Data Analysis in Astronomy and Physics (SoSe22)

Exercise Set 4

Submission by: Mahak Sadhwani, Xiongxiao Wang, Sakshi Pahujani

Due: **10:00 2 May 2022**
Discussion: **13:00 6 May 2**
Online submission at via in the directory Exercises / Übungen -> Submission of Exercises / Rückgabe des Übungsblät

1. Samples & Sampling Simulation [60 points]

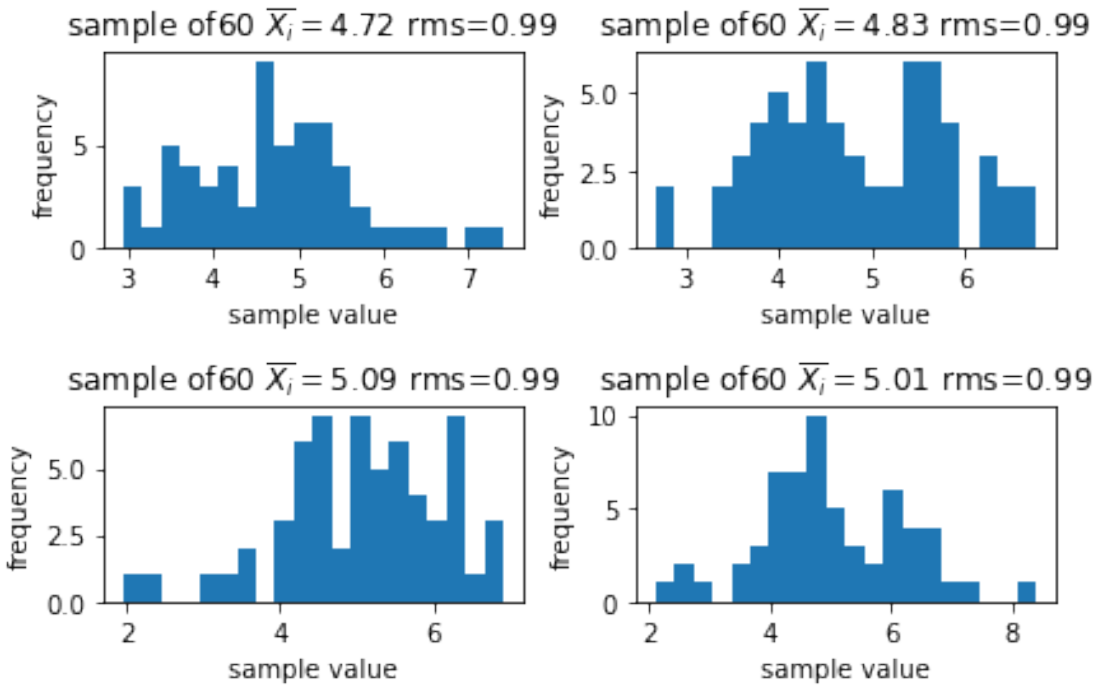
Write a numerical simulation code that reproduces the behaviour of the following simulation. Draw N samples of size n from a population with μ = 5 and σ = 1. Compute the confidence intervals (CI) of each sample distribution and the sampling distribution and visualize it in a similar manner. Test your simulation against your theoretical expectations. Make sure to test your result and to confirm that the result is as expected. (i.e. About 5 % of the samples not overlapping with the confidence interval, or 5% of all simulations with the population mean not part of the confidence interval.) **60 Points**

> According to the question, we draw N=100 samples of size n=40 from a normal distribution with $\mu=5$ and $\sigma=1$ by function **data = np.random.normal(mu,sigma,(N,n))** .

```
import numpy as np
import array
from scipy.stats import t
import matplotlib.pyplot as plt
from math import sqrt

mu, sigma = 5,1
N, n=100, 60
dof = n-1
confidence = 0.95
t_crit = np.abs(t.ppf((1-confidence)/2,dof))
print(t_crit)
CI=[]
data = np.random.normal(mu,sigma,(N,n))
f,a = plt.subplots(2,2)
a = a.ravel()
sample_means=[]
for idx,ax in enumerate(a) :
    m = data[idx].mean()
    s = data.std()
    CI.append((m-s*t_crit/sqrt(n),m+s*t_crit/sqrt(n)))
    sample_means.append(m)
    ax.hist(data[idx],bins=20)
    ax.set_title('sample of'+str(n)+' $\
overline{X_i}=$'+str(round(m,2))+' rms='+str(round(s,2)))
```

```
    ax.set_xlabel('sample value')
    ax.set_ylabel('frequency')
for idx in range(4,N):
    m = data[idx].mean()
    s = data.std()
    CI.append((m-s*t_crit/sqrt(n),m+s*t_crit/sqrt(n)))
    sample_means.append(m)

plt.tight_layout()
```

```
2.00099537704821
```



