Betting on Catalan numbers

Dylan Da Silva Moreira, Hukic Ibrahim

December 31, 2019

Contents

1 Introduction

A fair coin is flipped 100 times yielding a sequence $a = (a_1, \ldots, a_{100})$ of Heads or Tails. Player A realizes that if one thinks of Heads as $+1$ and Tails as -1, then for each k in [1, 100], $\sum_{i=1}^{k} a_i \geq 0$.
Furthermore, $\sum_{i=0}^{100} a_1 = 0$.
Player B thinks such a sequence is extremely unlikely and offers a bet to A with a $100:1$ odds that if they flip the coin again 100 times yielding a random sequence $x = (x_1, \ldots, x_{100})$, then x is not going to satisfy the properties of a. Should A accept the bet? What are the true odds for such an event? Is there a good graphical representation for the properties of a? What if there are $2n$ coin flips? Biased coin?

The solution to this problem will be structured in four parts. We start with a practical approach and do some programming and computation to get a feeling for the problem, after plotting our results we move over to find theoretical answer to our problem.Next we calculate the asked probability and check if it matches the probability value obtained in our experiment. Finally we will answer the questions regarding the betting game and what happens in the case of a biased coin.

2 Experiment (Programming)

2.1 Program

```
def verify(s):
  if sum(s) == 0:
     partial = 0
     for i in s:
         partial = partial + i
         if partial < 0:
            return False
     return True
   return False


def prob(n):
    count = 0
    for _ in range(n):
        seq = [random.choice([-1,1]) for _ in range(l)]
        if verify(seq):
           count += 1
    return count/n
```

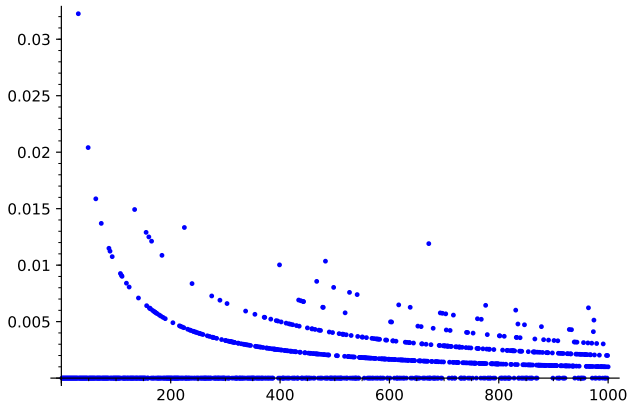
The first function **verify** takes in a sequence s and checks if the given sequence verifies our conditions by outputting a boolean. The second function **prob** computes n sequences of length l and counts how many of these sequences verify our conditions by outputting it's probability.

```
v = [(x,prob(x))  for x in range(1,n+1)]
```

We save the result in a list v of tuples up to n times we want to repeat the experiment. Now one only needs to plot the list using the `list_plot(v)` command.

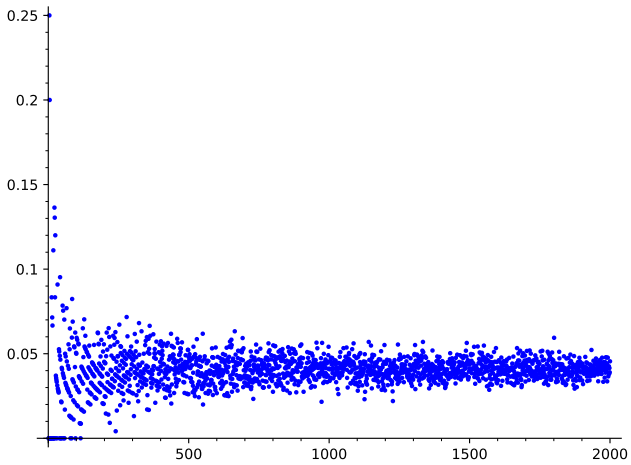
2.2 Results

The x-axis represents the n times we repeat the experiment and the y-axis the value of the **prob** function.



