Chance, Choice, and Change
or
Can Random Events Follow Laws?

ISC1057

Janet Peterson and John Burkardt

Computational Thinking

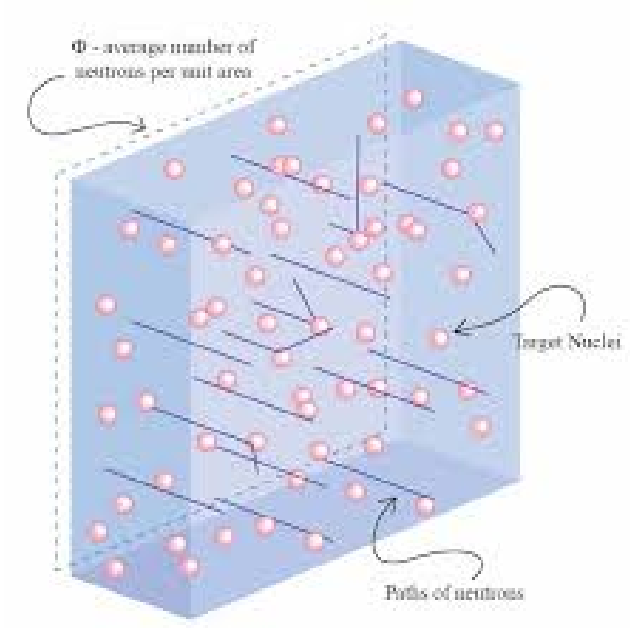
Fall Semester 2016

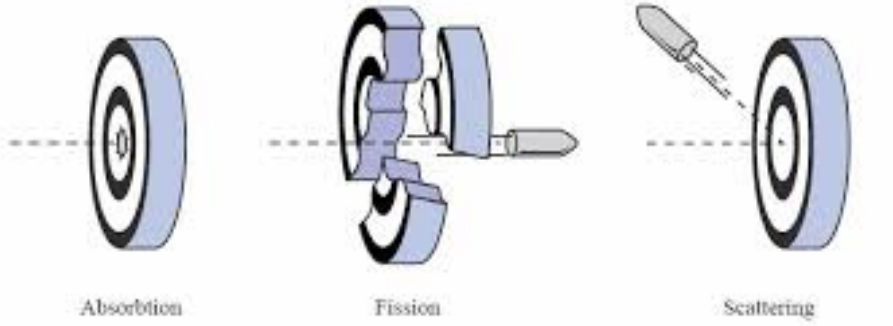
In this section of the course, we will look at things that seem to have an element of chance or randomness.

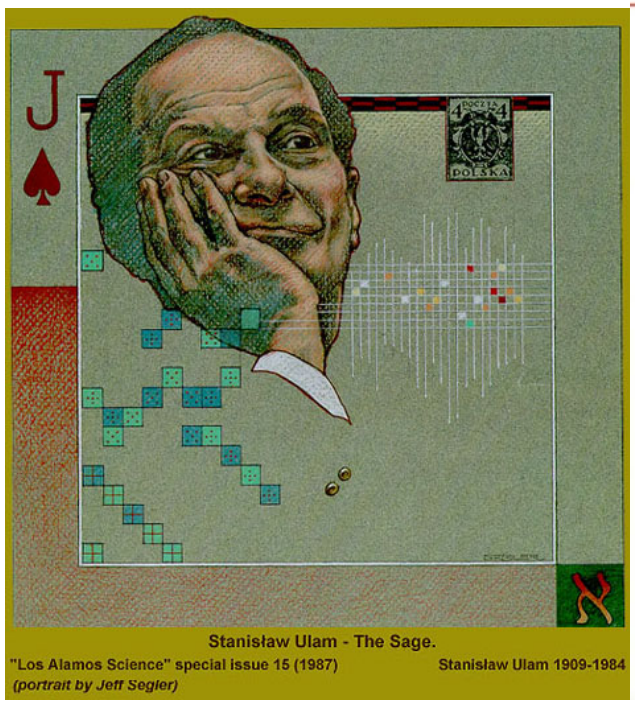
We will try to understand how randomness can be created, understood, and measured.

We will look at common instances of randomness, especially games of chance.

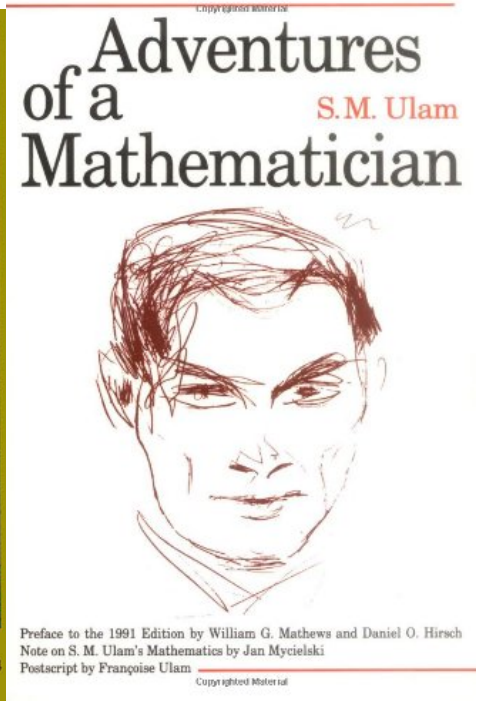
We will talk about measuring the kinds of randomness, and how it is possible for a computer to simulate random behaviors.







Stanisław Ulam - The Sage.
"Los Alamos Science" special issue 15 (1987) Stanisław Ulam 1909-1984
(portrait by Jeff Segler)

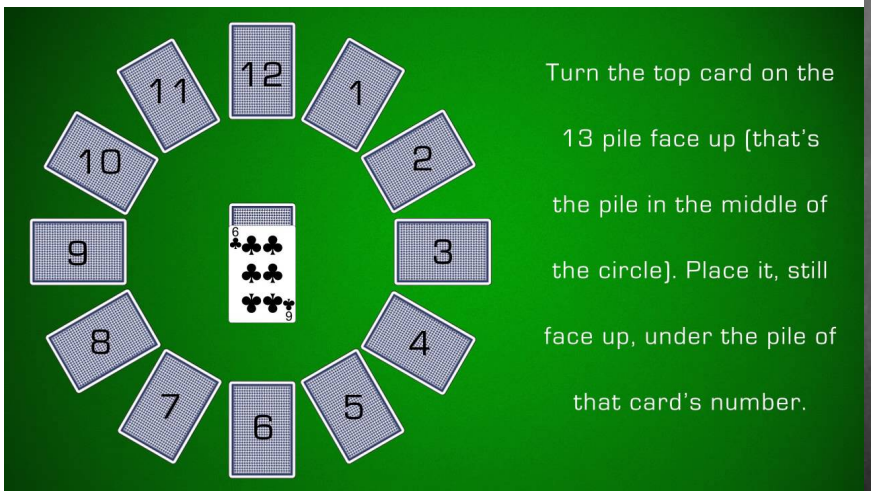


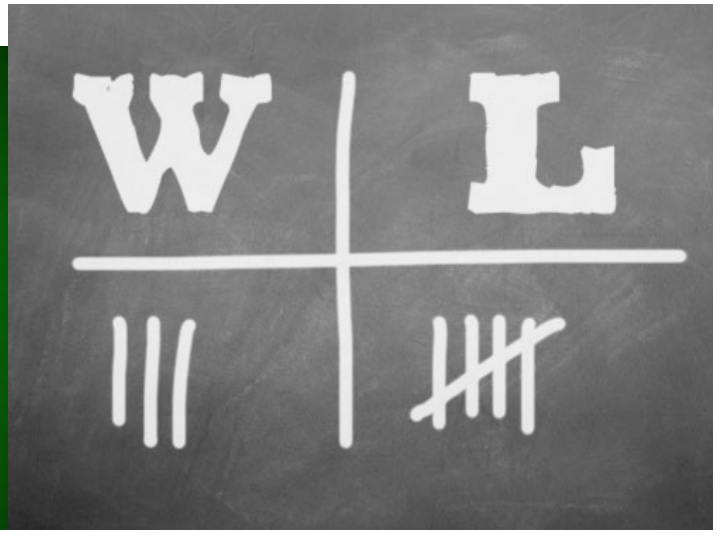
In 1946, Stanlislaw Ulam was a physicist working at Los Alamos, trying to predict the behavior of a beam of neutrons that penetrate a material.

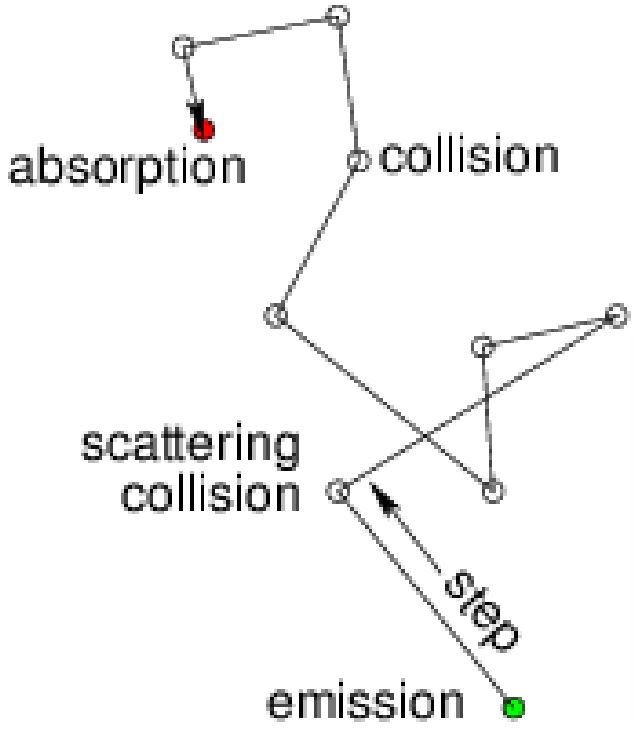
If a neutron collides with an atom in the material, it can be absorbed, be deflected at a new angle and speed, or cause the atom to split.

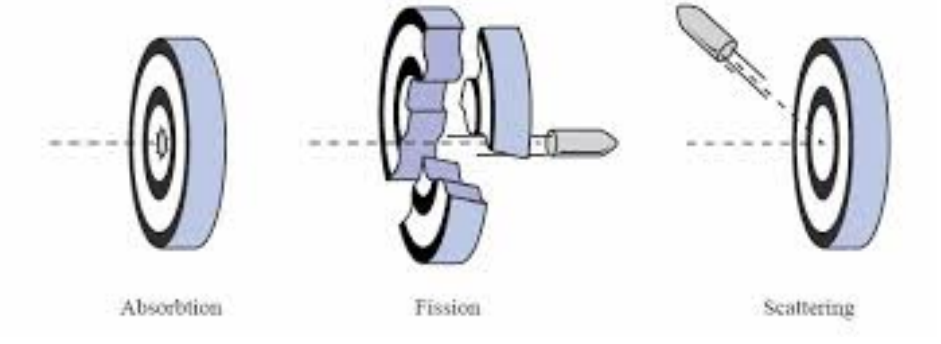
For one neutron and one collision, Ulam could estimate the probability of each event. The problem was that a single neutron might cause many collisions, with the outcome different each time.

Getting a useful answer meant estimating the behavior of many neutrons and many collisions, which seemed impossible to calculate.


12
1
2
3
4
5
6
7
8
9
10
11
Turn the top card on the 13 pile face up (that's the pile in the middle of the circle). Place it, still face up, under the pile of that card's number.


W
L


absorption
collision
scattering collision
step
emission


Absorbtion
Fission
Scattering

After playing the card game of clock solitaire many times, Ulam noticed that he didn't win very often. Rather than trying to figure out the probability of winning, he counted up his wins and losses until he saw a pattern that he could explain.

A few days later, he realized that rather than trying to solve the neutron problem exactly, he could get good results if he could simply "play the neutron game" and keep statistics.

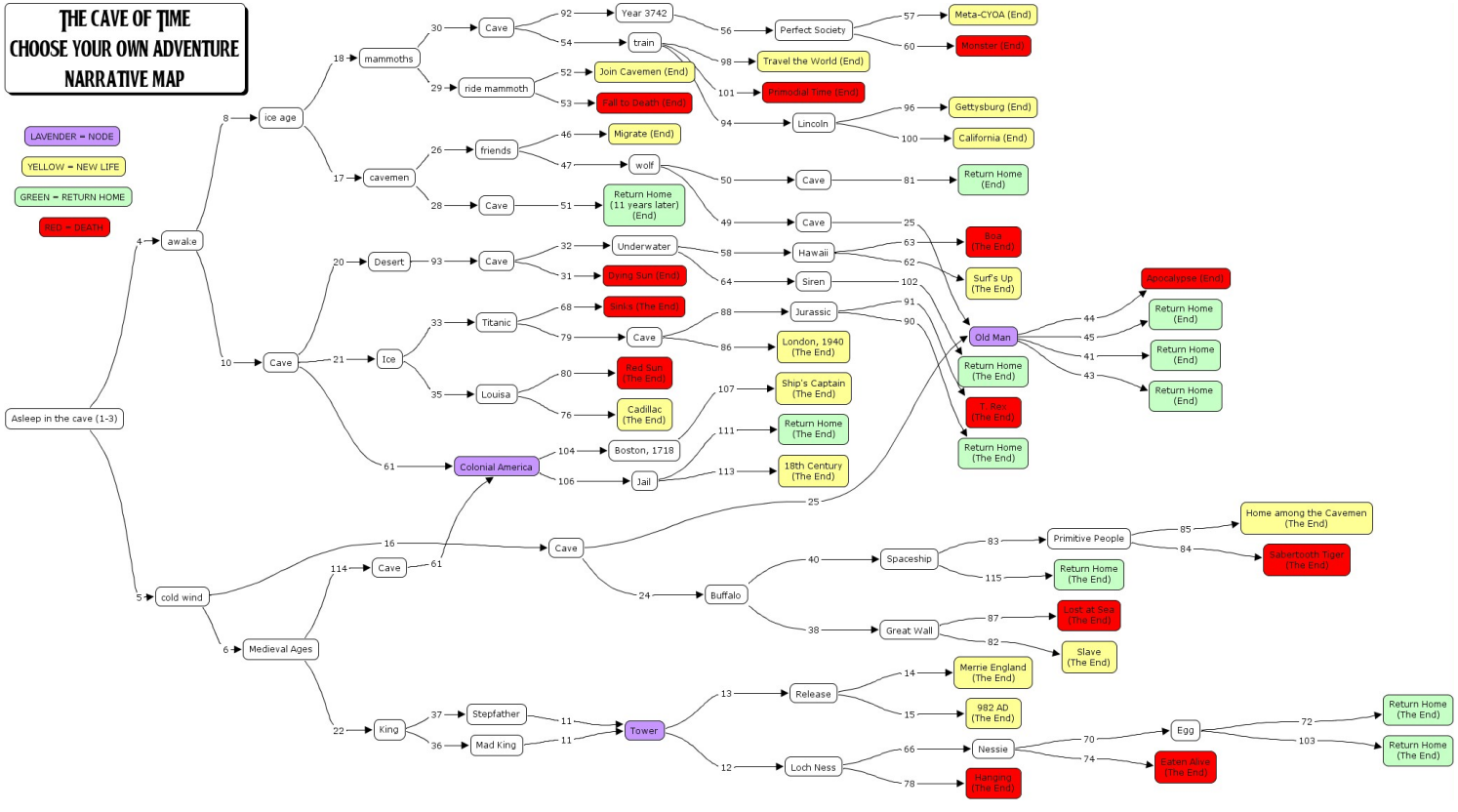
For each play of the game, he started a neutron heading into the material, and ...essentially... rolled dice to determine how far it would travel before colliding, and what would happen at the collision.

After a few games played on paper, it was clear the method was useful, and the newly invented computer was used to play and record the results of thousands and millions of such games.

Physicists were surprised that sometimes they could solve a problem by flipping coins.

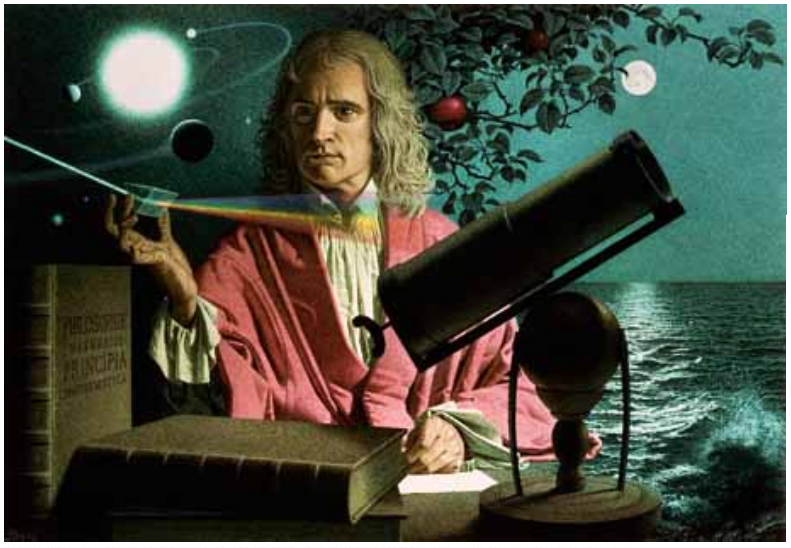

Something has happened!
A strong gust of wind blows across the board, carrying everyone with it! All players have landed on a random square. Most inconvenient.
OK

