Statistics

Federico Coccarelli 1845722

December 2022

1 Introduction

The random walk is one of the most fundamental models in probability theory, demonstrating profound mathematical properties. It has a broad range of applications in various scientific fields such as physics, chemistry and biology. Formally, a random walk is any process that follows a procession of random steps in a mathematical space such as a lattice or graph. The direction and magnitude of these steps can follow varying probabilistic distributions, leading to interesting properties to study. Random walks can be found within diverse areas of mathematics and science, from animal foraging and cell movement to ferromagnetism and long-chain polymers. As a result of this, they are of great interest to academics, with results in the field having countless initially unforeseen applications. There are many different classes of random walk, with each class harboring an extensive array of interesting properties and applications.

2 Random Walks

2.1 One Dimensional Simple Random Walks

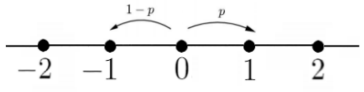
The simple random walk, or drunkards walk, is a fundamental system in probability theory, with applications such as gambling and electric networks. The simple random walk is particularly important because it forms the basis upon which more complicated walks can be later defined. This section will demonstrate some key theorems and properties about the simple random walk. Key information that will be covered is the expected position of the walk and the distribution that governs it.
To construct the simple random walk in one dimension, let $X_1, X_2, ..., X_n$ denote independent and identically distributed random variables s.t.

$$X_i = \begin{cases} +1 \text{ with probability p} \\ -1 \text{ with probability q} \end{cases}$$

With p $\in (0, 1)$ and q = 1 - p. Let S_n denote the sum

$$\sum_{i=1}^{n} X_i$$

The random variables X_i represent individual steps, and Sn represents the position of the walker after exactly n steps.



For simplicity, let the step-size of the random walker be 1, and let the origin be $S_0 = 0$.
The first question one might ask is what is the expected location of the simple random walk after n steps?

Theorem 2.1 $E[S_n] = n(p - q)$
Proof. An elementary result from probability theory states that if $X_1, X_2, ..., X_n$ are random variables then

$$E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i]$$

therefore,

$$E[S_n] = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] = nE[X_1]$$

Where the last equality is a result of $X_1, X_2, ..., X_n$ being identically distributed. This gives

$$E[S_n] = n[(1)p + (-1)q] = n(p - q)$$

Remark. If $p = q = 1/2$, then we have that $E[S_n] = 0$, i.e. the expected position of the walker will be at the origin for any number of steps n. This kind of simple random walk is called a symmetric random walk.

What is the probability, P(m, N), that the walker will be at position m after N steps?
Let p be the probability of a right step and $1 - p = q$ be the probability of a left step. There are many ways to obtain such a path when $m < N$ but each path is independent thus we can simply add up their probabilities. We know that the walker must start from 0 and we also know that the walker must make $n_1 = m + n_2$ steps to the right and n_2 steps to the left as $n_1 + n_2 = N$. It follows that $n1 = \frac{1}{2}(N + m)$ and $n2 = \frac{1}{2}(N + m)$. Now, since each path is independent, we know that there must be n_1 factors of p and n_2 factors of q. Using this we get:

$$p^{n_1} q^{n_2} = p^{\frac{1}{2}(N+m)} p^{\frac{1}{2}(N-m)}$$

which is simply the probability of any one path. We must multiply that by the total number of ways a path of this sort can occur which is:

$$\frac{N!}{n_1! n_2!} = \frac{N!}{n_1!(N! - n_1!)}$$

Thus we ultimately obtain:

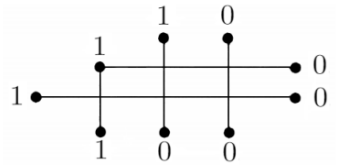
$$P(m, N) = \frac{N!}{(\frac{N+m}{2})!(\frac{N-m}{2})!} p^{\frac{1}{2}(N+m)} p^{\frac{1}{2}(N-m)}$$

That is clearly a binomial distribution.
This result applied to a symmetric distribution make it possible to show, in the following table, the probability to travel a specific distance in after N steps:

