Code task 1

Steven Maharaj
695281

August 21, 2019

a

The `LCG_alogorithm` in the `LCG` module implements the LCG alogrithim.

```python
def LCG_alogorithm(a,b,m,seed = 2,iterations=20):
    """"
    Implements the LCG alogrithm
    Outputs: an array of random numbers of size iterations x 1
    """
    x = np.zeros(iterations)
    x[0] = seed

    for i in range(1,iterations):
        x[i] = (a*x[i-1] + b)%m
    return x
```

Listing 1: LCG.LCG_alogrithm

`a_test.py` tests the algorithm.

```python
from LCG import LCG_alogorithm

print(LCG_alogorithm(a =2,b=0,m=7))
```

Listing 2: `a_test.py`

`a_test.py` yields the following output

```
[2. 4. 1. 2. 4. 1. 2. 4. 1. 2. 4. 1. 2. 4. 1. 2. 4. 1. 2. 4.]
```

b

The `check_valid` in the `LCG` module check if a sequence of random numbers is valid.

```python
def check_valid(x,m):
    """Checks if a sequence x is vaild

    Inputs: x - sequence
            m

    Output : True if valid
             False if not valid"""
    n_non_repeating = np.argwhere(x[0] == x).flatten()[1]
    if n_non_repeating==m-1:
        vaild = True
    else:
```

```
        vaild = False

    return vaild
```

Listing 3: `LCG.check_valid`

`b_test.py` checks the valid value of a when $b = 0, m = 7$.

```
from LCG import LCG_alogorithm, check_valid

b,m = 0,7
for a in range(1,m):
    x = LCG_alogorithm(a,b,m,seed = 2,iterations=20)
    if check_valid(x,m):
        print(f"a = {a} is valid")
```

Listing 4: `b_test.py`

We find that

```
a = 3 is valid
a = 5 is valid
```

c

For the valid values of a, values 1 to 6 is are uniform distributed like a real dice. However, the numbers repeat after $m-1$ realisations . This makes the random numbers predictable thus the model specified in part b is inadequate.

d

`d_test.py` checks the number valid when $b = 0, m = 997$.

```
from LCG import LCG_alogorithm, check_valid

b,m = 0,997
count = 0
for a in range(1,m):
    x = LCG_alogorithm(a,b,m,seed = 2,iterations=2000)
    if check_valid(x,m):
        count += 1

print(f"There are {count} vaild")
```

Listing 5: `d_test.py`

```
There are 328 vaild
```

e

`d_test.py` simulates dice throws for $a = 825, b = 0, m = 997$.

```
from LCG import LCG_alogorithm

a,b,m = 825,0,997
x = LCG_alogorithm(a,b,m,seed = 2,iterations=50)
dice_throws = (x%6)+1
print(dice_throws)
```

Listing 6: `e_test.py`

e_test.py yield the follow output.

```
[3. 6. 4. 1. 6. 2. 6. 4. 2. 2. 1. 3. 3. 1. 3. 5. 5. 3. 1. 3. 6. 6. 2. 3.
 6. 3. 2. 2. 4. 5. 2. 3. 4. 3. 1. 3. 3. 6. 1. 2. 4. 5. 2. 6. 3. 5. 4. 1.
 1. 5.]
```

1 f

f_test.py performs 10000 iterations.

```python
from LCG import LCG_alogorithm,prob_of_two_sixs
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("darkgrid")

a,b,m = 825,0,997
x = LCG_alogorithm(a,b,m,seed = 2,iterations=10000)
dice_throws = (x%6)+1

probabilty_of_two_sixs = prob_of_two_sixs(x)

print("For 10000 dice rolls")
print(f'The empirical probabilty of rolling two sixs is {probabilty_of_two_sixs}')

# Plots
hist = sns.countplot(x=dice_throws)
plt.xlabel("Dice values")
plt.ylabel("Freqency")
plt.title("Dice throws")
plt.savefig("fig/dice_freqency",dpi=250)
plt.show(hist)
```

Listing 7: f_test.py

Figure 1 shows that the empirical probabilities are uniform. For example the probability of rolling a 1 and the probability of rolling a 5 are the same.