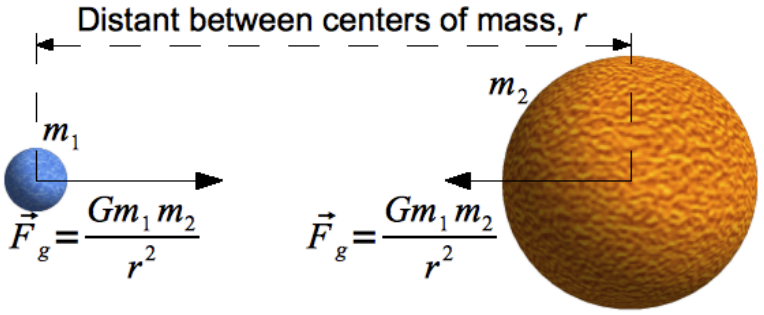

15+ BIG CHANCE
CREDIT
WIN
22
00

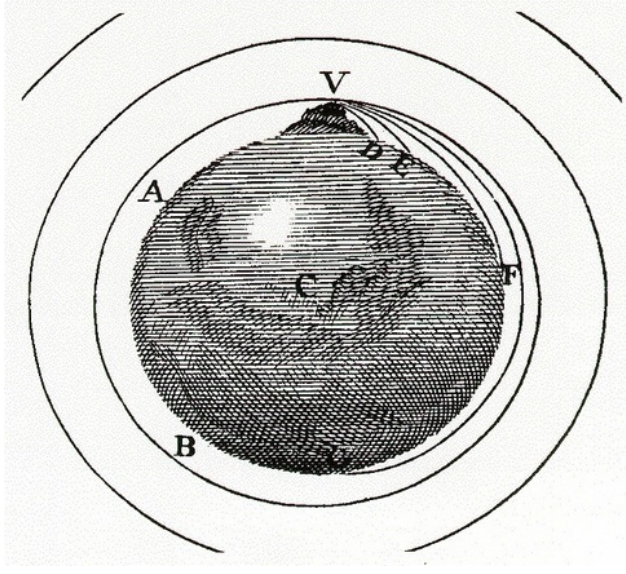

THE CAVE OF TIME
CHOOSE YOUR OWN ADVENTURE
NARRATIVE MAP
LAVENDER = NODE
YELLOW = NEW LIFE
GREEN = RETURN HOME
RED = DEATH
Asleep in the cave (1-3)
4
awake
8
ice age
18
mammoths
30
Cave
92
Year 3742
56
Perfect Society
57
Meta-CYOA (End)
60
Monster (End)
54
train
98
Travel the World (End)
101
Primodial Time (End)
94
Lincoln
96
Gettysburg (End)
100
California (End)
29
ride mammoth
52
Join Cavemen (End)
53
Fall to Death (End)
17
caveman
26
friends
46
Migrate (End)
47
wolf
50
Cave
81
Return Home (End)
49
Cave
25
28
Cave
51
Return Home (11 years later) (End)
10
Cave
20
Desert
93
Cave
32
Underwater
58
Hawaii
63
Boa (The End)
62
Surf's Up (The End)
31
Dying Sun (End)
64
Siren
102
91
90
21
Ice
33
Titanic
68
Sinks (The End)
79
Cave
88
Jurassic
86
London, 1940 (The End)
35
Louisa
80
Red Sun (The End)
76
Cadillac (The End)
107
Ship's Captain (The End)
111
Return Home (The End)
113
18th Century (The End)
61
Colonial America
104
Boston, 1718
106
Jail
Old Man
44
Apocalypse (End)
45
Return Home (End)
41
Return Home (End)
43
Return Home (End)
Return Home (The End)
T. Rex (The End)
Return Home (The End)
25
5
cold wind
16
Cave
114
Cave
61
Cave
24
Buffalo
40
Spaceship
83
Primitive People
85
Home among the Cavemen (The End)
84
Sabertooth Tiger (The End)
115
Return Home (The End)
38
Great Wall
87
Lost at Sea (The End)
82
Slave (The End)
6
Medieval Ages
22
King
37
Stepfather
36
Mad King
11
11
Tower
13
Release
14
Merrie England (The End)
15
982 AD (The End)
12
Loch Ness
66
Nessie
70
Egg
72
Return Home (The End)
103
Return Home (The End)
74
Eaten Alive (The End)
78
Hanging (The End)

In this discussion, we will consider some questions that are suggested by Ulam's discovery (which is now known as the Monte Carlo Method):

- What is special about a random event?
- Can we understand how some random events are "chosen" to occur?
- Are there underlying patterns in random events?
- Can we tell if a random event is actually biased?
- How can it be possible for a computer to behave randomly?
- Can random events be used, not just to play games or gamble, but to answer new scientific questions?




Distant between centers of mass, r
m_1
m_2
$\vec{F}_g = \frac{Gm_1m_2}{r^2}$
$\vec{F}_g = \frac{Gm_1m_2}{r^2}$


V
A
B
C
D
E
F


h = 5.46
k = 5.33
a = -0.24
IT WENT IN!
Line Equation: y = a*(x-h)^2 + k
$y = -0.24(x - 5.46)^2 + 5.33$

We often assume that the task of a scientist is to determine laws that are followed by natural objects and events.

One of the greatest such achievements, the rules of planetary motion, took thousands of years to work out, and resulted in Newton's law of gravity.

Newton's law for the moon also applies to basketballs.

It allows us to calculate the future location of an object of any size, moving at any speed, if we know a few pieces of initial information.

Newton's laws seemed to suggest a clockwork universe, that could be easily understood once we had worked out all the laws.

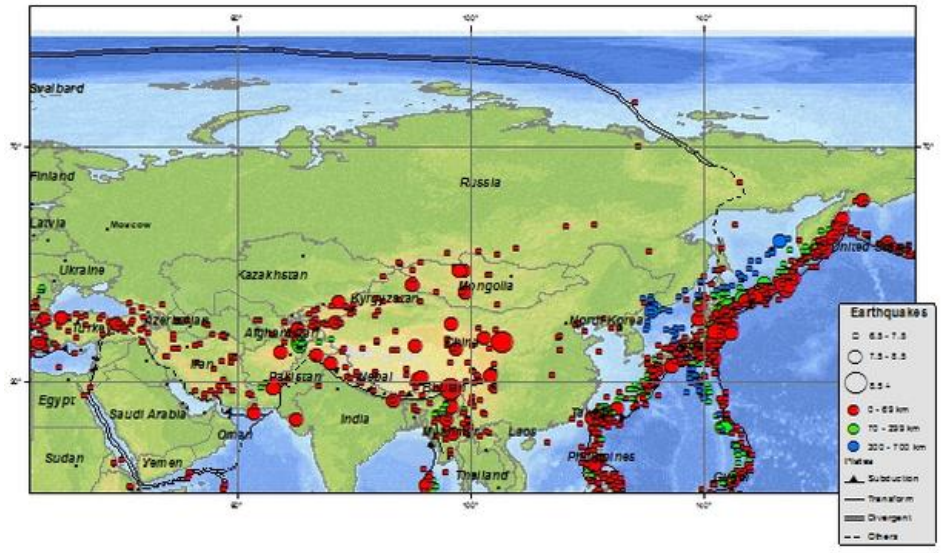

POWER BALL
POWERPLAY

Although scientists look for lawful behavior, there's at least one area where predictability is the last thing wanted - *games of chance*!

The reason we roll the dice, or shuffle the cards, or spin the wheel is to try to make it impossible to predict the outcome.

Shuffling, spinning or otherwise scrambling information to avoid prediction is sometimes called randomization, and the results, such as the sum of two dice, or a poker hand, or the lottery number, is called a random event.

When we say an event is random, we usually simply mean that, before it happened, we had no way of predicting that particular result.


Svalbard
Finland
Latvia
Moscow
Ukraine
Russia
Kazakhstan
Mongolia
Kyrgyzstan
Egypt
Saudi Arabia
Oman
Yemen
Sudan
India
Laos
Thailand
Earthquakes


m&m's


PANTHERS
3

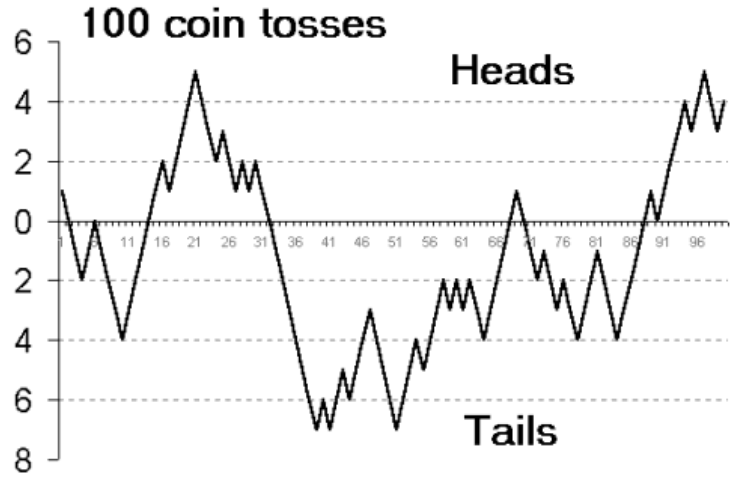
Examples of things that seem to be truly random:

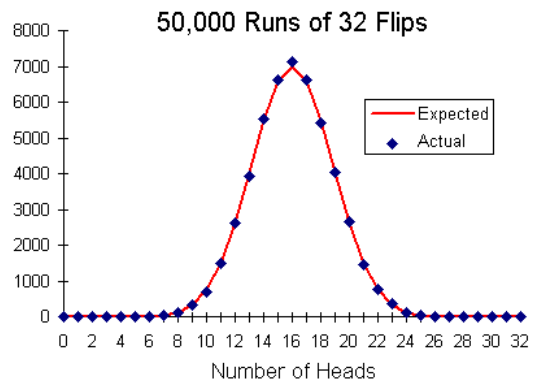
- (football) - who wins the toss of a coin;
- (biology) - baby girl or baby boy;
- (geology) - location and strength of next earthquake;
- (gambling) - the Power Ball lottery number);
- (safety) - the number of fatal traffic accidents per year;
- (candy) - the number of red, orange, blue, green, yellow and brown M&M's in a package;
- (manufacturing) - the number of hours a light bulb will last;
- (weather) - the number of named hurricanes in a season;
- (medicine) - the chances that a patient will be cured;

Coin Toss Experiment Run 10 x 10 Times (H=Head; T=Tails)

Group \ Trial	1	2	3	4	5	6	7	8	9	10
1	T	H	T	T	H	H	T	H	H	H
2	H	T	H	T	T	T	H	T	H	H
3	T	T	H	T	T	T	H	T	H	H
4	H	T	H	T	H	T	H	T	H	T
5	T	H	H	H	H	H	T	H	H	H
6	H	H	H	T	T	H	T	T	T	T
7	H	T	H	T	T	T	H	H	T	T
8	T	T	T	H	T	H	T	T	H	T
9	T	T	H	H	T	H	H	T	H	H
10	T	H	H	T	H	T	T	T	T	H

H	49
T	51



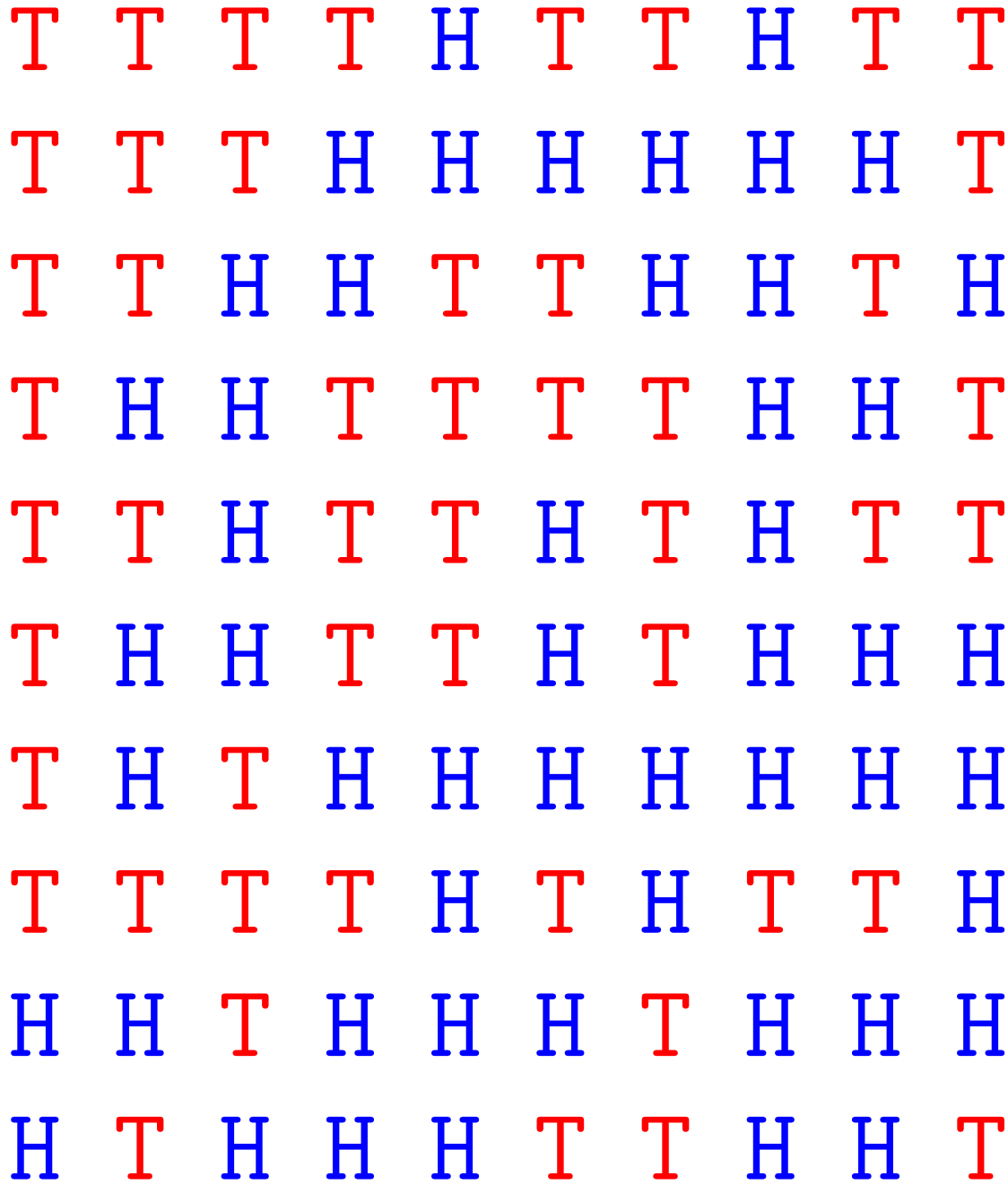


Outcomes	Frequency
Heads	26
Tails	24
Total	50

The simplest random event is a coin toss. We say heads and tails are *even odds*, or *50-50* or *equally likely*. The technical term is that heads and tails have uniform=(equal) probability.

This doesn't help us to predict what happens on a single coin toss, but if we toss a coin many times...or many coins all at once, we do predict that the number of heads and tails will be roughly the same.

We make this judgment based on our observations of actual experiments. Let's take an example of such an experiment and see what insight we can gain.

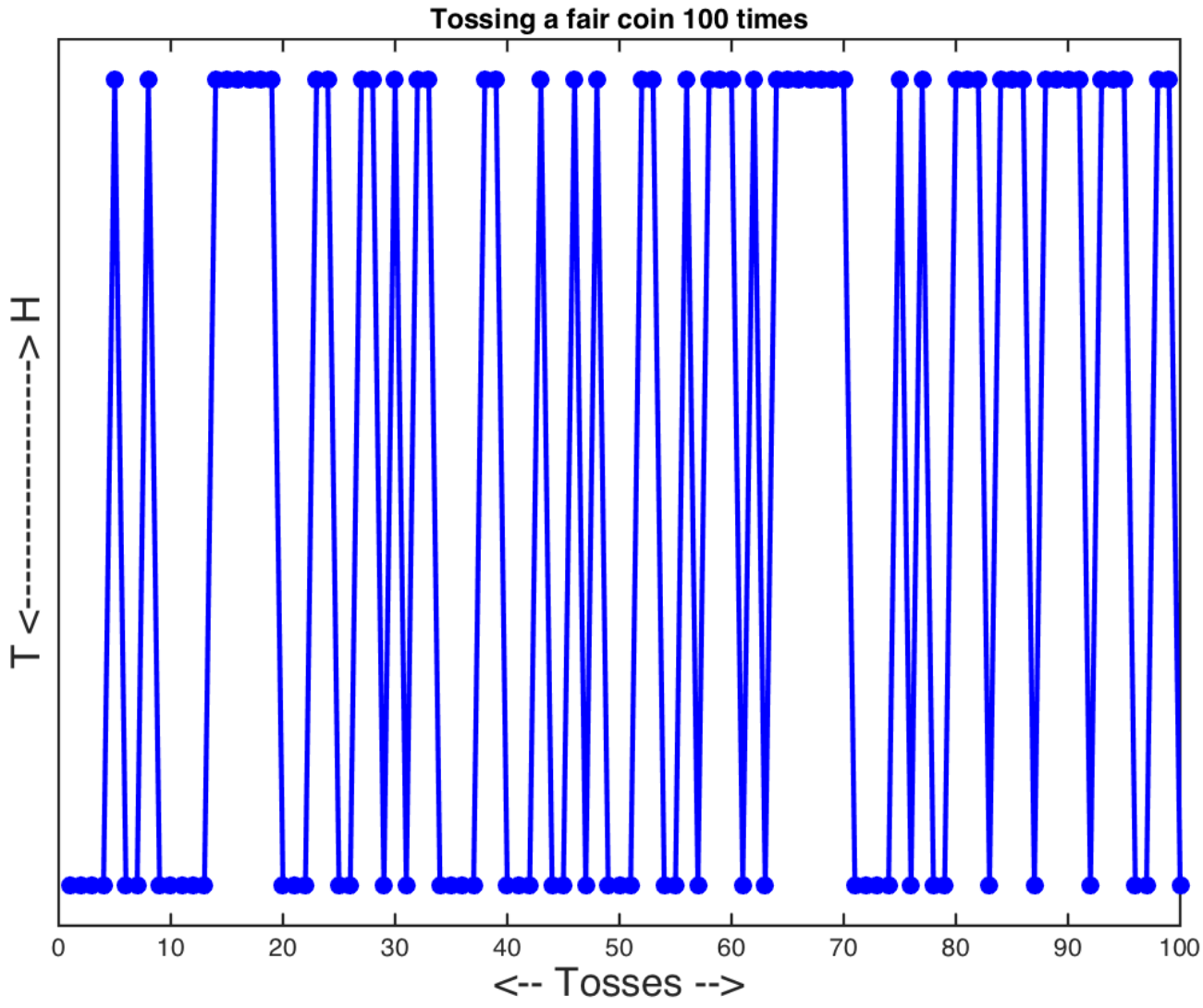

T T T T H T T H T T
T T T H H H H H H T
T T H H T T H H T H
T H H T T T T H H T
T T H T T H T H T T
T H H T T H T H H H
T H T H H H H H H H
T T T T H T H T T H
H H T H H H T H H H
H T H H H T T H H T

Because we wrote the data as a table, rather than a list, our eye can take it all in at once.

Some patterns, like the actual number of H's and T's are real; it seems like there are about the same number of each, but it's hard to judge.

Other patterns are accidental. Do you see that column 1 is mostly T's? Do you see many horizontal strings of H's? If we changed the shape of the table, many of these patterns would change or disappear.

Especially if the list of symbols becomes very long, we may want to find other ways of displaying the data.


