Project 3

Wenjun Li

February 11, 2020

Question 1

A components manufacturer delivers a batch of 125 microchips to a parts distributor. The distributor checks for lot conformance by counting the number of defective chips in a random sampling (without replacement) of the lot. If the distributor finds any defective chips in the sample they reject the entire lot. Suppose that there are six defective units in the lot of 125 microchips. Simulate the lot sampling to estimate the probability that the distributor will reject the lot if it tests five microchips. What is the fewest number of microchips that the distributor should test to reject this lot 95% of the time?

Results and Analysis

There are two sub questions in this experiment. First, we need to estimate the probability that the distributor will reject the lot. Second, we need to find the fewest number of microchips that the distributor need to test to reject the lot 95% of the time.

To do the experiment, I generate a sequence of 0, 1, ..., 125 as the 126 microchips and I view 0, 1, 2, 3, 4 and 5 as defective microchips. To check if the lot is rejected, I sample 5 numbers uniformly from 0, 1, ..., 125 without replacement and if any of them are less or equal to 5 (i.e. there are defective microchips in the sample), then the lot is rejected; otherwise, the lot is accepted.

For the first part, I simulate this experiment for 10 trails and there are 1000 times simulation in each trail (i.e. there are 10,000 single simulation in total). I compute the reject rate for each trail and then take an average on 10 trails to output the final reject rate. The 10 trail reject rate are: 20.9%, 21.8%, 23.6%, 21.7%, 21.3%, 22.9%, 22.7%, 22.1%, 22.7%, 21.6%, and the averaged reject rate is:

$$\textbf{The final reject rate} = \textbf{22.13\%}$$

For the second part, in order to find the fewest number, I again simulate the experiment for 1000 times in each trail. In this 1000 simulations, I record the smallest sample amount that make the total number of 'reject' great than 950. The experiment is repeated for 10 trails and the final result is obtained by averaging the 10 trail results. Figure 1. displays the histogram of the results.

$$\textbf{The fewest number to reach 95\% reject rate} = \lceil 48.3 \rceil = \textbf{49}$$

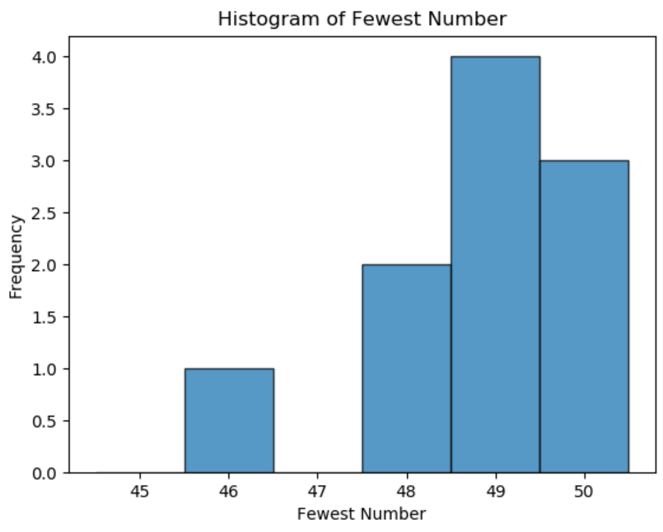


Figure 1: Histogram of The Fewest Number of 10 Trails.

Python Code

Listing 1: Code of Question 1

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


# part (a)
reject_hist = []
N = 1000
N_trails = 10
for j in range(N_trails):
    reject = 0
    for i in range(N):
        test_samples = random.sample(range(126), 5)
        if np.min(test_samples) <= 5:
            reject += 1
    reject_prob = reject / N
    reject_hist.append(reject_prob)

print(reject_hist)
print(np.mean(reject_hist))


```

```
# part (b)
fewest_number = []
for j in range(N_trails):           # loop to repeat the experiment several times
    reject = 0
    for i in range(6, 126):            # loop to find the fewest number, starting
        from 6
        reject = 0
        for k in range(N):
            test_samples = random.sample(range(126), i)     # sample i chips
            if np.min(test_samples) <= 5:
                reject += 1
        if reject >= 950:
            fewest_number.append(i)
            print(i)
            break

