Section5

Elliott Ashby

November 14, 2022

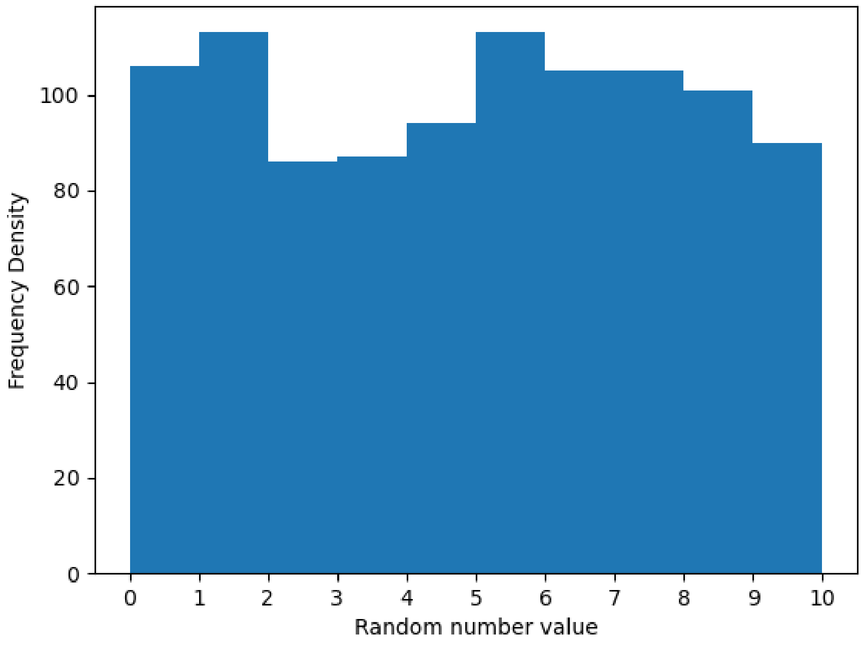
1 q1

The code first makes a list of 10 zeros then generates random numbers between 1 and 10 and increments the respective index of the list by 1. Therefore what is printed is a count of how many times each number is generated.

2 q2

We generate 1000 random numbers in order to minimise the random aspect of the code, the more random numbers are generated the more obvious the actual pattern appears to be.

