r09546042_ _DA_HW02

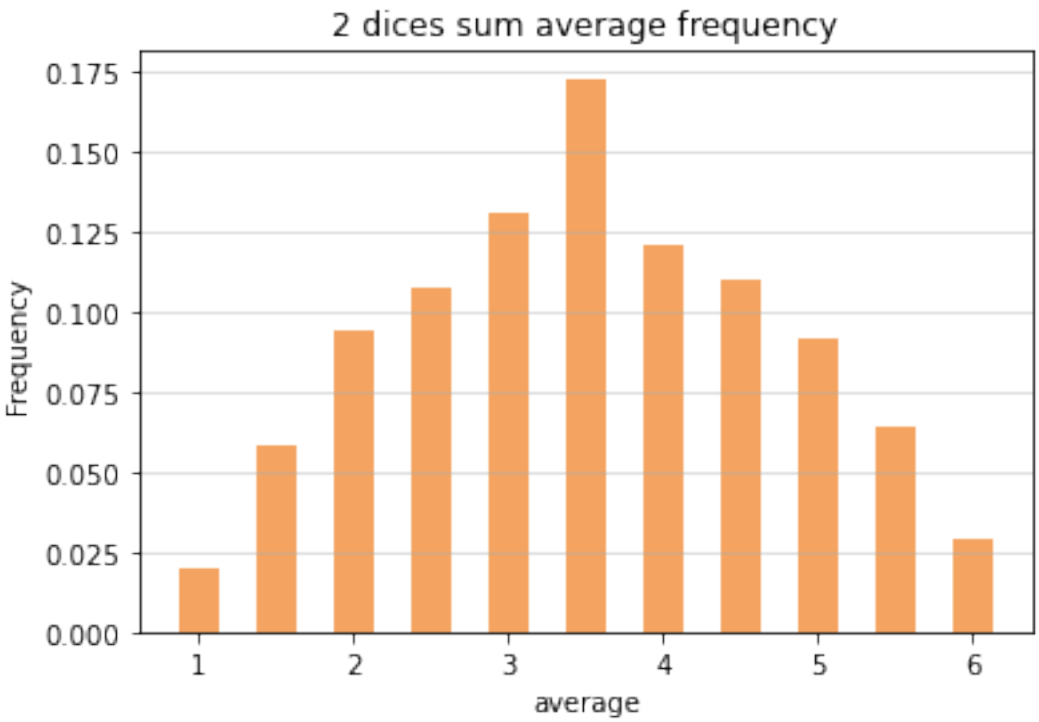
March 7, 2021

1 Q1. Dice average probability chart

[1]:
```
import random
import matplotlib.pyplot as plt
import numpy as np
```

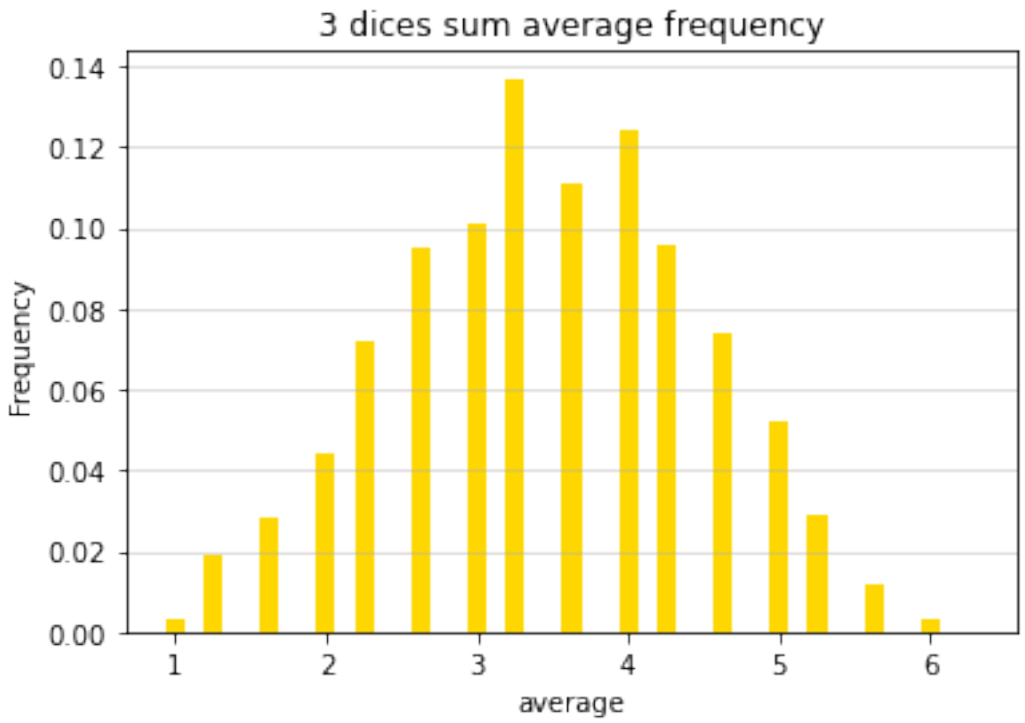
[2]:
```
values = (np.random.randint(1, 7, 1000)+np.random.randint(1, 7, 1000))/2
plt.xlabel('average')
plt.ylabel('Frequency')
plt.title('2 dices sum average frequency')
plt.grid(axis='y', alpha=0.5)
weights = np.ones_like(values)/float(len(values))
plt.hist(values, bins = np.arange(1,6.5,0.25), align = 'left', bottom = 0,␣
 ↪color = "sandybrown",weights=weights)
```

[2]:
```
(array([0.02 , 0.   , 0.058, 0.   , 0.094, 0.   , 0.108, 0.   , 0.131,
        0.   , 0.173, 0.   , 0.121, 0.   , 0.11 , 0.   , 0.092, 0.   ,
        0.064, 0.   , 0.029]),
 array([1.  , 1.25, 1.5 , 1.75, 2.  , 2.25, 2.5 , 2.75, 3.  , 3.25, 3.5 ,
        3.75, 4.  , 4.25, 4.5 , 4.75, 5.  , 5.25, 5.5 , 5.75, 6.  , 6.25]),
 <a list of 21 Patch objects>)
```



[3]:
```
values = (np.random.randint(1, 7, 1000)+np.random.randint(1, 7, 1000)+np.random.
 ↪randint(1, 7, 1000))/3
plt.xlabel('average')
plt.ylabel('Frequency')
plt.title('3 dices sum average frequency')
plt.grid(axis='y', alpha=0.5)
weights = np.ones_like(values)/float(len(values))
plt.hist(values, bins = np.arange(1,6.5,0.125), align = 'left', bottom = 0,␣
 ↪color = "gold",weights=weights)
```