The figures above are the first four examples of sample distribution of size 60, $\overline{X_i} \approx \mu = 5, rms \approx \sigma = 1$,which confirm that the result is as expected

```
samples_means=np.array(sample_means)
```

To find the sampling distribution, we use the means of 100 samples above as sample and calculate the mean and standard error of sampling distribution.

The formula to calculate mean and standard error of sampling distribution is

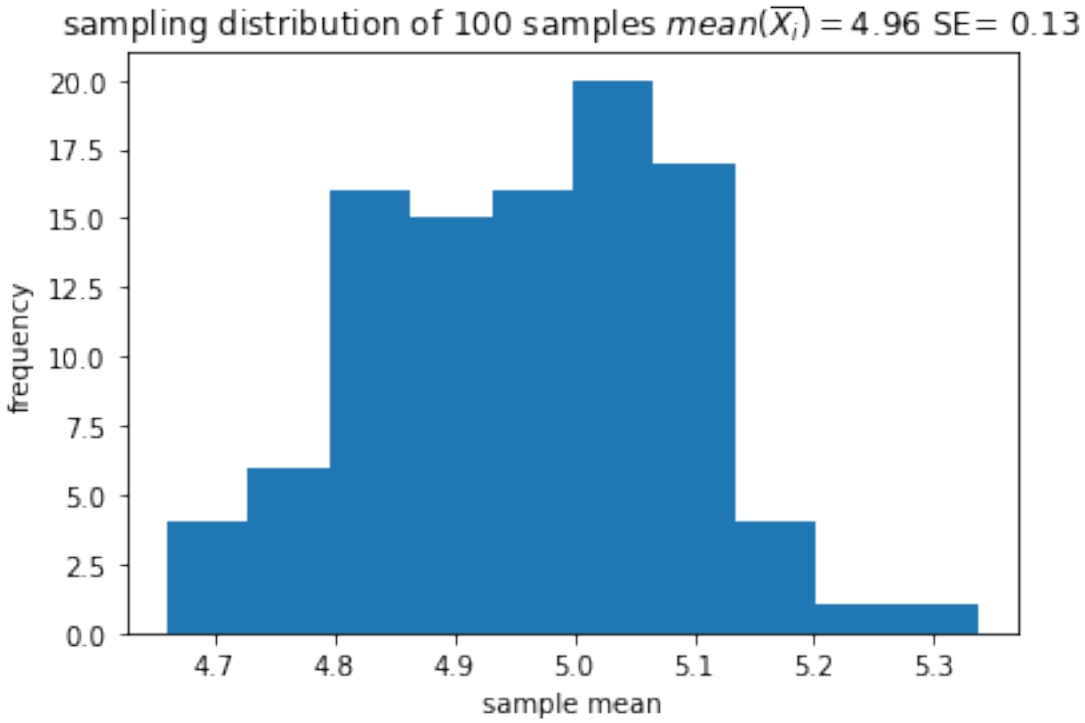
$$mean\left(\overline{X_i}\right)=\sum\frac{\overline{X_i}}{N}$$

$$SE(standarderror)=\sqrt{\sum\frac{\left(\overline{X_i}-mean\left(\overline{X_i}\right)\right)^2}{N}}$$

```
plt.hist(samples_means,bins=10)
print(samples_means.mean(),samples_means.std())
sampling_mean= samples_means.mean()
sampling_std= samples_means.std()
plt.title('sampling distribution of '+str(N)+' samples $mean(\
overline{X_i})=$'+str(round(sampling_mean,2))+' SE=
'+str(round(sampling_std,2)))
plt.xlabel('sample mean')
plt.ylabel('frequency')
```

```
4.963513661848113 0.12835408598660067
```

```
Text(0, 0.5, 'frequency')
```



The figure above is sampling distribution, the sampling distribution $mean(\overline{X_i}) \approx \mu$, and $SE \ll \sigma$

> To calculate the confidence intervals (CI) of each sample distribution, we set confidence =0.95,

Step 1: Subtract 1 from your sample size. 60 – 1 = 59. This gives you degrees of freedom, which you'll need in step 3.

