Probability Simulations

Moti Ben-Ari

http://www.weizmann.ac.il/sci-tea/benari/

February 18, 2023

Contents

Introduction

Simulations are an excellent way of understanding probability, especially, the behavior of process of long duration. These programs enable the user to perform experiments by varying the parameters of problems and analyzing the results, both printed and displayed in graphs. A level of knowledge of probability equivalent to the first few chapters of [2] or [6] is assumed.

The simulations are of processes known as *Markov chains*, where the next state of the system depends only on the current state and not on the history of how the process got to the current state. These problems appear in probability textbooks [2, 6] and in much greater detail in [4, 1, 3, 5].

Parameters such as the probability can be modified interactively in order to see how the outcome depends on the values of the parameters.

Section 1 presents the *Gambler's ruin* while Section 2 presents the one-dimensional *Random walk*.

Technical notes

The programs are written in the Python 3 language and use the `matplotlib` to generate the graphs. Parameters directly related to the problems, such as the probability of success, can be modified interactively. Others, related to the simulation, such as the number of steps in a simulation and the properties of the histograms, are defined in a module `configuration.py` that can be modified.

You need to install the Python (`https://www.python.org/downloads/`) although a knowledge of Python programming is not necessary.

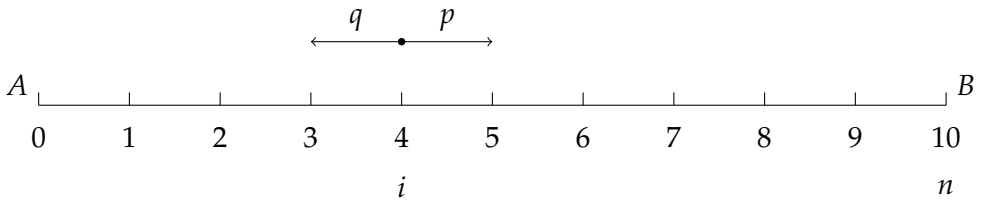
To run in the Visual Studio Code environment, ensure that the `Code Runner` extension is installed. I recommend that in the extension settings disable `Preserve Focus` and enable `Run In Terminal`.

To run in the IDLE or Thonny environments, change the configuration constant `CLOSE` to `True`. When the simulation is run multiple times, you will have to close each figure before running a new simulation.

1 Gambler's Ruin

Problem Two players A and B compete in a contest. There is an initial finite capital of n units: A has i and B has $n - i$. They repeatedly play a game where the probability that A wins is p and the probability that B wins is $q = 1 - p$. The loser gives one unit to the winner. When one player has all n units the contest is finished.

1. Given initial parameters (p, n, i), what is the probability that A wins?
2. What is the expected duration of the game?



The most extensive presentation the gambler's ruin is in [5, Chapter 2] which includes the solution to the expected duration of the contest. Note that Privault asks for the probability that A is ruined, that is, that B wins. I follow other references which ask for A's probability of winning.

1.1 Theoretical results

Given (p, n, i) the probability that A wins the contest is:

$$P_A(p, n, i) = \left(\frac{1 - r^i}{1 - r^n} \right) ,$$

where $r = q/p$. By symmetry, the probability that B wins is:

$$P_B(p, n, i) = \left(\frac{1 - (1/r)^{n-i}}{1 - (1/r)^n} \right) .$$

There are separate solutions for $p \neq 1/2$ and $p = 1/2$. For $p \neq 1/2$ the expected duration of the contest is:

$$E_{duration}(p, n, i) = \frac{1}{q - p} \left(i - n \frac{1 - r^k}{1 - r^n} \right) .$$

For $p = 1/2$ the expected duration of the contest is:

$$E_{duration}(p, n, i) = i(n - 1) .$$

Of course the duration does not depend on which player wins. If A wins, the contest terminates for B also, and conversely.

1.2 Program structure

`configuration.py` contains declarations of variables which are intended to be constant.

`gambler_plot.py` contains the functions for plotting the histogram of the duration of the contests. If the simulation is run for multiple probabilities or initial values, a graph of the proportion of wins is also displayed.

`gamblers_ruin.py` is the main program which obtains the parameters, runs the simulations, prints the output and calls the plotting functions.