Tossing a fair coin 100 times
T <----------> H
0
10
20
30
40
50
60
70
80
90
100
<-- Tosses -->

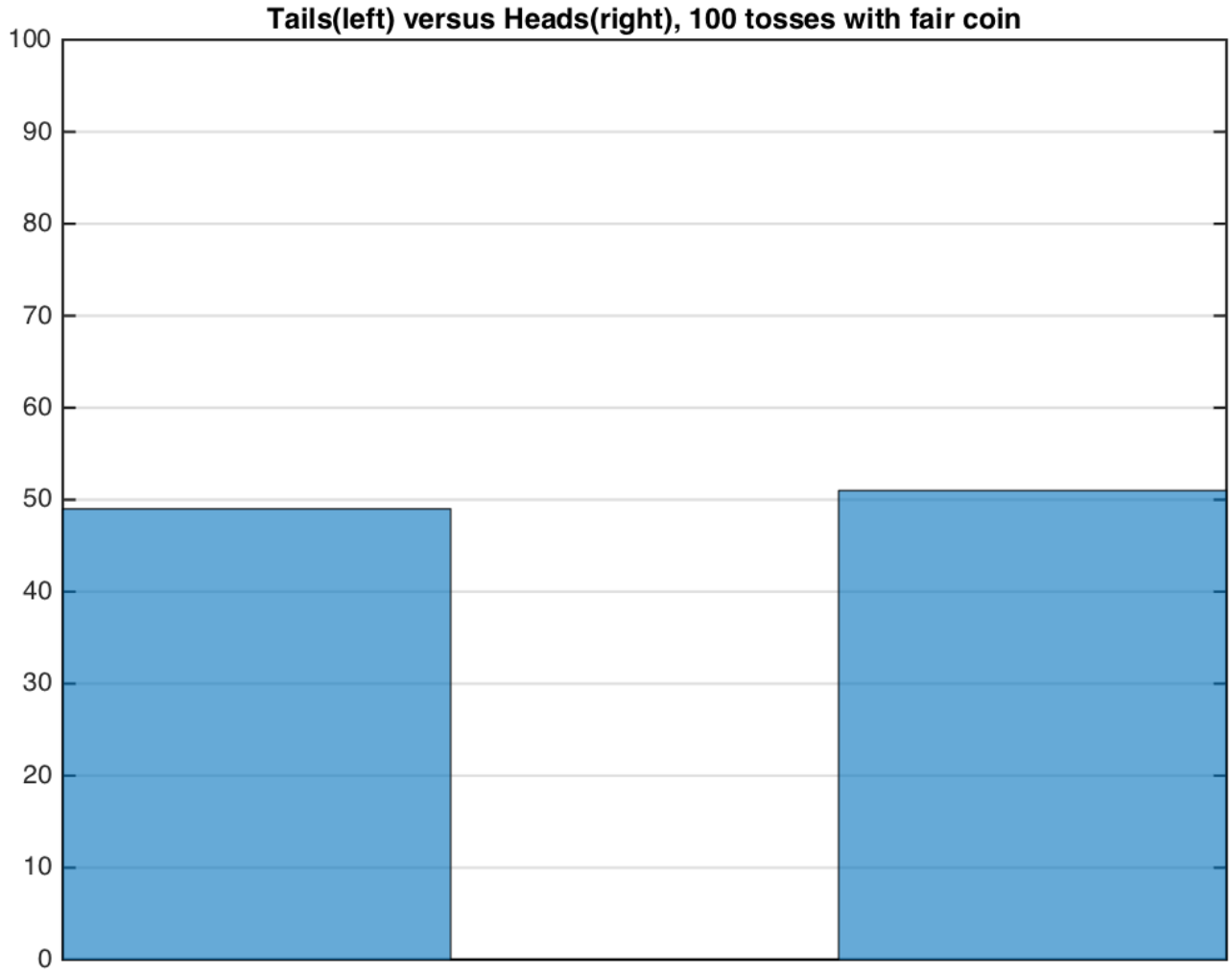
If we want to show all the coin toss results, in the exact order they occurred, we can make make a line plot, which connects pairs of data points.

Most plot programs expect numbers to work with. One way to do this is to represent T by -1 and H by +1.

Most plot program work with *pairs* of numeric values. For our data, we'll number the tosses, and pair the number with the result.

Symbol:	T	T	T	T	H	T	T	H	T	T	...
Toss:	1	2	3	4	5	6	7	8	9	10	...
Value:	-1	-1	-1	-1	+1	-1	-1	+1	-1	+1	...

So the plotting program would work with pairs (1,-1), (2,-1), (3,-1), (4,-1), (5,+1), (6,-1), and so on.

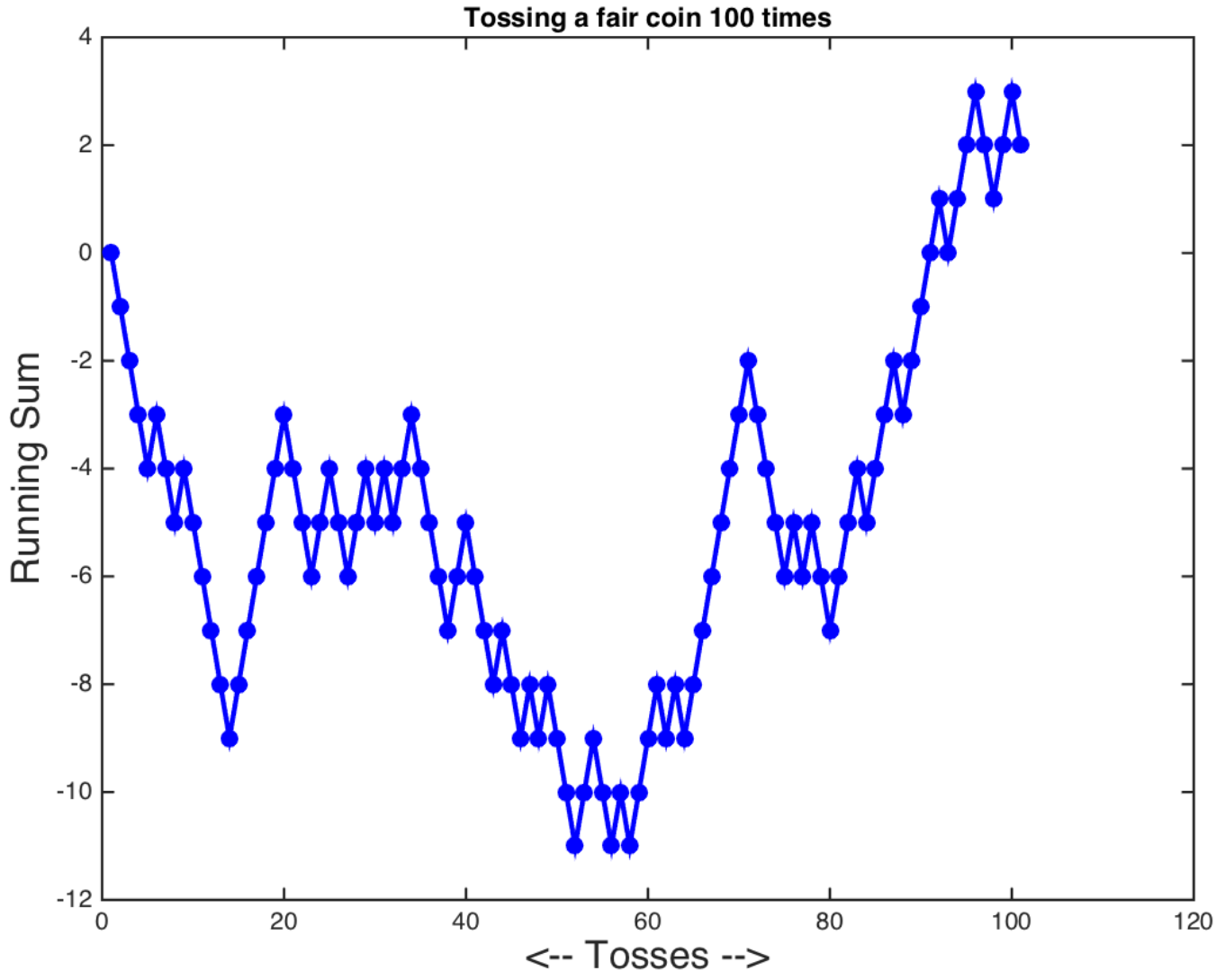

Tails(left) versus Heads(right), 100 tosses with fair coin
100
90
80
70
60
50
40
30
20
10
0

The simple plot does store all the data, but it's still very hard for the eye to read.

One of the obvious questions we may wonder is how many heads and tails did we encounter? To answer that question, we don't need to see the entire history of the experiment, just the frequency of each result.

We can make a bar chart where the height of the left and right bars represents the number of tails and heads in our experiment.

Such a chart is easy to read, even if we consider hundreds or thousands of coin tosses.


Tossing a fair coin 100 times
Running Sum
<-- Tosses -->
4
2
0
-2
-4
-6
-8
-10
-12
0
20
40
60
80
100
120

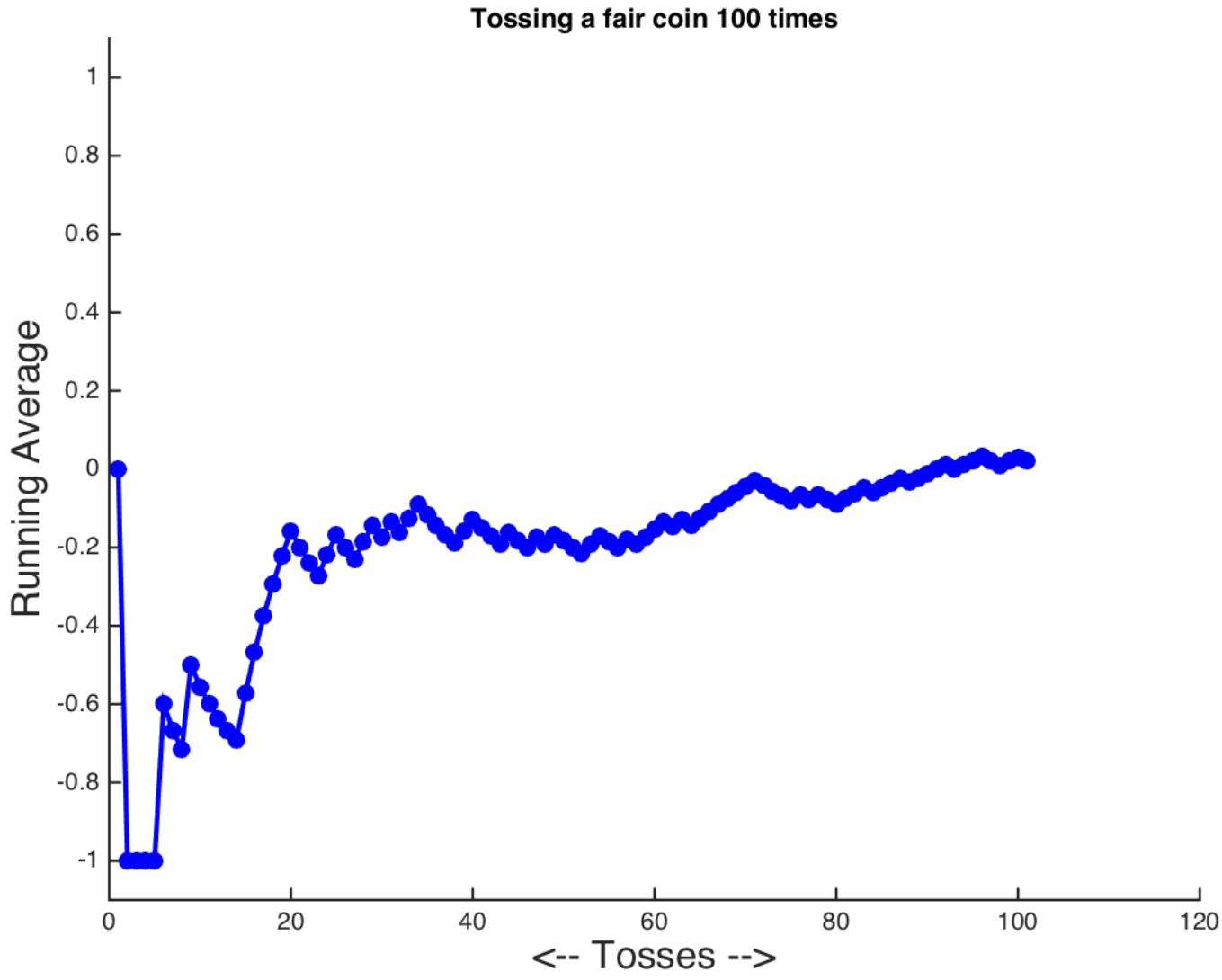
Suppose that, in this experiment, we won a dollar each time a head came up, and paid a dollar for each tail.

Then something that would be very interesting to us would be the *running sum* of the results, that is, our current winnings or losses.

We compute a running sum by adding up all the preceding values. For our data, this process would look like this:

Symbol:	T	T	T	T	H	T	T	H	T	T	...
Toss:	1	2	3	4	5	6	7	8	9	10	...
Value:	-1	-1	-1	-1	+1	-1	-1	+1	-1	+1	...
Sum:	-1	-2	-3	-4	-3	-4	-5	-4	-5	-4	...

We seem to be losing badly at the beginning, but the plot shows how we come out ahead when the game ends at toss 100, winning a total of $2.


Tossing a fair coin 100 times
Running Average
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
<-- Tosses -->

If we're playing for money, we might wonder whether the game is fair or not. As with most random events, the answer to such a question is never certain, but it becomes more and more obvious as we repeat the experiment many times.

A fair way to judge the coin tossing game is to compute the *running average* of the results. After each toss, divide the current total winnings or losses by the number of tosses. We'd expect the running averages would tend to zero.

We compute a running average by averaging the sum.

Symbol:	T	T	T	T	H	T	T	H
Toss:	1	2	3	4	5	6	7	8
Value:	-1	-1	-1	-1	+1	-1	-1	+1
Sum:	-1	-2	-3	-4	-3	-4	-5	-4
Ave:	-1/1	-2/2	-3/3	-4/4	-3/5	-4/6	-5/7	-4/8
Ave:	-1.00	-1.00	-1.00	-1.00	-0.60	-0.66	-0.71	-0.50

While the pattern is not so obvious in the first 10 values, it does seem to be decreasing. In fact, the average quickly drops to zero. This seems to be a fair game, and a fair coin.

Coin Toss Experiment Run 10 x 10 Times (H=Head; T=Tails)

Group \ Trial	1	2	3	4	5	6	7	8	9	10
1	T	H	T	T	H	H	T	H	H	H
2	H	T	H	T	T	T	H	T	H	H
3	T	T	H	T	T	T	H	T	H	H
4	H	T	H	T	H	T	H	T	H	T
5	T	H	H	H	H	H	T	H	H	H
6	H	H	H	T	T	H	T	T	T	T
7	H	T	H	T	T	T	H	H	T	T
8	T	T	T	H	T	H	T	T	H	T
9	T	T	H	H	T	H	H	T	H	H
10	T	H	H	T	H	T	T	T	T	H

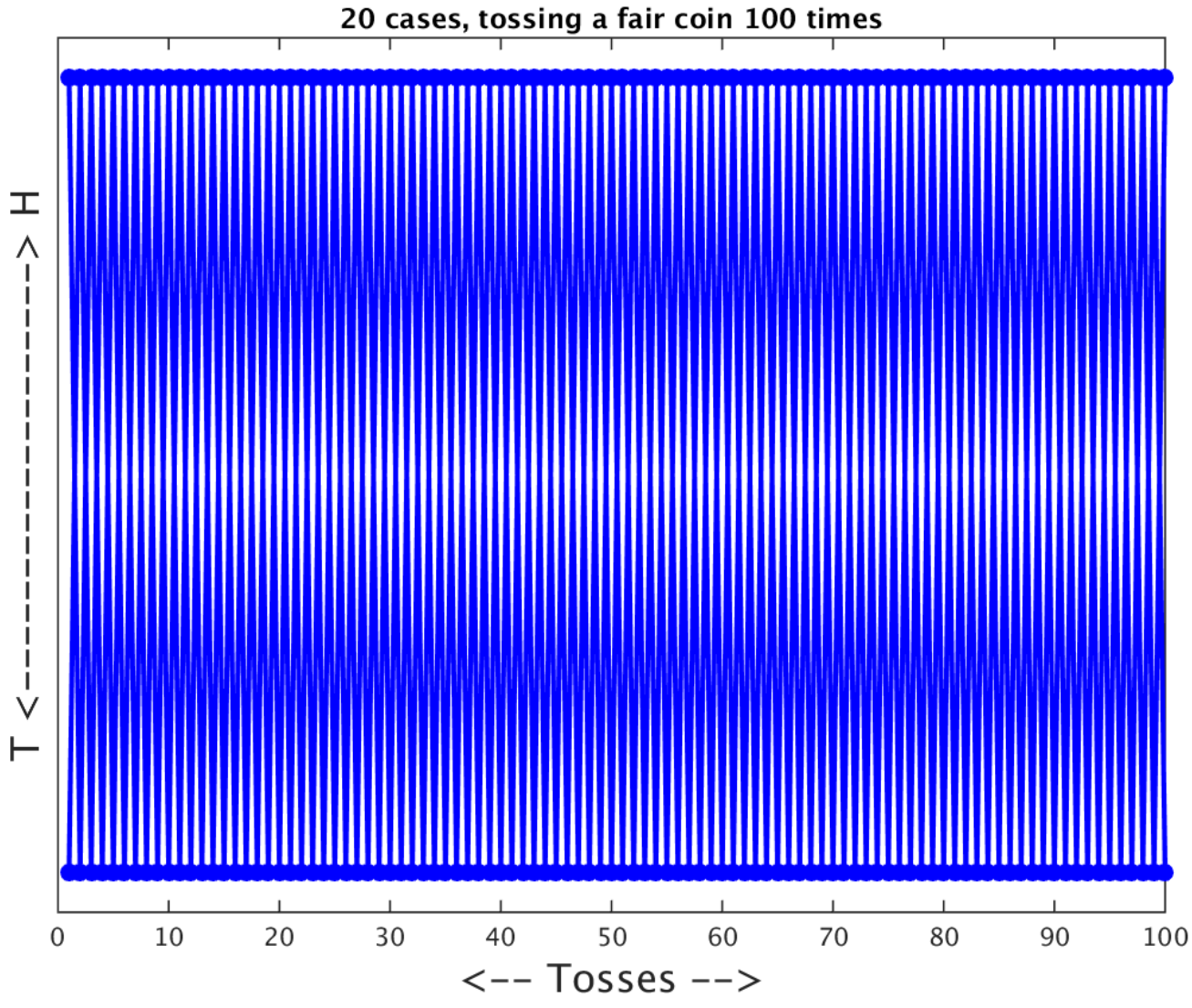
H	49
T	51

Ten students were each asked to flip a coin ten times and record the results. In the table, each row represents the results for a particular student. Because there are only 10 students and 10 flips, the table is compact.

But suppose one student did all the flips, and worse yet, suppose it was 1000 flips? What are our choices for presenting the information? Remember, we have to see the data in order to find any patterns in it.

Certainly, one option is simply a list of 1000 H's and T's, but this would not be easy to study.

