Transition Sequences for 2-state Markov Chains

Markov Chains

Introduction

A Markov chain is a process without memory. Every possible step in a process can be assigned to an element in a matrix called transition matrix. In the following, I will only and only look at 2 step Markov chains.
To characterize a Markov chain we need a transition matrix and the initial probabilities. The initial probabilities will tell us the odds of the Markov chain starting with state 1 or 2.
A transition matrix looks like:

```
In[1]:= 𝒯 = {{a, 1 - a}, {1 - b, b}};
        𝒯 // MatrixForm
```

Out[2]//MatrixForm=

$$\begin{pmatrix} a & 1-a \\ 1-b & b \end{pmatrix}$$

Such a transition matrix says that the probability of a step from state 1 to 1 (not switching) is “a”. The probability of switching from state 1 to 2 is “1-a” etc. The initial probabilities on the other hand are 2x1 matrices.

```
In[3]:= ℐ = {c, 1 - c};
        ℐ // MatrixForm
```

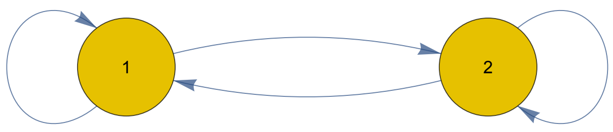
Out[4]//MatrixForm=

$$\begin{pmatrix} c \\ 1-c \end{pmatrix}$$

This matrix reveals that the probability of the system starting at state 1 is “c” while the probability of it starting at state 2 is “1-c”
A discrete Markov chain can be visualized using graphs:

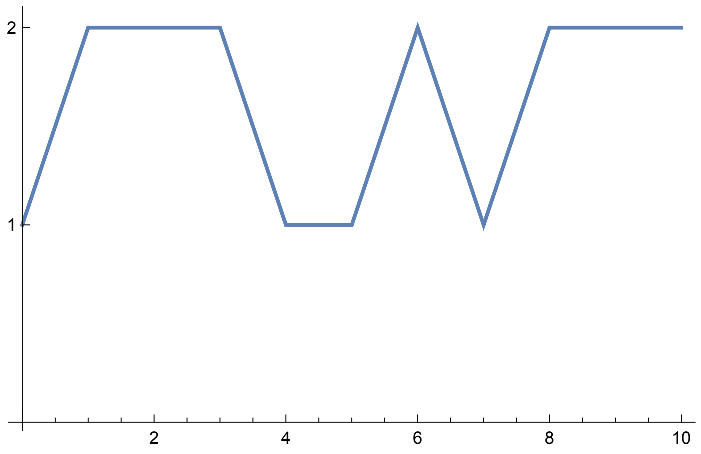
```
In[5]:= chain = DiscreteMarkovProcess[ℐ, 𝒯];
        Graph[chain]
```

Out[6]=

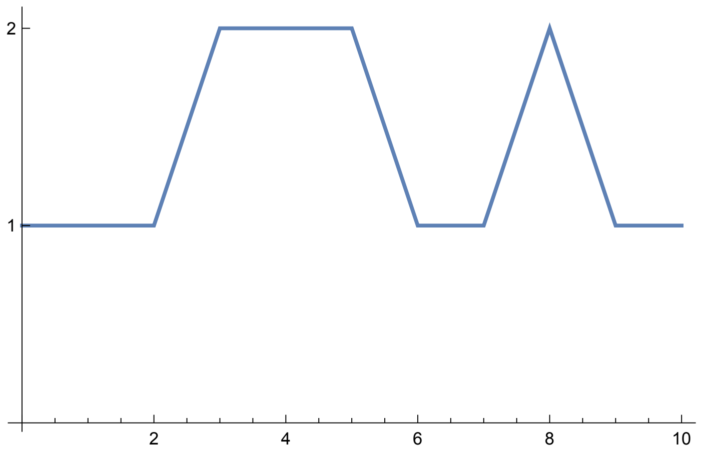


We can run simulation and see how a random process characterized by a Markov chain looks like.

```
In[7]:=  a = 0.7;
         b = 0.8;
         c = 0.5;

In[10]:= chain1 = DiscreteMarkovProcess[ℐ, 𝒯];
         sim1 = RandomFunction[chain1, {0, 10}];
         ListLinePlot[sim1, Ticks → {Automatic, {1, 2}}]

Out[12]=
```



```
In[13]:= a = 0.8;
         b = 0.6;
         c = 0.5;

In[16]:= chain1 = DiscreteMarkovProcess[ℐ, 𝒯];
         sim1 = RandomFunction[chain1, {0, 10}];
         ListLinePlot[sim1, Ticks → {Automatic, {1, 2}}]

Out[18]=
```



Question!

These are only two simulation for a 9step Markov chain. From now on I show the number of steps with N (so here N=9) Attention! 9 steps are equivalent to 10 data points.
Now the question that rises is the probability of sequences. **What sequence is the most probable**

<u>sequence knowing the transition matrix and initial probabilities?</u>

Investigation

To keep everything simple let' s look at the Markov chains with N = 4. All the possible sequences in such scenario would be :

```
In[19]:= N4allseq = Tuples[{1, 2}, 5]
Out[19]=
{{1, 1, 1, 1, 1}, {1, 1, 1, 1, 2}, {1, 1, 1, 2, 1}, {1, 1, 1, 2, 2}, {1, 1, 2, 1, 1},
 {1, 1, 2, 1, 2}, {1, 1, 2, 2, 1}, {1, 1, 2, 2, 2}, {1, 2, 1, 1, 1}, {1, 2, 1, 1, 2},
 {1, 2, 1, 2, 1}, {1, 2, 1, 2, 2}, {1, 2, 2, 1, 1}, {1, 2, 2, 1, 2}, {1, 2, 2, 2, 1},
 {1, 2, 2, 2, 2}, {2, 1, 1, 1, 1}, {2, 1, 1, 1, 2}, {2, 1, 1, 2, 1}, {2, 1, 1, 2, 2},
 {2, 1, 2, 1, 1}, {2, 1, 2, 1, 2}, {2, 1, 2, 2, 1}, {2, 1, 2, 2, 2},
 {2, 2, 1, 1, 1}, {2, 2, 1, 1, 2}, {2, 2, 1, 2, 1}, {2, 2, 1, 2, 2},
 {2, 2, 2, 1, 1}, {2, 2, 2, 1, 2}, {2, 2, 2, 2, 1}, {2, 2, 2, 2, 2}}
```

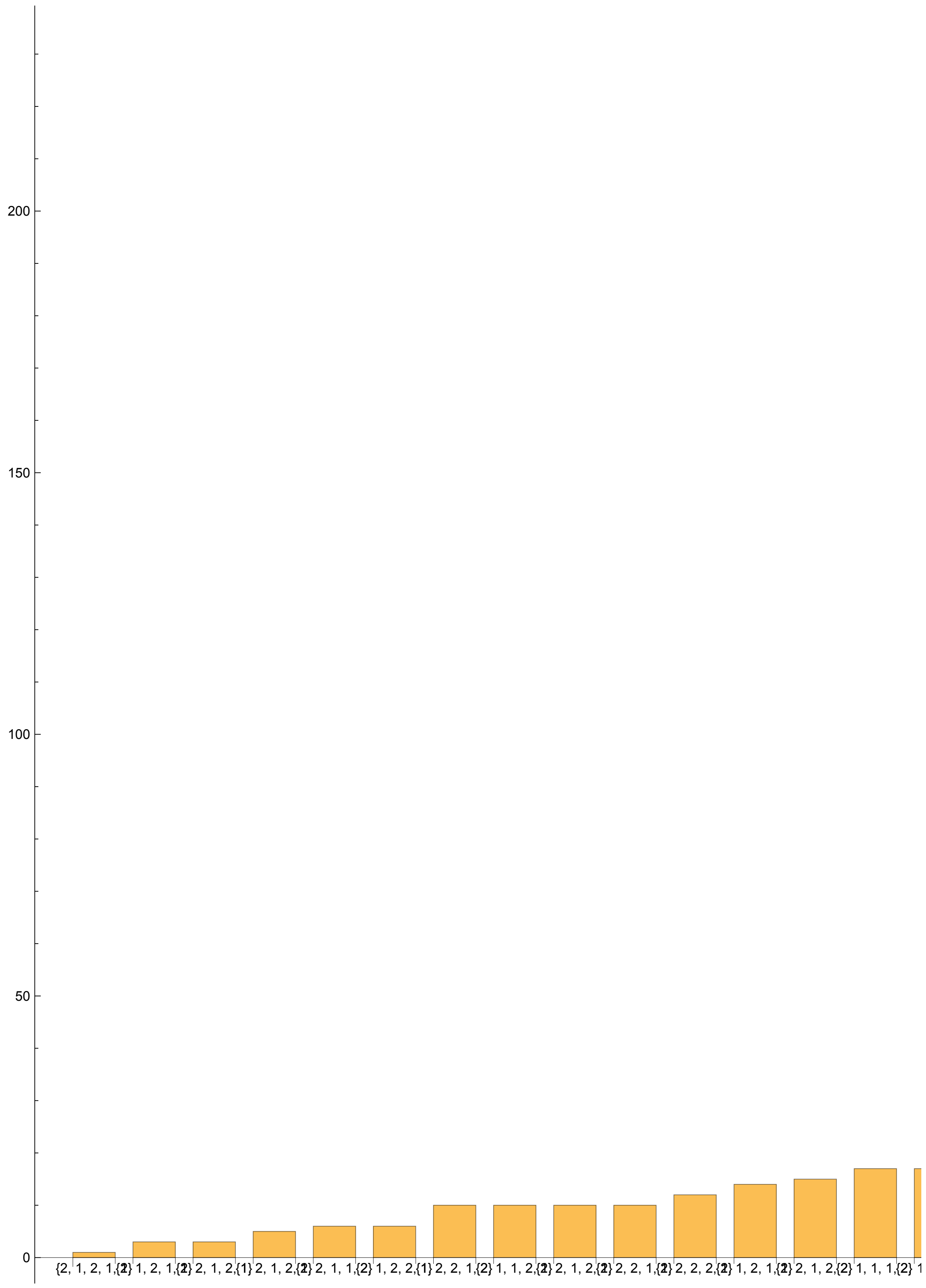
If I run a simulation of a Markov Process what sequence would show up more?

```
In[20]:= numRuns = 1000;
```

