Gabriel BUGINGA

Machine Learning
CPS 863
Lista 1

October 3, 2019

Date Performed:	October 3, 2019
Institution:	PESC/COPPE/UFRJ
Instructors:	Edmundo de Souza e Silva
	Rosa M. M. Leão
	Daniel Sadoc Menasché

Problem 1

1: For answering it precisely, we're going to model it as a Markov Chain, turning into a simple case of random walk. The index set is $T = \{0, 1, 2, 3, \ldots\}$; for the state space let our point of view be from, for example, Maria, then $\mathcal{X} = \{-100, 0, 100, \ldots, 9.900, 10.000, 10.100, 10.200, \ldots, \}$; initial state $X_0 = 10.000$; transition probabilities, noting that It is a homogeneous chain, $p_{ij} = \mathbb{P}(X_{n+1} = i \pm j | X_n = i) = 0.5, \forall i \in \mathcal{X}$. The relevant transition is $p_{0,0}(2n) = \begin{pmatrix} 2n \\ n \end{pmatrix} 0.5^{2n} \sim \frac{1}{\sqrt{n\pi}}$, then $\sum_n p_{0,0}(2n) = \infty$ which by a standard Markov Chain result [1] makes the whole chain **recurrent** because all states **communicate**, meaning that:

$$\mathbb{P}(X_n = 0 \text{ for some } n \geq 1 | X_0 = 10.000) = 1 \tag{1}$$

Finally, based on this probability we can assert that one of the players will lose: **"(X)Um dos 2 perdera todo o dinheiro"**.

2: The following table shows the simulations for a maximum of 5000 rounds, repeated for 5 times (as required), code at :

Last round (5000 if no winners)	Joao	Maria
5000	9000.0	11000.0
5000	5200.0	14800.0
5000	4200.0	15800.0
4656	20000.0	0.0
1850	0.0	20000.0

Problem 2

1, **2**: From the code at the last section:

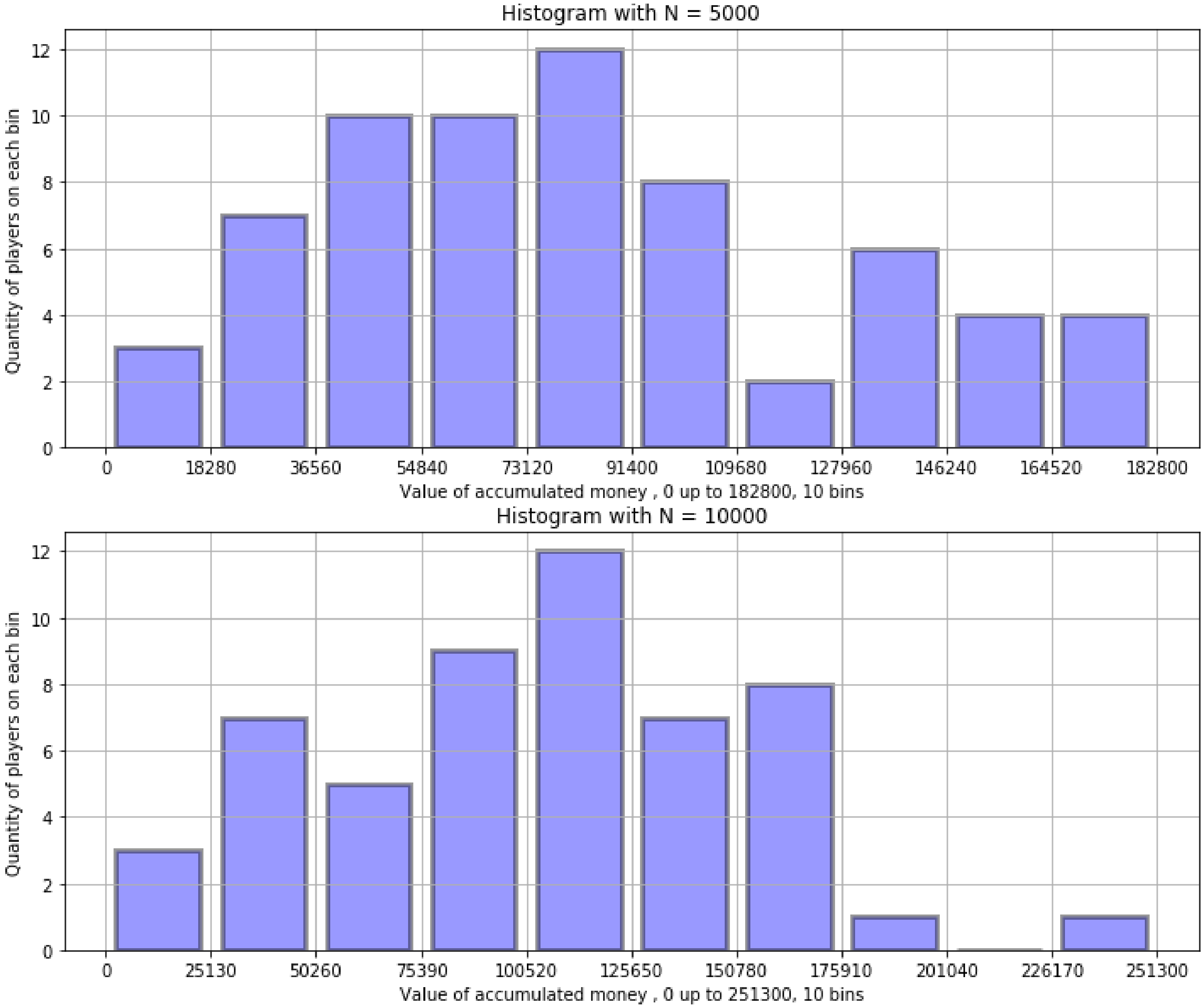

Figure 1: Histograms generated for Problems 2.

