EE511 Assignment 2

Zechen Ha

February 5, 2020

Question 1

Simulate sampling uniformly (how many?) on the interval [-3,2].

Question 1a

Generate a histogram of the outcomes.

Code

It is easy to use numpy.random tools to generate a list of uniform random samples. The sample size is 1000. Use tools in matplotlib to draw histogram of the sample.

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

sample_size = 1000
sample_num = 1
sample = np.random.rand(sample_size)*5-3
# print(sample)

plt.hist(sample,rwidth=0.9,bins=10,edgecolor='black',alpha=0.8,range
    =(-3,2))
plt.xlim(min(sample),max(sample))
plt.grid()
plt.xlabel('random variable value',fontsize=12)
plt.ylabel('count',fontsize=12)
plt.title('Histogram for outcomes',fontsize=15)
plt.show()
```

Listing 1: Question1(a) Code

Result

It is obvious to conclude from the figure that the uniform random variable has almost the same number of samples in each bins, and it is close to 100. I chose 1000 samples and it is large enough to show the distribution.

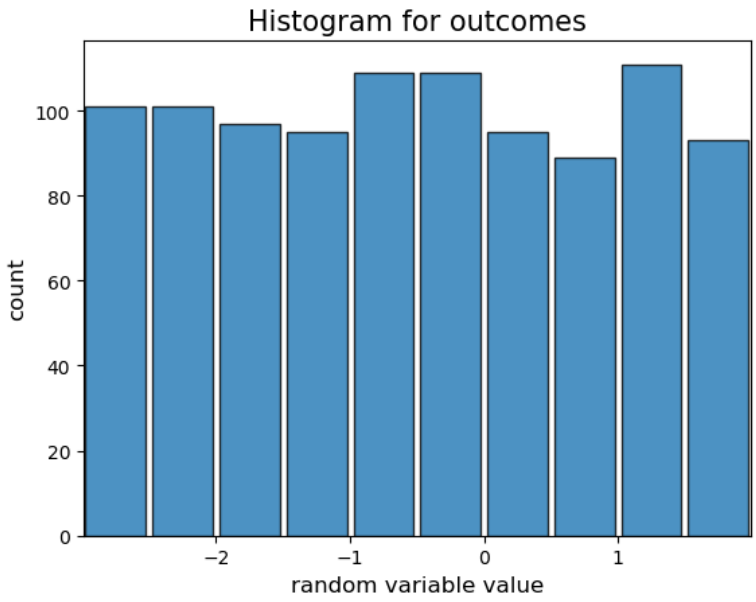

Figure 1: Histogram of uniform random sample

Question 1b

Compute the sample mean and sample variance for your samples. How do these values compare to the theoretical values? If you repeat the experiment will you compute a different sample mean or sample variance?

Code

Generate a list of uniform samples range from -3 to 2, and calculate the sample mean and sample variance.

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

sample_size = 1000
sample_num = 1
# sample = np.random.rand(sample_size)*5-3
sample_mean = []
sample_variance = []

for samples in range(sample_num):
    sample = np.random.rand(sample_size)*5-3
    mean = np.sum(sample)/sample_size
    sample_mean.append(mean)
    variance = np.sum(np.square(sample-mean))/(sample_size-1)
    sample_variance.append(variance)
```

```
16
17 print(sample_mean)
18 print(sample_variance)
```

Listing 2: Question1(b) Code

Result

From statistic theory, we have known that uniform random variable in the interval [a,b]. It has the mean of $(a+b)/2$ and variance of $(b-a)^2/12$. For the question, the expected value of mean and variance are:

Mean=$(2-3)/2 = -0.5$

Variance=$(2-3)^2/12 = 2.0833$

The formula of the sample mean and sample variance are:

Sample mean:

$$\bar{X}_n = \frac{1}{n}\sum_{k=1}^{n} X_k$$

Sample variance:

$$S_x^2 = \frac{1}{n-1}\sum_{k=1}^{n}(X_k - \bar{X}_n)^2$$

I get the table 1 after calculation.

-	Sample(1000)			Theoretical
Mean	-0.572	-0.452	-0.442	-0.5
Variance	2.036	2.137	2.076	2.0833

Table 1: Comparison table

We can conclude from the table that the sample mean and sample variance are very close to the theoretical value when I have a size of 1000 sample, which is large enough to show statistically. When I repeat the experiment, the value changes but it still close to the theoretical value.