For $n = 1000$ and $l = 100$

We can see that the probability value does converge as we increase the number of tries. To have a better overview let us increase n , for computational power reasons we have to decrease the length of the sequence l so the program doesn't take too long.



For $n = 2000$ and $l = 10$

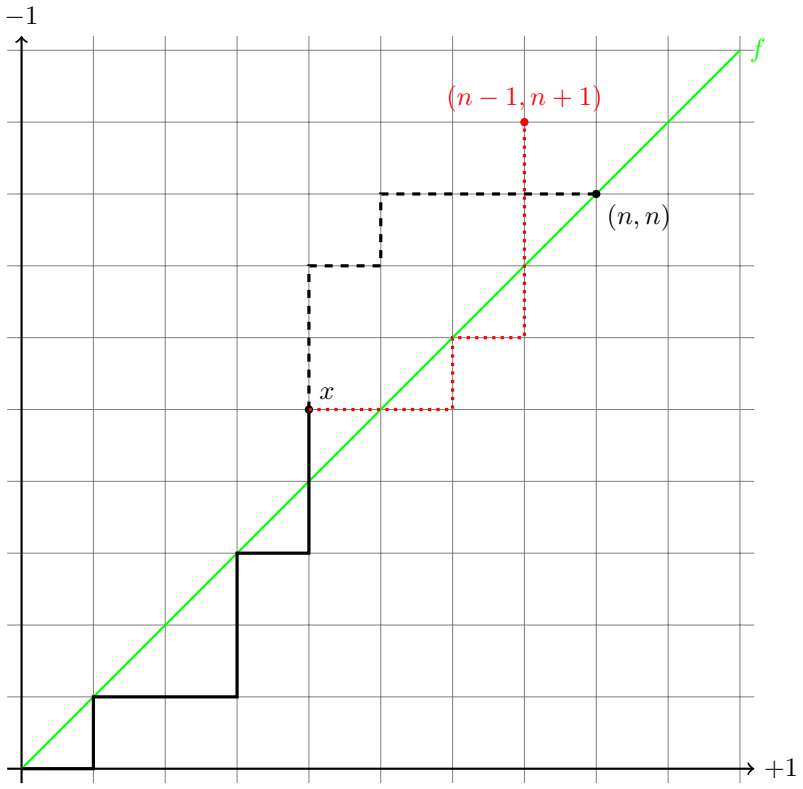
Here we can see that the probability value does converge from both sides to a certain value as we expected.

In both cases one can estimate the value to be around 0.002 for $l = 100$ and 0.05 for $l = 10$. The next step will be to determine this exact value and if does match our approximation from the experiment.

3 Theoretical approach

3.1 Preparation

Let us redefine our +1,−1 random sequence into *rightward* (−1) and *upward* (+1) steps. The solution to our coin flip problem are all the paths that end on the straight line $f(x) = x$ (sum equals to zero) and don't cross the line at any given moment (sum being always positive).



Claim. *The number of paths satisfying the above conditions is given by:*

$$\boxed{\binom{2n}{n} - \binom{2n}{n+1} \quad , n \in \mathbb{N}}$$

Remark. A point on the line f is denoted by (n, n) where $n = \frac{l}{2}$ with l being the length of the sequence.

Proof. To end up at the point (n, n) one has to choose n *upward* (or equivalently *rightward*) steps out of $2n$ total steps. So the number of paths satisfying this condition is equal to :

$$\binom{2n}{n}$$

Considering this number still contains paths which will end on the line but cross it at a given moment we need to exclude them. To find out the number of "bad paths" we will flip the portion of a path after it has crossed the line for the first time. By "flipping" one means interchanging all *upwards* and *rightward* steps. The section of a bad path which hasn't been flipped contains one more *upwards* step than *rightward* steps, hence the remaining path contains one more *rightward* step than *upward* step because it end on the line f. Since there are $2n$ steps and the flipping process doesn't change the total amount of steps , every flipped bad path will ends at the point $(n-1, n+1)$. Thus the number of bad paths is given by the possible permutations of $n+1$ *upward* steps (or equivalently, $n-1$ rightward steps) for a total of $2n$ steps :