1.3 Running the simulations

The program runs the simulations in a loop, each time asking the user how to run it. You can run the same simulation again with the saved parameters, enter new parameters, or run a sequence of simulations for a range of probabilities or initial values.

A typical output is as follows:

```
Probability = 0.45, capital = 20, initial = 8
Wins = 789, losses = 9211, limits exceeded = 0
Proportion of wins     = 0.0789
Probability of winning = 0.0732
Average duration  = 65
Expected duration = 65
```

The results of the simulation are very close to the theoretical probability and expected duration. The proportion of wins and the histogram of the durations are shown in Figures 1, 2. The vertical lines are the average durations.

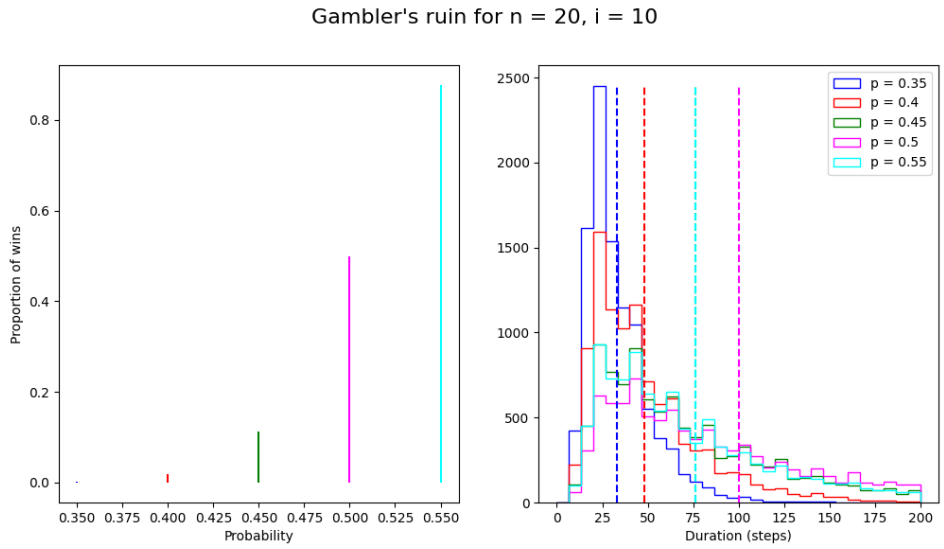


Figure 1: Proportion of wins and histogram for $n = 20, i = 10$ and multiple probabilities

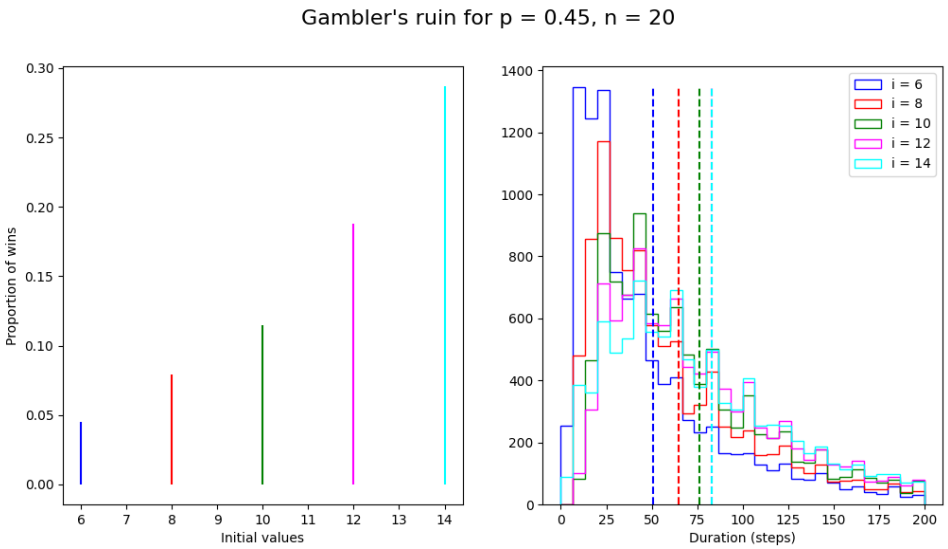
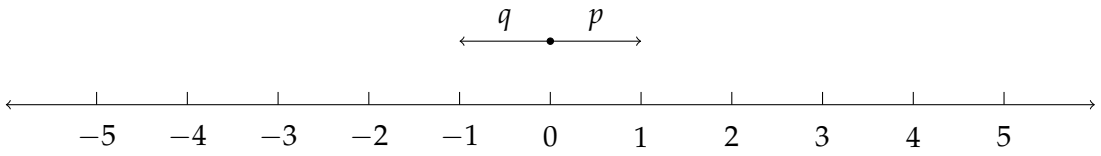