Here I run my simulations numRuns times!

```
In[21]:= sim1 = Table[RandomFunction[chain1, {0, 4}]["States"], {numRuns}];
         sim1 = Flatten[sim1, 1];
         countListSim1 = {#, Count[sim1, #]} & /@ N4allseq;
         countListSim1 = SortBy[countListSim1, Last];
         BarChart[countListSim1⟦All, 2⟧,
          ChartLabels → (ToString /@ countListSim1⟦All, 1⟧)]
```

Out[25]=

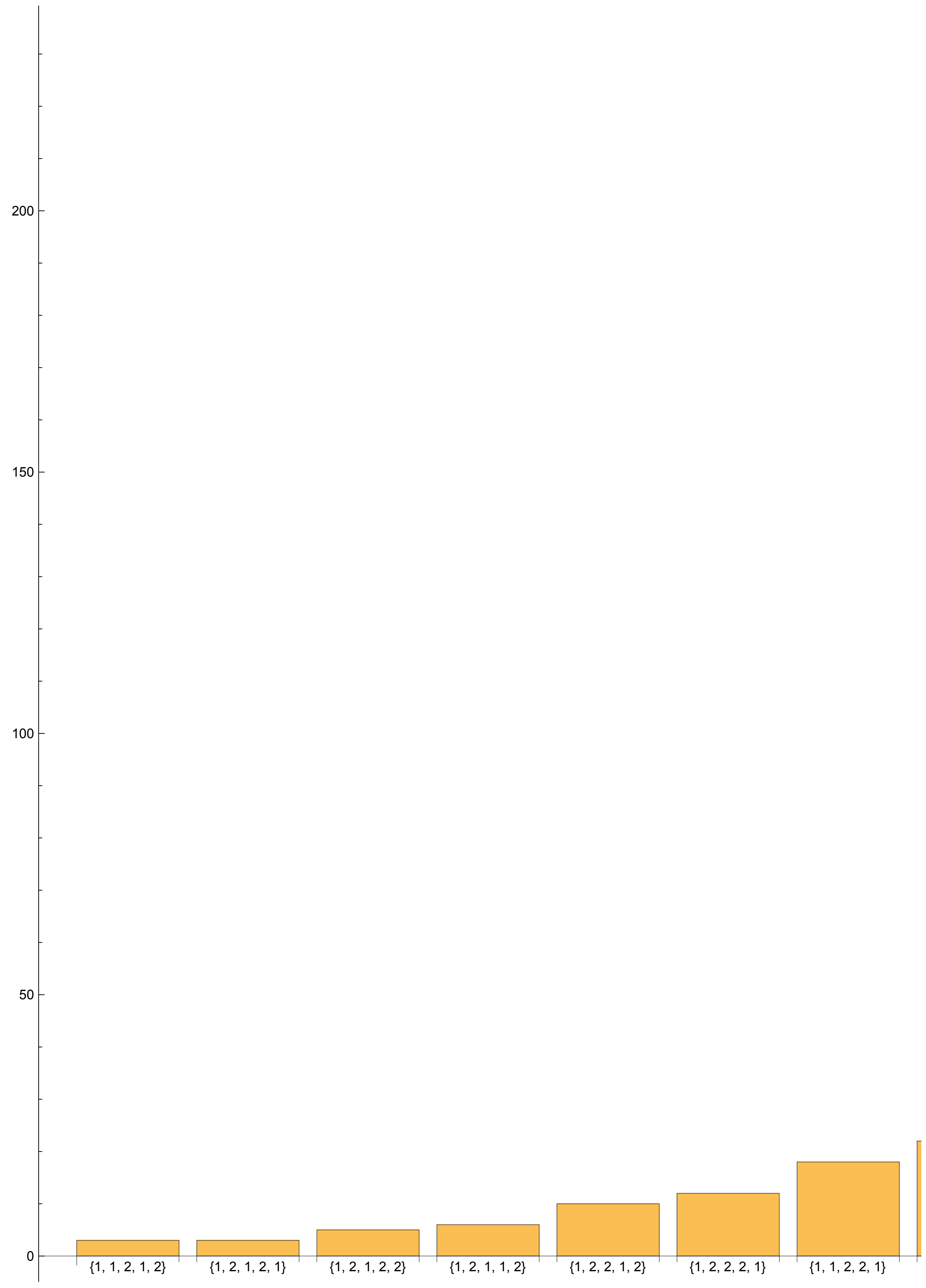


As we can see the amount of times the sequence {1,1,1,1} is showing up is higher than any other sequence. However, it doesn't necessarily mean that this sequence has the higher probability to show up. The reason is that there is only one way for a sequence to always stay at state to but there are multiple ways a sequence can switch. To simplify let us only look at the sequences starting with the first state. We can always adjust everything for scenarios where the sequence starts with the second state.

```
In[26]:= N4allseq1 = Select[N4allseq, #⟦1⟧ == 1 &]
Out[26]=
{{1, 1, 1, 1, 1}, {1, 1, 1, 1, 2}, {1, 1, 1, 2, 1}, {1, 1, 1, 2, 2},
 {1, 1, 2, 1, 1}, {1, 1, 2, 1, 2}, {1, 1, 2, 2, 1}, {1, 1, 2, 2, 2},
 {1, 2, 1, 1, 1}, {1, 2, 1, 1, 2}, {1, 2, 1, 2, 1}, {1, 2, 1, 2, 2},
 {1, 2, 2, 1, 1}, {1, 2, 2, 1, 2}, {1, 2, 2, 2, 1}, {1, 2, 2, 2, 2}}
```

```
In[27]:= countListSim1 = {#, Count[sim1, #]} & /@ N4allseq1;
         countListSim1 = SortBy[countListSim1, Last];
         BarChart[countListSim1⟦All, 2⟧,
          ChartLabels → (ToString /@ countListSim1⟦All, 1⟧)]
```

Out[29]=

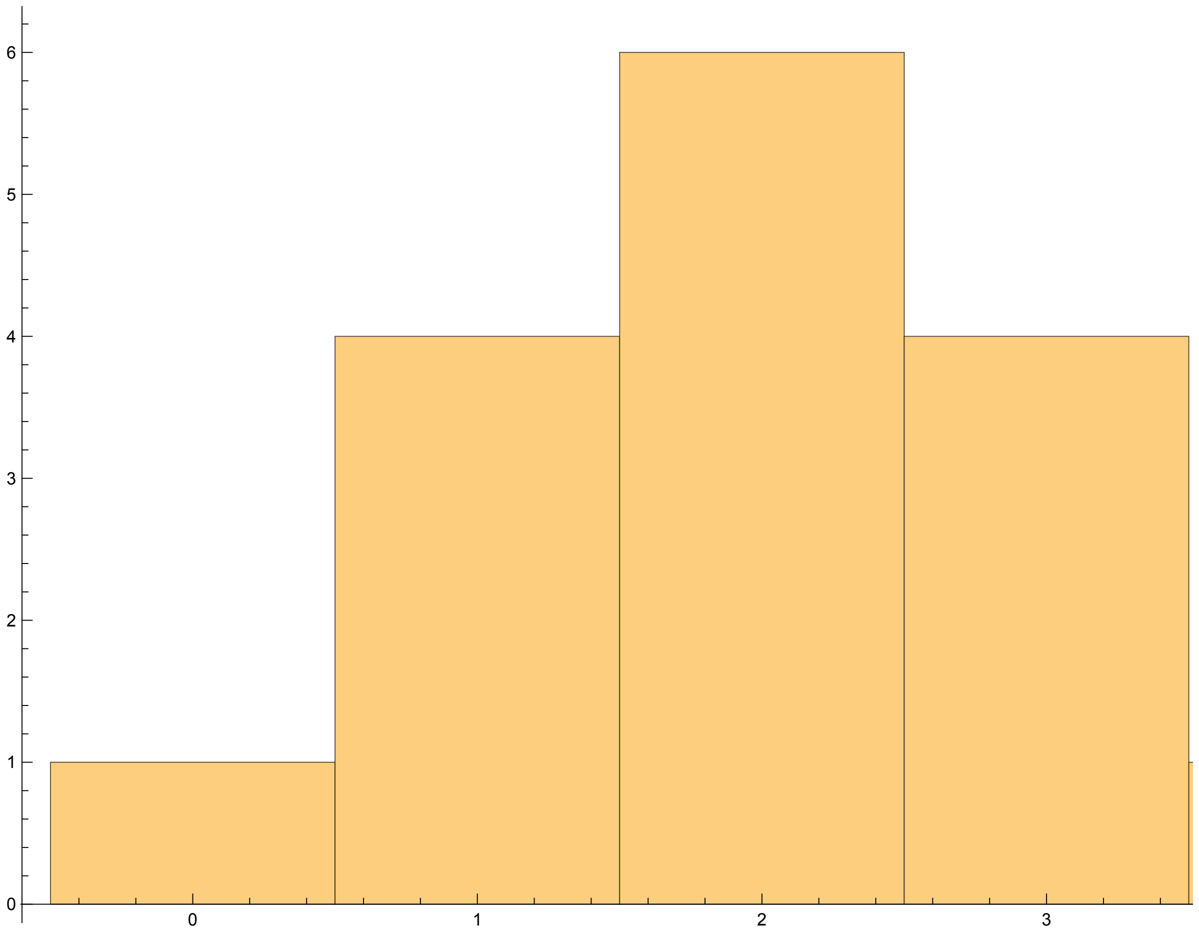


Now let’s define a function that counts the amount of jumps (transitions) that the sequences go through.

```
In[30]:= CountJumps[sequence_] := Count[Differences[sequence], x_ /; Abs[x] == 1];
```

```
In[31]:= numberOfJumpsExample = Map[CountJumps, N4allseq1];
         Histogram[numberOfJumpsExample, Automatic]
```