3: For $N = 5000$ the fraction without money to play: 86.8%, and the fraction owning 90%: 8.2%. For $N = 10000$: 89.4% and 6.8%.

4: Yes, we can estimate the cumulative distribution function via the broadly known statistics method called the **empirical distribution function** $\widehat{F}_n$, where we put a $\frac{1}{n}$ mass at each data point

X_i, based on [1]:

$$\widehat{F}_n(x) = \frac{\sum_{i=1}^{n} I\left(X_i \leq x\right)}{n}$$
$$I\left(X_i \leq x\right) = \begin{cases} 1 & \text{if } X_i \leq x \\ 0 & \text{if } X_i > x \end{cases} \tag{2}$$

Problem 3

1: Let U_A and U_B be the events of Joao choosing the respective urns, and $2blue$ getting 2 blue balls, then we can apply Bayes theorem [2]:

$$\begin{aligned} P(U_A|2blue) &= \frac{P(2blue|U_A) \cdot P(U_A)}{P(2blue|U_A) \cdot P(U_A) + P(2blue|U_B) \cdot P(U_B)} \\ &= \frac{\frac{1}{4} \cdot \frac{249}{999} \cdot \frac{1}{2}}{\frac{1}{4} \cdot \frac{249}{999} \cdot \frac{1}{2} + \frac{1}{10} \cdot \frac{99}{999} \cdot \frac{1}{2}} \\ &= 0.8627 \end{aligned}$$

We can say that we do have confidence on our bet, given that the posterior probability of knowing the 2 blue balls is near 90%.

2: For this case $P(U_A) = 0.15$ and $P(U_B) = 0.85$, then we reapply Bayes:

$$\begin{aligned} P(U_A|2blue) &= \frac{P(2blue|U_A) \cdot P(U_A)}{P(2blue|U_A) \cdot P(U_A) + P(2blue|U_B) \cdot P(U_B)} \\ &= \frac{0.15 \cdot \frac{249}{999} \cdot \frac{1}{2}}{\frac{1}{4} \cdot \frac{249}{999} \cdot 0.15 + \frac{1}{10} \cdot \frac{99}{999} \cdot 0.85} \\ &= 0.5259 \end{aligned}$$

Differently from before, now we're not so sure, the probability is near 50% so we change and don't bet or trust it any more.

Problem 4

1: Check **Problem 1: 1** for the modelling.

2:The mean time of the random walk according to [2] is $m(x) = (B - x)(x - A)$, for a game starting at x and ending by breaking through $(-A, B)$. then $m(0) = (100)(100) = 10000$ is the mean time for any player to loose.

Code in Python

Problem 1

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
players = np.linspace(10000,10000,2)
N = 5000
simulations , results = np.arange(5),[]
for sim in simulations:

    players,n_final = np.linspace(10000,10000,2),0
    for n in np.arange(N):
        throw = np.random.randint(2,size=1) #It wins or not
        if throw==1:
            players[0] = players[0] + 100
            players[1] = players[1] - 100
        if throw==0:
            players[0] = players[0] - 100
            players[1] = players[1] + 100
        #print (players)
        n_final = n + 1
        if np.max(players<=0):
            break

    results.append([n_final] + players.copy().tolist())

df = pd.DataFrame(results,columns = ['Last round (5000 if no winners)',\
                                     'Joao', 'Maria' ])
print (df.to_latex(index=False))
```

Problem 2

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

plt.figure(figsize=(12,10))
results = []
for subprob,N in enumerate([5000,10000]):
    players = np.linspace(10000,10000,500)
    mask = players >= 0
    for n in np.arange(N):
        for k,p in enumerate(players):
            if mask[k]:
                throw = np.random.randint(2,size=len(players))
                for i,t in enumerate(throw):
                    if i>k and mask[i]:
                        if t==1:
                            players[k] = players[k] + 100
                            players[i] = players[i] - 100
                        if t==0:
```

```
                            players[k] = players[k] - 100
                            players[i] = players[i] + 100

        #Mask the players with value less than zero after a whole round.
        mask = players >= 0
    results.append(players)
    #Now we plot the results
    bins = np.linspace(0,np.amax(players),11)
    hist, bins = np.histogram(players, bins=bins)
    width = np.diff(bins)
    center = (bins[:-1] + bins[1:]) / (2)

    plt.subplot(2,1,subprob+1)
    plt.bar(center, hist, align='center', color = "blue", width=0.8*width\
            ,alpha=0.40,edgecolor = "black",lw="3")
    plt.grid(True)
    plt.xticks(bins)
    plt.ylabel("Quantity of players on each bin")
    plt.xlabel("Value of accumulated money , 0 up to %d,\
    10 bins" % np.amax(players))
    plt.title("Histogram with N = %d" % N)
plt.show()

for t in results:
    fraction_lost = np.sum(t<=0)/t.size
    t = np.sort(t)
    fraction_acc = 0

    for k,v in enumerate(t):
        if (np.sum(t)*0.9)>=np.sum(t[k:]):
            fraction_acc =t[k:].size/t.size
            break
    print ("Fraction Lost : %f | Fraction owning 90%%: %f"\
           %(fraction_lost, fraction_acc))
```

References

[1] L. Wasserman, *All of statistics: a concise course in statistical inference.* Springer Science & Business Media, 2013.

[2] A. N. Shiryaev, "Probability-1," 2016.