Karandikar_Prajakta_project2

February 5, 2020

1 Problem 1

2 Explanation:

Here, we have taken 1000 random samples which are uniformly distributed between -3 and 2. These unifrom random samples are generated using the np.random.uniform() function. Thereafter, we plot a histogram of these samples which is obtained using the function plt.hist(). From the histogram, we can therefore determine whether the samples are uniformly distributed or not.

3 Code:

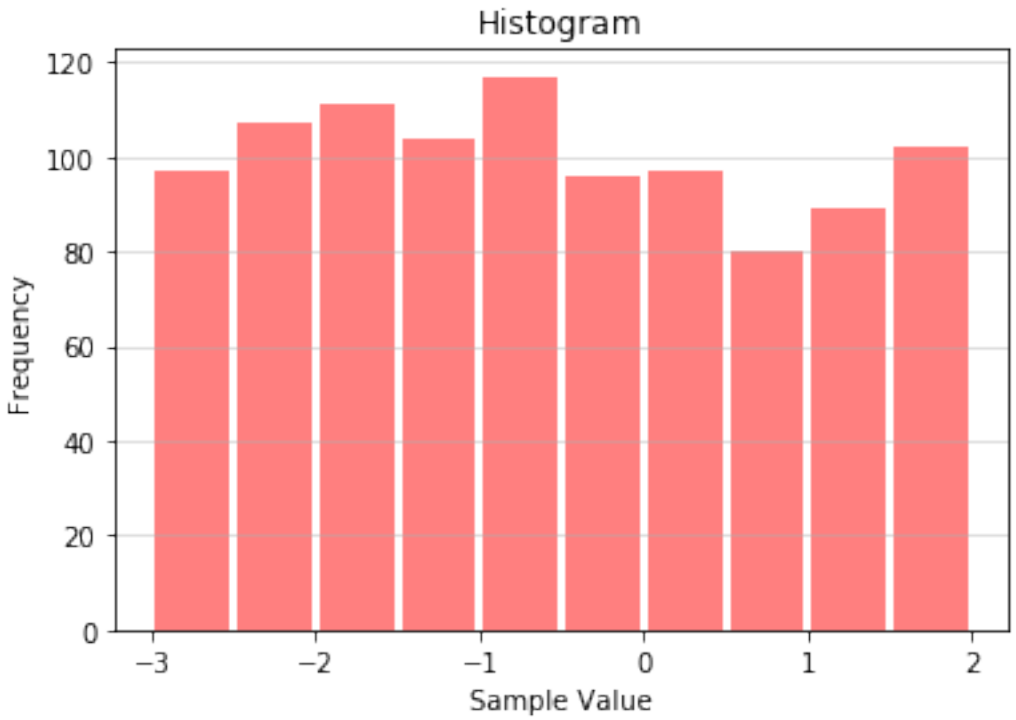
[6]:
```
import numpy as np
import scipy
import matplotlib.pyplot as plt
```

[ ]:
```
# Q1 part a
```

[51]:
```
#Generate the sample list
NumofSamples = 1000
a= -3
b= 2
Sample =np.array(np.random.uniform(a,b,NumofSamples))
```

[8]:
```
n, bins, patches = plt.hist(Sample, bins=10, color='red',
                            alpha=0.5, rwidth = 0.9)
plt.grid(axis='y', alpha=0.5)
plt.xlabel('Sample Value')
plt.ylabel('Frequency')
plt.title('Histogram')
```

[8]:
```
Text(0.5, 1.0, 'Histogram')
```



4 Q1 part b

[46]:
```
#find the mean and variance from samples as well as theoretical
sample_500 = np.array(np.random.uniform(a,b,500))
sample_5000 = np.array(np.random.uniform(a,b,5000))
sample_10000 = np.array(np.random.uniform(a,b,10000))
Sample_mean_1000 = np.mean(Sample)
sample_mean_500 = np.mean(sample_500)
sample_mean_5000 = np.mean(sample_5000)
sample_mean_10000 = np.mean(sample_10000)
Mean_th = (b+a)/2
sample_var_500 = np.var(sample_500)
Sample_var_1000 = np.var(Sample)
sample_var_5000 = np.var(sample_5000)
sample_var_10000 = np.var(sample_10000)
Var_th = ((b-a)**2)/12
print('Theoretical Mean = {}'.format(Mean_th))
print('Mean of the 500 Sample = {}'.format(sample_mean_500))
print('Mean of the 1000 Sample = {}'.format(Sample_mean))
print('Mean of the 5000 Sample = {}'.format(sample_mean_5000))
print('Mean of the 10000 Sample = {}'.format(sample_mean_10000))
print('Theoretical Variance = {}'.format(Var_th))
```

```
print('Variance of the 500 Samples = {}'.format(sample_var_500))
print('Variance of the 1000 Samples = {}'.format(Sample_var))
print('Variance of the 5000 Samples = {}'.format(sample_var_5000))
print('Variance of the 10000 Samples = {}'.format(sample_var_10000))
```

```
Theoretical Mean = -0.5
Mean of the 500 Sample = -0.5995112021462796
Mean of the 1000 Sample = -0.5719247657727567
Mean of the 5000 Sample = -0.4909303518548999
Mean of the 10000 Sample = -0.5111554181396285
Theoretical Variance = 2.0833333333333335
Variance of the 500 Samples = 2.232582183410149
Variance of the 1000 Samples = 2.049279860869078
Variance of the 5000 Samples = 2.105207829219062
Variance of the 10000 Samples = 2.0640226600464397
```