Step 2: Subtract the confidence level from 1, then divide by two. (1 – .95) / 2 = 0.025

Step 3: use the function **np.abs(t.ppf((1-confidence)/2),dof)** to calculate the t. For 39 degrees of freedom (df) and α = 0.025, my result t is 2.00099537704821.

step 4: to calculate the confidence interval, we need to use the formula $\overline{X}\pm t \frac{s}{\sqrt{n}}$,where s is standard deviation of each sample and n is the size of sample. The corresponding $t_crit/sqrt(n)$,m+s$t_crit/sqrt(n)))

```
CI.append((mu-sqrt(sigma)*t_crit/sqrt(dof),mu+sqrt(sigma)*t_crit/
sqrt(dof))) #theoretical expectations of confidence interval
samples_means=np.append(samples_means,5)
CI
```

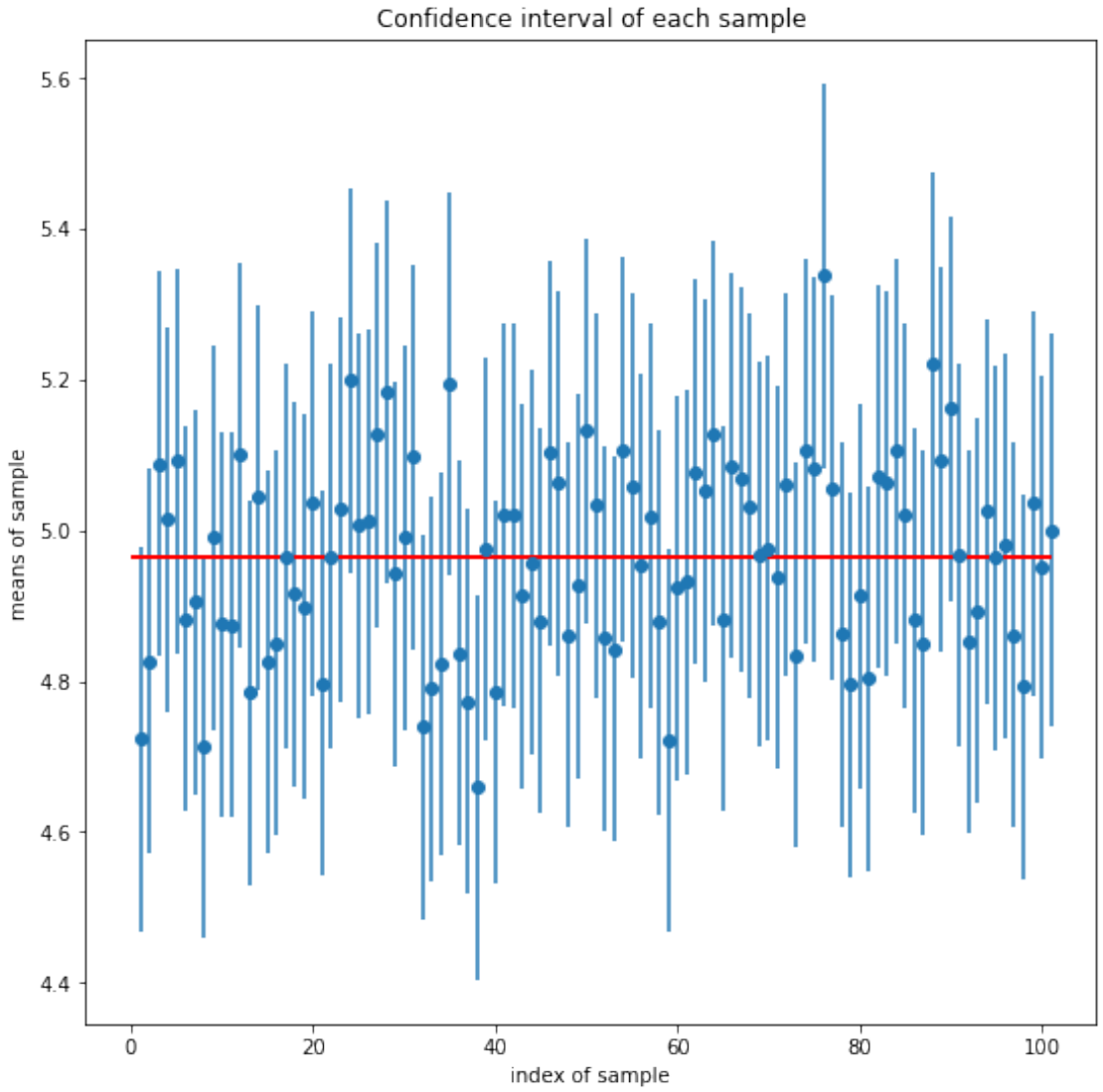
```
[(4.46833332821212, 4.977986519982469),
 (4.571758824939736, 5.081412016710085),
 (4.833262625332127, 5.342915817102476),
 (4.7591953002559535, 5.268848492026303),
 (4.836555074379654, 5.346208266150003),
 (4.627498753122494, 5.137151944892843),
 (4.6502295778130645, 5.1598827695834135),
 (4.458512515946065, 4.968165707716414),
 (4.735331089839731, 5.24498428161008),
 (4.620392393032718, 5.130045584803067),
 (4.619496648050337, 5.129149839820686),
 (4.845137114452245, 5.354790306222594),
 (4.529195859843986, 5.038849051614335),
 (4.788803011083813, 5.298456202854162),
 (4.570594971959064, 5.080248163729413),
 (4.5958308012222595, 5.105483992992609),
 (4.710664098604986, 5.220317290375335),
 (4.660844370818553, 5.170497562588902),
 (4.642906724127899, 5.152559915898248),
 (4.780846165403819, 5.290499357174168),
 (4.54226664496604, 5.051919836736389),
 (4.709914215889232, 5.219567407659581),
 (4.772839625927119, 5.282492817697468),
 (4.943377989771953, 5.453031181542302),
 (4.751253031643224, 5.260906223413573),
 (4.756844620395483, 5.266497812165832),
 (4.872099939452963, 5.381753131223312),
 (4.9285820031973415, 5.438235194967691),
 (4.6870864881570204, 5.1967396799273695),
 (4.735006836157755, 5.244660027928104),
 (4.841782756750226, 5.351435948520575),
 (4.484273692571739, 4.993926884342088),
 (4.535242238499422, 5.044895430269771),
 (4.567654985124259, 5.077308176894608),
 (4.939228927702278, 5.448882119472627),
 (4.5819088552268985, 5.091562046997248),
 (4.517702809354113, 5.027356011244625),
 (4.404564177672076, 4.914217369442425),
 (4.720117481552046, 5.229770673322395),
 (4.530374012414462, 5.040027204184811),
```