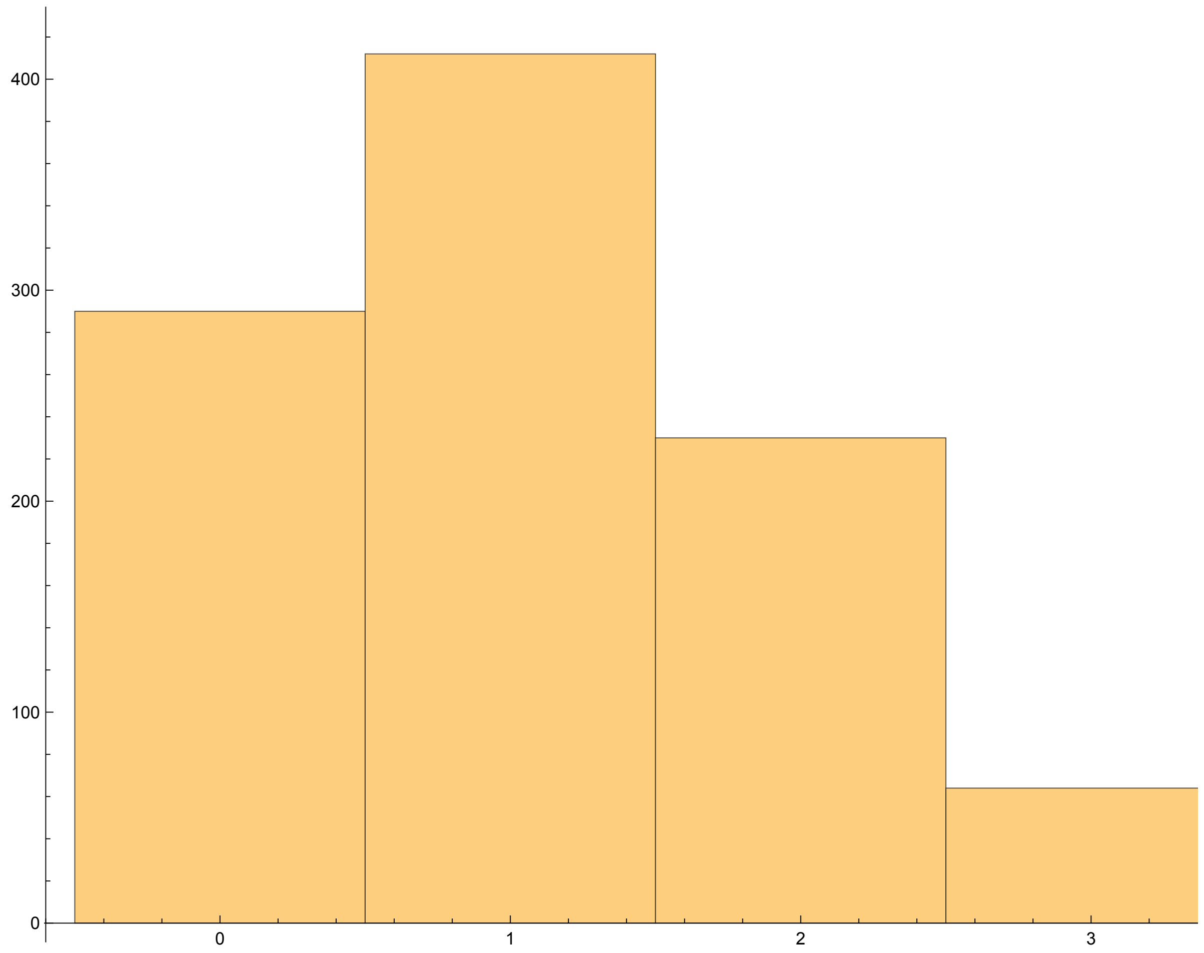
Out[32]=



There are only one way to have 0 jumps, 4 ways to have one jump, 6 ways to have two jumps, 4 ways to have three jumps and one way to have 4 jumps. I can bring the simulation on the Histogram too:

In[33]:=
```
numberOfJumpsSimExample = Map[CountJumps, sim1];
Histogram[numberOfJumpsSimExample, Automatic]
```

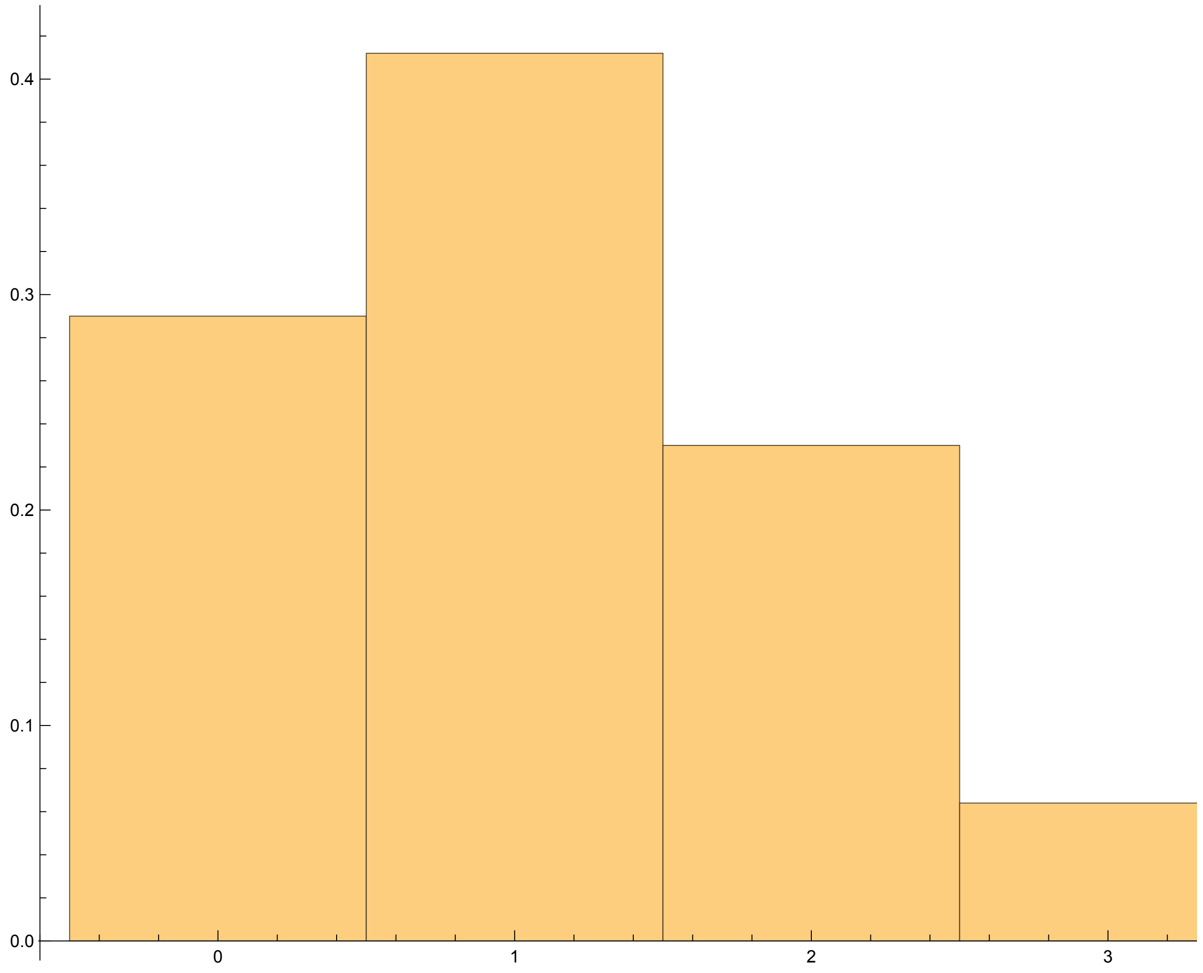
Out[34]=



Or in the sense of probabilities:

```
In[35]:= Histogram[numberOfJumpsSimExample, Automatic, "Probability"]
```

Out[35]=



Showing clearly that only because a sequence has shown up more often it doesn’t mean that it is the most probable outcome.
So it would make more sense to talk about the number of transitions when talking about Markov chains and not a specific sequence of events.

Counting

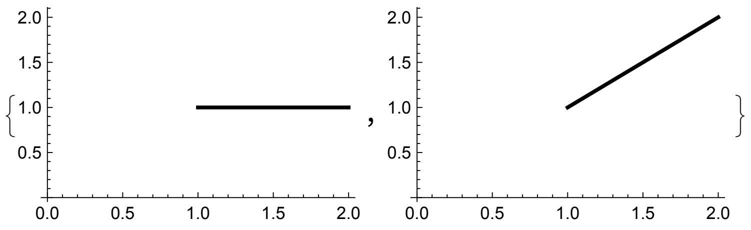
To calculate the probabilities, it is necessary to count the number of possible combinations and their possible relations. To get a sense let’s look as some of these combinations:

```
In[36]:= N1allseq = Tuples[{1, 2}, 2];
         N1allseq1 = Select[N1allseq, #[[1]] == 1 &];

In[38]:= N2allseq = Tuples[{1, 2}, 3];
         N2allseq1 = Select[N2allseq, #[[1]] == 1 &];

In[40]:= N3allseq = Tuples[{1, 2}, 4];
         N3allseq1 = Select[N3allseq, #[[1]] == 1 &];
```