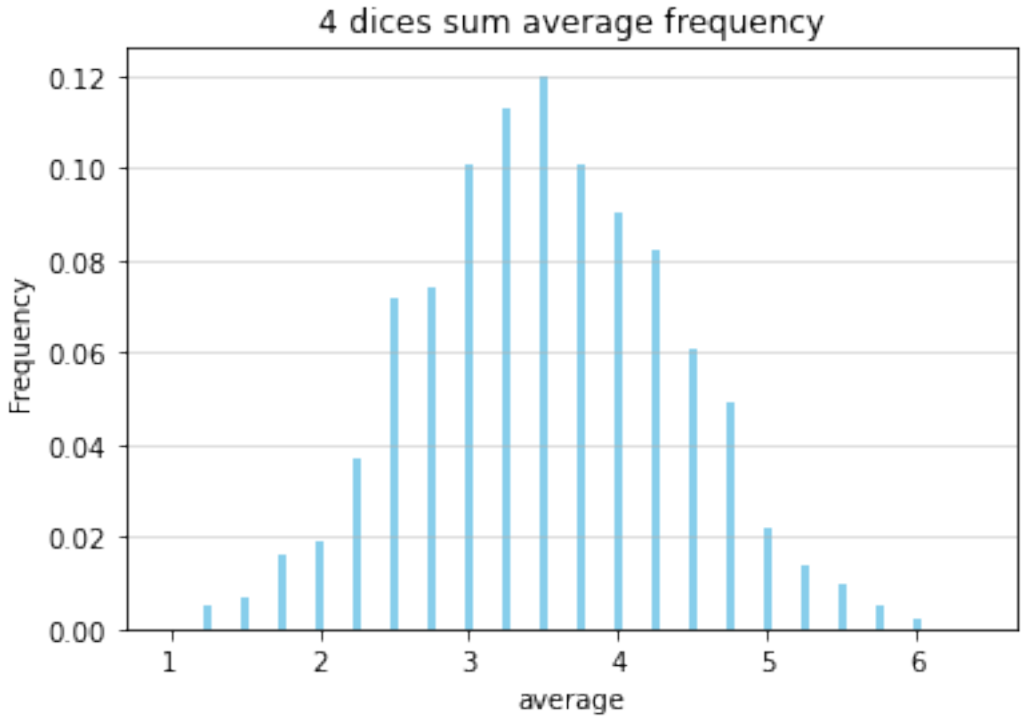
[3]:
```
(array([0.003, 0.   , 0.019, 0.   , 0.   , 0.028, 0.   , 0.   , 0.044,
        0.   , 0.072, 0.   , 0.   , 0.095, 0.   , 0.   , 0.101, 0.   ,
        0.137, 0.   , 0.   , 0.111, 0.   , 0.   , 0.124, 0.   , 0.096,
        0.   , 0.   , 0.074, 0.   , 0.   , 0.052, 0.   , 0.029, 0.   ,
        0.   , 0.012, 0.   , 0.   , 0.003, 0.   , 0.   ]),
 array([1.   , 1.125, 1.25 , 1.375, 1.5  , 1.625, 1.75 , 1.875, 2.   ,
        2.125, 2.25 , 2.375, 2.5  , 2.625, 2.75 , 2.875, 3.   , 3.125,
        3.25 , 3.375, 3.5  , 3.625, 3.75 , 3.875, 4.   , 4.125, 4.25 ,
        4.375, 4.5  , 4.625, 4.75 , 4.875, 5.   , 5.125, 5.25 , 5.375,
        5.5  , 5.625, 5.75 , 5.875, 6.   , 6.125, 6.25 , 6.375]),
 <a list of 43 Patch objects>)
```



[4]:
```
values = (np.random.randint(1, 7, 1000)+np.random.randint(1, 7, 1000)+np.random.
 ↪randint(1, 7, 1000)+np.random.randint(1, 7, 1000))/4
plt.xlabel('average')
plt.ylabel('Frequency')
plt.title('4 dices sum average frequency')
plt.grid(axis='y', alpha=0.5)
weights = np.ones_like(values)/float(len(values))
plt.hist(values, bins = np.arange(1,6.5,0.0625), align = 'left', bottom = 0,␣
 ↪color = "skyblue",weights=weights)
```