(4.76566765056951, 5.275320842339859),
(4.764908763532497, 5.274561955302846),
(4.6577556964925195, 5.167408888262869),
(4.702681313570322, 5.212334505340671),
(4.624507203627497, 5.134160395397846),
(4.847257573327668, 5.356910765098017),
(4.807782094695013, 5.317435286465362),
(4.606192193817476, 5.115845385587825),
(4.670836917097734, 5.180490108868083),
(4.877438129281651, 5.387091321052),
(4.778361527087814, 5.288014718858163),
(4.6022766242633555, 5.111929816033704 6),
(4.587089899659633, 5.096743091429982),
(4.851762524934205, 5.361415716704554),
(4.803800850244628, 5.313454042014977),
(4.698020295955326, 5.207673487725675),
(4.763310682236782, 5.272963874007131),
(4.623382280074068, 5.133035471844417),
(4.466277788643667, 4.975930980414016),
(4.6686749593189525, 5.178328151089302),
(4.676134202017152, 5.185787393787501),
(4.821940557146294, 5.331593748916643),
(4.797604499109486, 5.307257690879835),
(4.873673291703405, 5.383326483473754),
(4.627681134294528, 5.137334326064877),
(4.830138399921221, 5.33979159169157),
(4.81257122660185, 5.322224418372199),
(4.777183378131991, 5.28683656990234),
(4.712555865145518, 5.222209056915867),
(4.721180195633825, 5.230833387404174),
(4.682500968384878, 5.192154160155227),
(4.805471036265268, 5.315124228035617),
(4.578808055521489, 5.088461247291838),
(4.849712073740881, 5.35936526551123),
(4.826487685274345, 5.336140877044694),
(5.082790727864014, 5.592443919634363),
(4.800960336994141, 5.31061352876449),
(4.607126110848959, 5.1167793026193085),
(4.540576973333114, 5.050230165103463),
(4.658263191959439, 5.167916383729788),
(4.548216581592305, 5.057869773362654),
(4.816286946517697, 5.325940138288046),
(4.807322645414155, 5.316975837184504),
(4.850557674996203, 5.360210866766552),
(4.765089577041411, 5.27474276881176),
(4.625610294611095, 5.135263486381444),
(4.595330684602193, 5.104983876372542),
(4.96493290990166, 5.474586101672009),
(4.838617995612772, 5.348271187383121),
(4.906376903678512, 5.416030095448861),

```
 (4.711921210656203, 5.221574402426552),
 (4.597167262969226, 5.106820454739575),
 (4.638186145291051, 5.1478393370614),
 (4.770404575311815, 5.280057767082164),
 (4.70861743412976, 5.218270625900109),
 (4.724719307138257, 5.2343724989086065),
 (4.606235803820949, 5.115888995591298),
 (4.5379617280677325, 5.0476149198380815),
 (4.780296613988215, 5.289949805758564),
 (4.695998841764247, 5.205652033534596),
 (4.7394925909844305, 5.2605074090155695)]
```