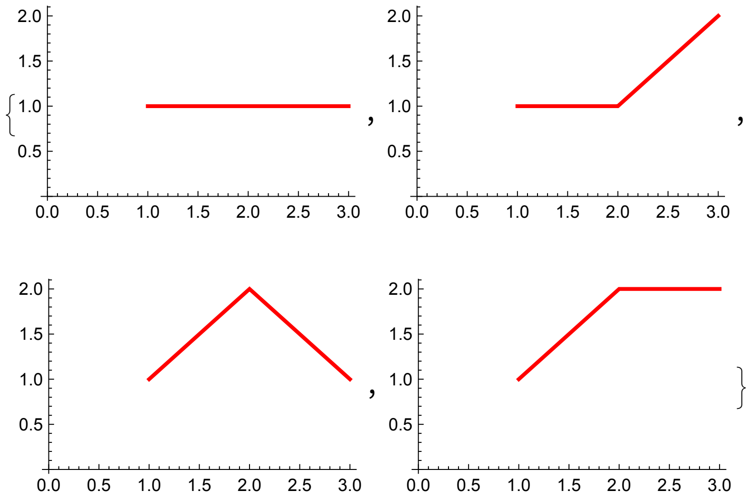
Plotting these possibilities:

```
In[42]:= plotsN1 = ListLinePlot[#, PlotStyle → Black] & /@ N1allseq1
         plotsN2 = ListLinePlot[#, PlotStyle → Red] & /@ N2allseq1
         plotsN3 = ListLinePlot[#, PlotStyle → Blue] & /@ N3allseq1
         plotsN4 = ListLinePlot[#,  PlotStyle → Brown] & /@ N4allseq1
```

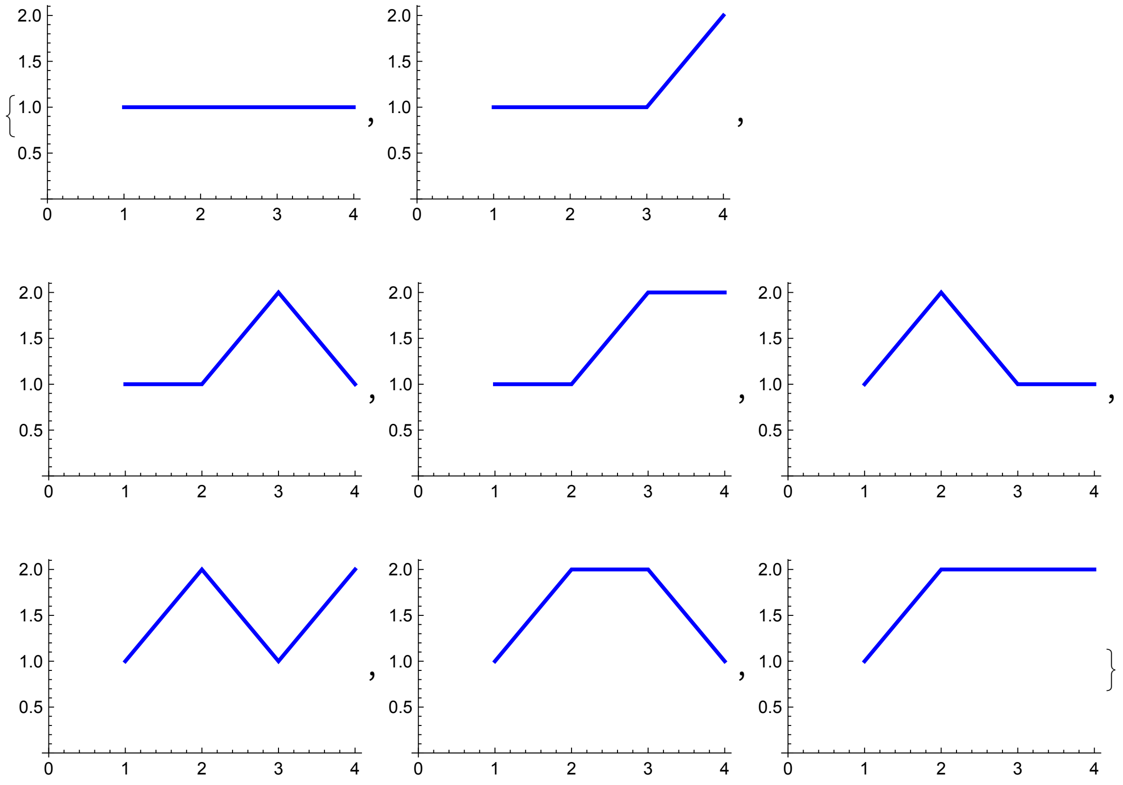
Out[42]=



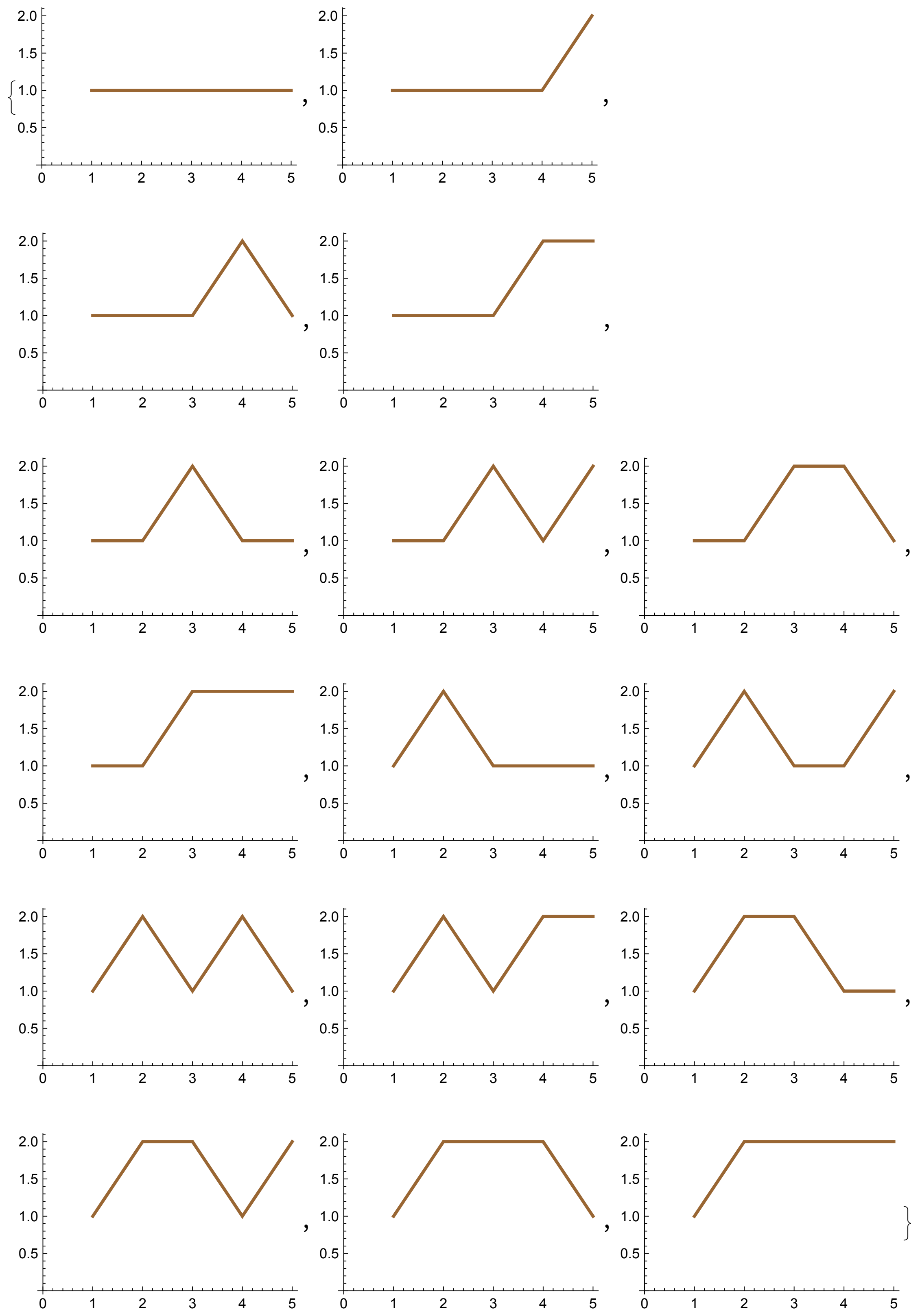
Out[43]=



Out[44]=



Out[45]=



We can categorize these combinations by the amount of jumps they make! If we do that we come across the famous Pascal triangle.

In[46]:=
```
PascalTriangle[n_] :=
  NestList[{1, Sequence @@ Plus @@@ Partition[#, 2, 1], 1} &, {1}, n - 1];
```

```
In[47]:= Column[Grid[{#}, ItemSize → 3] & /@
           (PascalTriangle[7] /. x_Integer ⧴ Text[Style[x, Large]]), Center]
Out[47]=
```

```
                  1
                1   1
              1   2   1
            1   3   3   1
          1   4   6   4   1
        1   5  10  10   5   1
      1   6  15  20  15   6   1
```

In this way we can count the possibilities of transitions. N=number of steps and k=number of transitions:

N=0, k=0 --> 1

N=1, k=0 --> 1

        k=1 --> 1

N=2, k=0 --> 1

        k=1--> 2

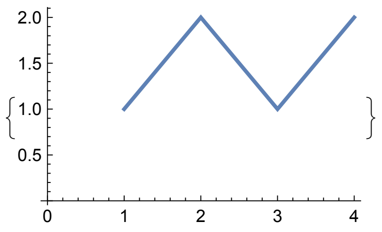
        K=2 -->1

        etc...

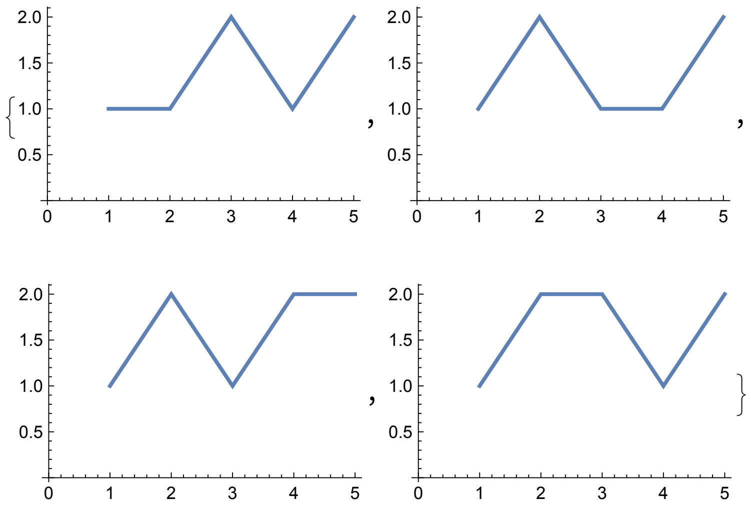
However, as I will demonstrate it is more valuable to categorize this possibilities not from the number of steps to the number of transition but vise versa.