As observed from the results obtained above, it can be concluded that as the number of samples taken increase, the difference between the theoretical and the calculated mean decreases. Moreover, since the data is being randomly generated, it is likely that the mean and variance of the data generated each time will vary.However, the theoretical mean and variance of the same interval is the same as the interval remains same.

5 Q1 part c

[54]:
```
import bootstrapped.bootstrap as bts
import bootstrapped.stats_functions as bts_stat

mean_1 = bts.bootstrap(Sample, stat_func = bts_stat.mean)
print(mean_1)
var_1 = bts.bootstrap(Sample, stat_func = bts_stat.std)
print(var_1)
```

```
-0.47422369266867037    (-0.5663347827829985, -0.38329044043533833)
1.4750399527554632    (1.437189582678026, 1.5160970918134513)
```

6 Problem 2

7 Explanation:

Here, we are generating a random sequence of 1000 samples. In order to compute the covariance between Xk and X(k+1), we are shifting the generated random sequence by 1 to the right using the np.roll() function. Elements from the original and shifted sequence are therefore compared to calculate the covariance using the np.cov() function.

8 Q2 part a

[55]:
```
# define the number of samples and generate a uniform random vector
N = 10000
X = (np.random.rand(1,N))
#Compute covariance
X_k = (np.roll(X,1))
covar = np.cov(X,X_k)
print('The Covariance of X_k,X_k+1 is:')
print(covar)
```

```
The Covariance of X_k,X_k+1 is:
[[0.08450312 0.00090518]
 [0.00090518 0.08450312]]
```

In the output obtained above, we can observe that the values of covariance are very close to 0. As a result, the random variables are uncorrelated. However, they need not be independent. For independence, the correlation coefficient has to be 0.

9 Q2 part b

[56]:
```
X_k_1 =np.array(np.roll(X,-1))
X_k_2 =np.array(np.roll(X,-2))
X_k_3 =np.array(np.roll(X,-3))
Y =(X -(2*X_k_1) + ((X_k_2)/2) -(X_k_3))
print('The covariance of COV[X_k, Y_k] is ')
Covar = np.cov(X,Y)
print(Covar)
```

```
The covariance of COV[X_k, Y_k] is
[[0.08450312 0.08288836]
 [0.08288836 0.52368792]]
```

[ ]:
```
The output obtained above shows that all execpt one value are close to 0. This␣
 ↪implies that X and Y are uncorrelated.
```

10 Problem 3

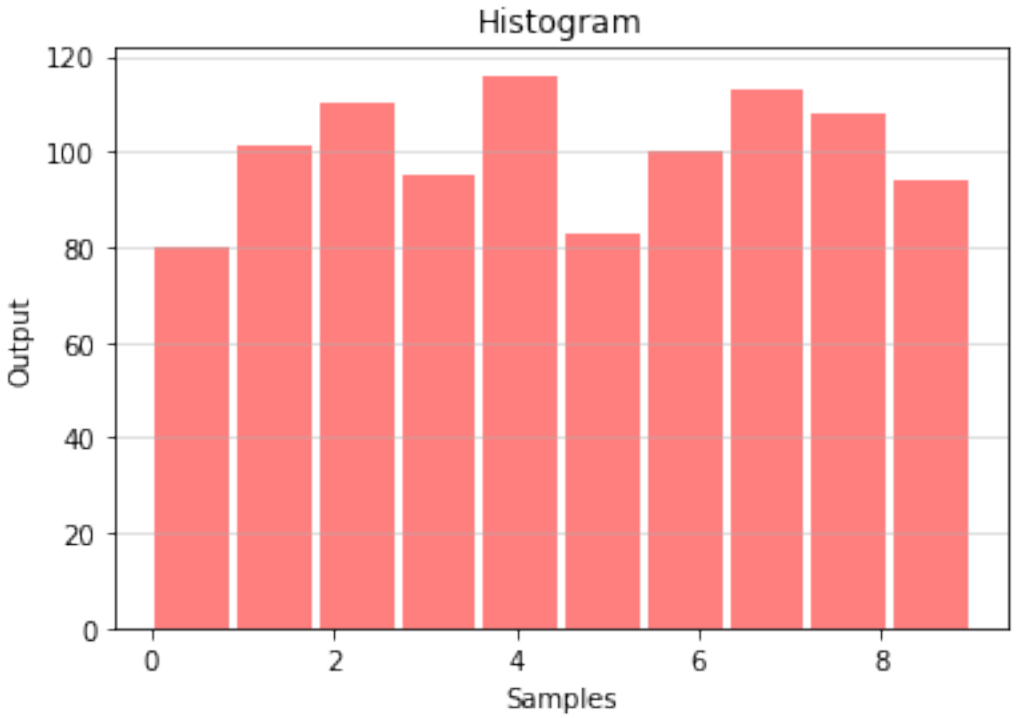
11 Q3 part a

In this part, 1000 random values are generated. Since we want sampling with replacement, we are using the np.random.randint() function. Thereafter, the histogram for these sample values is generated.