The results above is confidence interval of 100 samples.

```
plt.figure(figsize=(9,9))
plt.title('Confidence interval of each sample')
plt.xlabel('index of sample')
plt.ylabel('means of sample')
plt.errorbar(x=np.arange(1,N+2),
             y=samples_means,
             yerr=[(top-bot)/2 for top,bot in CI],
             fmt='o')
plt.hlines(xmin=0, xmax=N+1,
           y=data.mean(),
           linewidth=2.0,
           color="red");
```



The figure above is the confidence interval of 100 samples, which blue line is the confidence interval of the sample and blue dots is the mean of the sample. Red line is the mean of sampling distribution.

The far right blue line is theoretical confidence interval. Compare the theoretical CI with the samples' CI, the result is as expected.

2. Poisson statistics [40 points]

Perseids are a meteor showers associated with the comet Swift–Tuttle. In astronomy, the zenithal hourly rate (ZHR) of a meteor shower is the number of meteors a single observer would see in an hour of peak activity. Assume that today the Perseids have a ZHR = 120. Assume that the occurrence of every single meteor is distributed according to Poisson statistics.

a. Find the probability that no meteor is observed during a given minute **10 Points**

The occurence of every single meteor obey Poisson distribution,

$$P(x)=\frac{\mu^{x}.e^{-\mu}}{x!}$$

.The mean number in one minute $\mu=\frac{120}{60}=2$, so the probability that no meteor is observed during one minute $P(0)=\frac{2^{0}.e^{-2}}{0!}$ = 0.1353

b. What is the expected number of meteors occurring in two minutes **10 Points**

The expected number of meteors in two minutes,¿x>¿$\frac{120\times 2}{60}=4$

c. Find the probability that this expected number actually are observed in a given two-minute period. **10 Points**

