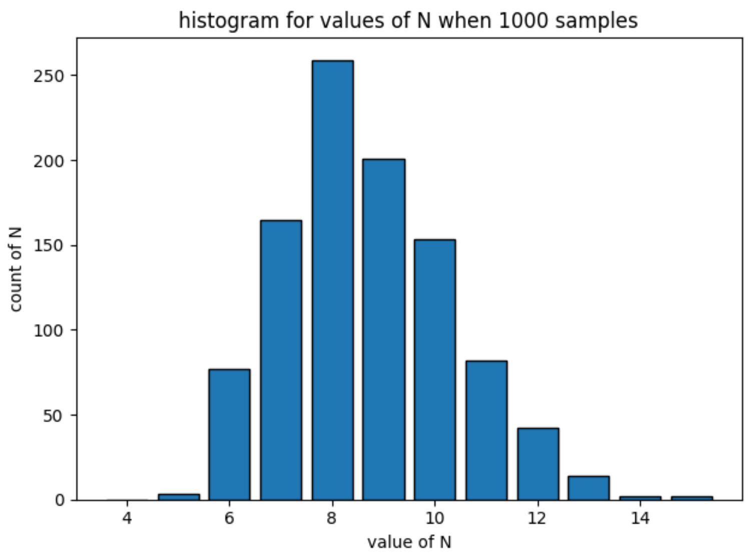
Result:

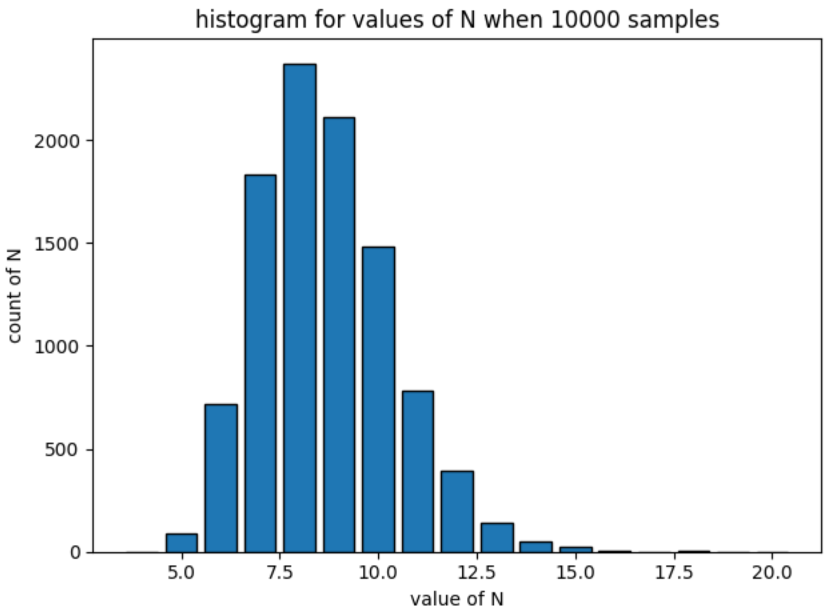
when using 100 samples, E[N] = 8.74

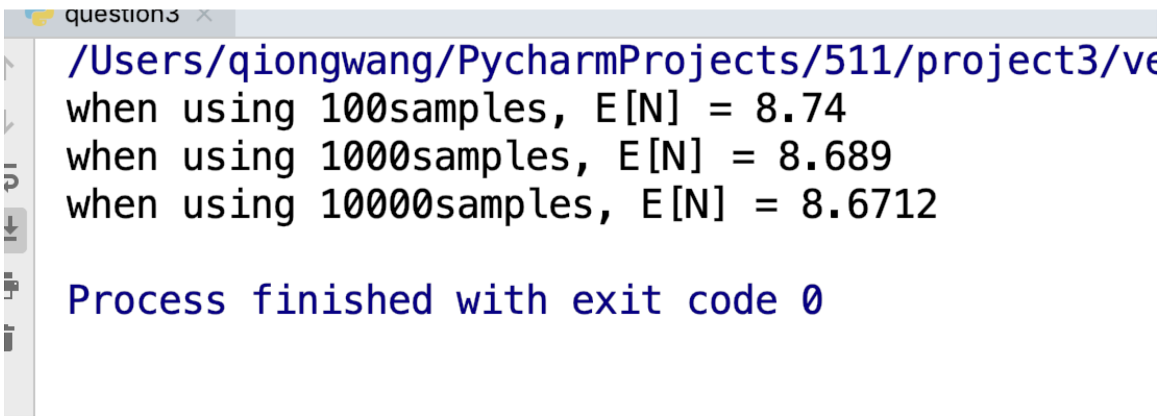
when using 1000 samples, E[N] = 8.689

when using 10000 samples, E[N] = 8.6712









```
/Users/qiongwang/PycharmProjects/511/project3/ve
when using 100samples, E[N] = 8.74
when using 1000samples, E[N] = 8.689
when using 10000samples, E[N] = 8.6712

Process finished with exit code 0
```