Question 1c

Compute the bootstrap confidence interval (what width?) for the sample mean and sample standard deviation.

Code

According to the defination of bootstrap method, generate 1000 samples of size 1000. Calculate the sample mean and sample variance of each sample and sort them in a non-descending

way. When I want to get the 95% confidence interval, I choose the 26th and 975th in the sorted list as the confidence interval boundary.

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

sample_size = 1000
sample_num = 1000
sample_mean = []
sample_variance = []
sample_standard_deviaion = []

for samples in range(sample_num):
    sample = np.random.rand(sample_size)*5-3
    mean = np.sum(sample)/sample_size
    sample_mean.append(mean)
    variance = np.sum(np.square(sample-mean))/(sample_size-1)
    sample_variance.append(variance)
    sample_standard_deviaion.append(np.sqrt(variance))

sample_mean.sort()
sample_variance.sort()
sample_standard_deviaion.sort()

# 95% confidence interval bootstrap
print(sample_mean[25],sample_mean[974])
print(sample_variance[25],sample_variance[974])
print(sample_standard_deviaion[25],sample_standard_deviaion[974])
```

Listing 3: Question 1(c) Code

Result

The result for this question is:

-	95% CI
Sample mean	[-0.595, -0.417]
Sample standard deviation	[1.402, 1.483]

Table 2: 95% confidence interval result

The theoretical value of sample standard deviation is

$$S_x = \sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(X_k - \bar{X}_n)^2} = 1.442$$

We can see that the statistical mean and standard deviation fall into the confidence interval. With 95% confidence interval, I am 95% certain that the statistical value will lie within the interval.

Question 2

Produce a sequence X by drawing samples from a standard uniform random variable.

Question 2a

Compute $Cov[X_k, X_{k+1}]$. Are X_k and X_{k+1} uncorrelated? What can you conclude about the independence of X_k and X_{k+1}?

Code

Generate a sequence of uniform random variable of size 1000, and shift rightward by one step. Calculate the co-variance with the formula:

$$Cov[X, Y] = E[(X - \mu_x)(Y - \mu_y)]$$

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

sample_size = 1000
X_k = np.random.rand(sample_size)
X_k1 = X_k[1:]
X_k = X_k[:-1]
sample_size = sample_size-1

X_k_mean = sum(X_k)/sample_size
X_k1_mean = sum(X_k1)/sample_size
cov = sum((X_k-X_k_mean)*(X_k1-X_k1_mean))/sample_size
print(cov)
```

Listing 4: Question 2(a) Code

Result

The result from the code is:

Sample co-variance between X_k and X_{k+1} = 0.0003

It is easy to understand with statistic theory. The X_k is independent with X_{k+1}, and is two random variable are independent, they are uncorrelated. The experiment has shown the theory. The co-variance is very close to zero, impling that they are uncorrelated.

Question 2b

Compute a new sequence Y where: $Y[k] = X[k] - 2X[k-1] + 0.5X[k-2] - X[k-3]$. Assume $X[k] = 0$ for $k \leq 0$. Compute $Cov[X_k, Y_k]$. Are X_k and Y_k uncorrelated?

Code

Generate a sequence of uniform random variables of size 1000, and shift rightward to generate $X[k-1], X[k-2], X[k-3]$,and get Y_k with the given formula. Calculate the co-variance of X and Y with the formula:

$$Cov[X, Y] = E[(X - \mu_x)(Y - \mu_y)]$$

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

sample_size = 1000
X_k = np.random.rand(sample_size)
X_k1 = X_k[1:-2]
X_k2 = X_k[2:-1]
X_k3 = X_k[3:]
X_k = X_k[:-3]
Y = X_k-2*X_k1+0.5*X_k2-X_k3
sample_size = sample_size-3