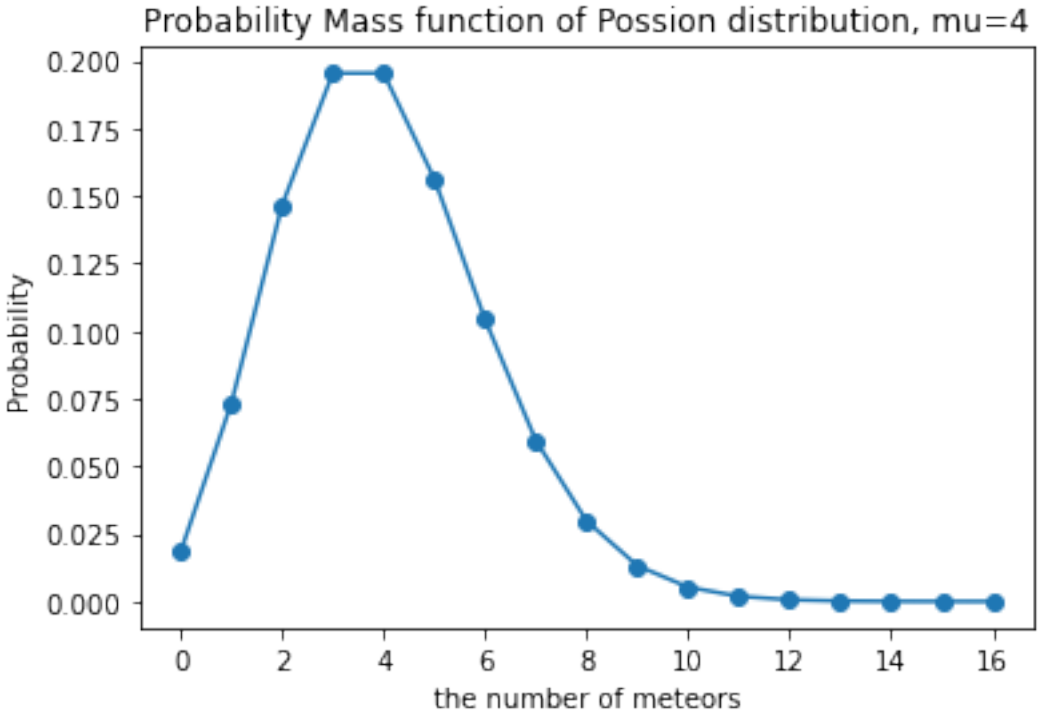
The probability that 4 meteors are observed in a given two-minute $P(4)=\frac{4^{4}.e^{-4}}{4!}=0.1953$

d. Plot a histogram of the probabilities for the number of meteors for each 2 minute period. **10 Points**

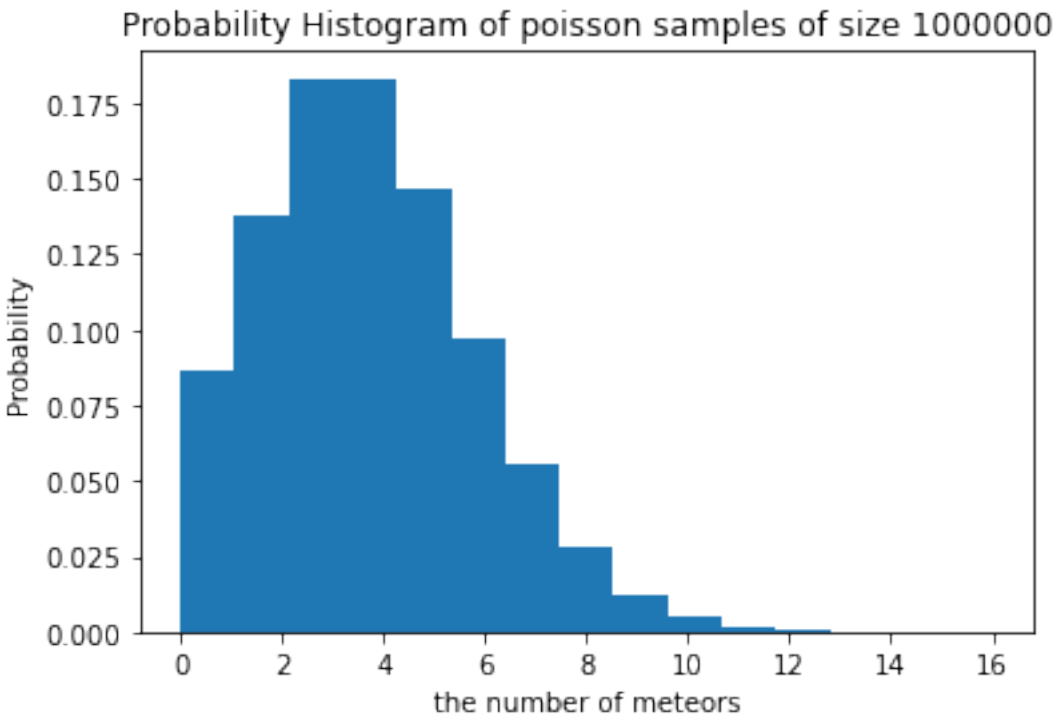
```
from scipy.stats import poisson
k = np.arange(0, 17)
pmf = poisson.pmf(k, mu=4)
pmf = np.round(pmf, 5)
for val, prob in zip(k,pmf):
    print(f"k-value {val} has probability = {prob}")
plt.plot(k, pmf, marker='o')
plt.xlabel('the number of meteors')
plt.ylabel('Probability')
plt.title('Probability Mass function of Possion distribution, mu=4')

plt.show()
```

```
k-value 0 has probability = 0.01832
k-value 1 has probability = 0.07326
k-value 2 has probability = 0.14653
k-value 3 has probability = 0.19537
k-value 4 has probability = 0.19537
k-value 5 has probability = 0.15629
k-value 6 has probability = 0.1042
k-value 7 has probability = 0.05954
k-value 8 has probability = 0.02977
k-value 9 has probability = 0.01323
k-value 10 has probability = 0.00529
k-value 11 has probability = 0.00192
k-value 12 has probability = 0.00064
k-value 13 has probability = 0.0002
```

```
k-value 14 has probability = 6e-05
k-value 15 has probability = 2e-05
k-value 16 has probability = 0.0
```