Step	-5	-4	-3	-2	-1	0	1	2	3	4	5
0						1					
1					1/2		1/2				
2				1/4		2/4		1/4			
3			1/8		3/8		3/8		1/8		
4		1/16		4/16		6/16		4/16		1/16	
5	1/32		5/32		10/32		10/32		5/32		1/32

2.2 Two Dimensional Simple Random Walks

The simple random walk on Z^2 has four choices at every vertex, the ± X direction, and the ± Y direction. This process is analogous to two simple random walks on Z where each step you flip a coin to determine which of the two walks takes the next step. Thus the y co-ordinate follows the same behavior as the one dimensional case with k steps, and the x co-ordinate follows the same behaviour of a one-dimensional random walk of (n - k) steps, where the number of steps in the y direction, k, follows a binomial distribution with p = 0.5.
The simple random walk on a finite subset of Z^2 is particularly useful in modelling temperature gradients on finite domains. To begin, take a finite subset of Z^2 and set boundary points as 1 or 0, as in the Figure below.



Then take a simple random walker starting in the interior, taking steps until it hits one of the absorbing boundary points. The expected value of the random walk's boundary point once it has finished is equivalent to the probability of it finishing at a 1, denoting a win. To calculate p(x, y), there are two numerical methods, the Monte-Carlo Method and the Method of Relaxation.

2.2.1 The Monte-Carlo Method

The Monte-Carlo method comprises of simulating a large number of simple random walks, beginning at the different interior points, and estimating p(x, y) as the ratio of successful random walks to the total number of random walks. This method is guaranteed to converge to the true solution due to the Law of Large Numbers. This method requires a large number of simulations and is thus expected to be the less efficient method.

2.2.2 The Method of Relaxation

The next method of approximating p(x, y) is the Method of Relaxation. This method takes advantage of the fact that our solution is harmonic, and thus has the averaging property:

$$p(x,y) = \frac{p(x+1,y) + p(x-1,y) + p(x,y+1) + p(x,y-1)}{4}$$

The algorithm takes a point in the interior, next to the boundary, and replaces its value for the mean of the four points adjacent to it. The next step is to chose a neighbouring point and do the same process, until all of the interior points have been 'averaged' exactly once. After many iterations, it approximately demonstrates the averaging property and is consequently a good approximation for p(x, y).

3 Brownian Motion

Brownian Motion was the real start to the study of random walks. However, Brownian motion covers many more cases than just that of the simple symmetric walk. Brownian motion is inspired by the movement of particles living in a three-dimensional space as opposed to the two directions we limited the simple random walk to. Thus the mathematical representation of the various scenarios quickly becomes more complex.
In this section we will discuss the basic definition of Brownian motion, as well as a construction of Brownian motion from a simple random walk.

Definition 2.1. The stochastic process B = B(t), t $\geq$ 0 is a Brownian motion if 1. The starting position is always zero so B(0) = 0,
2. B(t) has independent increments, i.e. for all times $0 \leq t_1 \leq t_2$ the increments $B(t_2) - B(t_1)$ and $B(t_1) - B(t_0)$ are independent random variables,
3. For all t $\geq$ 0 and h $>$ 0, the increments $B(t + h) - B(t)$ are normally distributed with expectation zero and variance h,
4. The function $t \to B(t)$ is continuous.

Brownian motion is the limit of the simple random walk where the step-size is getting smaller, and the 'speed' of the walk is increasing proportionally. To demonstrate this, we will first show that

$$B_n(t) = \frac{\sum_{1 \leq i \leq \lfloor nt \rfloor}(X_i)}{\sqrt{n}}$$

satisfies all of the conditions (1-4) for Brownian motion when $n \to \infty$.
1. $B_n(0) = 0$ by definition.
2. Let $t_1 < t_2$ be fixed and consider

$$B_n(t_1) = \frac{\sum_{1 \leq i \leq \lfloor nt_1 \rfloor}(X_i)}{\sqrt{n}}$$

and

$$B_n(t_2) - B_n(t_1) = \frac{\sum_{\lfloor nt_1 \rfloor < i \leq \lfloor nt_2 \rfloor}(X_i)}{\sqrt{n}}$$

The two sums contain mutually exclusive subsets of the sequence $X_1, X_2, ...$ and since the sequence is i.i.d., $B_n(t_1)$ and $B_n(t_2) - B_n(t_1)$ are independent.