If we really want to see all the data, in the order in which it occurred, we can make a plot with 1,000 spikes, going up for heads and down for tails.


20 cases, tossing a fair coin 100 times
T <-------> H
0
10
20
30
40
50
60
70
80
90
100
<-- Tosses -->

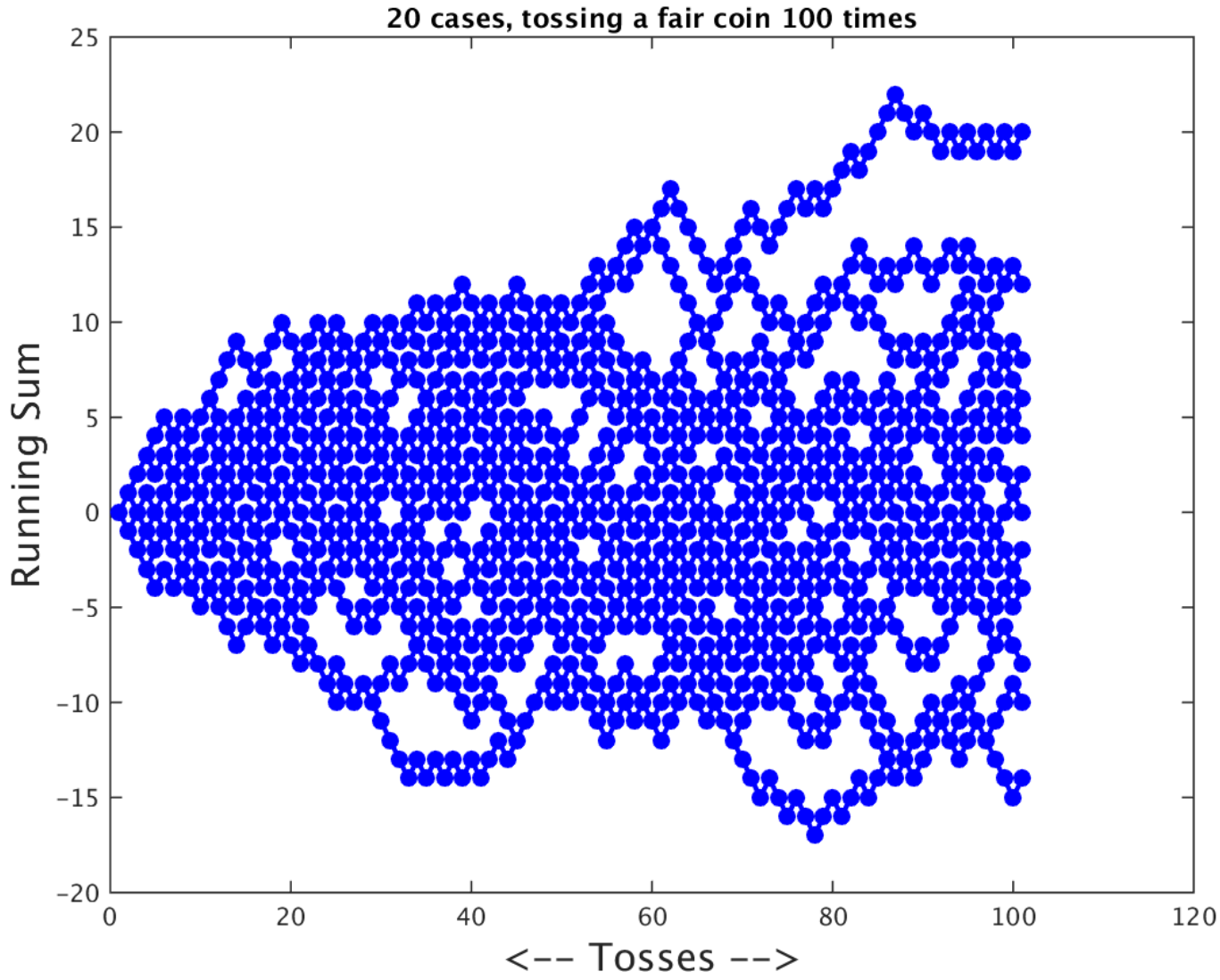
Let's suppose we have 20 students who do 100 coin tosses each.

How can we present this data in a way that will make patterns clear?

Our first attempt is simply to plot all the results, that is, for each student, we plot a zig-zag line that is $+1$ for each head and -1 for each tail.

The resulting plot seems useless, a blue cloud. It does suggest, though, that from one toss to the next, we have every possible pattern occurring (H-¿H, H-¿T, T-¿H, T-¿T).

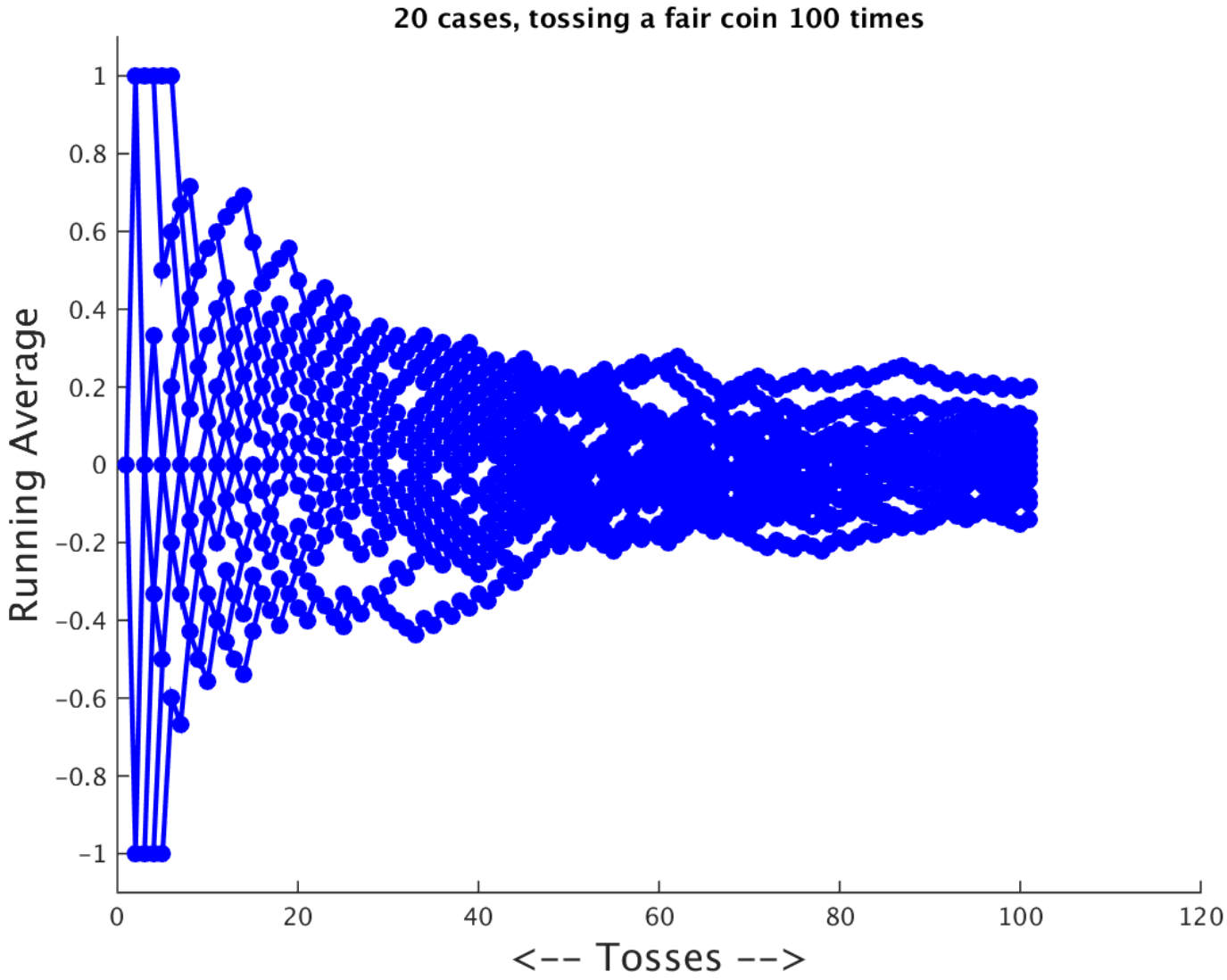
We only see this because the experiment was carried out 20 times.


20 cases, tossing a fair coin 100 times
Running Sum
<-- Tosses -->
25
20
15
10
5
0
-5
-10
-15
-20
0
20
40
60
80
100
120

If we plot the running sum, this tells us how much we would be winning or losing, if each coin toss was worth \$1.

But when we have 20 experiments, we can't see individual results well. However, we do notice something interesting. The average winnings tend to slowly drift apart, but there seems to be as much chance to be ahead as behind.

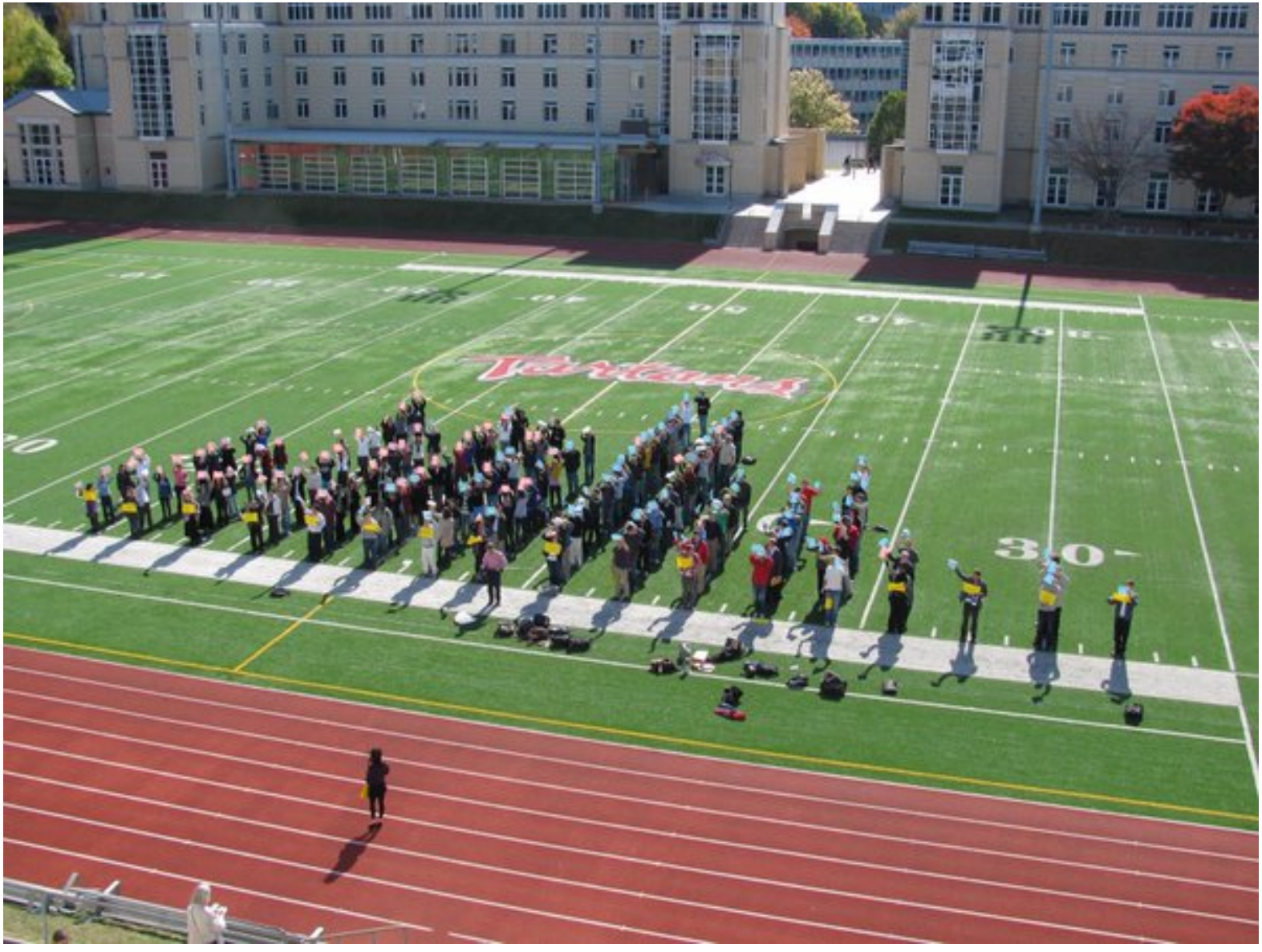
However, one thing you may notice is the tendency of most of the sums to stay near the middle.


20 cases, tossing a fair coin 100 times
Running Average
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
<-- Tosses -->

If we plot the running average, this tells us how much we would be winning or losing, on average, per toss.

This plot actually has useful information, because it shows that, for all 20 experiments, the running average tends to drop down towards zero.

It's hard to see the details, and to have a stronger certainty we'd want to take more tosses, but the running average experiment suggests that we have used a fair coin.

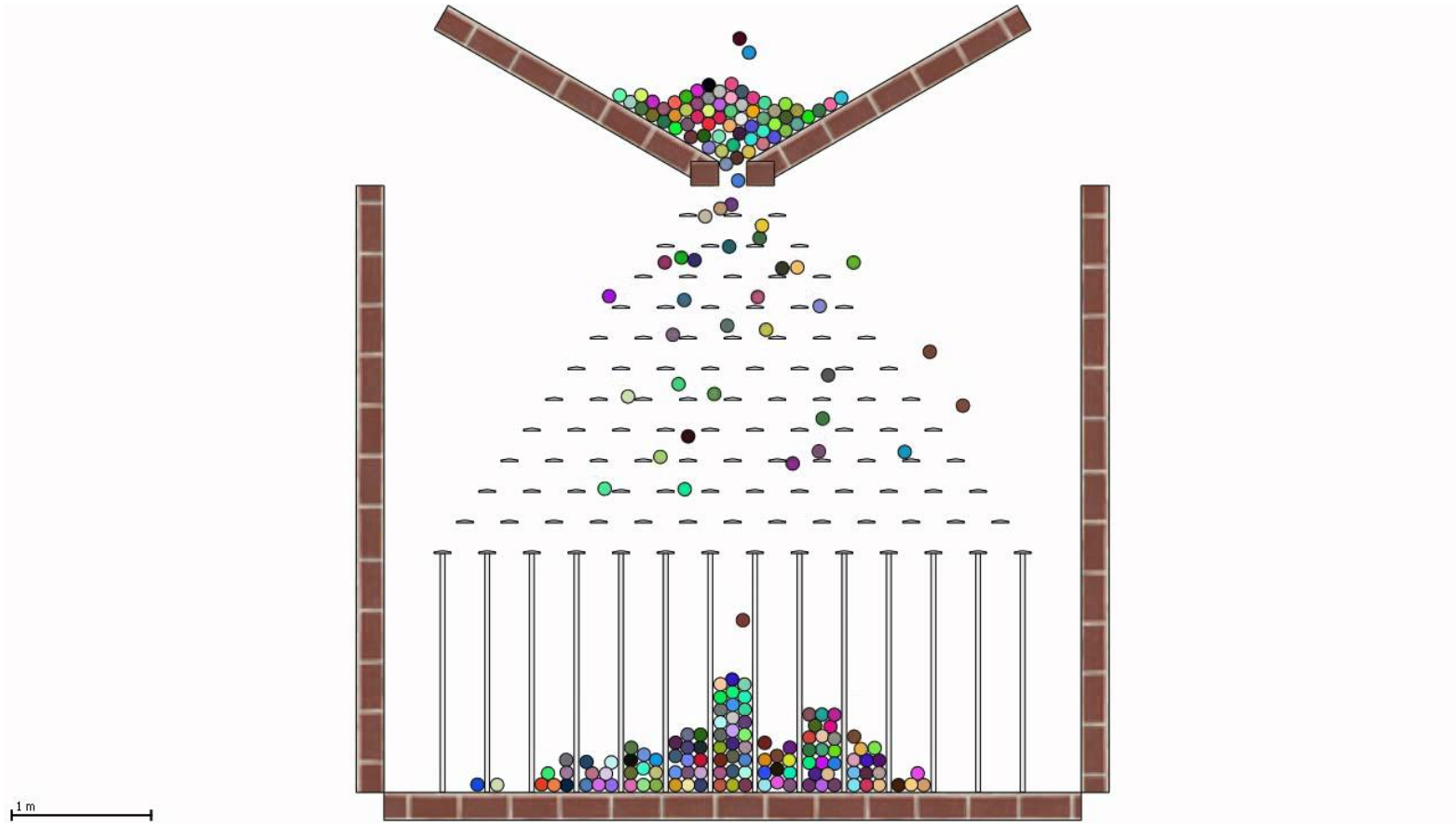


When we looked at 20 running sums plotted together, it seemed as though the curves tended to stay near the center, and only a few curves wandered far away.

This was not an accident, it's a feature of the coin tossing experiment, and of many natural phenomena.

It turns out that results in the middle are much more likely that those at the extremes, and that this becomes more and more true as we increase the number of coin tosses.

This may seem strange, since the coin has no preference for heads or tails. How does repeatedly tossing an unbiased or fair coin seem to produce a biased distribution of results?


1 m

Another example of a random event occurs when a marble drops down onto an obstructing peg, and must swerve left L or right R, as a tossed coin will show heads or tails.

If there are levels of pegs, that's like tossing the coin several times.

Following the marble's path to its final location is just like tracing the running sum. As the picture suggests, the marbles have a very strong tendency to end up near the middle.

This behavior, often observed in nature, is called *the central tendency* or *the return to the average*.