4. Question 4

Analysis: In this question, I will determinate p firstly, for any random variable x, $p(x) \in [0, 1]$, and $\sum p(x) = 1$, so, for $p_j = \frac{p}{j}$ j = 1,2 ……. 60, $\sum_{j=1}^{60} p_j = 1$, I calculate p firstly. $N_{60} = \min\{k: X_k = 60\}$, so, I use a uniform random variable U to simulate p_j, $U < p/60$, stands for $X = 60$. I use a loop to simulate the trial 10000 times and record the result, plot the histogram and calculate the mean and variance. Theoretically, $E[N_{60}] = 279.79$, $var[N_{60}] = 78563$.

Code:

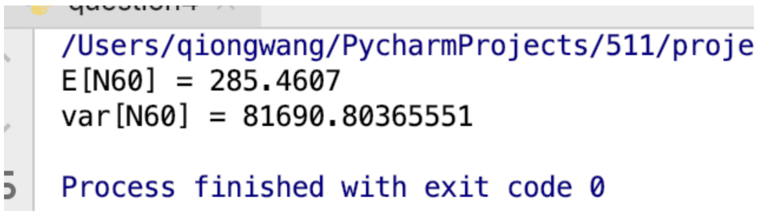
```
import numpy as np
import matplotlib.pyplot as plt

s = 0
for i in range(1, 61):
    s += (1/i)
p = 1/s

result = []
for i in range(10000):
    counter = 0
    while True:
        flip = np.random.uniform(0, 1)
        counter += 1
        if flip < p/60:
            result.append(counter)
            break

print('E[N60] =', np.mean(result))
print('var[N60] =', np.var(result))

plt.figure()
plt.hist(result, bins=range(max(result)+1), rwidth=0.8, color='r',
align='left')
plt.xlabel('Value of N60')
plt.ylabel('Frequency')
plt.title('Histogram for Distribution of N60')
plt.show()
```

Result:

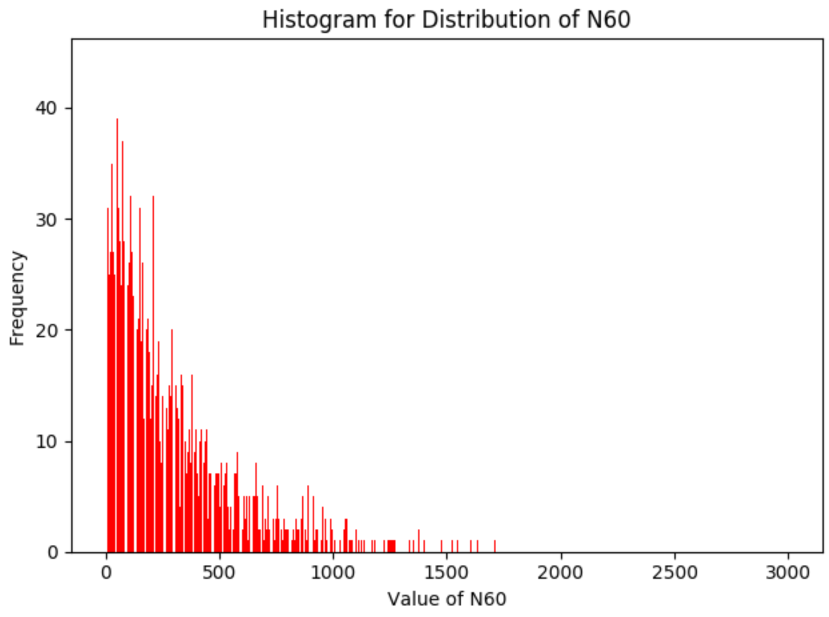
The result of simulation is: E[N60] = 285.4607 var[N60] = 81690.80365551

Theoretical value: $E[N_{60}] = 279.79$, $var[N_{60}] = 78563$.