3. By the Central Limit Theorem, for every $0 < h \leq t$, the distribution of

$$\frac{\sum_{\lfloor nt \rfloor < i \leq \lfloor n(t+h) \rfloor}(X_i)}{\sigma\sqrt{n\lfloor (t+h) \rfloor - \lfloor nt \rfloor}}$$

converges to the standard normal distribution as $n \to \infty$. The difference between $\lfloor nt \rfloor - n\lfloor t+h \rfloor$ is at most 1 and becomes negligible as $n \to \infty$, thus we write

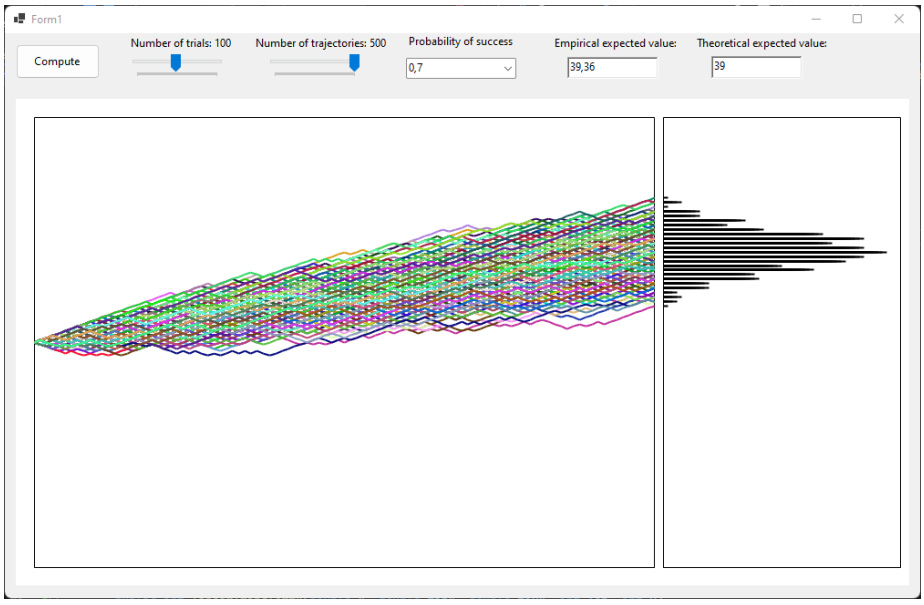
$$\frac{\sum_{\lfloor nt \rfloor < i \leq \lfloor n(t+h) \rfloor}(X_i)}{\sigma\sqrt{nh}} = \frac{B_n(t+h) - B_n(t)}{\sigma\sqrt{h}}$$

4. Following the above reasoning, with a very small h the difference $|B_n(t+h) - B_n(t)|$ can become arbitrarily small as $n \to \infty$ with probability 1. Thus $B_n(t)$ becomes a continuous function with probability 1 as $n \to \infty$.

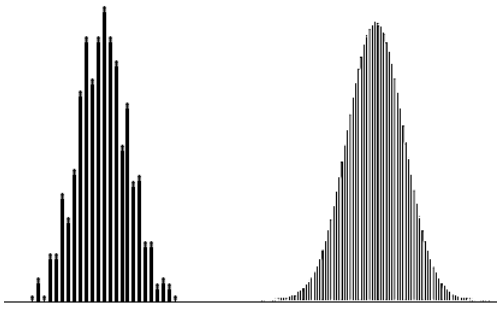
Therefore, $B := lim_{n\to\infty} B_n(t)$ is a Brownian motion.

4 Simulation

In this section is show a simulation of a simple random walk made using C#. The figure shows the paths of 500 simple random walks with $p = 0.7$ along 200 steps.



The result shows how the empirical average arrival distance is a good approximation of the theoretical arrival distance given by Theorem (2.1).
In the following figure we can notice the similarities between the final positions of the simulation and a binomial distribution.



