

module-1

August 27, 2021

1 Module 1

1.1 Setup

```
[1]: import numpy as np
      %matplotlib inline
      import matplotlib.pyplot as plt
      from scipy.stats import rv_discrete
      import os
```

1.2 Win / Loss

```
[2]: win_prob = 20/100
      loss_prob = 80/100
      win_prob + loss_prob == 1
```

[2]: True

1.3 Exam Scores

```
[3]: scores = {"50-60": 20, "61-80": 30, "81-100": 50}
```

```
[4]: tot = sum(scores.values())
      probs = np.divide(list(scores.values()),tot)
      print(probs)
```

[0.2 0.3 0.5]

```
[5]: sum(probs) == 1
```

[5]: True

1.4 Coin Toss

```
[6]: np.random.choice(["heads", "tails"])
```

[6]: 'heads'

1.5 Roll Dice

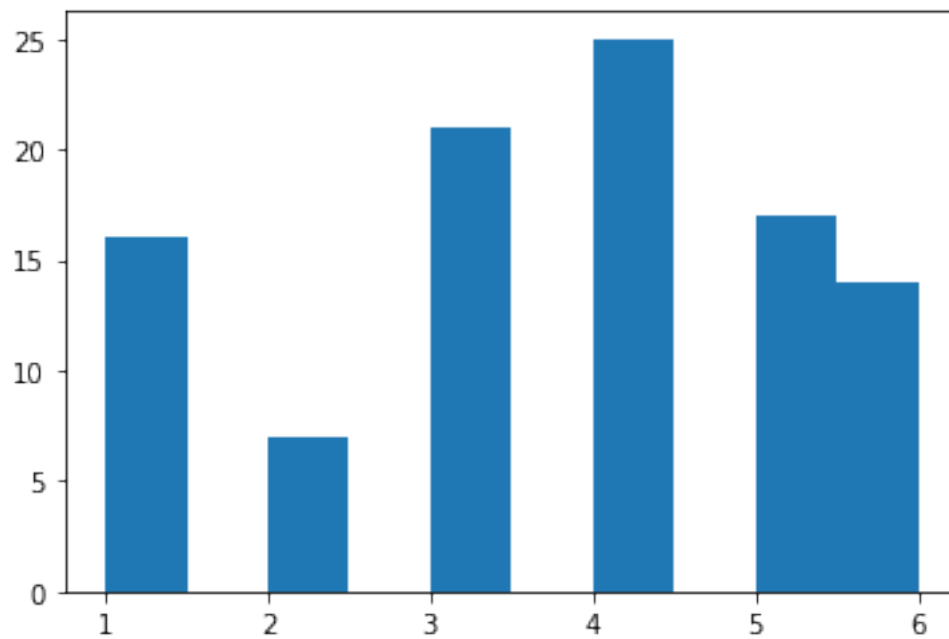
```
[7]: np.random.randint(1,6)
```

```
[7]: 3
```

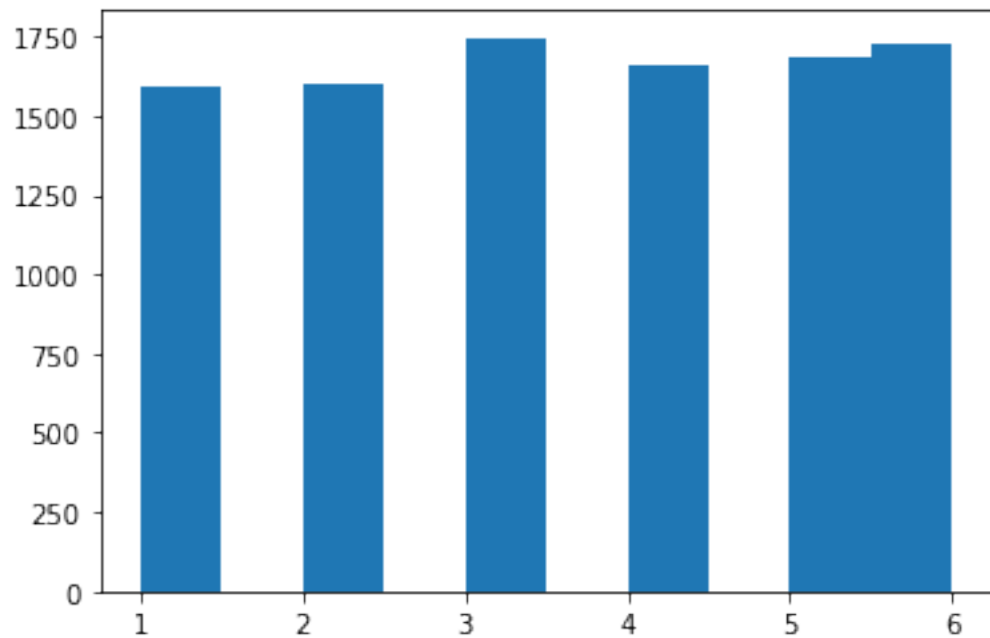
```
[8]: def roll_dice(n):  
      return np.random.randint(1, 7, n)
```

```
[9]: rolls = [roll_dice(n) for n in [100, 10000, 1000000]]
```

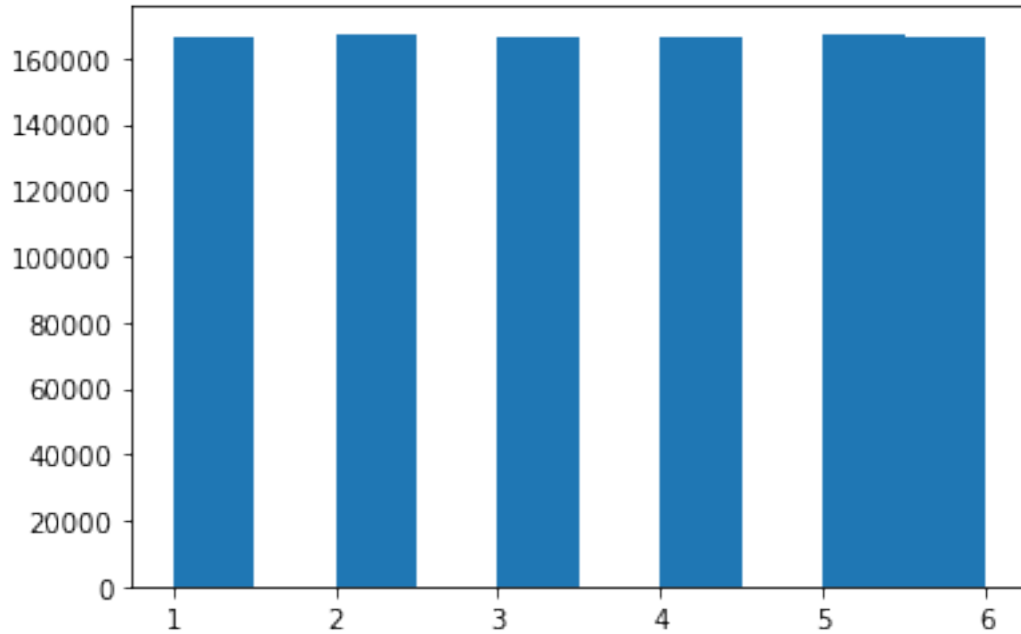
```
[10]: plt.hist(rolls[0]);
```



```
[11]: plt.hist(rolls[1]);
```



```
[12]: plt.hist(rolls[2]);
```



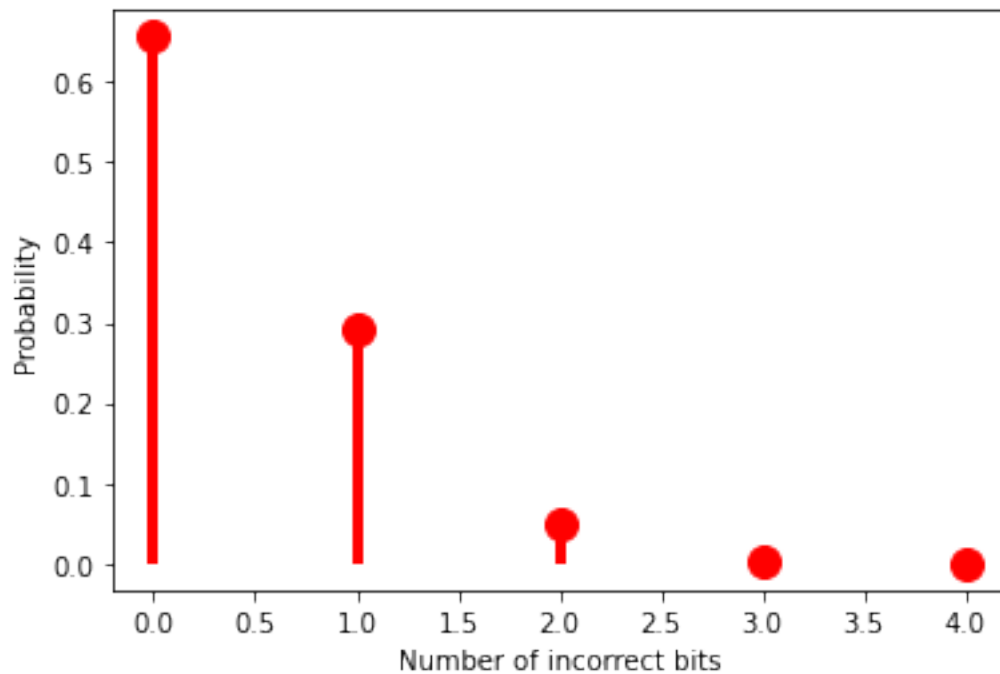
```
[13]: p = 1/6  
di_probs = {i:p for i in range(1,7)}
```

```
[14]: sum(di_probs.values())
```

```
[14]: 0.9999999999999999
```

1.6 Digital Channel (Ex 3.5)

```
[38]: # modified from https://docs.scipy.org/doc/scipy/reference/generated/scipy.  
      ↪ stats.rv_discrete.html  
xk = np.arange(5)  
pk = [0.6561, 0.2916, 0.0486, 0.0036, 0.0001]  
custm = rv_discrete(name='custm', values=(xk, pk))  
  
fig, ax = plt.subplots(1, 1)  
ax.plot(xk, custm.pmf(xk), 'ro', ms=12, mec='r')  
ax.vlines(xk, 0, custm.pmf(xk), colors='r', lw=4)  
plt.xlabel("Number of incorrect bits")  
plt.ylabel("Probability")  
plt.show()
```



1.6.1 Cumulative Sum

```
[39]: print(np.cumsum(pk))
```

```
[0.6561 0.9477 0.9963 0.9999 1.    ]
```

1.6.2 Expectation Value

```
[40]: mu = np.dot(xk, pk)
      print(mu)
```

0.4

1.6.3 Variance

```
[41]: var = sum([p*(x-mu)**2 for x, p in zip(xk, pk)])
      print(var)
```

0.36000000000000001

Note that `np.var()` calculates the population variance with equal weights (assuming default arguments for `np.var()`). Notice that the result is incorrect.

```
[42]: np.var(pk)
```

```
[42]: 0.06357454
```

Helper Function for Mean and Variance

```
[43]: def dist_mean_var(xk, pk):
      mu = np.dot(xk, pk)
      var = sum([p*(x-mu)**2 for x, p in zip(xk, pk)])
      return mu, var
```

1.6.4 Standard Deviation

```
[44]: np.sqrt(var)
```

```
[44]: 0.60000000000000001
```

```
[45]: mu, var = dist_mean_var(xk, pk)
      print(mu, var)
```

0.4 0.36000000000000001

1.7 NiCd Battery (3.3.6)

```
[24]: battery = {0: 0.17, 2: 0.35, 3: 0.33, 4: 0.15}
      charges = list(battery.keys())
      vals = list(battery.values())
      cdf = np.cumsum(vals)
      print(cdf)
```

[0.17 0.52 0.85 1.]

```
[25]: n = 10000
      x = np.linspace(0, 5, num=n)
```

```

c1 = (x >= 0) & (x < 2)
c2 = (x >= 2) & (x < 3)
c3 = (x >= 3) & (x < 4)
c4 = x >= 4
conds = [c1, c2, c3, c4]
pw = np.pieceswise(x, conds, cdf)

```

```

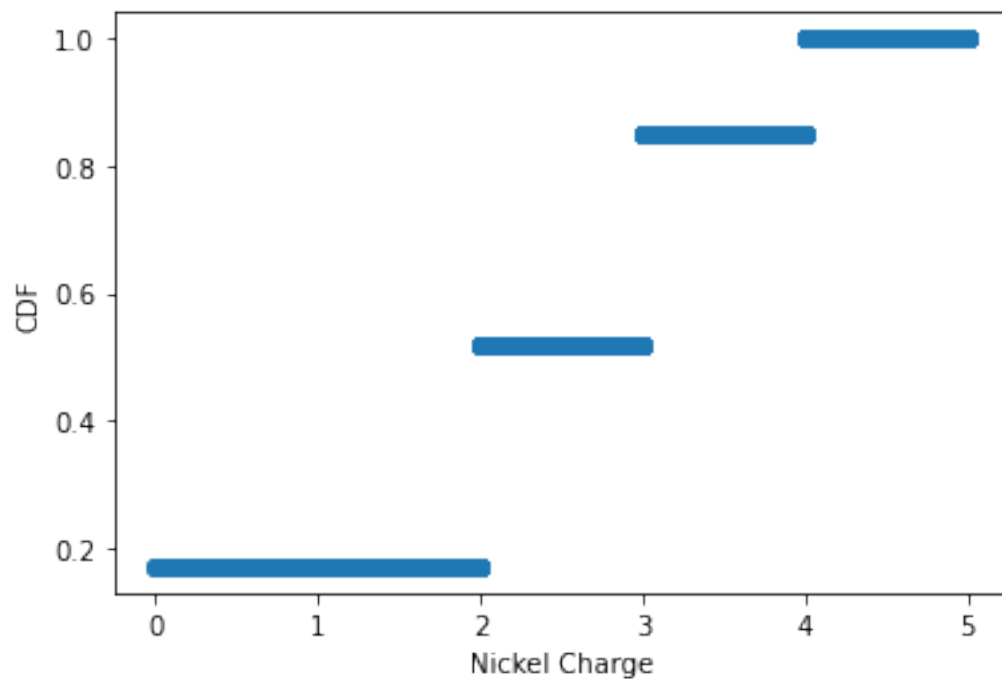
[26]: plt.scatter(x,pw)
plt.xlabel("Nickel Charge")
plt.ylabel("CDF")

```

```

[26]: Text(0, 0.5, 'CDF')

```



```

[27]: mu, var = dist_mean_var(charges, vals)
sigma = np.sqrt(var)
print(mu, sigma)

```

```

2.29 1.2559060474414476

```

2 Code Graveyard

```

[28]: n = 10000
x = np.linspace(0, 100, num=n)
c1 = (x >= 50) & (x <= 60)

```

```

c2 = (x >= 61) & (x <= 80)
c3 = (x >= 81) & (x <= 100)
conds = [c1, c2, c3]
vals = np.array([20/(60-50), 30/(80-61), 50/(100-81)])/n

```

```

[29]: pw = np.piecewise(x, conds, vals)
      print(pw)

```

```

[0.          0.          0.          ... 0.00026316 0.00026316 0.00026316]

```

```

[30]: sum(pw)

```

```

[30]: 0.9999999999999853

```

```

[31]: [sum(c*pw) for c in conds]

```

```

[31]: [0.200000000000000367, 0.29999999999999941, 0.50000000000000163]

```

```

[32]: conds = [0, 1, 2, 3, 4]
      vals = [0.6561, 0.2916, 0.0486, 0.0036, 0.0001]
      digi_chan = {cond: val for cond, val in zip(conds, vals)}
      print(digi_chan)

```

```

{0: 0.6561, 1: 0.2916, 2: 0.0486, 3: 0.0036, 4: 0.0001}

```

3 Print to PDF

```

[ ]: os.system("jupyter nbconvert --to pdf module-1.ipynb")

```