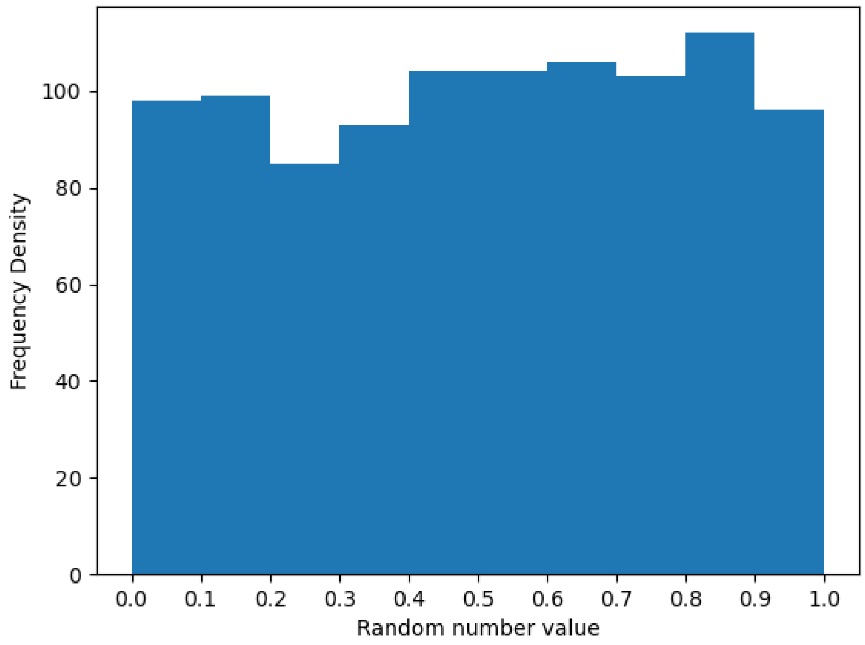
3 q3



4 q4

```
def randbin1(totalrandomnumbers, totalbins, average=1):
    m = zeros(totalbins, int)
    for i in range(totalrandomnumbers):
        n = sum([random.random() for x in range(average)
            ↪ ]) / average
        k = int(totalbins * n)
        m[k] = m[k] + 1
    return m

import matplotlib.pyplot as plt
from numpy import zeros, random


def randbin(totalrandnumbers, numofbins):
    m = zeros(numofbins, int)
    for i in range(totalrandnumbers):
        n = random.randint(numofbins)
        m[n] = m[n] + 1
    return m
```

```
def randbin1(totalrandomnumbers, totalbins, average=1):
    m = zeros(totalbins, int)
    for i in range(totalrandomnumbers):
        n = sum([random.random() for x in range(average)
            ↪ ]) / average
        k = int(totalbins * n)
        m[k] = m[k] + 1
    return m


def normalgen(totalrandomnumbers):
    m: list = []
    for i in range(totalrandomnumbers):
        k = random.normal()
        m.append(k)
    return m


if __name__ == '__main__':
    maxrandnumbers = 1000
    bins = 10
    plt.bar([x/10 for x in range(bins)], randbin1(
        ↪ maxrandnumbers, bins), width=0.1, align='edge')
    plt.xticks([x/10 for x in range(bins + 1)])
    plt.xlabel('Random␣number␣value')
    plt.ylabel('Frequency␣Density')
    plt.savefig('./q5_4.png')
```

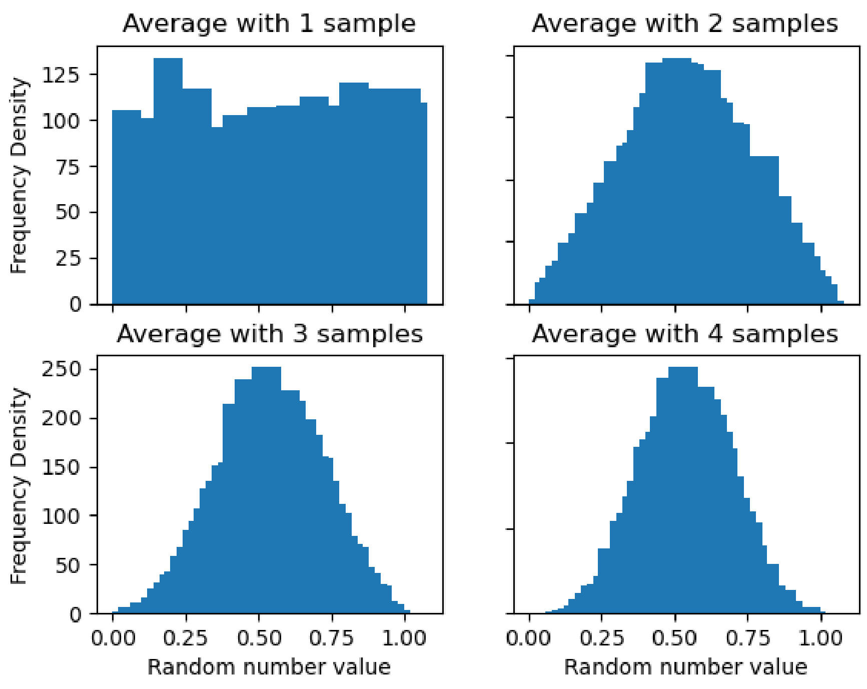


5 q5

```
def randbin1(totalrandomnumbers, totalbins, average=1):
    m = zeros(totalbins, int)
    for i in range(totalrandomnumbers):
        n = sum([random.random() for x in range(average)
            ↪ ]) / average
        k = int(totalbins * n)
        m[k] = m[k] + 1
    return m
```

```
from q5_4 import randbin1
import matplotlib.pyplot as plt


def generatedata(mean, totalnums, bins):
    data = randbin1(totalnums, bins, mean)
    datarange = [x/bins for x in range(bins)]
    return data, datarange


if __name__ == '__main__':
    x = []
    for i in range(1, 5):
        x.append(generatedata(i, 5000, 50))
```

```
fig, axs = plt.subplots(2, 2)
axs[0, 0].bar(x[0][1], x[0][0], width=0.1, align='
    ↪ edge')
axs[0, 0].set_title('Average with 1 sample')
axs[0, 1].bar(x[1][1], x[1][0], width=0.1, align='
    ↪ edge')
axs[0, 1].set_title('Average with 2 samples')
axs[1, 0].bar(x[2][1], x[2][0], width=0.1, align='
    ↪ edge')
axs[1, 0].set_title('Average with 3 samples')
axs[1, 1].bar(x[3][1], x[3][0], width=0.1, align='
    ↪ edge')
axs[1, 1].set_title('Average with 4 samples')
for ax in axs.flat:
    ax.set(xlabel='Random number value', ylabel='
        ↪ Frequency Density')

for ax in axs.flat:
    ax.label_outer()

plt.savefig('./q5_5.png')
```



6 q6

Averages of random numbers converge at the center most value, this makes sense since if there are an equal number of numbers accross the whole range, their average is the center most value.