COMPUTATIONAL THINKING

Start at top of path

Repeat until bottom of path

    Toss coin

    If coin = Heads

        Go 1 step on RED path

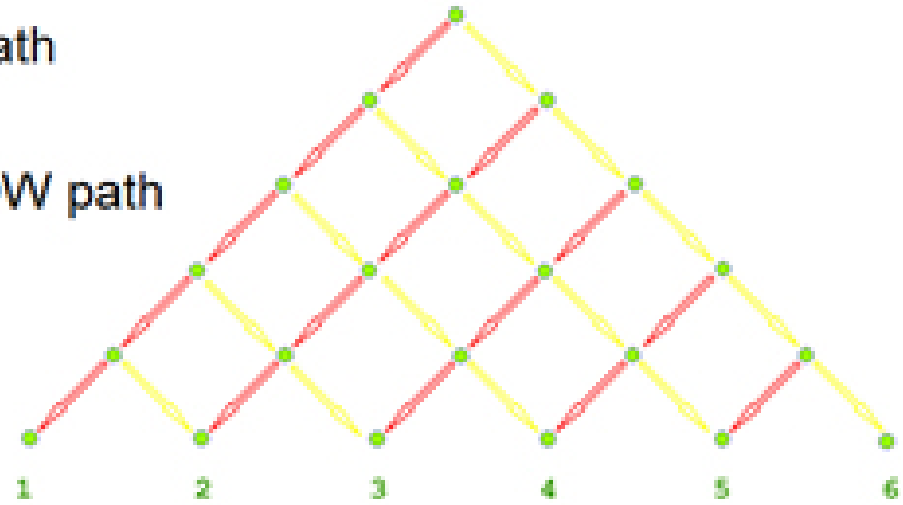
    Else

        Go 1 step on YELLOW path

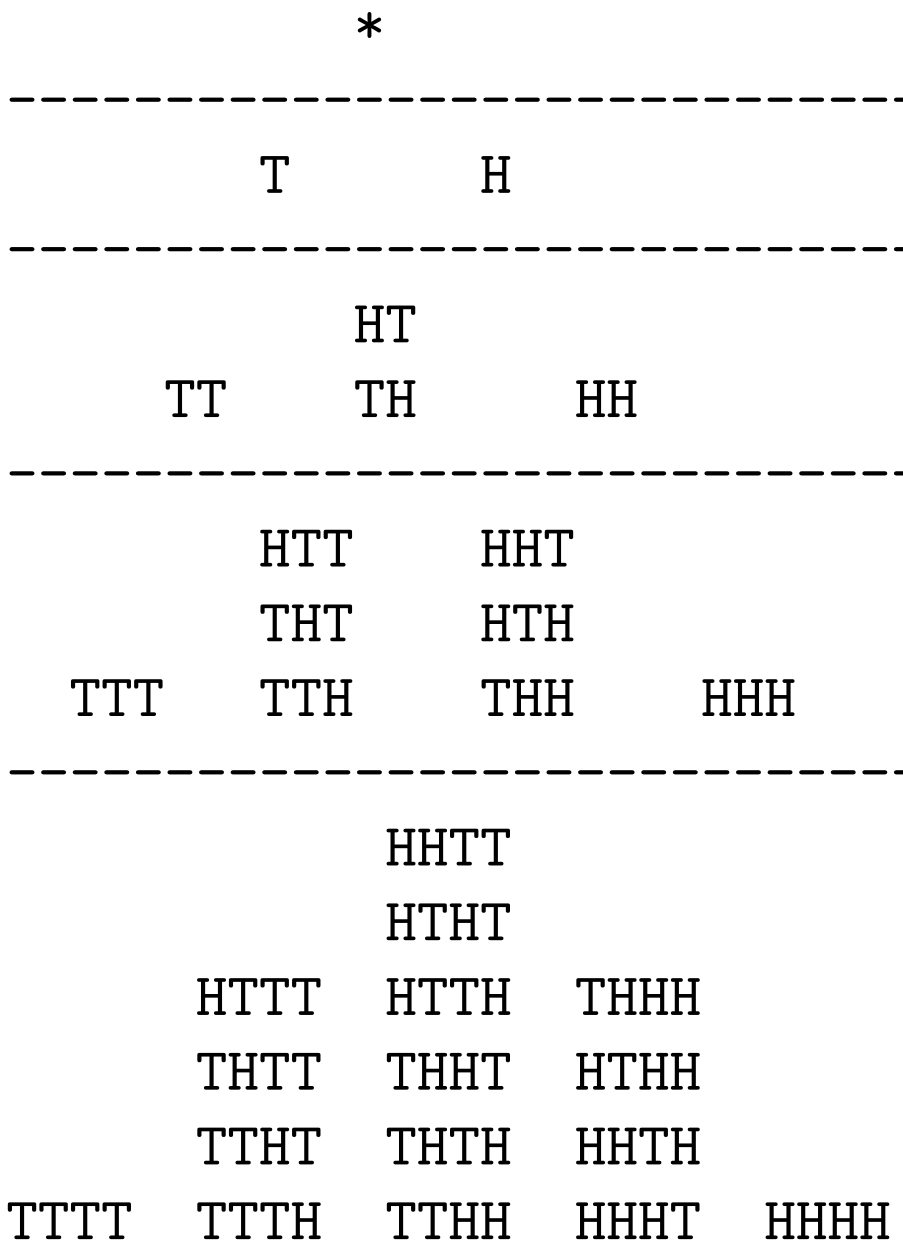
    End If

End Repeat

Circle your number



As a demonstration of this behavior, let's try this exercise.

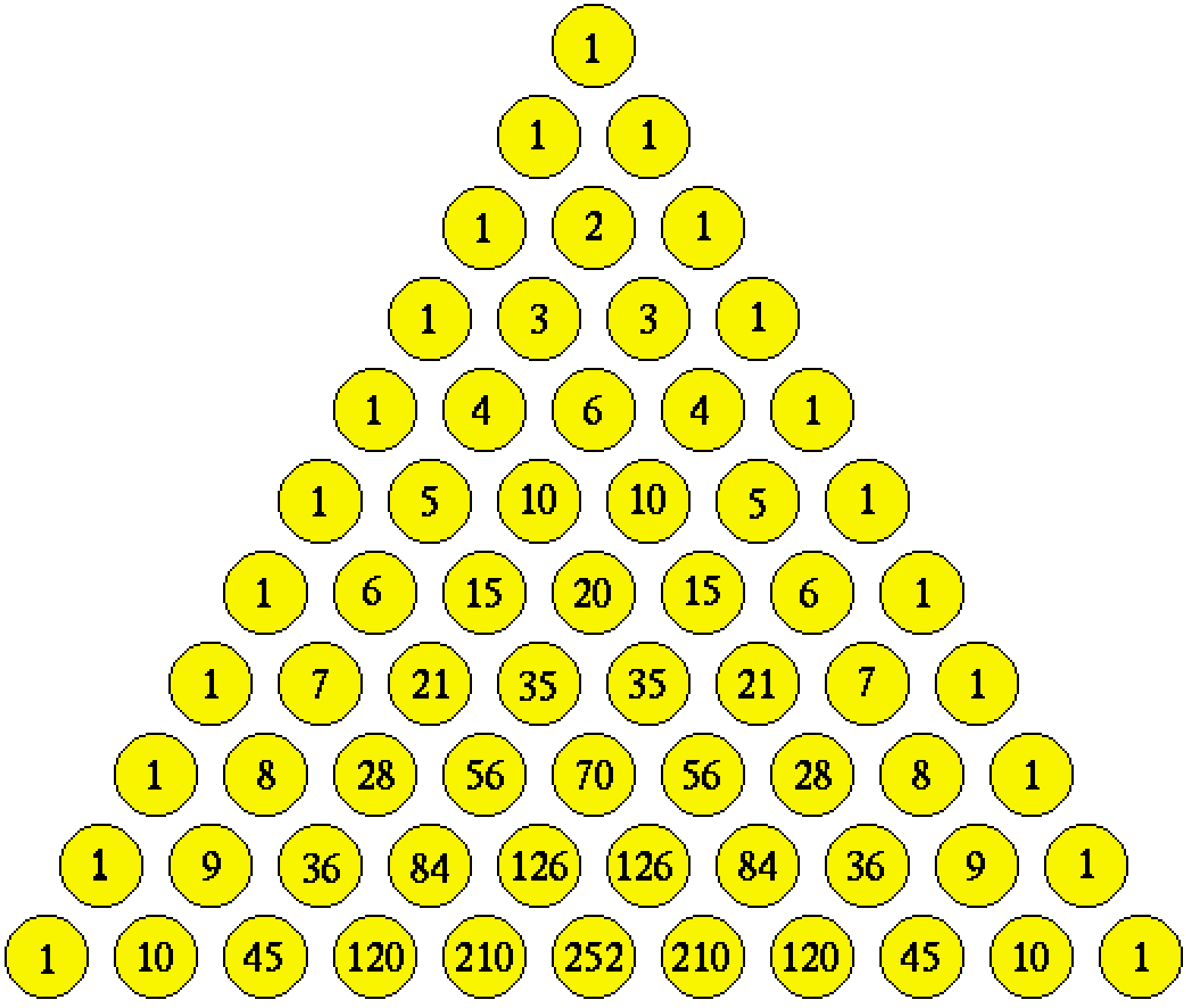

*
T H
HT
TT TH HH
HTT HHT
THT HTH
TTT TTH THH HHH
HHTT
HTHT
HTTT HTTH THHH
THTT THHT HTHH
TTHT THTH HHTH
TTTT TTTH TTHH HHHT HHHH

Now we can try to understand the pattern in our results.

After one toss, it's easy to see we have two equally likely results, T or H.

After another toss, we have four equally likely results, if we want to remember the order in which things occurred. But usually, we simply ask how many heads and tails we have. In that case, there is only one way to have two heads or tails, but two ways to have a head and a tail. When we toss again, there is only one way to have three tails, but three ways to have 2 tails and a head; we could have picked up the head on the first, second or third toss.

And as we go down the table, there are more and more ways to get answers near the middle, whereas it is hard to stay at the extremes unless you always get heads or always get tails.


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
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1

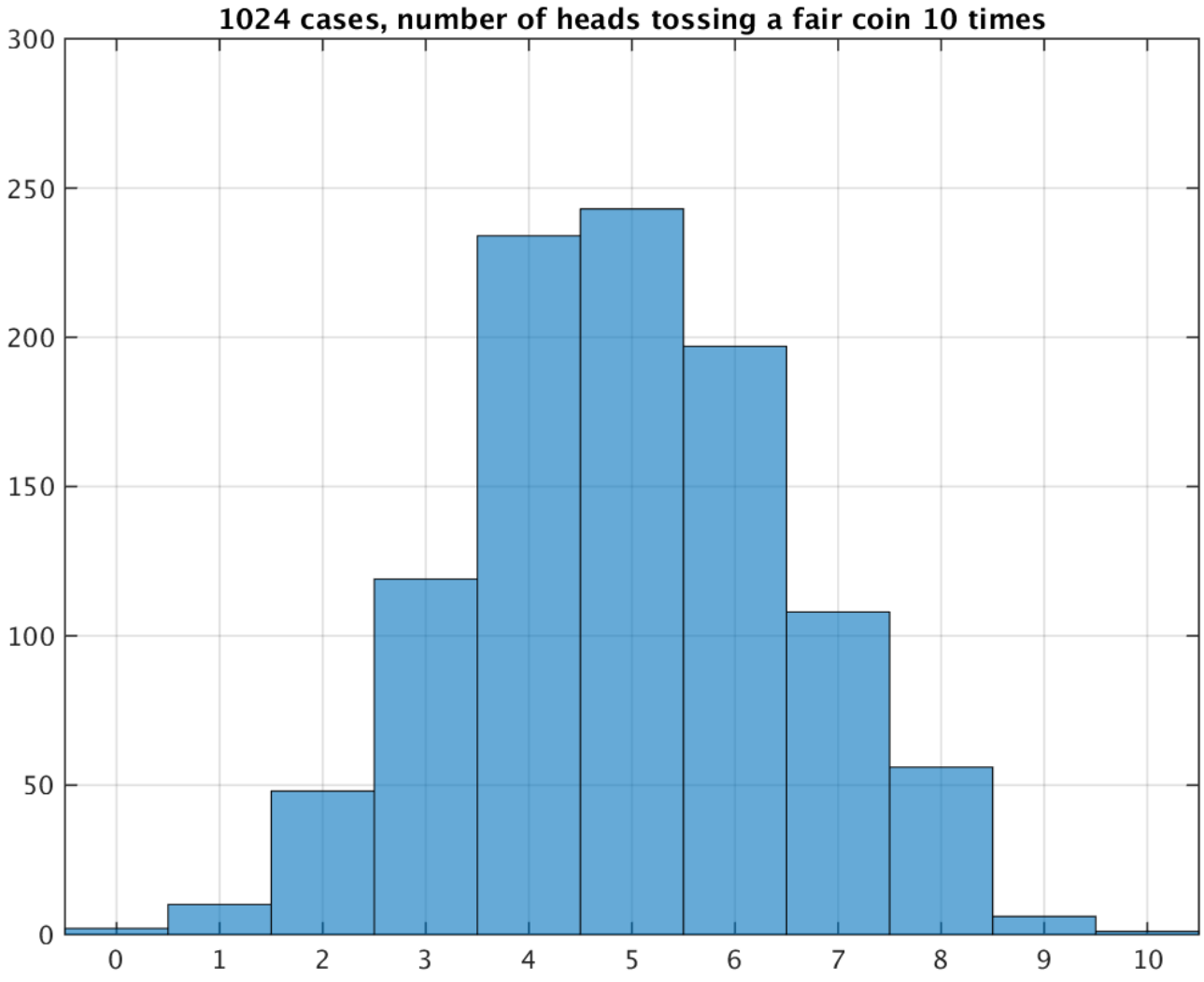
In going from row to row of our triangle, there is a simple pattern.

If we simply write down the **number** of ways we can get a given result, we have what is called Pascal's triangle.

As we move down the triangle, you can see that in each row, the first and last values are always 1, while the numbers towards the center continue to grow larger.

In each row, the numbers count the number of ways of reaching that result. The more ways, the more likely the result.

This reflects the fact that repeated coin tosses tend to result in about the same number of heads and tails, and that in a pachinko game, the balls will most often end up near the middle.


1024 cases, number of heads tossing a fair coin 10 times
300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8
9
10

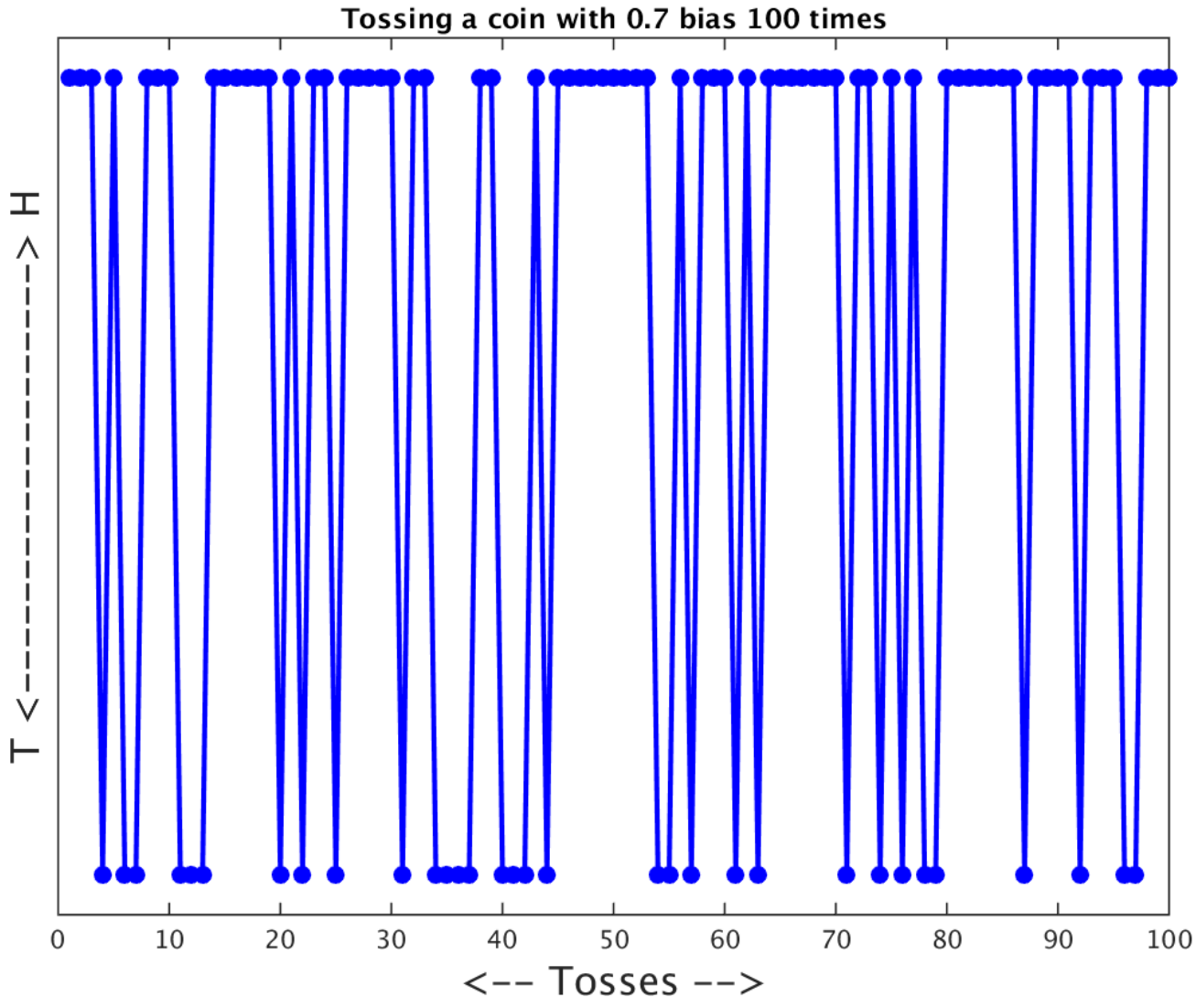
In order to see patterns in a random process, we have to observe it many times.

So we can imagine that a single experiment is to toss a coin ten times, and count the number of heads.

To see that we tend to get about 5 heads most of the time, we can do this experiment many times.

For this test, with 1,024 experiments, we definitely see the tendency to get results near the middle of the range.