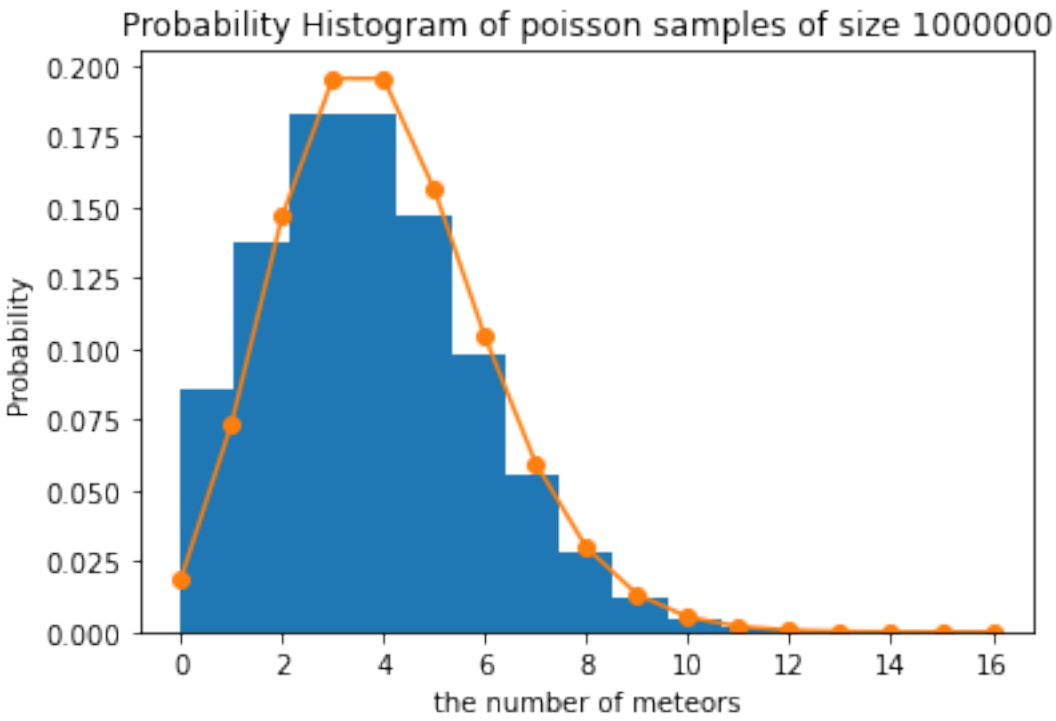



```
rvs = poisson.rvs(4,size=1000000)
fig, ax = plt.subplots(1, 1)
ax.hist(rvs,bins=15,range=(0,16),density=1)
plt.title('Probability Histogram of poisson samples of size 1000000')
plt.xlabel('the number of meteors')
plt.ylabel('Probability')

Text(0, 0.5, 'Probability')
```




```
rvs = poisson.rvs(4,size=1000000)
fig, ax = plt.subplots(1, 1)
ax.hist(rvs,bins=15,range=(0,16),density=1)
plt.title('Probability Histogram of poisson samples of size 1000000')
k = np.arange(0, 17)
pmf = poisson.pmf(k, mu=4)
pmf = np.round(pmf, 5)

plt.plot(k, pmf, marker='o')
plt.xlabel('the number of meteors')
plt.ylabel('Probability')
```

```
Text(0, 0.5, 'Probability')
```


Probability Histogram of poisson samples of size 1000000
Probability
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
2
4
6
8
10
12
14
16
the number of meteors