7 q7

```
def normalgen(totalrandomnumbers):
    m: list = []
    for i in range(totalrandomnumbers):
        k = random.normal()
        m.append(k)
    return m

from q5_4 import normalgen


if __name__ == '__main__':
    total: int = 9000000
    normaldist: list = normalgen(total)
    inrangecount: int = 0
    for x in normaldist:
        if x > -2.0 and x < 2.0:
            inrangecount += 1
        else:
            pass
    probinrange: float = inrangecount / len(normaldist)
    proboutofrange: float = 1 - probinrange
    print(f'Chance that x lies outside -2 < x < 2 is: {
        ↪ round(proboutofrange * 100, 4)}%')
```

Simply count how many of the randomly generated numbers fall in the range and find that as a fraction of the total numbers generated. This is the percentage of numbers within the range, all numbers outside of that range is the opposite value, so 1 - that value gives you the percentage of randomly generated numebers outside of the range.

8 q8

```
from numpy import random


def generatedata(totalnums: int, mean: int):
    data = random.poisson(mean, totalnums)
    return data
```

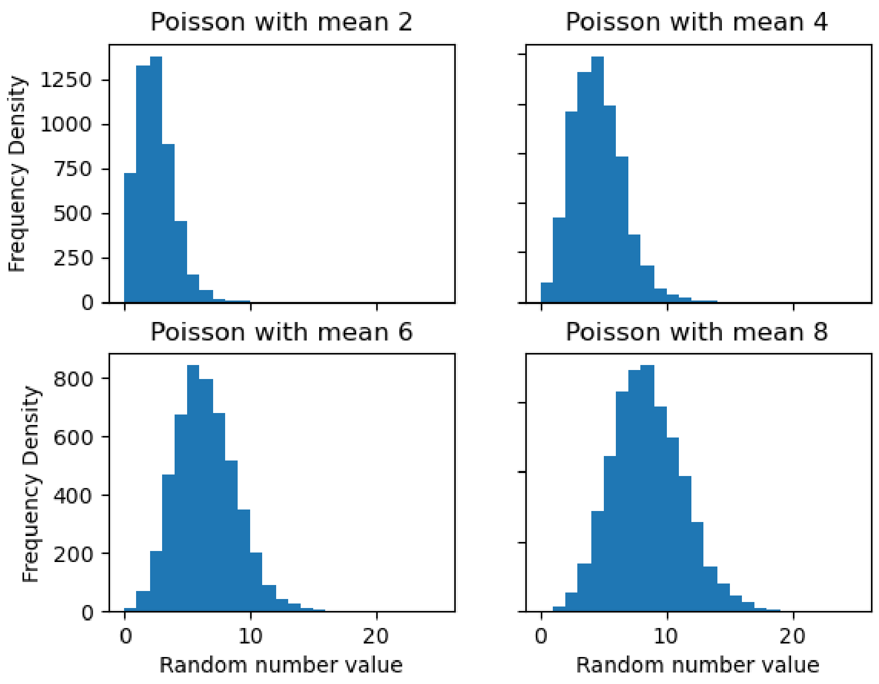
```
if __name__ == '__main__':
    poissondist = generatedata(100000, 15)
    datamean = sum(poissondist) / len(poissondist)
    print(f'The average of 100000 counts of random.
        ↪ poisson with a mean of 15 is {datamean}')
```

This results in an output of around: 15 ± 0.01, Which confirms that the poisson data set of mean 15 actually has a mean of 15.

9 q9

```
from numpy import random, zeros
import matplotlib.pyplot as plt


def poissonsetgen(total: int, bins: int, mean: float):
    m = zeros(bins, int)
    n1 = random.poisson(mean, total)
    n = [x for x in n1 if x < bins]
#   a = mean / bins
    for i in n:
        #         k = int(i * a)
        m[i] = m[i] + 1
    datarange = [x for x in range(bins)]
    return m, datarange


if __name__ == '__main__':
    x = []
    for i in [2, 4, 6, 8]:
        x.append(poissonsetgen(5000, 25, i))

    fig, axs = plt.subplots(2, 2)
    axs[0, 0].bar(x[0][1], x[0][0], width=1, align='edge'
        ↪ )
    axs[0, 0].set_title('Poisson with mean 2')
    axs[0, 1].bar(x[1][1], x[1][0], width=1, align='edge'
        ↪ )
    axs[0, 1].set_title('Poisson with mean 4')
    axs[1, 0].bar(x[2][1], x[2][0], width=1, align='edge'
        ↪ )
    axs[1, 0].set_title('Poisson with mean 6')
    axs[1, 1].bar(x[3][1], x[3][0], width=1, align='edge'
        ↪ )
    axs[1, 1].set_title('Poisson with mean 8')
    for ax in axs.flat:
        ax.set(xlabel='Random number value', ylabel='
            ↪ Frequency Density')

    for ax in axs.flat:
```

```
    ax.label_outer()

plt.savefig('./q5_9.png')
```



10 q10

As the mean increases, the histogram shifts to the right as the mean becomes the value with the largest count, in addition the poisson width increases. A poisson with a low mean is skewed with a tail only extending to the right. As the mean increases the distribution spreads out to become more symmetric. This is because in a poisson distribution, the mean is equal to the variance.