Figure 2: Proportion of wins and histogram for $p = 0.45, n = 20$ and multiple initial values

2 Random Walk

Problem A particle is placed at the origin of the x-axis. It repeatedly takes steps: right with probability p and left with probability $q = 1 - p$.

1. What is the probability that the particle will return to the origin?
2. What is the expected duration until the particle returns to the origin?



The clearest presentation of one-dimensional random walk is in [3], but the derivation of the expected duration is in [5].

2.1 Theoretical results

By symmetry, without loss of generality let the first step be to the right.

The particle can only return to the origin after an even number of steps. Assume that $p = 1/2$. Let S_{2m} be the position of the particle after $2m$ steps. Then:

$$P(S_{2m} = 0) = \binom{2m}{m} \frac{1}{2^{2m}},$$

which by Stirling's formula is:

$$P(S_{2m} = 0) \approx \frac{1}{\sqrt{\pi m}}.$$

It can now be proved that the probability of a return to the origin is 1.

For $p \leq 1/2$, P_{origin}, the probability of a return to the origin, is 1 and for $p \geq 1/2$ the probability is (Figure 3):

$$P_{origin} = \frac{q}{p} = \frac{1-p}{p}.$$

E_{origin}, the expected duration until the first return to the origin, is infinite for $p \geq 1/2$ while for $p < 1/2$ it is:

$$E_{origin} = \frac{1}{q-p} = \frac{1}{1-2p}.$$

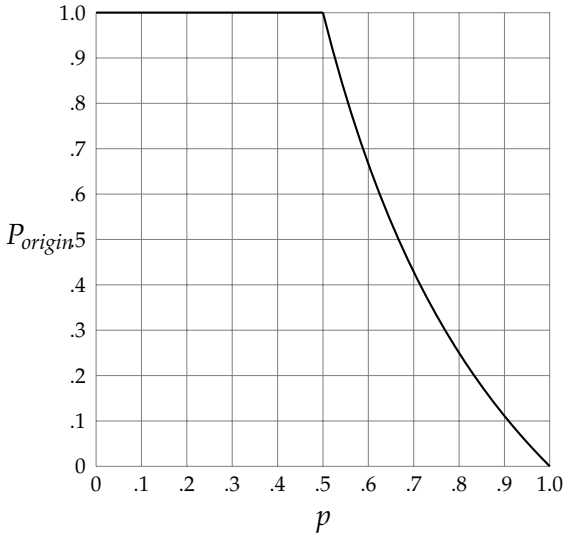


Figure 3: Graph of P_{origin}

2.2 Program structure

`configuration.py` contains declarations of variables which are intended to be constant.

`random_walk_plot.py` contains the functions for plotting a graph of the proportion of simulations that return to the origin and the mean duration if the simulation is run for multiple probabilities or limits.

`random_walk.py` is the main program which obtains the parameters, runs the simulations, prints the output and calls the plotting functions.

2.3 Running the simulations

The program runs the simulations in a loop, each time asking the user how to run it. You can run the same simulation again with the saved parameters, enter new parameters, or run a sequence of simulations for a range of probabilities or limits.

