More Markov Chains

Prompt

Original Prompt can be found here. A copy of the prompt along with the completed exercise can be found under /Applications.

Summary

Part 1

You roll a fair six-sided die 6 times. Whatever the results, you paint those on the sides of a blank die. So, if your rolls were 4, 5, 2, 6, 1, 1, then your new die has no 3's and two 1's. Then, you repeat the process with your new die — roll it 6 times and paint the results on a blank die. Eventually, you'll roll the same number 6 times, at which point the process stops. Let T = the number of rounds (of 6 rolls each) that you perform until stopping. (If your 6 rolls in the first round all result in the same number, then you stop with $T = 1$.)

Part 2

In this part you will create and solve your own problems involving Markov chains. You have lots of flexibility, but your problems should include at least one part that requires:

- A simulation
- An analytical solution that uses either first step analysis/absorption
- An analtyical solution that uses stationary distributions/long run behavior

---

Application - Part 1

1.

Code and run a simulation to approximate the distribution of T and its expected value, without using Markov chains. Summarize the approximate distribution in an appropriate plot, and describe the distribution in 2-3 sentences (e.g., compute and interpret a few percentiles).

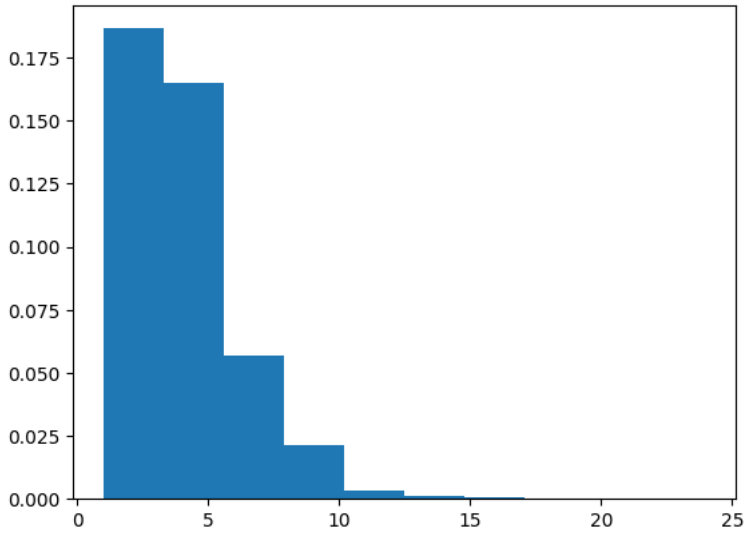
```
import numpy as np

def die_sim():
  # create containers for dice
  fresh_die = np.array([1, 2, 3, 4, 5, 6])
  current_die = fresh_die
  new_die = np.empty(6)

  #create flag for loop and variables
  same = False
  rolls = 6 # rolls corisponding to sides of dice
  T = 0 # number of dice created before all sides are the same

  while(not same):
    #show current die
    #print(current_die, "\n")

    # roll current die and create new die
    for i in range(0,rolls):
      result = np.random.choice(current_die)
      new_die[i] = result

    #increment die counter, check die side similarity
    T += 1
```

```
    current_die = new_die
    same = all(x == new_die[0] for x in new_die)

  #print(current_die, "\n", T)
  return T


die_sim()
```

```
     4
```

```
# run simulation multiple times
sims = 100000

sim_results = np.empty(sims)

for i in range(0,sims):
  sim_results[i] = die_sim()


import matplotlib.pyplot as plt

plt.hist(sim_results, density = True)
```

```
     (array([1.86600000e-01, 1.64969565e-01, 5.68086957e-02, 2.14913043e-02,
             3.21739130e-03, 1.16086957e-03, 4.43478261e-04, 5.65217391e-05,
             1.30434783e-05, 2.17391304e-05]),
      array([ 1. ,  3.3,  5.6,  7.9, 10.2, 12.5, 14.8, 17.1, 19.4, 21.7, 24. ]),
      <BarContainer object of 10 artists>)
```

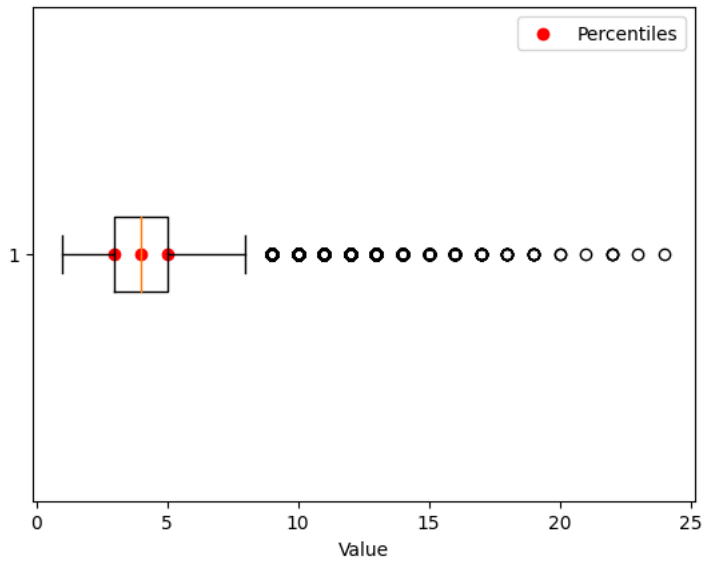



```
# compute expected value
ev = sim_results.mean()
std_dev = sim_results.std()
print("The average of dice created until the final with all the same sides is {:.2f} dice".format(ev))
print("The standard deviation was {:.2f} dice".format(std_dev))
```

```
     The average of dice created until the final with all the same sides is 4.22 dice
     The standard deviation was 1.89 dice
```

The distribution seems to be right skewed, with the bulk of the required rolls hovering around 2-5. The expected value of the distribution is about 3.5.

```
# calculate percentiles
percentiles = np.percentile(sim_results, [25, 50, 75])

# create a box plot with percentiles
plt.boxplot(sim_results, vert=False)
plt.scatter(percentiles, [1, 1, 1], color='red', marker='o', label='Percentiles')
plt.xlabel('Value')
plt.legend()
plt.show()
```




```
print("The percentiles are :",percentiles)
```

```
The percentiles are : [3. 4. 5.]
```

2.

Define a Markov chain that will help you find $E(T)$. Be sure to clearly define the state space. (Note: there are several ways to do this; any one is fine just be clear in your choice.)

3.

Determine the transition matrix for your Markov chain. You might want to compute a few of the transition probabilities by hand, but you'll probably need to write code to fill in the whole matrix.

4.

Use the transition matrix and tools from class to solve for $E(T)$. Compare to the simulated value from part 1.

5.

Use the transition matrix and tools from class to solve for the distribution of T, and display the distribution in an appropriate plot. Compare to the simulated distribution from part 1.

Application - Part 2