[4]:
```
(array([0.   , 0.   , 0.   , 0.   , 0.005, 0.   , 0.   , 0.   , 0.007,
        0.   , 0.   , 0.   , 0.016, 0.   , 0.   , 0.   , 0.019, 0.   ,
        0.   , 0.   , 0.037, 0.   , 0.   , 0.   , 0.072, 0.   , 0.   ,
        0.   , 0.074, 0.   , 0.   , 0.   , 0.101, 0.   , 0.   , 0.   ,
        0.113, 0.   , 0.   , 0.   , 0.12 , 0.   , 0.   , 0.   , 0.101,
        0.   , 0.   , 0.   , 0.09 , 0.   , 0.   , 0.   , 0.082, 0.   ,
        0.   , 0.   , 0.061, 0.   , 0.   , 0.   , 0.049, 0.   , 0.   ,
        0.   , 0.022, 0.   , 0.   , 0.   , 0.014, 0.   , 0.   , 0.   ,
        0.01 , 0.   , 0.   , 0.   , 0.005, 0.   , 0.   , 0.   , 0.002,
        0.   , 0.   , 0.   , 0.   , 0.   , 0.   ]),
 array([1.    , 1.0625, 1.125 , 1.1875, 1.25  , 1.3125, 1.375 , 1.4375,
        1.5   , 1.5625, 1.625 , 1.6875, 1.75  , 1.8125, 1.875 , 1.9375,
        2.    , 2.0625, 2.125 , 2.1875, 2.25  , 2.3125, 2.375 , 2.4375,
```

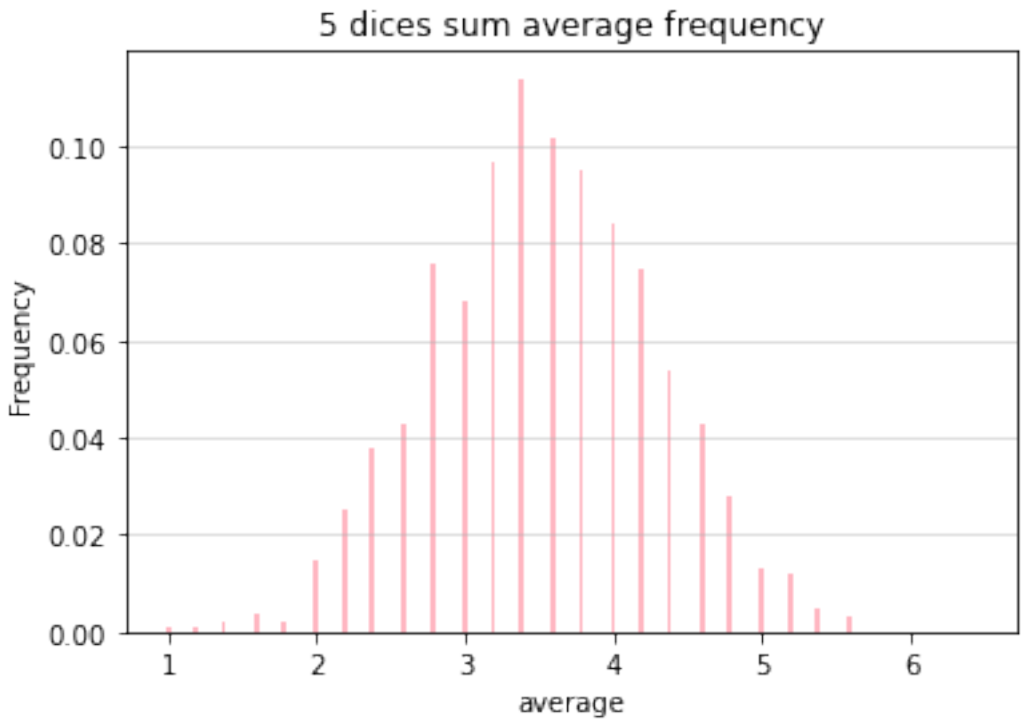
```
        2.5   , 2.5625, 2.625 , 2.6875, 2.75  , 2.8125, 2.875 , 2.9375,
        3.    , 3.0625, 3.125 , 3.1875, 3.25  , 3.3125, 3.375 , 3.4375,
        3.5   , 3.5625, 3.625 , 3.6875, 3.75  , 3.8125, 3.875 , 3.9375,
        4.    , 4.0625, 4.125 , 4.1875, 4.25  , 4.3125, 4.375 , 4.4375,
        4.5   , 4.5625, 4.625 , 4.6875, 4.75  , 4.8125, 4.875 , 4.9375,
        5.    , 5.0625, 5.125 , 5.1875, 5.25  , 5.3125, 5.375 , 5.4375,
        5.5   , 5.5625, 5.625 , 5.6875, 5.75  , 5.8125, 5.875 , 5.9375,
        6.    , 6.0625, 6.125 , 6.1875, 6.25  , 6.3125, 6.375 , 6.4375]),
 <a list of 87 Patch objects>)
```