$$\binom{n+1+n-1}{n+1} = \binom{2n}{n+1} = \binom{2n}{n-1}$$

By subtracting one finally obtains :

$$\binom{2n}{n} - \binom{2n}{n+1}$$

□

Let us develop this expression furthermore :

$$\begin{aligned}
\binom{2n}{n} - \binom{2n}{n+1} &= \frac{(2n)!}{n!\,(2n-n)!} - \frac{(2n)!}{(n+1)!\,(2n-n-1)!} \\
&= \frac{(2n)!}{n!\,(2n-n)(2n-n-1)!} - \frac{(2n)!}{(n+1)\,n!\,(2n-n-1)!} \\
&= \frac{(2n)!\,(n+1-2n+n)}{n!\,(2n-n-1)!\,(2n-n)(n+1)} \\
&= \frac{(2n)!}{n!\,(2n-n-1)!\,(2n-n)(n+1)} \\
&= \frac{(2n)!}{n!\,(2n-n)!\,(n+1)} = \frac{(2n)!}{(n+1)!\,n!} \\
&= \frac{1}{1+n}\binom{2n}{n}
\end{aligned}$$

One realizes that this is the formula for the **Catalan numbers**, which is a sequence of natural numbers given by :

$$\boxed{C_n = \frac{1}{1+n}\binom{2n}{n} = \frac{(2n)!}{(n+1)!\,n!} = \prod_{k=2}^{n} \frac{n+k}{k} \qquad \text{for } n \geq 0.}$$

3.2 The expected value

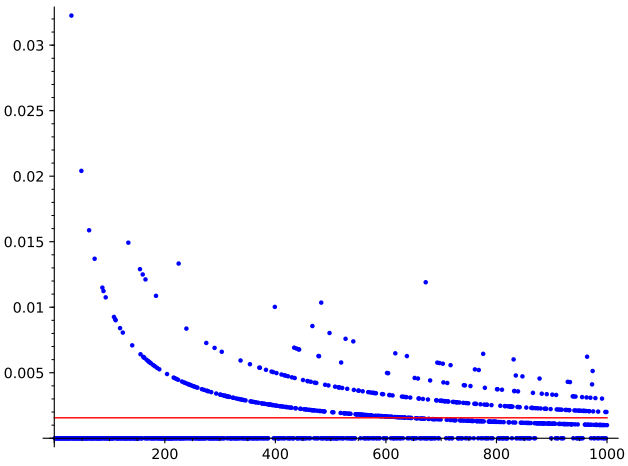
Now one can easily determine the probability of such a sequence to occur. Let's call this probability P_{100} for the sequence of length 100 and P_{10} for the sequence of length 10 :

$$P_{100} = \frac{C_{50}}{2^{100}} = 0.001560573282101544343 1 \simeq 0.00156 \tag{1}$$

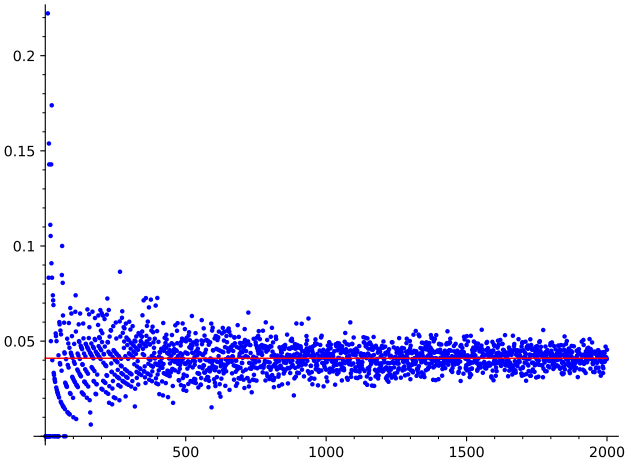
$$P_{10} = \frac{C_5}{2^{10}} = 0.041015625 \simeq 0.041 \tag{2}$$

Remark. Notice that the Catalan number C_{50} it the number of possibilities satisfying our condtions for a length $l = 100$ since we defined $n = \frac{l}{2}$. Analogously for C_5.