Let’s start with k=3 and increase the number of steps from there. Remember: Min(N)=k

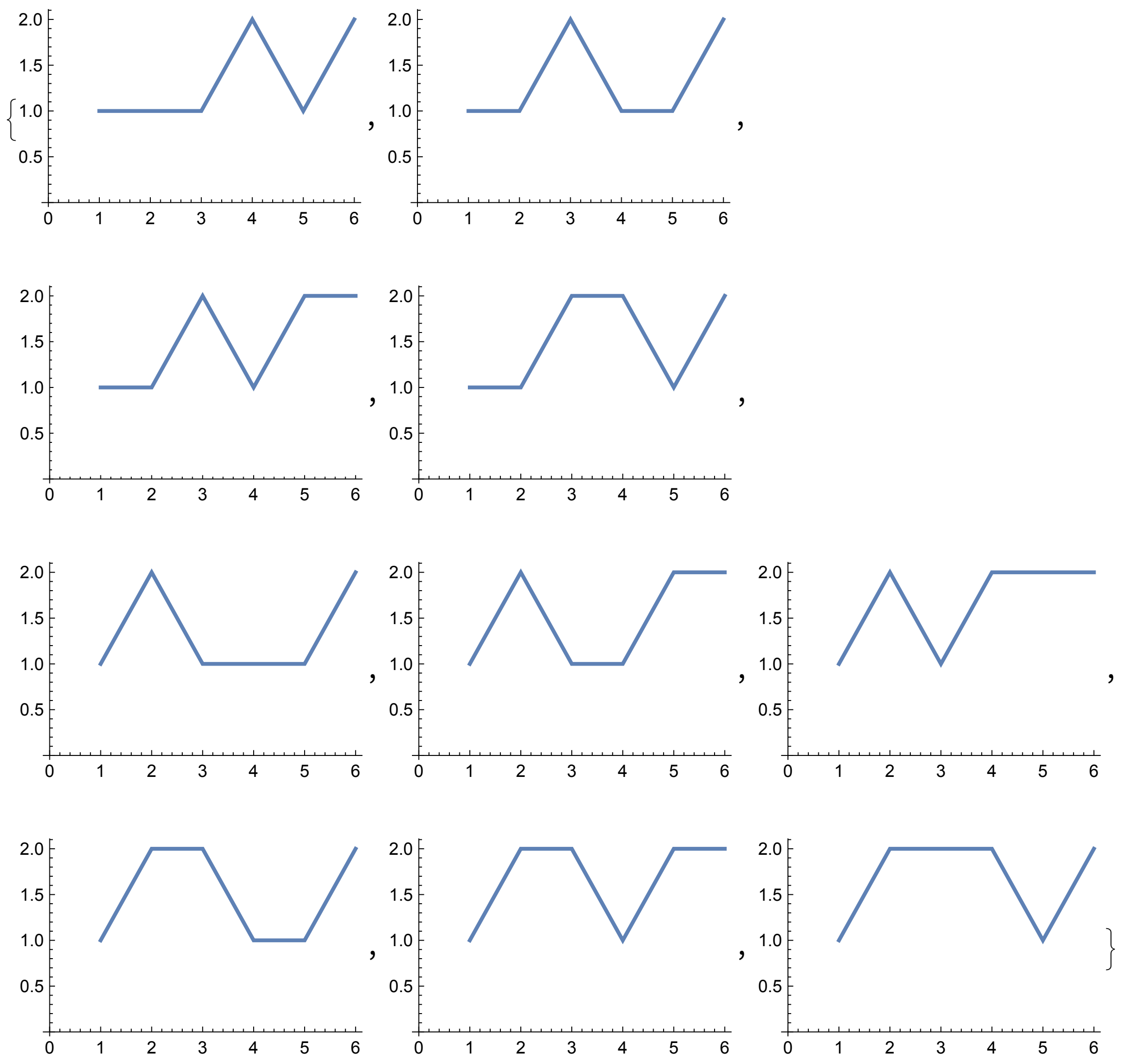
```
In[48]:= countk3N3allseq1 = Select[N3allseq1, CountJumps[#] == 3 &];
         plotk3N3allseq1 = ListLinePlot /@ countk3N3allseq1
Out[49]=
```



```
In[50]:= countk3N4allseq1 = Select[N4allseq1, CountJumps[#] == 3 &];
         plotk3N3allseq1 = ListLinePlot /@ countk3N4allseq1
Out[51]=
```



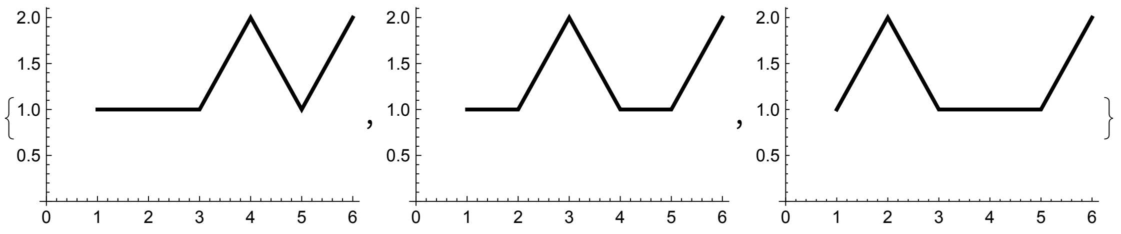
```
In[52]:= N5allseq = Tuples[{1, 2}, 6];
         N5allseq1 = Select[N5allseq, #⟦1⟧ == 1 &];
         countk3N5allseq1 = Select[N5allseq1, CountJumps[#] == 3 &];
         plotk3N5allseq1 = ListLinePlot /@ countk3N5allseq1
Out[55]=
```



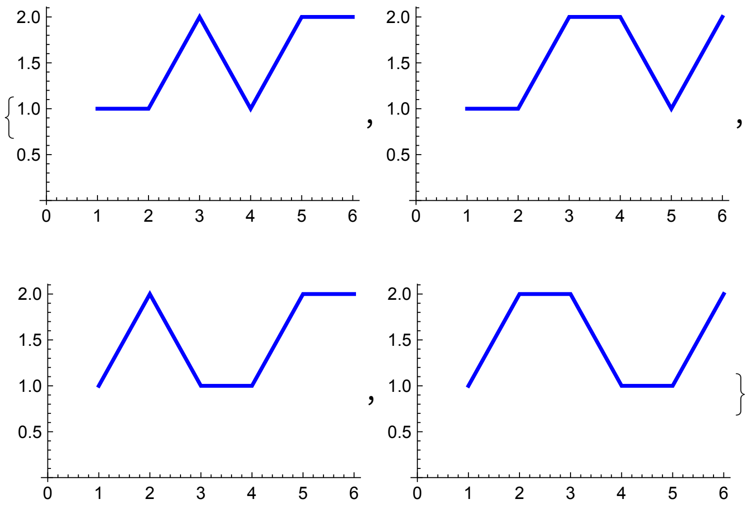
At this point, we can ask ourselves if any of these sequences will have the same probability to occur? if we look at the **plotk3N3allseq1** we can see that the sequence 1,1,2,1,2 has the same probability as 1,2,1,1,2. (think about the transition matrix. it will be c*a*(1-a)*(1-b)*(1-a) and c*(1-a)*a*(1-b)*(1-a) for these two sequences which are equal. The other two sequences also have the same probability. If we expand and look at **plotk3N5allseq1** we see we can there are sequences with equal probability to occur too. there we can have 3 groups. first group has 3 sequences, second group 4 and the last group again three sequences: see the following diagrams:

```
In[56]:= countk3N5allseq1a =
           Select[countk3N5allseq1, Count[#, 1] == 4 && Count[#, 2] == 2 &];
         countk3N5allseq1b =
           Select[countk3N5allseq1, Count[#, 1] == 3 && Count[#, 2] == 3 &];
         countk3N5allseq1c =
           Select[countk3N5allseq1, Count[#, 1] == 2 && Count[#, 2] == 4 &];
         plotsk3N5a = ListLinePlot[#, PlotStyle → Black] & /@ countk3N5allseq1a
         plotsk3N5b = ListLinePlot[#, PlotStyle → Blue] & /@ countk3N5allseq1b
         plotsk3N5c = ListLinePlot[#, PlotStyle → Red] & /@ countk3N5allseq1c
```

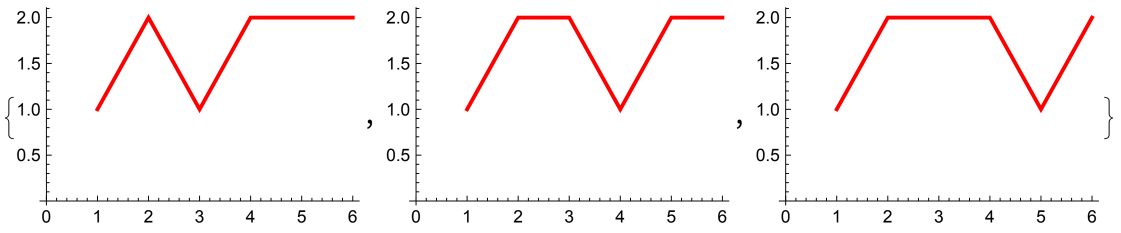
Out[59]=



Out[60]=



Out[61]=



To calculate the number of equivalent transitions the following calculator can be coded: Adjust the nJump, m and the total number nJump+n to calculate the sequences of equivalent transitions.

```
In[62]:= possibleCombination = Table[Tuples[{1, 2}, n], {n, 2, 21}];
         nMax = Length[possibleCombination];
         possibleCombination1 =
           Table[Select[possibleCombination⟦n⟧, #⟦1⟧ == 1 &], {n, nMax}];
```

```
In[65]:= nJump = 3;
         jumps =
           Table[Select[possibleCombination1⟦n⟧, CountJumps[#] == nJump &], {n, nMax}];

In[67]:= pascalTable = Table[
            Length /@ Table[Select[jumps⟦m⟧, Count[#, 1] == n && Count[#, 2] == m + 1 - n &],
               {n, 2, nJump + 6}], {m, 3, 10}];

In[68]:= Column[Grid[{#}, ItemSize → 3] & /@
           (pascalTable /. {x_Integer /; x ≠ 0 :→ Text[Style[x, Large]], 0 → ""}), Center]
Out[68]=
 1
 2   2
 3   4   3
 4   6   6   4
 5   8   9   8   5
 6  10  12  12  10   6
 7  12  15  16  15  12   7
 8  14  18  20  20  18  14   8

In[69]:= nJump = 4;
         jumps =
           Table[Select[possibleCombination1⟦n⟧, CountJumps[#] == nJump &], {n, nMax}];
         pascalTable = Table[
            Length /@ Table[Select[jumps⟦m⟧, Count[#, 1] == n && Count[#, 2] == m + 1 - n &],
               {n, 2, nJump + 6}], {m, 3, 10}];
         Column[Grid[{#}, ItemSize → 3] & /@
           (pascalTable /. {x_Integer /; x ≠ 0 :→ Text[Style[x, Large]], 0 → ""}), Center]
Out[72]=

     1
     2   3
     3   6   6
     4   9  12  10
     5  12  18  20  15
     6  15  24  30  30  21
     7  18  30  40  45  42  28
```

```
In[73]:= nJump = 5;
         jumps =
           Table[Select[possibleCombination1⟦n⟧, CountJumps[#] == nJump &], {n, nMax}];
         pascalTable = Table[
            Length /@ Table[Select[jumps⟦m⟧, Count[#, 1] == n && Count[#, 2] == m + 1 - n &],
              {n, 2, nJump + 6}], {m, 3, 10}];
         Column[Grid[{#}, ItemSize → 3] & /@
           (pascalTable /. {x_Integer /; x ≠ 0 :→ Text[Style[x, Large]], 0 → ""}), Center]
Out[76]=
```

```
1
3    3
6    9    6
10   18   18   10
15   30   36   30   15
21   45   60   60   45   21
```

```
In[77]:= nJump = 6;
         jumps =
           Table[Select[possibleCombination1⟦n⟧, CountJumps[#] == nJump &], {n, nMax}];
         pascalTable = Table[
            Length /@ Table[Select[jumps⟦m⟧, Count[#, 1] == n && Count[#, 2] == m + 1 - n &],
              {n, 2, nJump + 6}], {m, 3, 10}];
         Column[Grid[{#}, ItemSize → 3] & /@
           (pascalTable /. {x_Integer /; x ≠ 0 :→ Text[Style[x, Large]], 0 → ""}), Center]
Out[80]=
```

```
1
3    4
6    12   10
10   24   30   20
15   40   60   60   35
```

And so on. More if these tables can be found in ***my Notebook from 05.07.23***

The question I ask myself here is if there is any pattern for to find these numbers? The first thing I realized was that these triangles are not the best way to display these numbers and I change the way of writing these numbers. (Of course one could argue that we can stay with these tables and don’t change anything but for me it was easier to find the pattern after I changed the shape of these numbers!)

I push every column upwards so that:
1
2 2
3 4 3
4 6 6 4
would look like:
1 2 3 4
2 4 6
3 6
4
Now the factors are not on every row but rather on the anti-diagonal. If we do that for every one of these triangle we see a pattern. If we assume that these triangles are elements of a matrix the pattern is as following:
the element at “i x j” is the result of multiplication of the elements “i x 1” and “1 x j”
Now the question remains how do I calculate the first columns and rows.
First we see a cleat difference between the k=2*q and k=2*q+1 (even and odd) numbers of jumps.
If the number of jumps are odd the the elements on **1 x j and j x 1** are exactly the same. This is not the case for even numbers. At first the even numbers seem very random but they are not.
Let’s look at the first row of k=3:

```
In[81]:= Table[n, {n, 1, 10}]
Out[81]=
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

is just the natural numbers. This is the first column of k=4. The first row of k=4 is then just the series of the sum of natural numbers. It is very well-known that such series can be calculated:

```
In[82]:= Table[n * (n + 1) / 2, {n, 1, 10}]
Out[82]=
{1, 3, 6, 10, 15, 21, 28, 36, 45, 55}
```

which is then the first column of k=5. Since 5 is an odd number the first row is the same as the first column.
The first row if k=5 is then just the first column of k=6. The first row then is the series of the sums of the first column. It guessed the way to calculate the first column. it was:

```
In[83]:= Table[n * (n + 1) * (n + 2) / 6, {n, 1, 10}]
Out[83]=
{1, 4, 10, 20, 35, 56, 84, 120, 165, 220}
```

We can see the pattern already. I again guessed the first row of k=8:

```
In[84]:= Table[n * (n + 1) * (n + 2) * (n + 3) / 24, {n, 1, 10}]
Out[84]=
{1, 5, 15, 35, 70, 126, 210, 330, 495, 715}
```

All of these work and can be verified. However, the mathematical logic behind is still unknown to me. ***The patterns are written in my notebook from 06.07.23***

Probability combinations

Now the second part of the problem is to find all the probability combinations. Just a reminder that I still am discussing the combinations where the first state is the state 1.
k=3 means at least 3 steps. Starting from state 1 it means that we only have one way to distribute the steps UP, DOWN, UP. It doesn't really matter what is the number of points we will always have 2 UPs and only 1 DOWN. Remember the transition matrix:

```
In[85]:= Clear[a, b, c]
```

```
In[86]:= 𝒯 = {{a, 1 - a}, {1 - b, b}};
         𝒯 // MatrixForm
```

Out[87]//MatrixForm=

$$\begin{pmatrix} a & 1-a \\ 1-b & b \end{pmatrix}$$

2 UPs and 1 DOWN means: (1-a)(1-a)(1-b)
Any other combination will only be multiplied to. Let's say we have N=4. Now the only possibilities that are combined here are if one step will be non switching of the first or the second state:
(1-a)(1-a)(1-b)a
(1-a)(1-a)(1-b)b
The factors are calculated already in the last subsection: 3(1-a)(1-a)(1-b)a +3(1-a)(1-a)(1-b)b.
Not to forget that the initial probability has to be taken into account too: 3c [(1-a)(1-a)(1-b)a +(1-a)(1-a)(1-b)b]
If N=5 we will have:
(1-a)(1-a)(1-b)a^2
(1-a)(1-a)(1-b)ab
(1-a)(1-a)(1-b)b^2

We can find the following pattern.

```
In[88]:= Column[Table[Sum[a^(n - k) * b^k, {k, 0, n}], {n, 1, 6}]]
```

Out[88]=

$a + b$
$a^2 + a b + b^2$
$a^3 + a^2 b + a b^2 + b^3$
$a^4 + a^3 b + a^2 b^2 + a b^3 + b^4$
$a^5 + a^4 b + a^3 b^2 + a^2 b^3 + a b^4 + b^5$
$a^6 + a^5 b + a^4 b^2 + a^3 b^3 + a^2 b^4 + a b^5 + b^6$

And that is pretty much it....
We just have to multiply the correct factors with the correct probability combination.

Verification

```
In[89]:= ClearAll;
```

before moving any further let's see how good the current relation works:

an example N=9 meaning that the highest number of jumps k=9

```
In[90]:= combinationsN9 = Table[Sum[b^k*a^(n - k), {k, 0, n}], {n, 1, 8}];

In[91]:= combinationsN9 // Column
Out[91]=
```

$a + b$
$a^2 + a b + b^2$
$a^3 + a^2 b + a b^2 + b^3$
$a^4 + a^3 b + a^2 b^2 + a b^3 + b^4$
$a^5 + a^4 b + a^3 b^2 + a^2 b^3 + a b^4 + b^5$
$a^6 + a^5 b + a^4 b^2 + a^3 b^3 + a^2 b^4 + a b^5 + b^6$
$a^7 + a^6 b + a^5 b^2 + a^4 b^3 + a^3 b^4 + a^2 b^5 + a b^6 + b^7$
$a^8 + a^7 b + a^6 b^2 + a^5 b^3 + a^4 b^4 + a^3 b^5 + a^2 b^6 + a b^7 + b^8$

```
In[92]:= k2Factors = Table[n, {n, 8, 1, -1}]
Out[92]=
{8, 7, 6, 5, 4, 3, 2, 1}
```

starting with k=3
The factors are at:
1 x 7
2 x 6
3 x 5
4 x 4
5 x 3
6 x 2
7 x 1

```
In[93]:= k3Factors = Flatten[Table[{(Product[1 + i + m, {m, 0, 0}] / Factorial[1] *
                 Product[7 - i + m, {m, 0, 0}] / Factorial[1])}, {i, 0, 6}]];
```

For k=4
1 x 6
2 x 5
3 x 4
4 x 3
5 x 2
6 x 1

```
In[94]:= k4Factors = Flatten[Table[{(Product[1 + i + m, {m, 0, 0}] / Factorial[1] *
                 Product[6 - i + m, {m, 0, 1}] / Factorial[2])}, {i, 0, 5}]];
```

For k=5
1 x 5
2 x 4
3 x 3
4 x 2
5 x 1

```
In[95]:= k5Factors = Flatten[Table[{(Product[1 + i + m, {m, 0, 1}] / Factorial[2] *
                 Product[5 - i + m, {m, 0, 1}] / Factorial[2])}, {i, 0, 4}]];
```

k=6

1 x 4

2 x 3

3 x 2

4 x 1

```
In[96]:= k6Factors = Flatten[Table[{(Product[1 + i + m, {m, 0, 1}] / Factorial[2] *
                 Product[4 - i + m, {m, 0, 2}] / Factorial[3])}, {i, 0, 3}]];
```

For k=7

1 x 3

2 x 2

3 x 1

```
In[97]:= k7Factors = Flatten[Table[{(Product[1 + i + m, {m, 0, 2}] / Factorial[3] *
                 Product[3 - i + m, {m, 0, 2}] / Factorial[3])}, {i, 0, 2}]];
```

and k=8

1 x 2

2 x 1

```
In[98]:= k8Factors = Flatten[Table[{(Product[1 + i + m, {m, 0, 2}] / Factorial[3] *
                 Product[2 - i + m, {m, 0, 3}] / Factorial[4])}, {i, 0, 1}]];
```

```
In[99]:= k3Factors
         k4Factors
         k5Factors
         k6Factors
         k7Factors
         k8Factors
Out[99]=
         {7, 12, 15, 16, 15, 12, 7}
Out[100]=
         {21, 30, 30, 24, 15, 6}
Out[101]=
         {15, 30, 36, 30, 15}
Out[102]=
         {20, 30, 24, 10}
Out[103]=
         {10, 16, 10}
Out[104]=
         {5, 4}
```

Now it’s time to combine these factors with the probability combinations.

In[105]:=
```
combinationsN9List = List @@@ combinationsN9
```
Out[105]=

$\{\{a, b\}, \{a^2, a\,b, b^2\}, \{a^3, a^2\,b, a\,b^2, b^3\}, \{a^4, a^3\,b, a^2\,b^2, a\,b^3, b^4\}, \{a^5, a^4\,b, a^3\,b^2, a^2\,b^3, a\,b^4, b^5\}, \{a^6, a^5\,b, a^4\,b^2, a^3\,b^3, a^2\,b^4, a\,b^5, b^6\}, \{a^7, a^6\,b, a^5\,b^2, a^4\,b^3, a^3\,b^4, a^2\,b^5, a\,b^6, b^7\}, \{a^8, a^7\,b, a^6\,b^2, a^5\,b^3, a^4\,b^4, a^3\,b^5, a^2\,b^6, a\,b^7, b^8\}\}$

In[106]:=
```
K9 = c (1 - a) ^ 5 (1 - b) ^ 4
K8 = c (1 - a) ^ 4 (1 - b) ^ 4 (k8Factors⟦1⟧ * combinationsN9List⟦1⟧⟦1⟧ +
     k8Factors⟦2⟧ * combinationsN9List⟦1⟧⟦2⟧)
K7 = c (1 - a) ^ 4 (1 - b) ^ 3 (k7Factors⟦1⟧ * combinationsN9List⟦2⟧⟦1⟧ + k7Factors⟦2⟧ *
      combinationsN9List⟦2⟧⟦2⟧ + k7Factors⟦3⟧ * combinationsN9List⟦2⟧⟦3⟧)
K6 = c (1 - a) ^ 3 (1 - b) ^ 3 (Sum[k6Factors⟦i⟧ * combinationsN9List⟦3⟧⟦i⟧, {i, 1, 4}])
K5 = c (1 - a) ^ 3 (1 - b) ^ 2 (Sum[k5Factors⟦i⟧ * combinationsN9List⟦4⟧⟦i⟧, {i, 1, 5}])
K4 = c (1 - a) ^ 2 (1 - b) ^ 2 (Sum[k4Factors⟦i⟧ * combinationsN9List⟦5⟧⟦i⟧, {i, 1, 6}])
K3 = c (1 - a) ^ 2 (1 - b) (Sum[k3Factors⟦i⟧ * combinationsN9List⟦6⟧⟦i⟧, {i, 1, 7}])
K2 = c (1 - a) (1 - b) (Sum[k2Factors⟦i⟧ * combinationsN9List⟦7⟧⟦i⟧, {i, 1, 8}])
K1 = c (1 - a) (Sum[combinationsN9List⟦8⟧⟦i⟧, {i, 1, 9}])
K0 = c a ^ 9
```
Out[106]=

$(1-a)^5\,(1-b)^4\,c$

Out[107]=

$(1-a)^4\,(1-b)^4\,(5\,a+4\,b)\,c$

Out[108]=

$(1-a)^4\,(1-b)^3\,\left(10\,a^2+16\,a\,b+10\,b^2\right)\,c$

Out[109]=

$(1-a)^3\,(1-b)^3\,\left(20\,a^3+30\,a^2\,b+24\,a\,b^2+10\,b^3\right)\,c$

Out[110]=

$(1-a)^3\,(1-b)^2\,\left(15\,a^4+30\,a^3\,b+36\,a^2\,b^2+30\,a\,b^3+15\,b^4\right)\,c$

Out[111]=

$(1-a)^2\,(1-b)^2\,\left(21\,a^5+30\,a^4\,b+30\,a^3\,b^2+24\,a^2\,b^3+15\,a\,b^4+6\,b^5\right)\,c$

Out[112]=

$(1-a)^2\,(1-b)\,\left(7\,a^6+12\,a^5\,b+15\,a^4\,b^2+16\,a^3\,b^3+15\,a^2\,b^4+12\,a\,b^5+7\,b^6\right)\,c$

Out[113]=

$(1-a)\,(1-b)\,\left(8\,a^7+7\,a^6\,b+6\,a^5\,b^2+5\,a^4\,b^3+4\,a^3\,b^4+3\,a^2\,b^5+2\,a\,b^6+b^7\right)\,c$

Out[114]=

$(1-a)\,\left(a^8+a^7\,b+a^6\,b^2+a^5\,b^3+a^4\,b^4+a^3\,b^5+a^2\,b^6+a\,b^7+b^8\right)\,c$

Out[115]=

$a^9\,c$

These are the needed factors to calculate the probability histogram for a Markov process for c=1

In[116]:=
```
a = 0.6;
b = 0.8;
c = 1;
```

In[119]:=

```
K = {K0, K1, K2, K3, K4, K5, K6, K7, K8, K9};
Total[K]
```

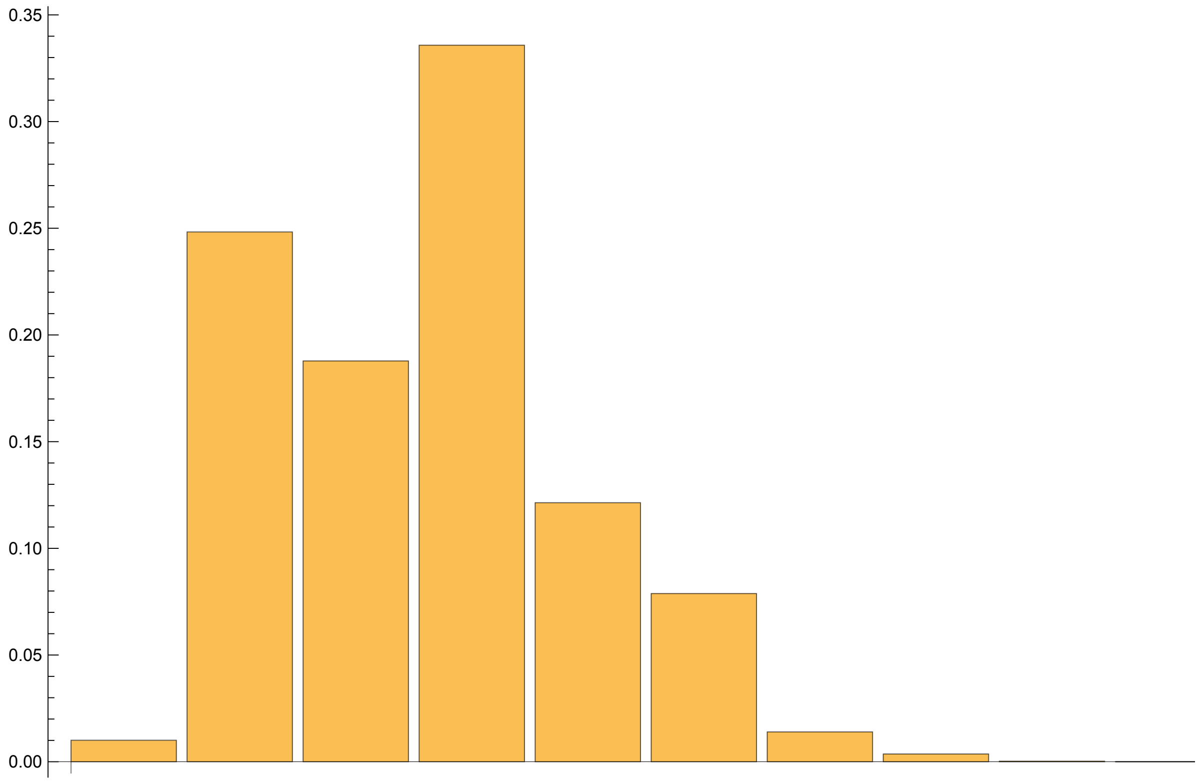
Out[120]=

```
1.
```

In[121]:=

```
BarChart[K, ChartLabels → Automatic]
```

Out[121]=

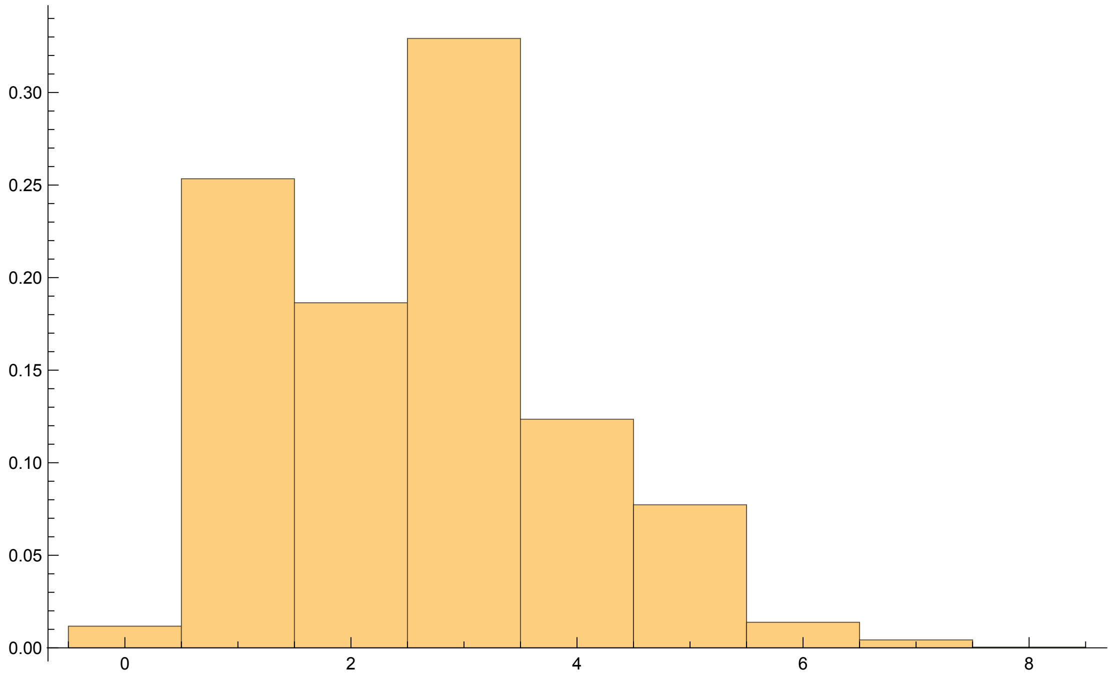


Let’s see how this predicted the result!

In[127]:=

```
process = DiscreteMarkovProcess[{1, 0}, 𝒯];
sim = Table[RandomFunction[process, {0, 9}]["States"], {10 000}];
sim = Flatten[sim, 1];
numberOfJumpsSimVerify = Map[CountJumps, sim];
Histogram[numberOfJumpsSimVerify, Automatic, "Probability"]
```

Out[131]=



Meaning that the approach works perfectly! The next step is to automate the calculation and to write down a mathematical expression where the factors are combined with the probability combinations

Now let’s look at N=99 (meaning 100 data points) to verify this process again!

In[132]:=

```
Clear[a, b, c]
```

In[133]:=

```
combinationsN99 = Table[Sum[b^k * a^(n - k), {k, 0, n}], {n, 1, 98}];
```

In[134]:=

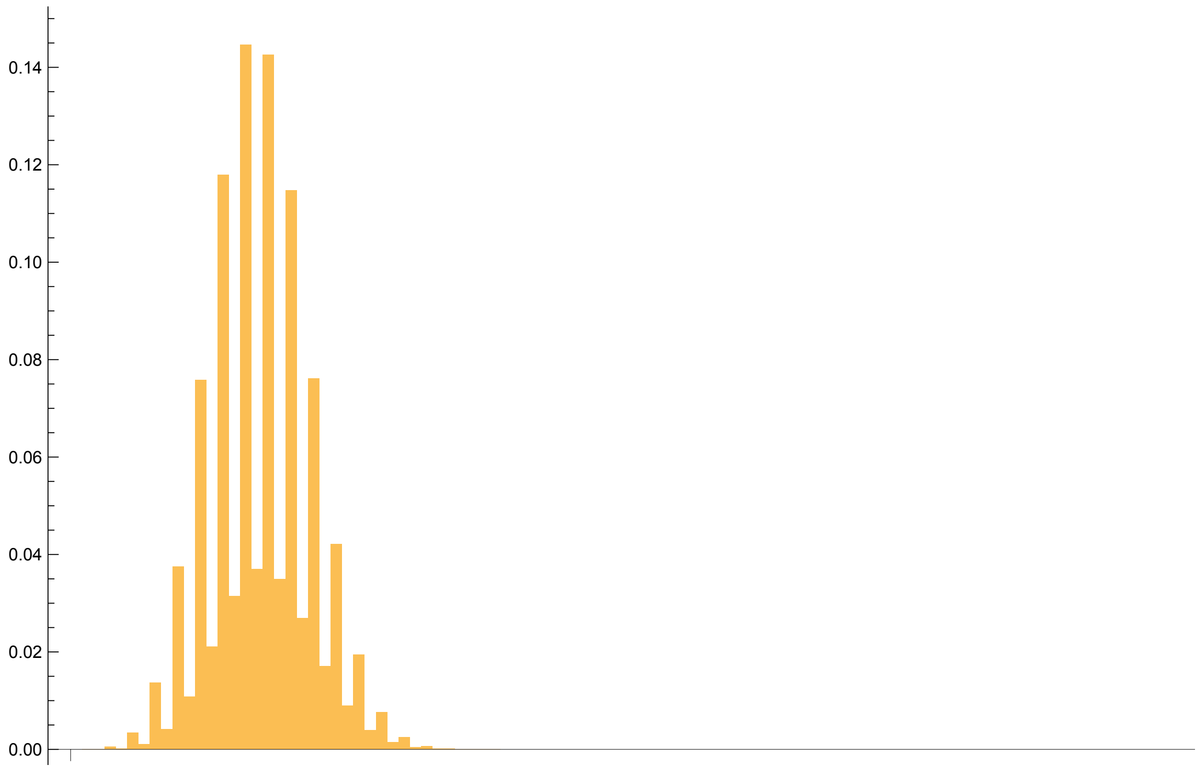
```
k1Factors99 = {Table[1, {n, 1, 99}]};
k2Factors99 = {Table[n, {n, 98, 1, -1}]};
kFactorsOdd =
  Flatten[Table[{Table[(Product[1 + i + m, {m, 0, j}] / Factorial[j + 1]) *
        (Product[(97 - 2 * j) - i + m, {m, 0, j}] / Factorial[j + 1]),
      {i, 0, 96 - 2 * j}]}, {j, 0, 48}], 1];
kFactorsEven =
  Flatten[Table[{Table[(Product[1 + i + m, {m, 0, j}] / Factorial[j + 1]) *
        (Product[(96 - 2 * j) - i + m, {m, 0, j + 1}] / Factorial[j + 2]),
      {i, 0, 95 - 2 * j}]}, {j, 0, 47}], 1];
```

```
In[138]:=
    combinedFactors = Join[kFactorsOdd, kFactorsEven, k2Factors99, k1Factors99];
    Factors99 = Sort[combinedFactors, Length[#1] > Length[#2] &];

In[140]:=
    combinationsN99 = Table[Sum[b ^ k * a ^ (n - k), {k, 0, n}], {n, 1, 98}];
    combinationsN99List = List @@@ combinationsN99;

In[142]:=
    K99Ex0a99 = Sort[
       Table[Sum[Factors99[[j]][[i]] * combinationsN99List[[99 - j]][[i]], {i, 1, 99 - j + 1}],
        {j, 98, 1, -1}], Length[#1] > Length[#2] &];
    multipliers99Ex0a99 = Table[
       (1 - a) ^ (Ceiling[i / 2]) * (1 - b) ^ (Floor[i / 2]), {i, 1, Length[K99Ex0a99]}];
    K99Ex0a99 = multipliers99Ex0a99 * K99Ex0a99;
    K99Ex99 = Prepend[K99Ex0a99, c * a ^ 99];
    K99 = Append[K99Ex99, c * (1 - a) ^ 50 (1 - b) ^ 49];

In[147]:=
    Length[K99Ex0a99]

Out[147]=
    98

In[148]:=
    a = 0.6;
    b = 0.9;
    c = 1;

In[151]:=
    Total[K99]

Out[151]=
    1.
```

In[152]:=

```
BarChart[K99, ChartLabels → Automatic]
```

Out[152]=



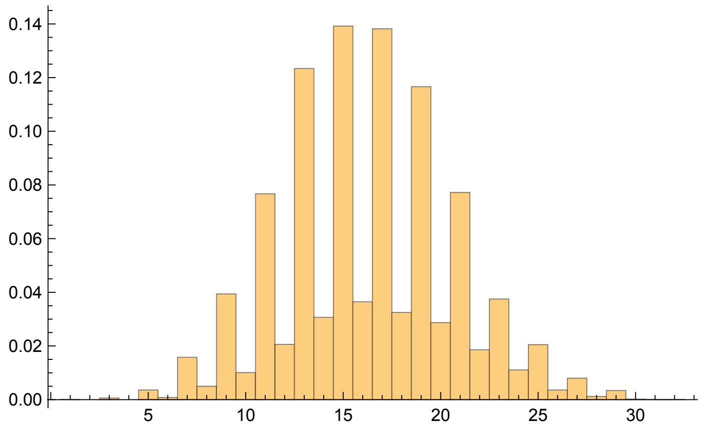
In[153]:=

```
process = DiscreteMarkovProcess[{1, 0}, 𝒯]
sim  = Table[RandomFunction[process, {0, 99}]["States"], {10 000}];
sim = Flatten[sim, 1];
numberOfJumpsSimVerify = Map[CountJumps, sim];
Histogram[numberOfJumpsSimVerify, {1}, "Probability"]
```

Out[153]=

```
DiscreteMarkovProcess[{1, 0}, {{0.6, 0.4}, {0.1, 0.9}}]
```

Out[157]=



However, we don't need to calculate the sum of these probabilities. We can calculate every single entry. And put it on a histogram with probability bins on the x-axis.

In[158]:=

```
Clear[a, b, c, Factors99];
```

```
In[159]:=
        Factors99 = Sort[combinedFactors, Length[#1] > Length[#2] &];
        kNew = Sort[combinationsN99List, Length[#1] > Length[#2] &];
        kNew = multipliers99Ex0a99 * kNew;
        kNew = Prepend[kNew, {c * a ^ 99}];
        kNew = Append[kNew, {c * (1 - a) ^ 50 (1 - b) ^ 49}];
        Factors99 = Prepend[Factors99, {1}];
        Factors99 = Flatten[Factors99];
        (*combineNewEx99a0=multipliers99Ex0a99*kNew;
        combineNewEx0=Prepend[combineNewEx99a0,{c*a^99}];
        combineNew=Append[combineNewEx0,{c*(1-a)^50(1-b)^49}];
        combineNew=Flatten[combineNew]*)

In[166]:=
        a = 0.6;
        b = 0.7;
        c = 1;

In[169]:=
        weightedData = WeightedData[Flatten[kNew], Factors99];
        Max[Flatten[kNew]]
Out[170]=
        2.64039 × 10^-16

In[171]:=
        Histogram[weightedData, {0.000000000000000000000001}]
Out[171]=
        $Aborted
```