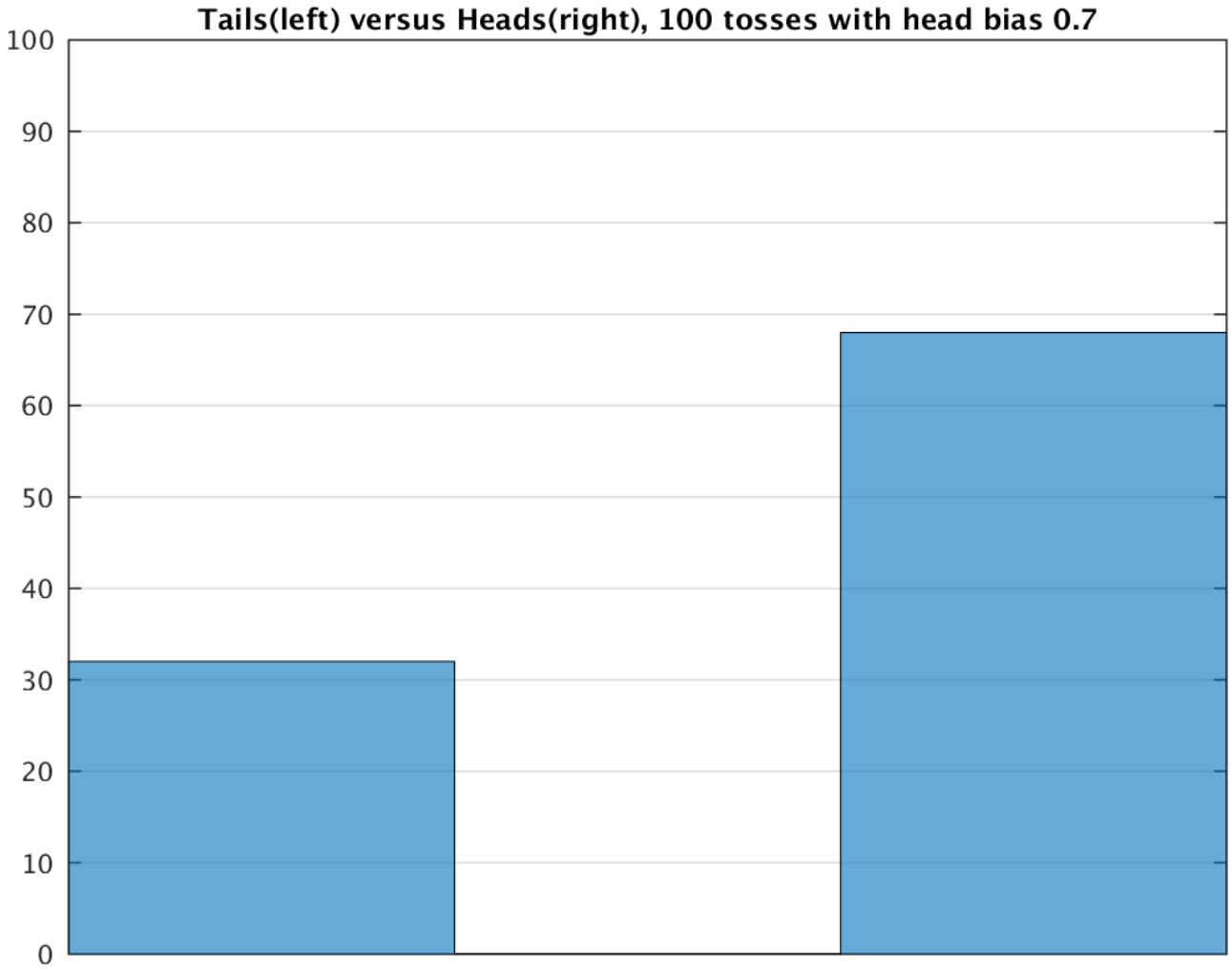
If we do it exactly 1,024 times, then the numbers we see in the bar chart should roughly match the numbers we saw in Pascal's triangle.


Tossing a coin with 0.7 bias 100 times
T <----> H
0
10
20
30
40
50
60
70
80
90
100
<-- Tosses -->

So far, we have assumed that the coin is fair.

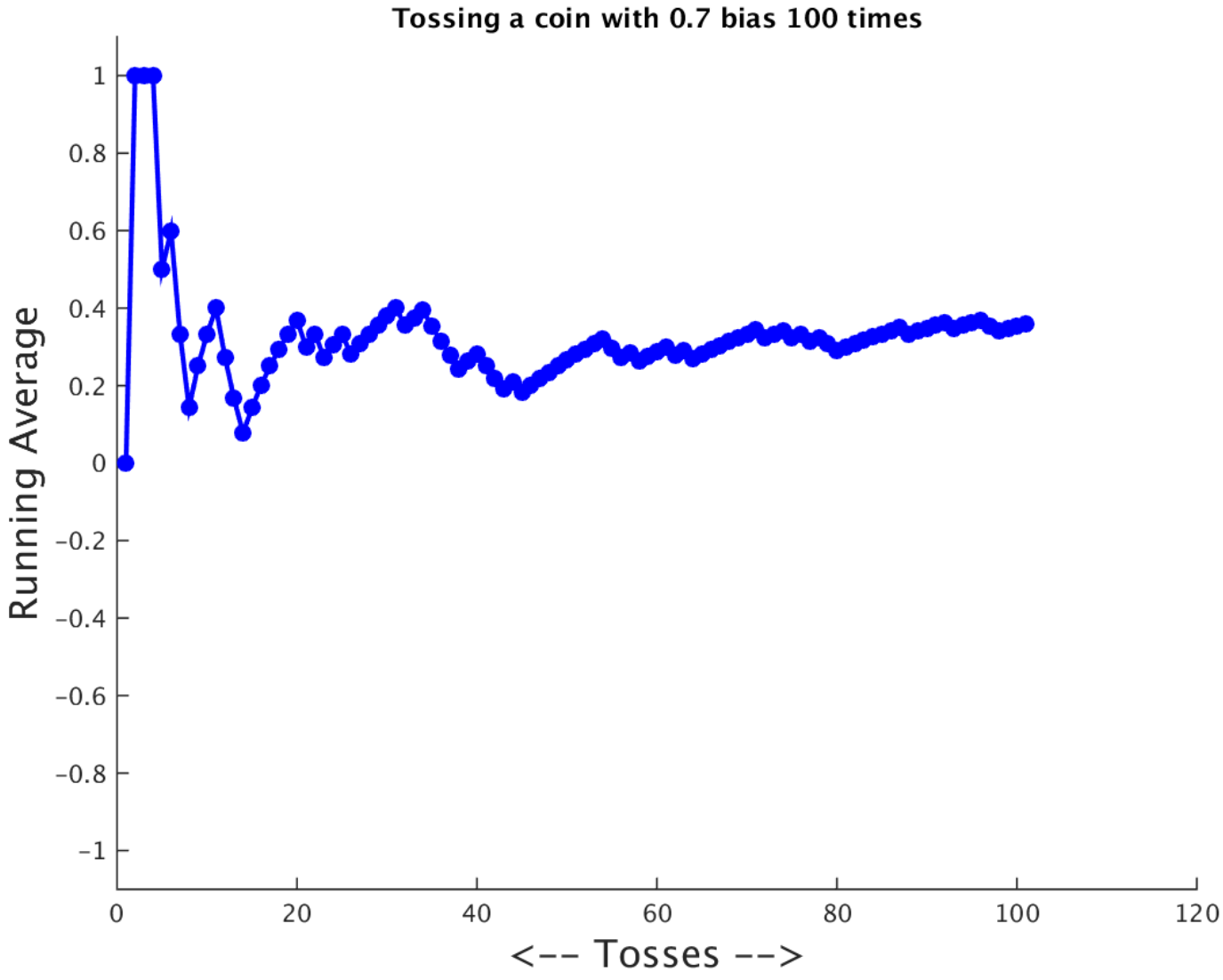
Suppose someone has replaced the fair coin with a trick coin that shows up heads much more often than tails?

If we make a plot of the results for 100 tosses, we may think we see something wrong, because there seem to be a number of streaks of heads showing up.


Tails(left) versus Heads(right), 100 tosses with head bias 0.7
100
90
80
70
60
50
40
30
20
10
0

In this case, doing a bar chart suggests that there is something very wrong.

This unfair coin has a tendency, in 10 tosses, to turn up heads 7 times, and tails 3 times. By plotting the results of 100 tosses, we can see and measure this behavior very clearly.

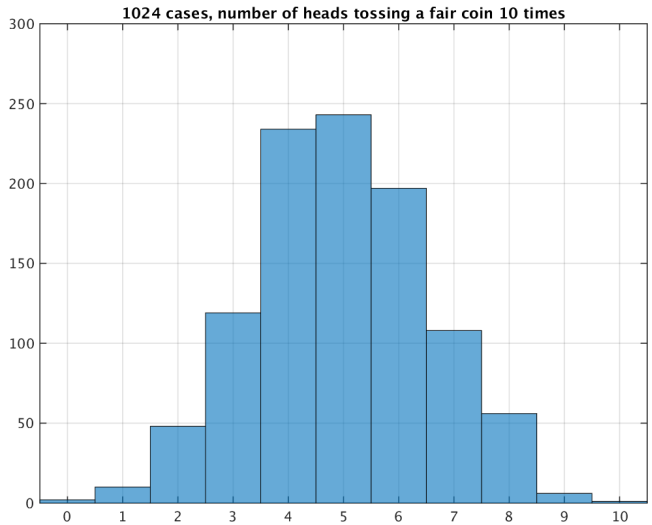

Tossing a coin with 0.7 bias 100 times
Running Average
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
<-- Tosses -->

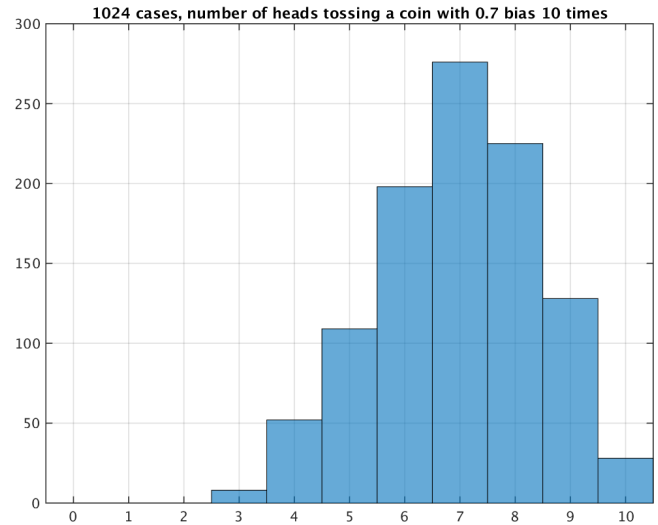
The running average of the results constantly updates an estimate of how much we are winning per coin toss.

This average may wiggle around at first, but it should eventually settle down.

In a fair game, the running average settles down to zero.

In this unfair game, we see an estimated value closer to 0.4, that is, on average, you can expect to make about 40 cents per toss over the long run.


1024 cases, number of heads tossing a fair coin 10 times
300
250
200
150
100
50
0
0 1 2 3 4 5 6 7 8 9 10


1024 cases, number of heads tossing a coin with 0.7 bias 10 times
300
250
200
150
100
50
0
0 1 2 3 4 5 6 7 8 9 10

If we compare the barcharts for 10 tosses of the uniform and biased coins. we see the effect of the bias.

We said the uniform tosses had a tendency to go to the center. For the biased coin, the tosses still have a tendency, but now the "center" is not a physical place (the middle). Ten tosses of a coin with bias 0.7 seems to tend towards 7 heads, which is actually what we should expect.

Using what we know, let's imagine a gambling situation that's a little more realistic.

Suppose there are two gamblers, Mr Alpha (A) and Mr Bond (B).

Mr A has \$3, and Mr B has \$7.

If a coin lands heads, Mr A takes \$1 from Mr B, while tails works the other way.

The two gamblers are going to play "all the way", that is, they will not stop until one of them is bankrupt.

```
A:  3  4  5  4  3  2  3  2  1  2  1  2  3  2  1  0
      H  H           H        H     H  H
               T  T  T     T  T     T        T  T  T
B:  7  6  5  6  7  8  7  8  9  8  9  8  7  8  9 10
```

In this example, A loses after 15 tosses, with 6 heads and 9 tails.

In fact, 9 - 6 = 3, because A started with $3,000 and he is guaranteed to lose as soon as there are 3 more tails than heads.

On the other hand, for B to lose, there have to be 7 more heads than tails, because he has $6,000 to start with.

As long as the coin is fair, it is much more likely that the current total will have an excess of 3 tails than an excess of 7 heads. So there is an important advantage in starting the game with more money than your opponent.

```
set A and B to initial values (such as 3 and 7)

if nobody is bankrupt then

  toss a coin (choose H or T randomly)

  keep track of the total number of tosses

  if we got an H
    transfer 1 from B to A
  otherwise
    transfer 1 from A to B
```

As long as we have a coin, or other randomizer, we can simulate this game.

To get some idea of how how the game works, we can play it many times, and look at the results.

We can replay the game with new values of A and B, and see what effect that has as well.

Things to ask:

- Given A and B, what is a typical length of the game?
- Given A and B, what are the chances A or B will win?

A	B	Average length	Maximum length	Probability A wins	Probability B wins
3	7	21	135	0.29	0.71
30	70	2,010	12,992	0.28	0.72
10	10	101	722	0.49	0.51
1	100	104	10,472	0.01	0.99

We simulated the gambling game 1,000 times, and recorded how long each game took, and who won. By averaging, we got the average length of a game, and the estimated probablities for A or B to win.

By looking at the numbers, you might be able to guess some of the following mathematical facts:

- the average length of a game should be A*B;
- A's probability of winning should be A/(A+B);
- no matter how the game begins, on average the players will end up even (wins and losses average out)

Even if A has \$1 and B has \$100, the fact is that A has a small chance of a big win, and B has a big chance of a small win, and these are equal.

It's time to go to Las Vegas and try some real gambling!

To keep things simple, let's look at a typical Las Vegas roulette.

There are 36 pockets numbered 1 to 36, half red and half black. There are also 0 and 00 pockets, which always win for the house.

Simple bets include choosing a number (which pays \$36 for a \$1 bet) or a color (which pays \$2 for a \$1 bet).

(When we say the payoff is \$36, we might instead say you win \$35 plus the \$1 back that you originally bet.)

These payoffs would actually be fair, except that anytime the wheel hits 0 or 00, the house always wins.

Results of 1,000 roulette spins, always betting \$10 on red:

Trial	36 pockets	37 pockets	38 pockets
1	-760	-540	-1,880
2	1660	-1,340	-2,080
3	-140	-1,540	-2,360
4	20	-1,320	-1,880
5	380	300	-1,880
6	1120	-660	-1,900
7	360	-2,380	-1,720
8	-60	120	-3,120
9	420	-1,520	-1,140
10	-960	-1,140	-2,980
-----	----	------	------
Total	-160	-2,720	-5,340

A "fair" roulette wheel would have 36 pockets, so that if you bet on 1 number, you would win $35 if you hit, and lose $1 if you missed. And if you bet on red, you would win $1 half the time and lose $1 half the time.

But on a 37 or 38 pocket roulette wheel, if a '0' or '00' turns up, you lose whether you bet red or black. This gives the casino an edge. How much of an edge would this be? We can experiment to get a feel for the difference that the extra pockets make.

In this table, we bet $10 on red 1000 times, and we do this 10 times. On the fair wheel, we actually come out ahead 6 times out of ten, and averaged a loss of $16 for 1,000 spins.

For the 37 pocket wheel, we only come out ahead twice, and our average loss is $272 in 1,000 spins.

On the 38 pocket wheel, we never come out ahead, and we about double our typical loss.

The two extra pockets make a huge difference in the long run!


SNAKES & LADDERS

The game of Snakes and Ladders may have originated in India.

The game is played on a 10x10 board of squares numbered 1 through 100.

Several players may compete, alternating turns.

To move, a player moves ahead as indicated by the roll of a die, with the following special cases:

- players essentially begin on square "0";
- exactly landing on 100 wins the game;
- if the move exceeds 100, some rules allow a win, others forfeit the move;
- landing on a snake moves the player backward;
- landing on a ladder moves the player ahead;

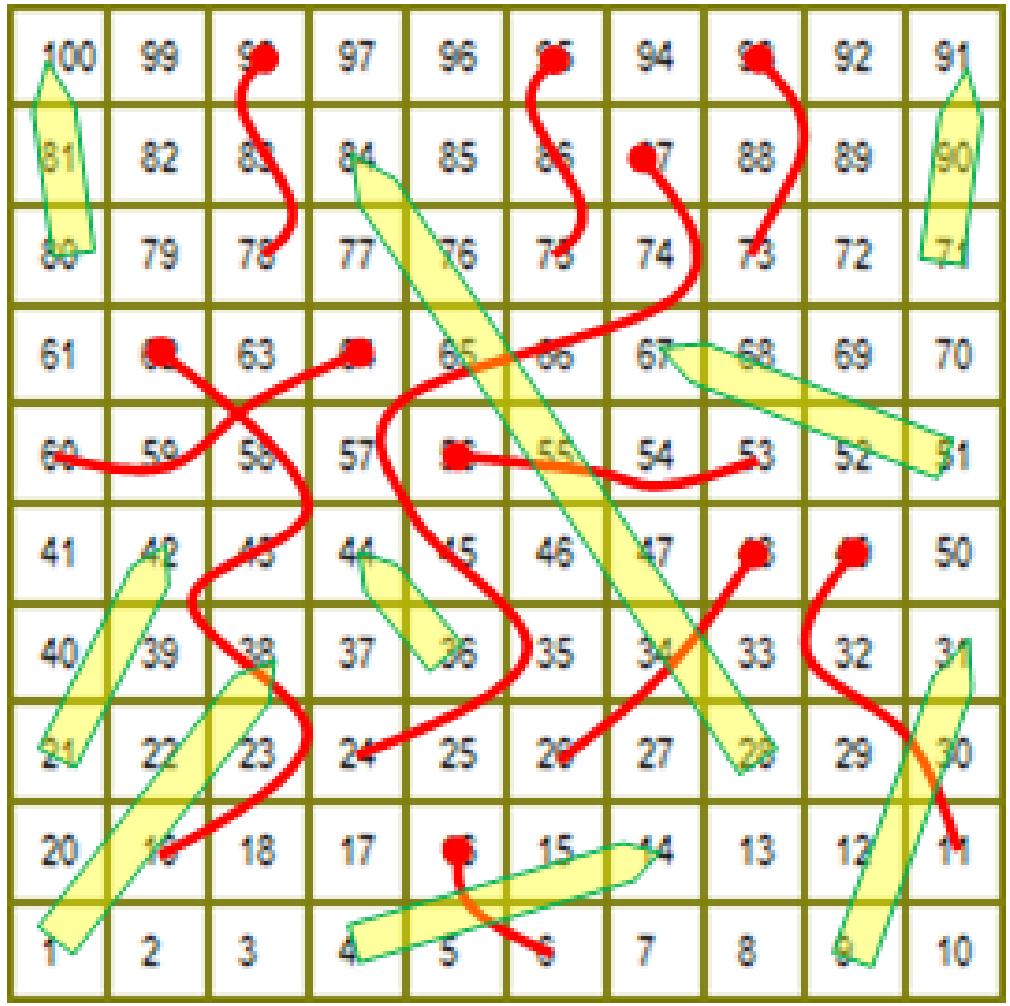

100 99 98 97 96 95 94 93 92 91
81 82 83 84 85 86 87 88 89 90
80 79 78 77 76 75 74 73 72 71
61 62 63 64 65 66 67 68 69 70
60 59 58 57 56 55 54 53 52 51
41 42 43 44 45 46 47 48 49 50
40 39 38 37 36 35 34 33 32 31
21 22 23 24 25 26 27 28 29 30
20 19 18 17 16 15 14 13 12 11
1 2 3 4 5 6 7 8 9 10

Snakes and Ladders is not a kind of gambling, but it is an elaborate process which involves random events (the dice).

Computer simulation is the use of computers to answer questions about physical or logical systems but making a model of the system and having the computer follow its rules.

We can use the computer to simulate a game of Snakes and Ladders. For simplicity, we'll only have one player, and the question we will ask is "How many turns will it take for the typical player to complete the game?"

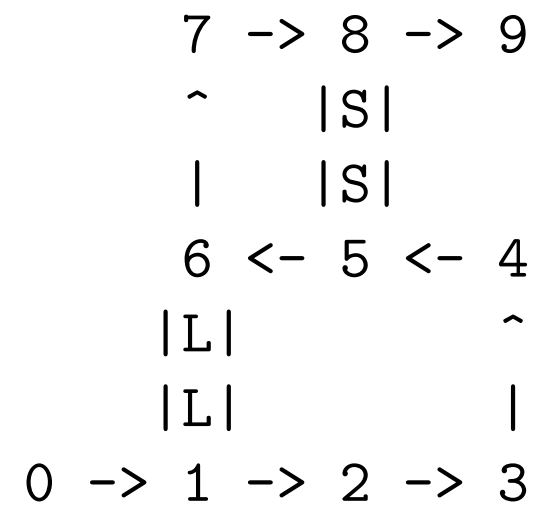
To answer that question, we'll have to make a model of the game, understand its rules, and keep track of the results of many games.


100 99 97 96 94 92 91
81 82 84 85 88 89 90
80 79 78 77 76 75 74 73 72
61 63 68 69 70
60 58 57 54 53 52
41 42 44 46 50
40 39 37 35 33 32
24 22 23 25 27 29
20 18 17 15 14 13 12 11
1 2 3 4 5 7 8 10

Looking closely at the board, we can see that our player can be in any one of 100 squares.

For convenience, we can add a square 0 (for "Start") and a square 101 (for "Finish).

Some squares are snakes and some are ladders. To model this, we just need to know if the square is a jump, and where such a square will jump to.



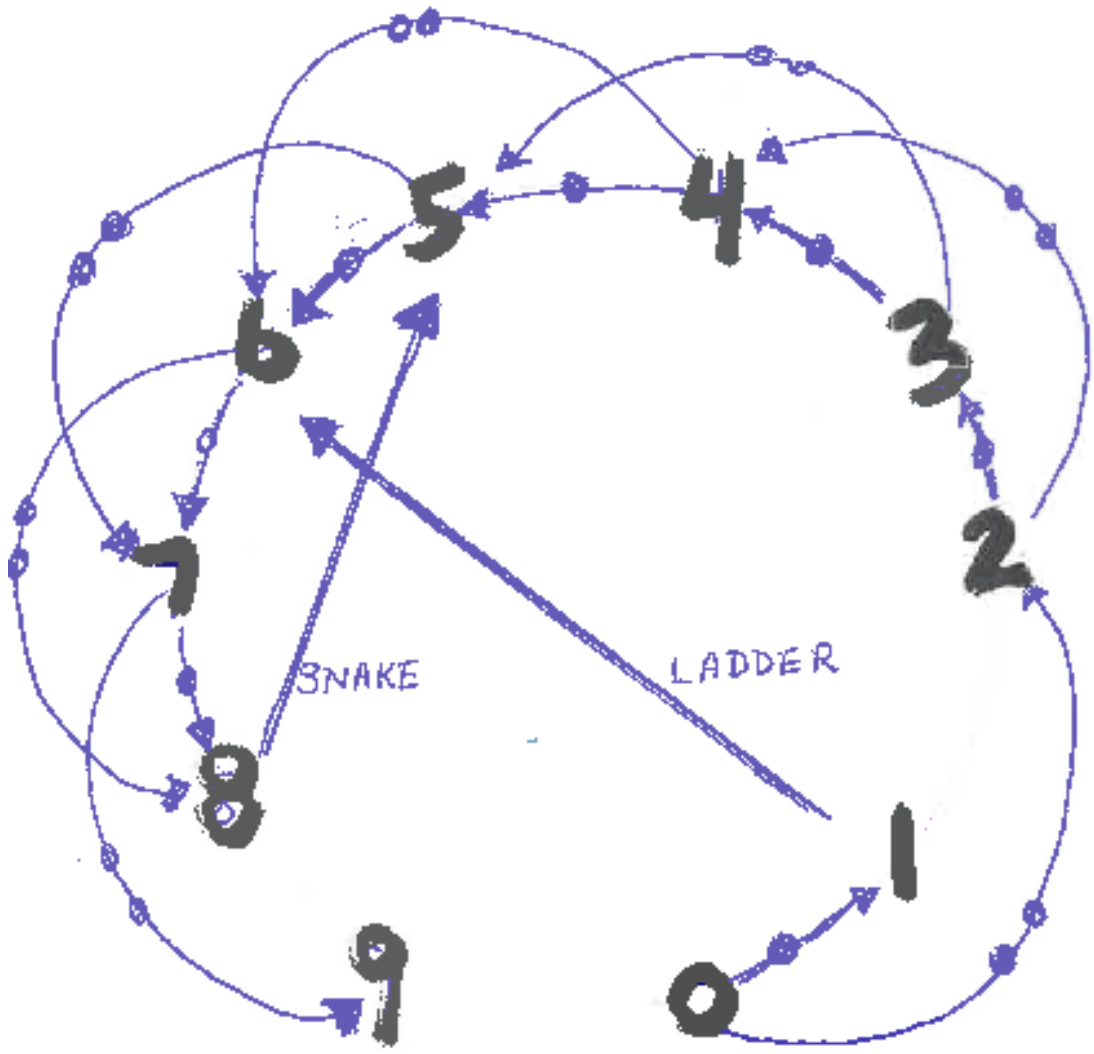
square	0	1	2	3	4	5	6	7	8	9
type?		L							S	
go to:	-	6	-	-	-	-	-	-	5	-

It might help to sketch a small version of our problem, where we only have a 3x3 board, and one ladder, from square 1 to square 6, and one snake, from square 8 back to 5.

To record where the snakes and ladders are, we can make a list of numbers.

```
          0 1 2 3 4 5 6 7 8 9      Ladder at 1 goes to 6,
type        L               S
connect  |0|6|2|3|4|5|6|7|5|9| <-- Snake at 8 goes to 5
```

and if you think about it, we don't really need to know whether a square is a snake or a ladder.


5
4
6
3
7
2
SNAKE
LADDER
8
1
9
0

Most dice have 6 sides, but for this tiny model, let's assume the die will only show a 1 or a 2.

Then we start at square 0, and roll the die repeatedly. This diagram shows, for each square, the snake, ladder or dice moves that are possible.

We can work out the least number of dice rolls necessary to end the game, but it's not clear what the longest game would be.

In fact, it is theoretically possible for the game to last forever, since we could do a loop like 5 -¿ 6 -¿ 7 -¿ 8 -¿ 5 over and over.

However, we can certainly estimate the average game length.

```
square = 0

if square is less than 9 then

  if square is snake or ladder then
    next square is other end of snake or ladder
  otherwise
    roll die for value of 1 or 2
    next square is ( square + die value )

  move from square to next square
```

Here is an idea of how we might simulate a single game of our tiny version of Snakes and Ladders.

We can easily include a line that keeps track of the number of dice rolls.

Then we repeat this procedure many times, and average that result.

This is what we will now do for the actual game of Snakes and Ladders.

Trial	Average	Shortest	Longest
1	39.8	7	175
2	39.2	7	185
3	38.7	7	158
4	39.5	7	205
5	38.5	7	187
6	38.3	7	198
7	39.5	8	207
8	38.8	7	176
9	39.9	7	185
10	39.0	7	242

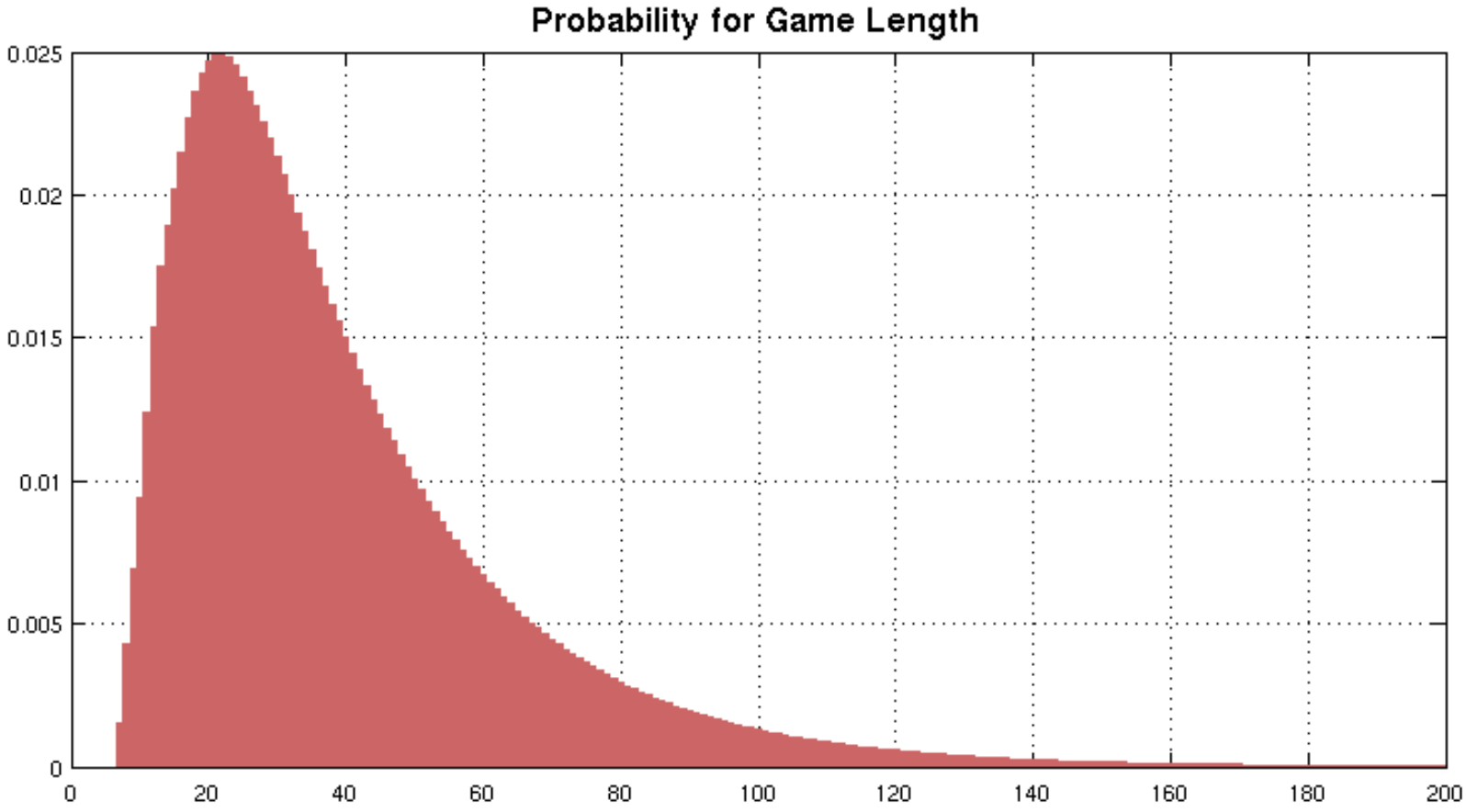
If we simulate 1,000 games, we can compute the average, shortest and longest games observed.

To feel comfortable with our results, we can do this 10 times, and see if our estimates change.

Note that the average and shortest seem reliable.

But the length of the longest game varies considerably. That is because it is always possible to have a very long game. But the longer the game, the more unlikely it is to actually happen. So these things are hard to observe.

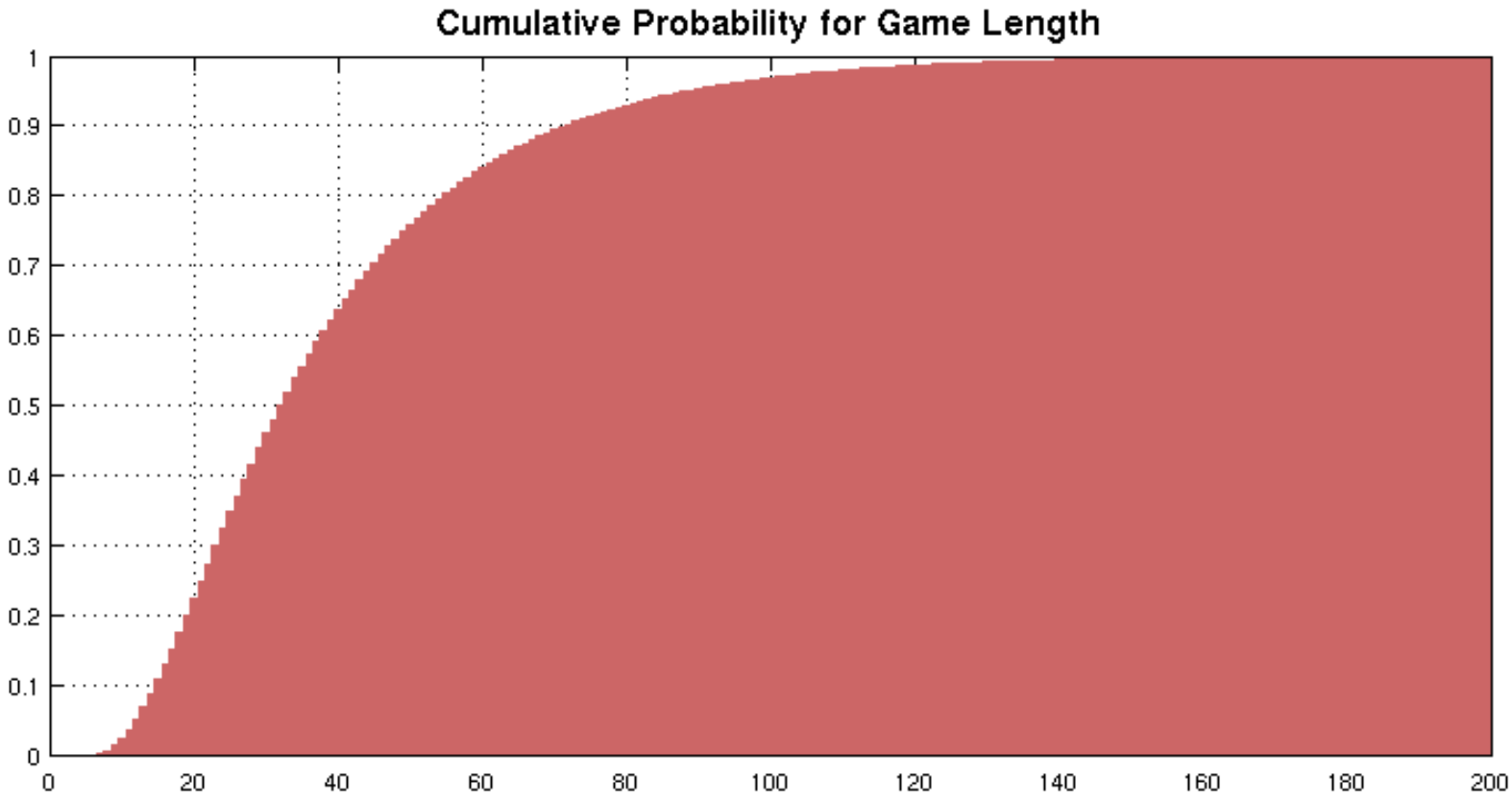
Very rare events are sometimes called "black swans", and they worry people who only have simulations to guide them.


Probability for Game Length
0.025
0.02
0.015
0.01
0.005
0
0
20
40
60
80
100
120
140
160
180
200

In fact, if we are more careful about keeping our records, we can estimate the probability that the game will take any number of moves.

That is, if we play the game 10,000 times, we can ask for the probability that the game will take exactly 39 moves to win (which is about the average number.) If we observe a 39 move game 247 times in our sample, we estimate the probability at $\frac{247}{10{,}000} \approx 0.025 = 2.5\%$ chance.

By converting frequency to relatively frequency, we estimate probability.


Cumulative Probability for Game Length
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20
40
60
80
100
120
140
160
180
200

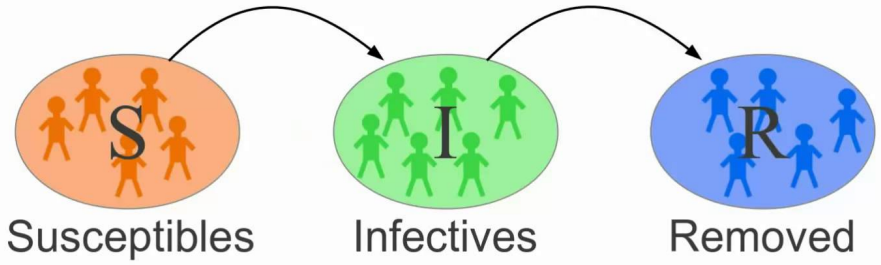
We know the average length of the game. We know the probability of any particular number of moves. But another useful piece of information is to be able to say the probability that the game will be done in at most a certain number of steps.

To estimate whether the game will be over in 39 steps or less, we again look at our results, and count all the games that take 39 steps or less and divide by the total number of games played.

As the number of steps increases, this value must steadily increase as well.

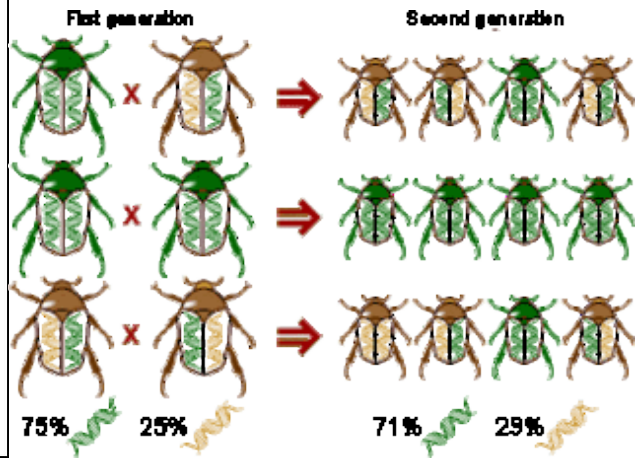
Now, by looking at the graph, we can answer the question "How soon will 90% of all the games be done?" ... looks like 70 moves.