A typical output is as follows:

```
Probability = 0.50, step limit   = 1000
Proportion returning to origin   = 0.977
Probability of return to origin  = 1.000
Proportion reaching limit        = 0.023
Mean duration (steps)            = 49
Expected duration (steps)        = infinity
```

The proportion of wins in the simulation are very close to the theoretical probability, but the mean duration is far from infinite because the step limit was too small. The proportion

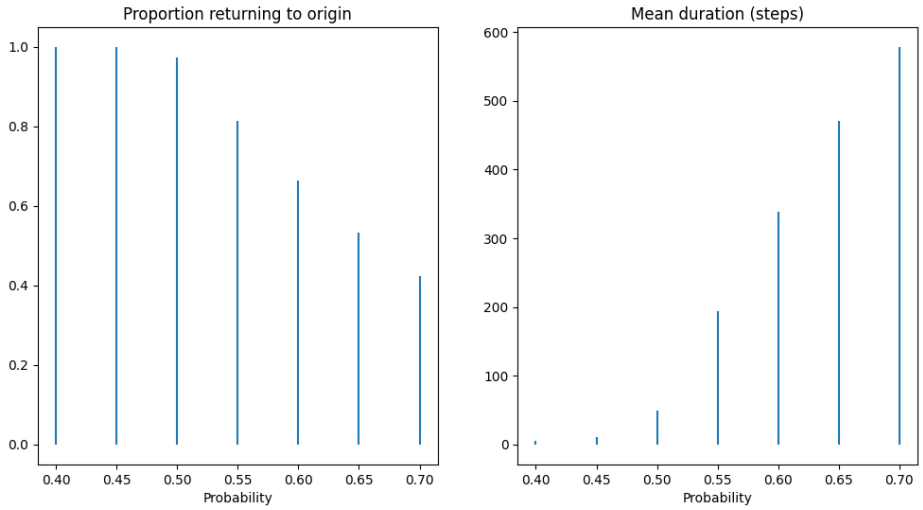


Figure 4: Proportion of wins and and mean durations for multiple probabilities

of wins and the mean durations are shown in Figures 4 and 5.

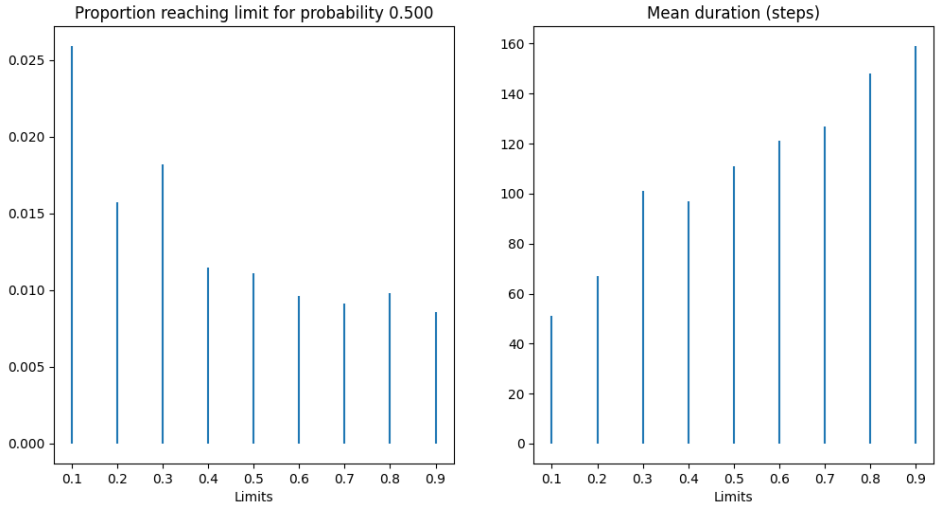


Figure 5: Proportion of wins and and mean durations for multiple limits

References

[1] Moti Ben-Ari. Mosteller's challenging problems in probability. `https://github.com/motib/probability-mosteller`.

[2] Joseph K. Blizstein and Jessica Hwang. *Introduction to Probability (Second Edition)*. CRC Press, 2019.

[3] K.C. Border. Lecture 16: Simple random walk. `http://www.math.caltech.edu/~2016-17/2term/ma003/Notes/Lecture16.pdf`, 2017.

[4] Frederick Mosteller. *Fifty Challenging Problems in Probability with Solutions*. Dover, 1965.

[5] Nicolas Privault. *Understanding Markov Chains: Examples and Applications (Second Edition)*. Springer, 2018.

[6] Sheldon Ross. *A First Course in Probability (Tenth Edition)*. Pearson, 2019.