plt.figure(1)
bins = np.arange(min(fewest_number)-1, max(fewest_number)+2, 1) - 0.5
plt.hist(fewest_number, bins, edgecolor='black', alpha=0.75)
plt.title('Histogram of Fewest Number')
plt.ylabel('Frequency')
plt.xlabel('Fewest Number')
plt.show()
```

Question 2

Suppose that 120 cars arrive at a freeway onramp per hour on average. Simulate one hour of arrivals to the freeway onramp: (1) subdivide the hour into small time intervals ($<$ 1 second) and then (2) perform a Bernoulli trial to indicate a car arrival within each small time-interval. Generate a histogram for the number of arrivals per hour. Repeat the counting experiment by sampling directly from an equivalent Poisson distribution by using the inverse transform method (described in class). Generate a histogram for the number of arrivals per hour using this method. Overlay the theoretical p.m.f. on both histograms. Comment on the results.

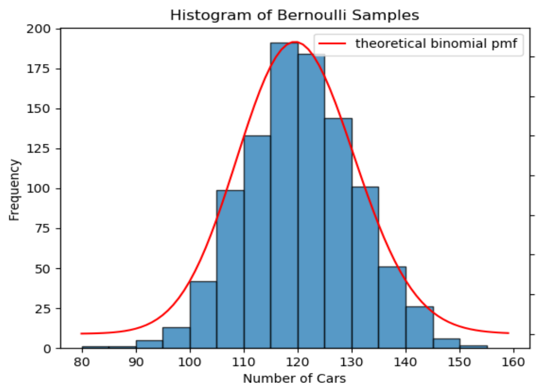
Results and Analysis

This question can be divided into two parts. First, we need to compute the theoretical probability mass function (pmf) of Poisson distribution and Binomial distribution. In this question, I divide an hour into N = 5000 small time intervals and perform a Bernoulli trail in each time interval. For Poisson, λ = 120. For Bernoulli, p = λ / N = 0.024. To make the simulation more stable and reliable, I repeat the simulation for 1000 times and count the frequency of number of cars that arrive per hour.

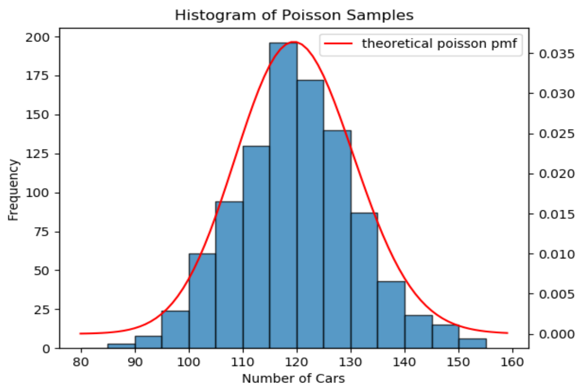
$$Poisson : \frac{\lambda^x * e^{-\lambda}}{x!} \tag{1}$$

$$Binomial : \lim_{n\to\infty} \binom{n}{k} * p^k * (1-p)^{n-k} \tag{2}$$

As we know, Binomial will converge to Poisson when n is infinity and n*p = λ. The results and plots are shown in Figure 2. As we can see from Figure 2, Binomial approximates Poisson very well when n is large (in this case, n=5000).



(a)



(b)

Figure 2: Histograms of (a)Binomial and (b)Poisson.

Python Code

Listing 2: Code of Question 2

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.special import comb


lamda = 120         # lambda of Poisson
trails = 1000       # number of simulation times
N = 5000           # number of small time intervals
p = lamda / N       # p in Bernoulli


# theoretical poisson pmf
p_pmf_range = range(lamda−40, lamda+40)
```

```
P_pmf = stats.poisson.pmf(p_pmf_range, lamda)


# theoretical bernoulli pmf
b_pmf_range = np.arange(lamda-40, lamda+40, 1)
B_pmf = [comb(N, k) * pow(p, k) * pow(1-p, N-k) for k in b_pmf_range]


# Poisson Samples
P_history = np.random.poisson(lamda, trails)

plt.figure(1)
bins = np.arange(80, 160, 5)
plt.hist(P_history, bins, edgecolor='black', alpha=0.75)
plt.title('Histogram of Poisson Samples')
plt.xlabel('Number of Cars')
plt.ylabel('Frequency')
ax = plt.twinx()
ax.plot(p_pmf_range, P_pmf, color='r', label='theoretical poisson pmf')
plt.legend()
plt.show()


# Bernoulli Approximation
B_history = []

for i in range(trails):
    u = np.random.rand(N, 1)
    bernoulliTrails = u < p
    B_history.append(np.sum(bernoulliTrails))

plt.figure(2)
bins = np.arange(80, 160, 5)
plt.hist(B_history, bins, edgecolor='black', alpha=0.75)
plt.title('Histogram of Bernoulli Samples')
plt.xlabel('Number of Cars')
plt.ylabel('Frequency')
ax = plt.twinx()
ax.plot(b_pmf_range, B_pmf, color='r', label='theoretical binomial pmf')
plt.legend()
plt.show()
```

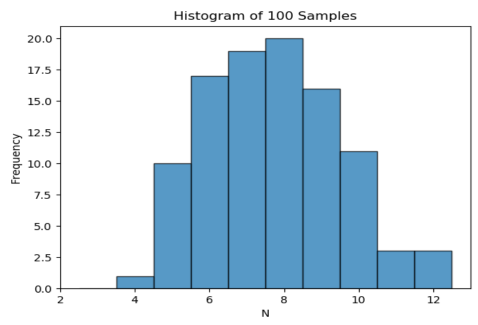
Question 3

Define the random variable $N = min\{n : \sum_{i=1}^{n} X_i > 4\}$ as the smallest number of uniform random samples whose sum is greater than four. Generate a histogram using 100, 1000, and 10000 samples for N. Comment on $E[N]$.

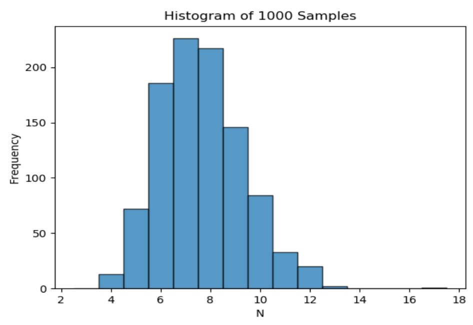
Results and Analysis

This question asks us to generate a uniformly distributed random variable until the sum of the sequence is greater than 4, and we need to repeat the experiment with 100, 1000 and 10000 samples. Since the random number is from standard uniform distribution, thus $E[x] = 0.5$. So, theoretically, we need at least $E[N] = 9$ random number to let their sum greater than 4. However, when N becomes larger, $E[N]$ will go to 8 because $E[8] = 8 * E[x] = 4$. Thus, $E[N]$ will finally converge between 8 and 9. The simulation results and plots are shown below.

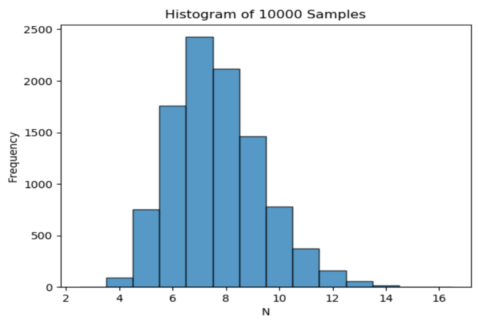
When N=100, E[N]=8.69
When N=1000, E[N]=8.664
When N=10000, E[N]=8.6753



(a)



(b)



(c)

Figure 3: Histograms of (a)100, (b)1000, and (c)10000 Samples.

Python Code

Listing 3: Code of Question 3

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


N = [100, 1000, 10000]
freq1 = []
freq2 = []
freq3 = []