Let's see if these values correspond to our estimations:



For $n = 1000$ and $l = 100$



For $n = 2000$ and $l = 10$

Indeed, in both cases they converge to our calculated value (red line).

4 Outcome

4.1 What should player A do ?

In this section we will discuss if player A should accept the bet, if not under which conditions should he play such that the bet would be in his favour.
To answer this question it suffices to calculate the *expected value* $E[X]$, where X is a random variable and represents the outcome of either "winning 100eur" or "loosing 1eur". Since one has only two possible outcomes this is fairly easy:

$$E[X] = 100\,P_{100} - (1 - P_{100}) = -0.84238209850774402135$$

Thus player A shouldn't accept the bet under a $100:1$ ratio. The answer to

the second part it suffices to solve the following inequality where m represents the winning amount of money:

$$m\,P_{100} - (1 - P_{100}) > 0 \iff m > \frac{(1 - P_{100})}{P_{100}} = 639.79015799459995134$$

Let us suppose cents coins are not accepted in this bet, then the bet is favourable for player A if he accepts to play under a ratio of $640:1$

4.2 Biased coin and Markov Model

To answer the question in the case of a biased coin we will use the Markov Model to visualize our results. In order to do that we will take advantage of the implemented function *hmm.DiscreteHiddenMarkovModel* in Sage:

a=hmm.DiscreteHiddenMarkovModel([[1/3,2/3],[1/2,1/2]], [[1,0],[0,1]], [0.5,0.5],[-1, 1])

```
[1/3,2/3]
[1/2,1/2]
designs the transition matrix
[1,0]
[0,1]
designs the emission matrix
[0.5,0.5] is the initial probabilty condition and [-1,1] are the respective outcom
```

for detailed description visit:
http://doc.sagemath.org/html/en/reference/stats/sage/stats/hmm/hmm.html
Notice that in order to make it a normal Markov model we use the identity matrix as the emission matrix.

Now we can use

```
a.sample(l)
```

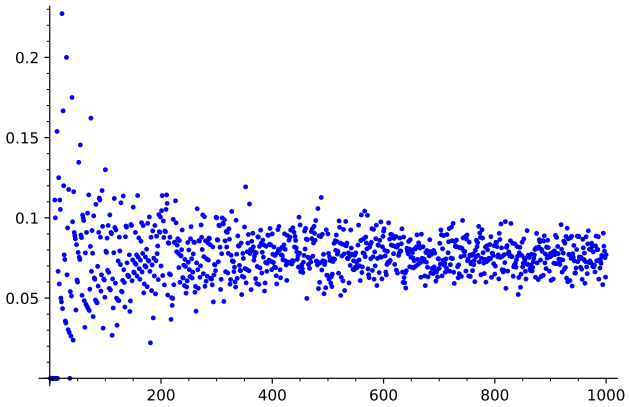
to sample a sequence of length l that obeys our biased coin,to write the following program:

```
def prob_markov(n):
        count = 0
        for _ in range(n):
          seq = a.sample(l)
          if verify(seq):
            count += 1

        return count/n
```

By saving this in a list as we did in section 2 and plotting it we get:

```
w = [(x,prob_markov(x))  for x in range(1,n+1)]
list_plot(w)
```



For $n = 2000$ and $l = 10$

One observes that the probability value of **prob_markov** function converges to a different value in the case of a biased coin.
The reader is welcome to play around with the program sheet and see how the graph changes accordingly to changes in the transitions matrix .

Reminder

We have created a interactive Jupyter sheet which contains all the programs and plots used , you are welcome to play around with. The sheet and the whole presentation has been put in Github repository which can be accessed HERE. In order to use this sheet make sure you are using the SageMath Kernel in Jupyter Notebook. It is also worth noticing that to compile the .tex file you must use the

```
pdflatex -synctex=1 -interaction=nonstopmode --shell-escape \%.tex
```

command in your texeditor.