For a real dice the probability of roll two sixes in a row is $1/36$. However, from our simulation

```
The empirical probabilty of rolling two sixs is 0.0
```

Therefore this suggests the algorithm can not sufficiently model a random dice roll for this choice of parameters.

g

If let $a = 858$ and us the same parameter from question f, the probabilities are not uniform. Figure 2 shows a clear non-uniform distribution. Thus, the values chosen for the parameters a, b, m are very important as they greatly impact how how 'random' the generated values from the LCG algorithm.

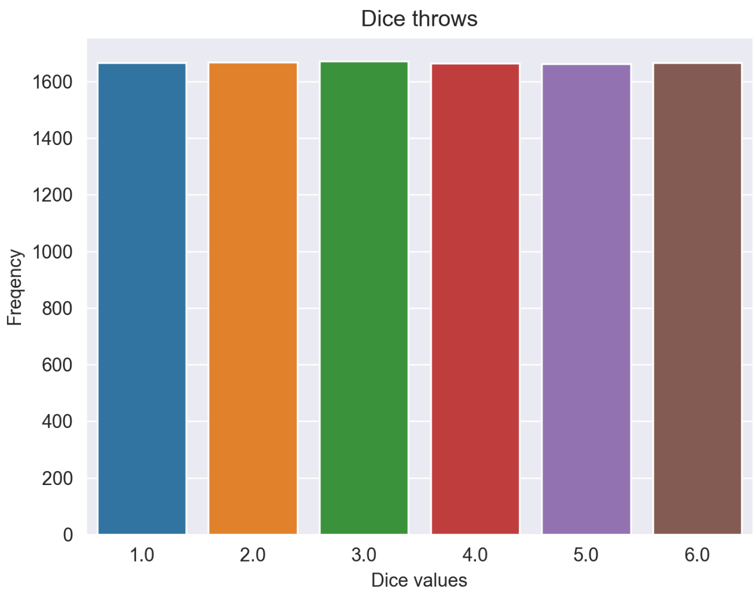

Figure 1: Frequency of each dice throw. Performed for 10000 iteration. a,b,m = 825,0,997.

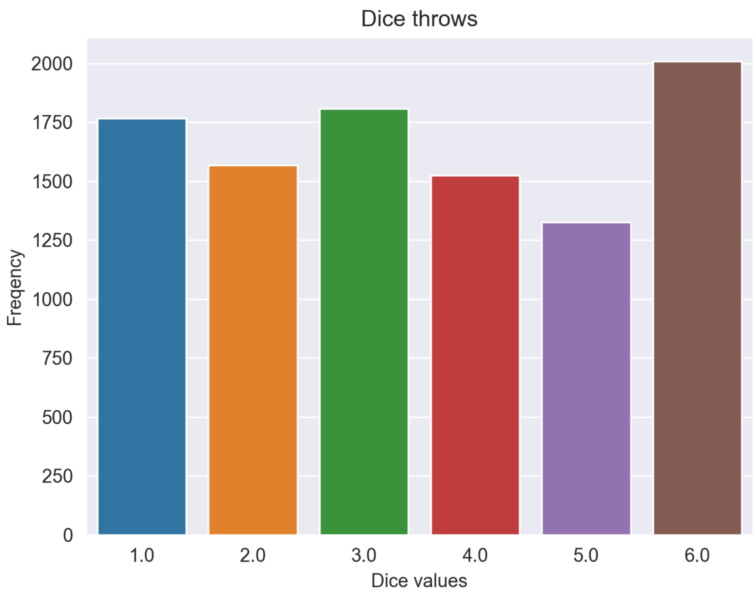

Figure 2: Frequency of each dice throw. Performed for 10000 iteration. a,b,m = 858,0,997.