[5]:
```
values = (np.random.randint(1, 7, 1000)+np.random.randint(1, 7, 1000)+np.random.
 ↪randint(1, 7, 1000)+np.random.randint(1, 7, 1000)+np.random.randint(1, 7,␣
 ↪1000))/5
plt.xlabel('average')
plt.ylabel('Frequency')
plt.title('5 dices sum average frequency')
plt.grid(axis='y', alpha=0.5)
weights = np.ones_like(values)/float(len(values))
plt.hist(values, bins = np.arange(1,6.5,0.03125), align = 'left', bottom = 0,␣
 ↪color = "lightpink",weights=weights)
```

[5]:
```
(array([0.001, 0.   , 0.   , 0.   , 0.   , 0.   , 0.001, 0.   , 0.   ,
        0.   , 0.   , 0.   , 0.002, 0.   , 0.   , 0.   , 0.   , 0.   ,
```

```
       0.   , 0.004, 0.   , 0.   , 0.   , 0.   , 0.   , 0.002, 0.   ,
       0.   , 0.   , 0.   , 0.   , 0.   , 0.015, 0.   , 0.   , 0.   ,
       0.   , 0.   , 0.025, 0.   , 0.   , 0.   , 0.   , 0.   , 0.038,
       0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.043, 0.   , 0.   ,
       0.   , 0.   , 0.   , 0.076, 0.   , 0.   , 0.   , 0.   , 0.   ,
       0.   , 0.068, 0.   , 0.   , 0.   , 0.   , 0.   , 0.097, 0.   ,
       0.   , 0.   , 0.   , 0.   , 0.114, 0.   , 0.   , 0.   , 0.   ,
       0.   , 0.   , 0.102, 0.   , 0.   , 0.   , 0.   , 0.   , 0.095,
       0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.084, 0.   , 0.   ,
       0.   , 0.   , 0.   , 0.075, 0.   , 0.   , 0.   , 0.   , 0.   ,
       0.054, 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.043, 0.   ,
       0.   , 0.   , 0.   , 0.   , 0.028, 0.   , 0.   , 0.   , 0.   ,
       0.   , 0.   , 0.013, 0.   , 0.   , 0.   , 0.   , 0.   , 0.012,
       0.   , 0.   , 0.   , 0.   , 0.   , 0.005, 0.   , 0.   , 0.   ,
       0.   , 0.   , 0.   , 0.003, 0.   , 0.   , 0.   , 0.   , 0.   ,
       0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ,
       0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ,
       0.   , 0.   , 0.   , 0.   ]),
array([1.     , 1.03125, 1.0625 , 1.09375, 1.125  , 1.15625, 1.1875 ,
       1.21875, 1.25   , 1.28125, 1.3125 , 1.34375, 1.375  , 1.40625,
       1.4375 , 1.46875, 1.5    , 1.53125, 1.5625 , 1.59375, 1.625  ,
       1.65625, 1.6875 , 1.71875, 1.75   , 1.78125, 1.8125 , 1.84375,
       1.875  , 1.90625, 1.9375 , 1.96875, 2.     , 2.03125, 2.0625 ,
       2.09375, 2.125  , 2.15625, 2.1875 , 2.21875, 2.25   , 2.28125,
       2.3125 , 2.34375, 2.375  , 2.40625, 2.4375 , 2.46875, 2.5    ,
       2.53125, 2.5625 , 2.59375, 2.625  , 2.65625, 2.6875 , 2.71875,
       2.75   , 2.78125, 2.8125 , 2.84375, 2.875  , 2.90625, 2.9375 ,
       2.96875, 3.     , 3.03125, 3.0625 , 3.09375, 3.125  , 3.15625,
       3.1875 , 3.21875, 3.25   , 3.28125, 3.3125 , 3.34375, 3.375  ,
       3.40625, 3.4375 , 3.46875, 3.5    , 3.53125, 3.5625 , 3.59375,
       3.625  , 3.65625, 3.6875 , 3.71875, 3.75   , 3.78125, 3.8125 ,
       3.84375, 3.875  , 3.90625, 3.9375 , 3.96875, 4.     , 4.03125,
       4.0625 , 4.09375, 4.125  , 4.15625, 4.1875 , 4.21875, 4.25   ,
       4.28125, 4.3125 , 4.34375, 4.375  , 4.40625, 4.4375 , 4.46875,
       4.5    , 4.53125, 4.5625 , 4.59375, 4.625  , 4.65625, 4.6875 ,
       4.71875, 4.75   , 4.78125, 4.8125 , 4.84375, 4.875  , 4.90625,
       4.9375 , 4.96875, 5.     , 5.03125, 5.0625 , 5.09375, 5.125  ,
       5.15625, 5.1875 , 5.21875, 5.25   , 5.28125, 5.3125 , 5.34375,
       5.375  , 5.40625, 5.4375 , 5.46875, 5.5    , 5.53125, 5.5625 ,
       5.59375, 5.625  , 5.65625, 5.6875 , 5.71875, 5.75   , 5.78125,
       5.8125 , 5.84375, 5.875  , 5.90625, 5.9375 , 5.96875, 6.     ,
       6.03125, 6.0625 , 6.09375, 6.125  , 6.15625, 6.1875 , 6.21875,
       6.25   , 6.28125, 6.3125 , 6.34375, 6.375  , 6.40625, 6.4375 ,
       6.46875]),
<a list of 175 Patch objects>)
```