[57]:
```
M= 10
num = 1000
```

```
samples = np.random.randint(0,M,num)
n, bins, patches = plt.hist(samples, bins=10, color='red',
                            alpha=0.5, rwidth = 0.9)
plt.grid(axis='y', alpha=0.5)
plt.xlabel('Samples')
plt.ylabel('Output')
plt.title('Histogram')
```

[57]:
```
Text(0.5, 1.0, 'Histogram')
```



12 Q3 part b

The chi-squared test is used for the goodness of the fit where the chi-square value is computed from the generated data.

[60]:
```
#find observed and expected values
num=1000
M=10
print('Observed values:')
print(n)

Expectedval = num/M
print("Expected value:")
```

```
print(Expectedval)

from scipy import stats
chisqrd = np.sum(((n-Expectedval)**2)/Expectedval)
c = stats.chi2.cdf(chisqrd,10)
chipdf = 1 - c
print('Chi-squared:')
print(chipdf)
```

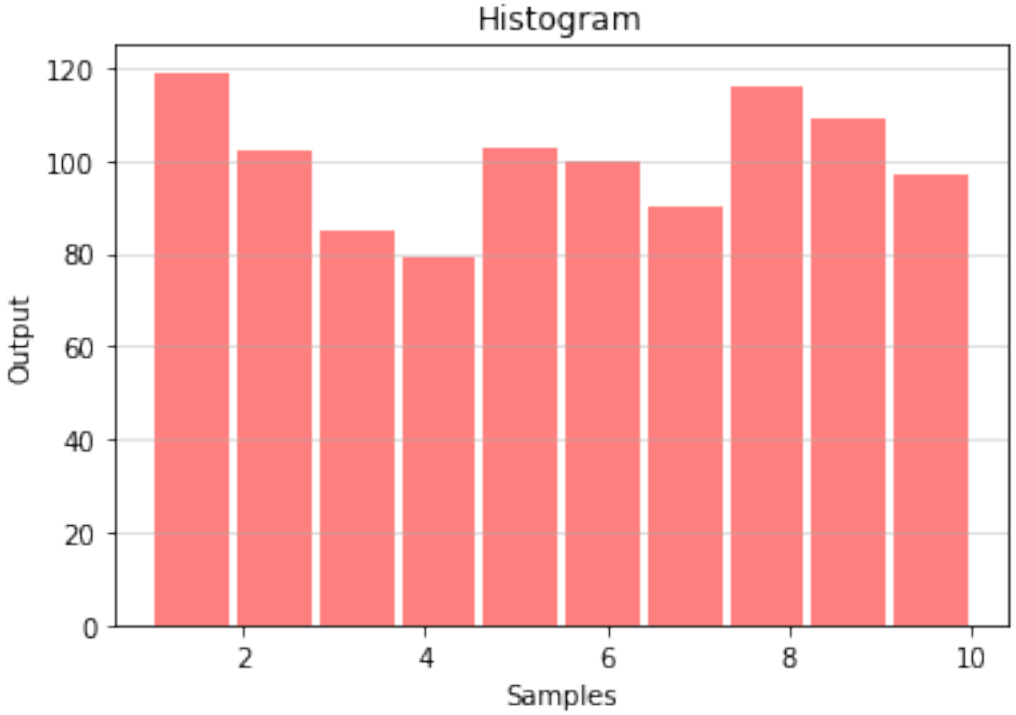
```
Observed values:
[ 80. 101. 110.  95. 116.  83. 100. 113. 108.  94.]
Expected value:
100.0
Chi-squared:
0.20215902661636254
```

13 Q3 part c

[62]:
```
#create new sample
num= 1000
M=10
new_samples = np.random.randint(1,11,num)
n1, bins, patches = plt.hist(new_samples, bins=10, color='red',
                             alpha=0.5, rwidth = 0.9)
plt.grid(axis='y', alpha=0.5)
plt.xlabel('Samples')
plt.ylabel('Output')
plt.title('Histogram')

#find observed and expected values
print('Observed values:')
print(n1)

Expectedval = num/M
print("Expected value:")
print(Expectedval)

from scipy import stats
new_chisqrd = np.sum(((n1-Expectedval)**2)/Expectedval)
c1 = stats.chi2.cdf(new_chisqrd,10)
new_chipdf = 1 - c1
print('Chi-squared New:')
print(new_chipdf)
```

```
Observed values:
[119. 102.  85.  79. 103. 100.  90. 116. 109.  97.]
Expected value:
```

```
100.0
Chi-squared New:
0.13724998184204107
```



[ ]: