Challenge 3

Programming Challenges

Use the `random` module to generate pseudo random numbers. For instance, `random.randrange(2)` produces (pseudo) random bits. To use this module, it is necessary to `import random`. It may also be helpful to `import numpy as np`.

In this numerical procedure, you will create multiple sequences of random variables. Let `X` and `Y` correspond to the roll of independent, fair dice. Let sum `S = X + Y` and max `M = max(X,Y)`. Compute the joint PMF of `S` and `M`.

```
NumberTrials = 100000
sequenceX = []
sequenceY = []
sequenceS = []
sequenceM = []

for TrialIndex in range(0, NumberTrials):
    sequenceX.append(random.randint(1, 6))
    sequenceY.append(random.randint(1, 6))
    sequenceS.append(sequenceX[TrialIndex] + sequenceY[TrialIndex])
    sequenceM.append(max(sequenceX[TrialIndex], sequenceY[TrialIndex]))
```

Then, look at the empirical distribution of the ratios of zeros and ones.

```
PMFofSM = np.zeros((13, 7))

for TrialIndex in range(0, NumberTrials):
    PMFofSM[sequenceS[TrialIndex], sequenceM[TrialIndex]] += 1

PMFofSM /= float(NumberTrials)
```

Write code to isolate the (empirical) conditional PMF of `S` given `M`. Explore how this empirical distribution changes as `N` increases: 10, 100, 1000, 10000. Can you guess the correct structure for the conditional PMF of `S` given `M`?