2 Q2. Kruskal's count cards game

[7]:
```
import numpy as np
print("Probabilities list of first 10 cards reach the same end in 10000 times:
 ↪\n")
for DeckOfCards in range(1, 3):
    for StepsOfFaceCards in range(1,11,2):
        numberOfCards = DeckOfCards * 52
        poker = np.
 ↪array(DeckOfCards*4*([1,2,3,4,5,6,7,8,9,10]+3*[StepsOfFaceCards]))
        epoch = 10000
        totalResult =[]
        for times in range(epoch) :
            np.random.shuffle(poker)
            ends = 0
            sameOrNot = 1
            for i in range(0,10):
                step = 0
                location = i
                step = poker[location]
                while location + step < numberOfCards:
                    location = location + step
```

```
                    step = poker[location]
                if location + step >= numberOfCards:
                    if i == 0:
                        end = location +1
                    if end != location +1:
                        sameOrNot = 0
                        break
            totalResult.append(sameOrNot)
        totalResult = np.array(totalResult)
        print(numberOfCards,"cards,", "face cards steps",StepsOfFaceCards,":
↪",np.sum(totalResult == 1)/epoch)
```

```
Probabilities list of first 10 cards reach the same end in 10000 times:

52 cards, face cards steps 1 : 0.8254
52 cards, face cards steps 3 : 0.7129
52 cards, face cards steps 5 : 0.5871
52 cards, face cards steps 7 : 0.4436
52 cards, face cards steps 9 : 0.3312
104 cards, face cards steps 1 : 0.9913
104 cards, face cards steps 3 : 0.9721
104 cards, face cards steps 5 : 0.9341
104 cards, face cards steps 7 : 0.8879
104 cards, face cards steps 9 : 0.8195
```