# 100 trails
for j in range(N[0]):
    sum_ = 0
    for i in range(N[0]):
        x = np.random.uniform(0, 1)
        sum_ += x
        if sum_ > 4:
            freq1.append(i+1)
            break
# 1000 trails
for j in range(N[1]):
    sum_ = 0
    for i in range(N[1]):
        x = np.random.uniform(0, 1)
        sum_ += x
        if sum_ > 4:
            freq2.append(i+1)
            break
# 10000 trails
for j in range(N[2]):
    sum_ = 0
    for i in range(N[2]):
        x = np.random.uniform(0, 1)
        sum_ += x
        if sum_ > 4:
            freq3.append(i+1)
            break


plt.figure(1)
bins = np.arange(min(freq1)-1, max(freq1)+2) - 0.5
```

```python
plt.hist(freq1, bins, edgecolor='black', alpha=0.75)
plt.title('Histogram of {} Samples'.format(N[0]))
plt.xlabel('N')
plt.ylabel('Frequency')
plt.show()

plt.figure(2)
bins = np.arange(min(freq2)-1, max(freq2)+2) - 0.5
plt.hist(freq2, bins, edgecolor='black', alpha=0.75)
plt.title('Histogram of {} Samples'.format(N[1]))
plt.xlabel('N')
plt.ylabel('Frequency')
plt.show()

plt.figure(3)
bins = np.arange(min(freq3)-1, max(freq3)+2) - 0.5
plt.hist(freq3, bins, edgecolor='black', alpha=0.75)
plt.title('Histogram of {} Samples'.format(N[2]))
plt.xlabel('N')
plt.ylabel('Frequency')
plt.show()


# print E[N]
print('When N=100, E[N]={}'.format(np.mean(freq1)))
print('When N=1000, E[N]={}'.format(np.mean(freq2)))
print('When N=10000, E[N]={}'.format(np.mean(freq3)))
```

Question 4

Produce a sequence $\{X_k\}$ where $p_j = \frac{p}{j}$ for $j = 1, 2, ..., 60$ where p is a constant for you to determine. [This is equivalent to spinning the minute hand on a clock and observing the stopping position if $P[stoponminutej] = \frac{p}{j}$]. Generate a histogram. Define the random variable $N_j = min\{k : X_k = j\}$. Simulate sampling from N_{60}. Estimate $E[N_{60}]$ and $Var[N_{60}]$. Compare your estimates with the theoretical values.

Results and Analysis

This question ask us to determine p, and as we know, the summation of all probability is equal to 1, *i.e.* $\sum_{j=1}^{60} p_j = 1$. Next, I sample a x from the standard uniform distribution to simulate N_{60}. If $x < p/60$, then I record the count. I simulate the experiment for 10000 times and plots the histograms as shown in Figure 4.

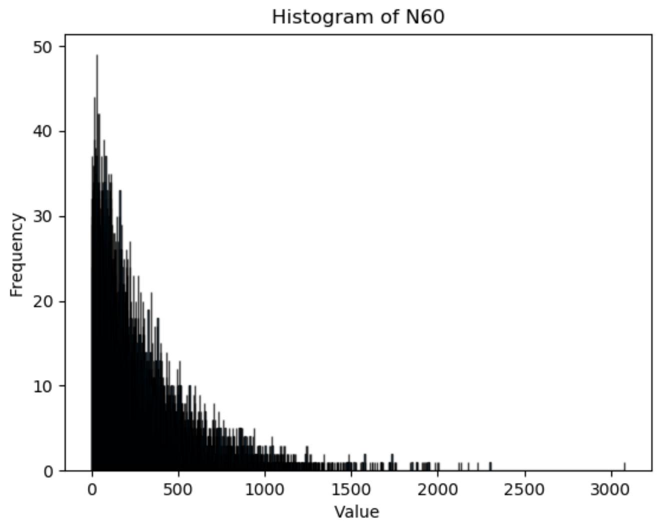


Figure 4: Histogram of N_{60} from 10000 Simulations.

Theoretically, $E[N_{60}] = \frac{1}{p_{60}} = \frac{1}{\frac{p}{60}} = 280.79$ and $Var[N_{60}] = 78563$. The comparison of theoretical values and sample values it listed below:

Theoretical: $E[N_{60}] = \mathbf{280.79}$**,** $Var[N_{60}] = \mathbf{78563}$
Sample: $E[N_{60}] = \mathbf{278.42}$**,** $Var[N_{60}] = \mathbf{76032}$

Python Code

Listing 4: Code of Question 4

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


# compute the value of p
sum_ = 0
for i in range(1, 61):
    sum_ += (1 / i)
p = 1 / sum_


# sampling from N60
history = []
N = 10000
for i in range(N):
    count = 0
    while True:
        x = np.random.uniform(0, 1)
        count += 1
        if x < p/60:
            history.append(count)
            break

print('p={}'.format(p))
print('E[60] = {}'.format(np.mean(history)))
print('V[60] = {}'.format(np.var(history)))

# plot
plt.figure(1)
bins = np.arange(min(history)-1, max(history)+2) - 0.5
plt.hist(history, bins, edgecolor='black', alpha=0.75)
plt.title('Histogram of N60')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

Question 5

Use the accept-reject method to sample from the following distribution p_j by sampling from the (non-optimal) uniform auxiliary distribution ($q_j = 0.05 for j = 1, 2, ..., 20$):

p1=p2=p3=p4=p5=0.06, p6=p9=0.15, p7=p10=0.13, p8=0.14

Generate a histogram and overlay the target distribution p_j. Compute the sample mean and sample variance and compare these values to the theoretical values. Estimate the efficiency of your sampler with the following ratio:

$$Efficiency = \frac{\#accepted}{\#accepted + \#rejected} \tag{3}$$

Compare your estimate of the efficiency to the theoretical efficiency given your choice for the constant c.

Results and Analysis

This question requires to use *Acceptance-Rejection Model* to generate discrete random numbers because it is difficult to obtain such numbers directly from p_i. The maximal probability in p_i is $p_6 = p_9 = 0.15$ and the minimal probability in q_j is 0.05. Thus I choose $c = 0.15/0.05 = 3$. Since the length of q_j is 20, which is longer than p_i, so, I pad p_i with ten zeros.

In the simulation, I first generate a random variable x from [0, 1] uniform distribution and then make it become a random integer in 1, 2, ..., 20. If c*x $<$ p/q, then accept the number; otherwise, reject the number. The theoretical efficiency is $1/c = 0.333$. In this experiment, I run the simulation for 10000 times and get the sample expectation and efficiency as below:

$$\mathbf{E[X] = 6.484}$$
$$\mathbf{Efficiency = 0.334}$$

The simulation result plot is shown in Figure 5, from which we can conclude that our *Acceptance-Rejection Model* generate the discrete random numbers very well, *i.e.* very close to the distribution of p_i.

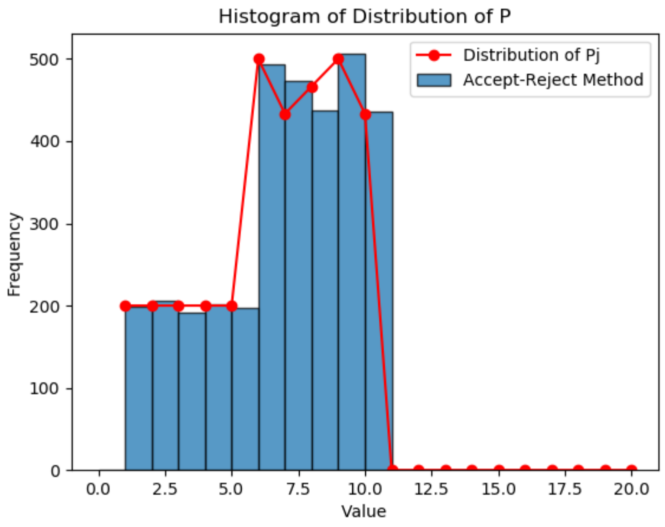


Figure 5: Histogram of Distribution of P

Python Code

Listing 5: Code of Question 5

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


p = np.array([0.06, 0.06, 0.06, 0.06, 0.06, 0.15, 0.13, 0.14, 0.15, 0.13, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0])
N = 10000
X = []

c = 3       # c = max(p) / min(q) = 0.15 / 0.05 = 3
```

```

for i in range(1, N):
    while True:
        j = int(1 + np.floor(20*np.random.rand()))      # get uniform j from {1, 2, ..., 20}
        if (c*np.random.rand()) < p[j-1]/0.05:
            X.append(j)
        break

print('Mean of X: {}'.format(np.mean(X)))
print('Efficiency = {}'.format(len(X)/N))

plt.figure(1)
plt.hist(X, bins=range(21), edgecolor='black', alpha=0.75, label='Accept-Reject Method')
plt.plot(range(1, 21), [x*N/c for x in p], marker='o', color='red', label='Distribution of Pj')
plt.title('Histogram of Distribution of P')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```