X_k_mean = sum(X_k)/sample_size
Y_mean = sum(Y)/sample_size
cov = sum((X_k-X_k_mean)*(Y-Y_mean))/sample_size
print(cov)
```

Listing 5: Question 2(b) Code

Result

The result of code is:

The sample co-variance between X_k and Y_k=0.09

The result is not close to zero, so they are correlated.

Question 3

Let M = 10. Simulate (uniform) sampling with replacement from the outcomes 0, 1, 2, 3, . . . , M-1.

Question 3a

Generate a histogram of the outcomes.

Code

Generate a sequence of uniform integers range of 0,1,2,3,...,M-1, and count the number of each elements. Draw the histogram of number of uniform integer.

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

M = 10
sample_size = 1000
sample = np.random.randint(M,size=sample_size)

plt.hist(sample,bins=10,rwidth=0.9,alpha=0.8,edgecolor='black')
plt.xlabel('Outcome Value',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.title('Histogram of uniform int random variable',fontsize=15)
plt.grid(axis='y',alpha=0.5)
plt.show()
```

Listing 6: Question 3(a) Code

Result

The histogram of the number of uniform integer random variable is:

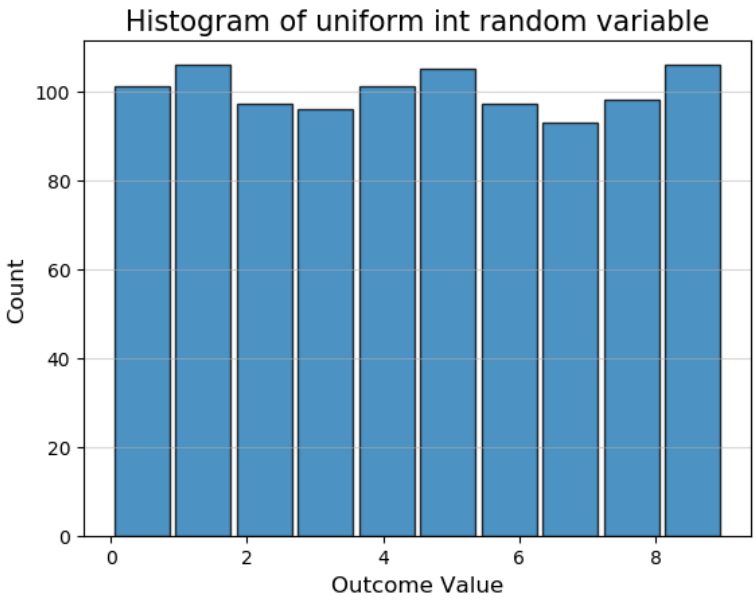

Figure 2: Histogram of uniform integer number

The size of the sample is 1000, and number of bins is 10. The value of each bin is close to

100, which fits the uniform distribution. We will calculate the good-of-fit in a more explicit way.

Question 3bc

b. Perform a statistical goodness-of-fit test to conclude at the95% confidence level if your data fits samples from a discrete uniform distribution 0, 1, 2, . . . , 9.
c. Repeat (b) to see if your data (the same data from b) instead fit an alternate uniform distribution 1, 2, 3,. . . , 10.

Code

Use tools in sci.stats to calculate the p-value and chi square value of the chi square test. The observed number can be counted from the sample, and the expected number is all 100.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as scis

M = 10
sample_size = 1000

#Question number 3(b) Code
sample = np.random.randint(10,size=sample_size)
#Question number 3(b) Code
#sample = np.random.randint(1,11,size=sample_size)

observed = []
for i in range(10):
    observed.append(sample[sample == i].size)
expected = 100*np.ones((10),dtype=np.int)

(kvalue,pvalue)=scis.chisquare(observed,expected)
print(kvalue,pvalue)
```

Listing 7: Question 3(bc) Code

Result

The solution of the two questions are similar. I just calculate the chi square value using the same observed list and different expected list. The result is as follow:

-	Question b	Question c
p-value	0.429	10^{-19}
chi square value	9.08	110.68

Table 3: Result for question bc

To analysis the result, I will concentrate on two values. In question b, the p value is 0.429, which is larger than 0.05, so the sample fits the uniform distribution range in (0,10) well. In another aspect, the chi square value is 9.08. After checking the table of critical value of chi square, I get the $\alpha = 0.05, degree = 9$ chi square value is equal to 16.919, which is larger than 9.08. We can also get the same conclusion.

In problem c, the p value is $10^(-19)$, which is much smaller than 0.05, so the chi square test fails. Also, the chi square value is 110.68, which is larger than 16.919, so we can get the same conclusion.