Compare to theoretical value, the error of mean is 1.8%, the error of variance is 4%



```
/Users/qiongwang/PycharmProjects/511/proje
E[N60] = 285.4607
var[N60] = 81690.80365551

Process finished with exit code 0
```



5. Question 5

Analysis: In this accept-reject method to sample from p_j, p = [0.06, 0.06, 0.06, 0.06, 0.06, 0.15, 0.13, 0.14, 0.15, 0.13], auxiliary distribution q = [0.05] * 20. The probability of $q_j = 0.05$, and the maximum probability of p is 0.15, so, I set c = 0.15/0.05 = 3. Then I generate a uniform random variable U and a random integer from 1 to 20, if $p_j/c*q_j > U$, accepted. I simulated 10000 time and get the result, and plot the histogram and calculate sample mean, sample variance, and efficiency.

Code:

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

accepted = []
p = [0.06]*5+[0.15, 0.13, 0.14, 0.15, 0.13]+[0]*10
q = [0.05]*20
c = 3
N = 10000

for i in range(N):
    j = np.random.randint(20)
    u = np.random.uniform(0, 1)
    if u < p[j]/(c*q[j]):
        accepted.append(j+1)

plt.figure()
plt.hist(accepted, bins=range(21), rwidth=0.8, edgecolor='black',
align='left', label='Accept-Reject Sample')
plt.plot(range(1, 21), [x*N/c for x in p], marker='o', color='r',
label='distribution pj')
plt.xlabel('Random Variable')
plt.ylabel('Frequency')
plt.title('Histogram for Distribution of p')
plt.legend()
plt.show()

print('sample mean =', np.mean(accepted))
print('sample variance =', np.var(accepted, ddof=1))
print('efficiency =', len(accepted)/N)
```

Result:

sample mean = 6.425060679611651

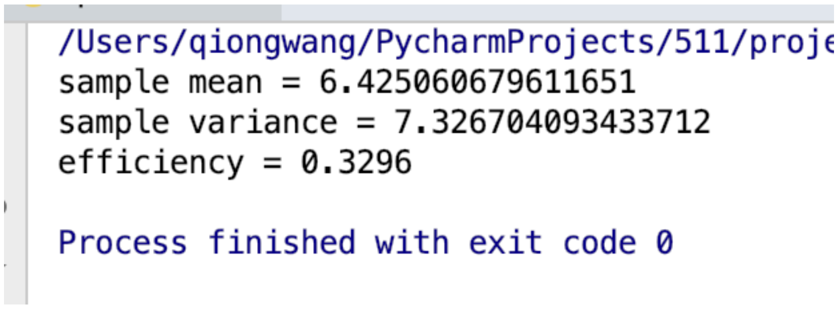
theoretical mean = 6.48

sample variance = 7.326704093433712

theoretical variance = 7.19

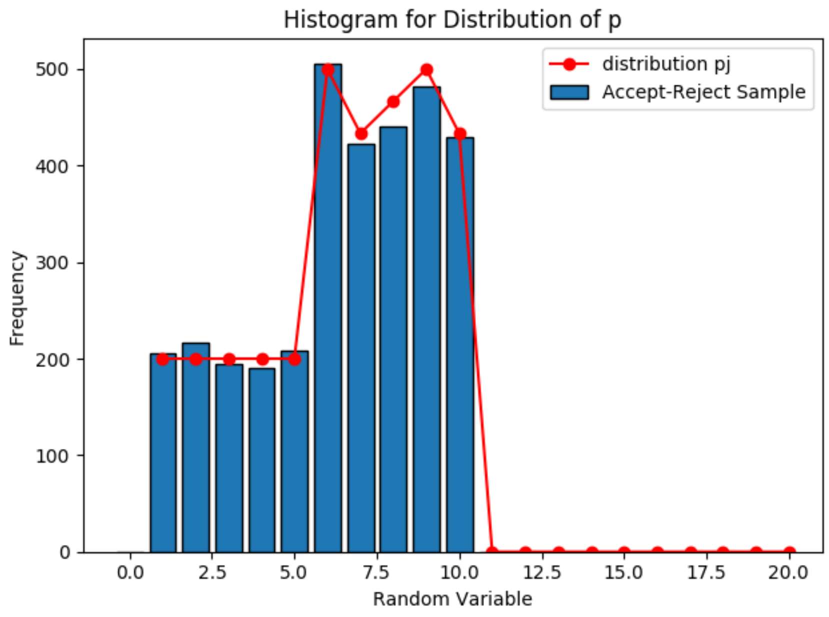
efficiency = 0.3296

theoretical efficiency = 0.33336



```
/Users/qiongwang/PycharmProjects/511/proje
sample mean = 6.425060679611651
sample variance = 7.326704093433712
efficiency = 0.3296

Process finished with exit code 0
```

In histogram, the red points are theoretical frequency of pj, the blue bar are experiment result, from the graph, I find that the result is good.


Histogram for Distribution of p
distribution pj
Accept-Reject Sample
Frequency
Random Variable