5 Implementation

```
namespace ProgettoTesi
{
    public partial class Form1 : Form
    {

        Bitmap b;
        Graphics g;
        Random random = new Random();
        Pen PenTrajectory = new Pen(Color.OrangeRed, 1);
        Pen PenHistogram = new Pen(Color.Black, 2);
        int lastX;
        int lastY;
        int[] results = new int[2000];

        public Form1()
        {
            InitializeComponent();
            comboBox1.Text = "0,5";
            b = new Bitmap(pictureBox1.Width,
                pictureBox1.Height);
            g = Graphics.FromImage(b);
            g.SmoothingMode =
                System.Drawing.Drawing2D.SmoothingMode.HighQuality;
            g.Clear(Color.White);

        }

        private void button1_Click(object sender, EventArgs e)
        {
            g.Clear(Color.White);
            int TrialsCount = trackBar1.Value;
            double successProbability;
            Boolean success = Double.TryParse(comboBox1.Text,
                out successProbability);
            if (!success || (success & (successProbability > 1
                || successProbability < 0))) successProbability
                = 0.5;
            int TrajectoryNumber = trackBar2.Value;

            Rectangle r = new Rectangle(20, 20, b.Width - 300,
                b.Height - 40);
            g.FillRectangle(Brushes.White, r);
            g.DrawRectangle(Pens.Black, r);
```

```
Rectangle r2 = new Rectangle(r.Right + 10, 20,
    260, b.Height - 40);
g.FillRectangle(Brushes.White, r2);
g.DrawRectangle(Pens.Black, r2);

double minX = 0;
double maxX = (double)TrialsCount;
double minY = 0;
double maxY = (double)TrialsCount;

Array.Clear(results, 0, results.Length);
double sum = 0;

for (int t = 0; t < TrajectoryNumber; t++)
{

    lastX = r.Left;
    lastY = r.Bottom;
    double Y = 0;
    PenTrajectory.Color =
        Color.FromArgb(random.Next(0, 256),
        random.Next(0, 256), random.Next(0, 256));
    PenTrajectory.Width = 2;
    for (int X = 1; X <= TrialsCount; X++)
    {
        random.NextDouble();

        if (random.NextDouble() <
            successProbability) Y = Y + 1;
        else Y = Y - 1;

        int xCord = linearTransformX(X, minX,
            maxX, r.Left, r.Width);
        int yCord = linearTransformY(Y, minY,
            maxY, r.Top, r.Height);

        g.DrawLine(PenTrajectory, lastX,
            lastY-r.Height/2, xCord, yCord -
            r.Height / 2);
        lastX = xCord;
        lastY = yCord;
        if (X == TrialsCount)
        {
            results[lastY] += 5; //histogram bar
                increases by 5px
```

```
                g.DrawLine(PenHistogram, r2.Left,
                    lastY - r.Height / 2, r2.Left +
                    results[lastY], lastY - r.Height /
                    2);

                sum += Y;

                pictureBox1.Image = b;
                pictureBox1.Update();
            }
            pictureBox1.Image = b;
        }

    }
    double mean =
        ((float)sum/(double)TrajectoryNumber);
    richTextBox1.Text = mean.ToString();
    richTextBox2.Text = ((int)(TrialsCount * (2*
        successProbability-1))).ToString();
}

public int linearTransformX(double X, double minX,
    double maxX, int Left, int W)
{
    return Left + (int)(W * ((X - minX) / (maxX -
        minX)));
}

public int linearTransformY(double Y, double minY,
    double maxY, int Top, int H)
{
    return Top + (int)(H - H * ((Y - minY) / (maxY * 2
        - minY)));
}

private void trackBar1_Scroll(object sender, EventArgs
    e)
{
    label1.Text = "Number of trials: " +
        trackBar1.Value.ToString();
}

private void trackBar2_Scroll(object sender, EventArgs
    e)
{
    label2.Text = "Number of trajectories: " +
```

```
            trackBar2.Value.ToString();
        }

        private void label4_Click(object sender, EventArgs e)
        {

        }
    }
```