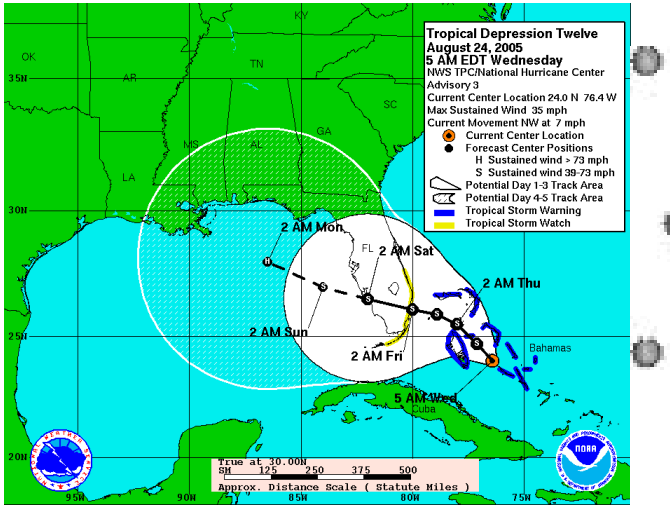
Determine dynamics of the SIR Model

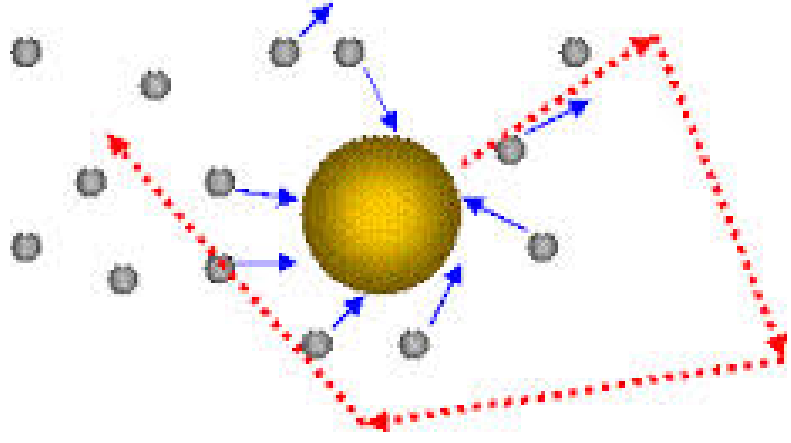


We must determine:

- The rate at which susceptible individuals get infected.
- The rate at which infected individuals recover (or die).



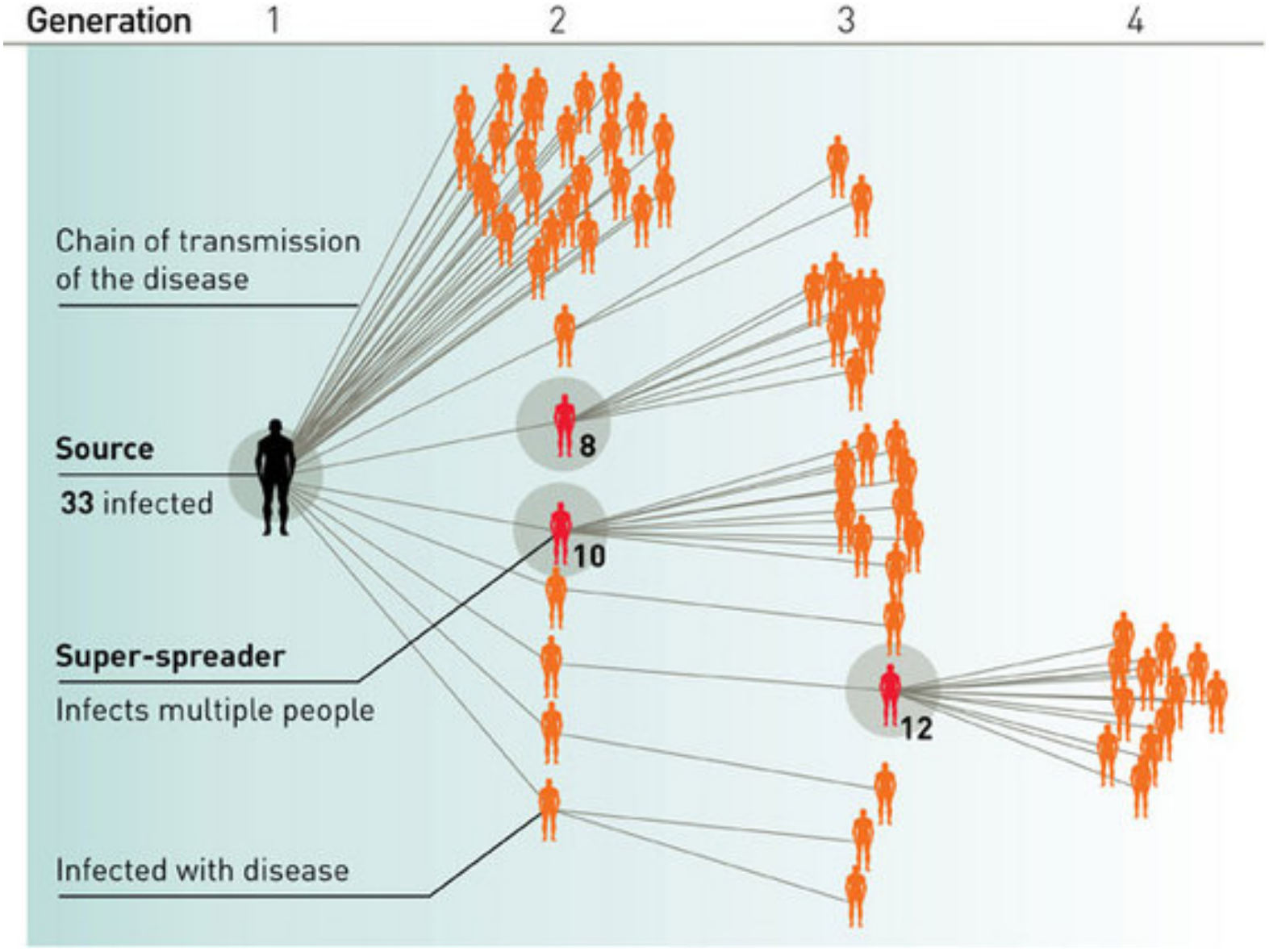




We looked at the game of Snakes and Ladders because it was a simple example with clear rules and some questions that made sense to ask.

The point is, though, that the very same ways of looking and analyzing a problem are used when a computational scientist tries to make a model of

- the spread of an epidemic disease;
- the mixing of genes in successive generations;
- the forecast of a hurricane's path;
- the random motion of pollen particles suspended in the air;
- the annual number and severity of earthquakes;
- how a rumor can spread or disappear over social media

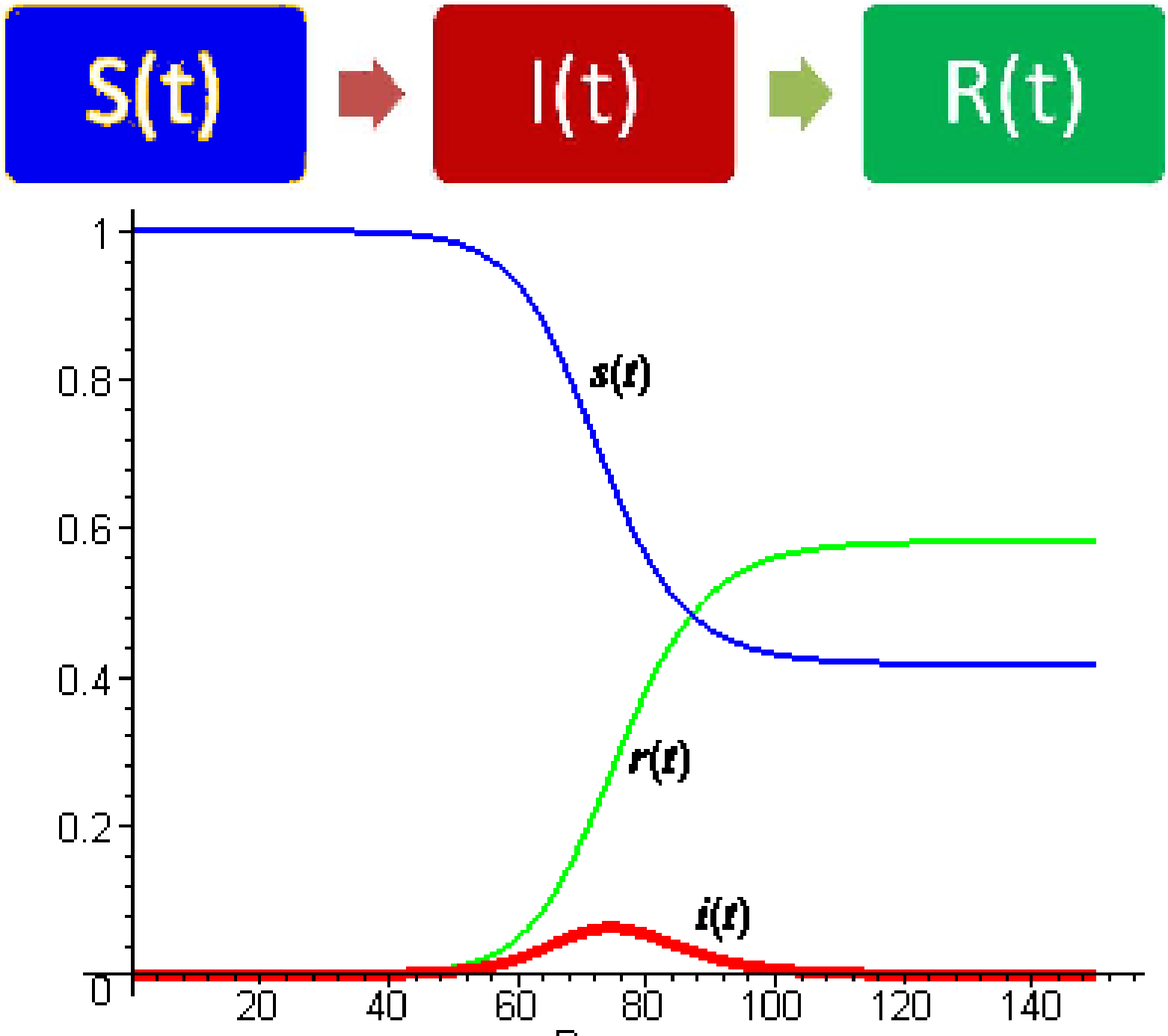

Generation
1
2
3
4
Chain of transmission of the disease
Source
33 infected
Super-spreader
Infects multiple people
Infected with disease
8
10
12

The transmission of disease is still a mystery.

If people are exposed to a sick person, some will also get sick. We don't know which people will get sick, we don't know why they get sick. The only thing we can do is observe, over and over again, the probability that a person will catch the disease after exposure.

Since we don't have rules, only probabilities, we can't predict the exact extent of a disease epidemic. But we can certainly use our probabilities to make simulations, which give us at least an idea of typical ways an epidemic might behave over time.

If we have a good estimate for the probabilities and other factors in disease transmission, we can use computer simulations to investigate the best strategies for controlling an epidemic.


S(t)
I(t)
R(t)
1
0.8
0.6
0.4
0.2
0
20
40
60
80
100
120
140
s(t)
r(t)
i(t)

The **SIR** model is one of the simplest disease transmission models.

It divides the populuation into:

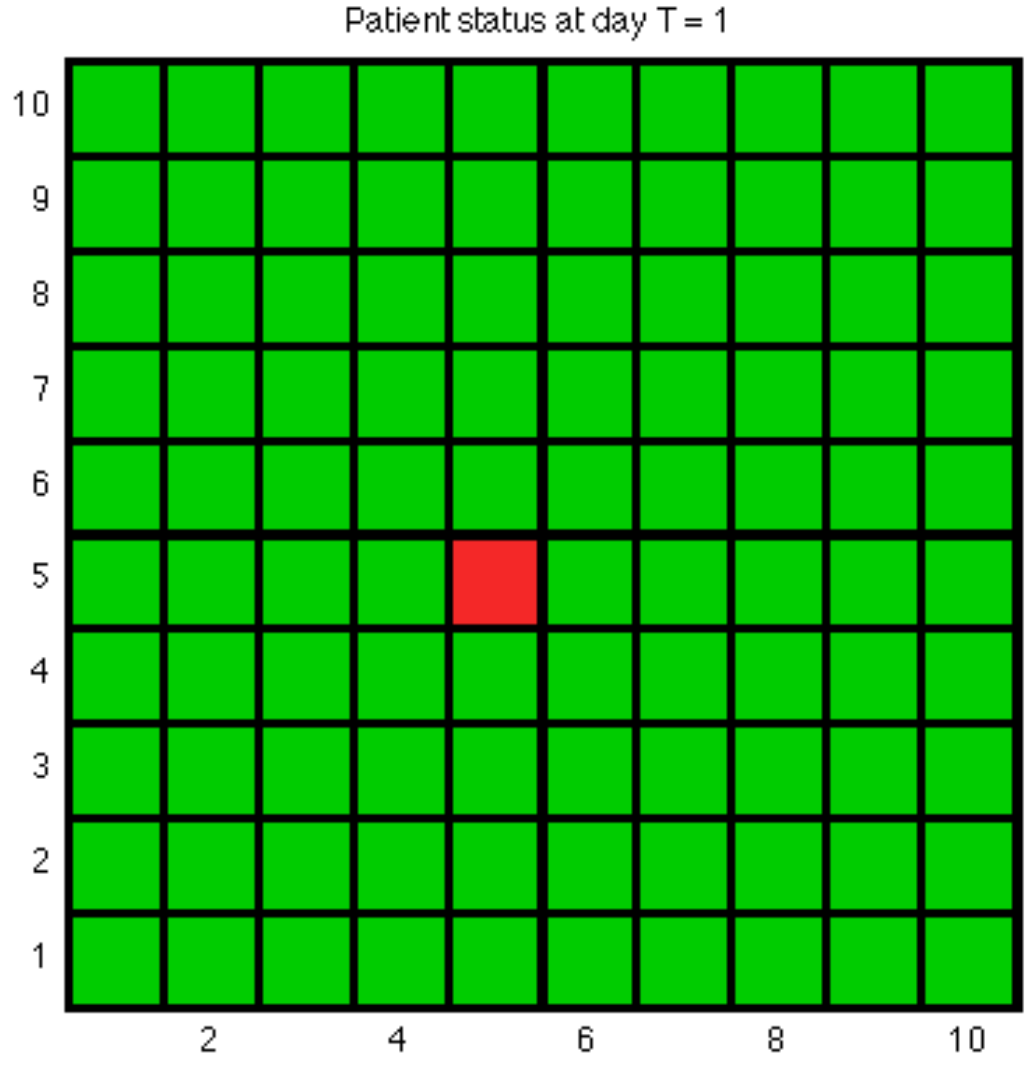
- **Susceptibles**, well people who can catch the disease;
- **Infecteds**, sick people who can spread the disease;
- **Recovereds**, people who had the disease, but recovered.

This is a reasonable model for diseases you only get once, such as measles and chicken pox.

```
repeat this from day 1 to day 50

  consider each person P

    if P is susceptible
      roll die to decide if P becomes infected
    otherwise, if P is infected
      and P has been sick for at least 4 days, P now rec
```

So that we have something to work with, let's make some choices.

- We run our simulation for 100 days;
- We start with a group of healthy people.
- We assume every healthy person has the same chance of getting sick on a given day;
- We assume every sick person gets better after 4 days;
- Our simulated day begins
- We consider the chance that each healthy person get sick;
- For each infected person, if they've been sick 4 days, they get better.
- We keep track of the days, the number of S, I and R people.
- We may ask whether everyone eventually got sick.


Patient status at day T = 1
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10

Our original idea is simply that everyone has the same chance of catching the disease. But we know very well that the chance of catching a disease is very low if you avoid sick people.

Let's look at a model that includes this idea of nearness and disease transmission.

Let's imagine a hospital with one huge room with 100 beds. People are in the hospital waiting to have babies, to get bones set, to have burns bandaged, and so on.

But now let's suppose exactly one patient has a new disease.

Then let's assume that healthy people in the hospital can only catch the disease if they are in a bed that is next to a sick person.

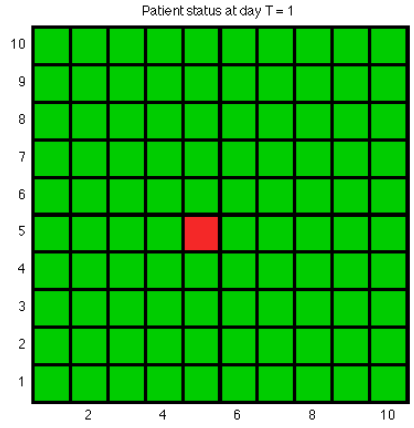
The other rules are the same:

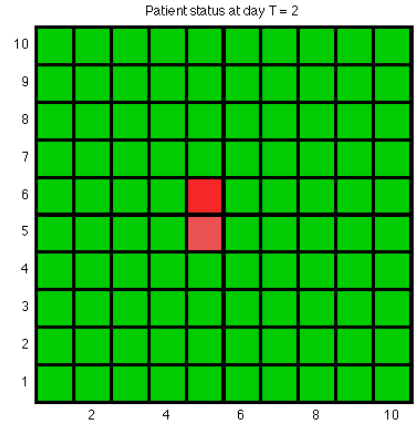
- a healthy person near a sick person has a probability of getting sick;
- a sick person gets better after 4 days.
- a recovered person can't get sick again.

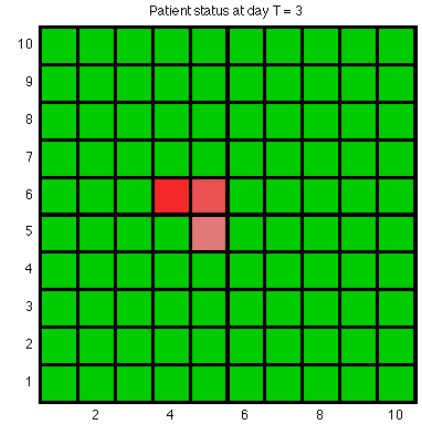
```
repeat this from day 1 to day 50

  consider each person P

    if P is susceptible
      if P has a sick neighbor (north, south, east, west)
        roll die to decide if P becomes infected
    otherwise, if P is infected
      and P has been sick at least 4 days, P recovers
```

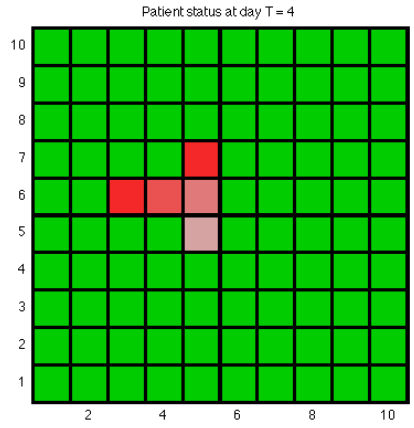
Now, in order to use the disease model with our hospital bed case, we simply have to check whether a susceptible person is actually in a bed directly next to a sick person.

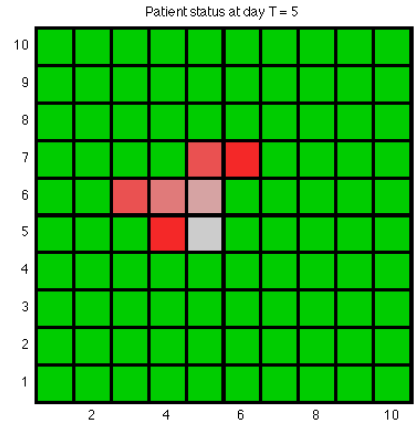
You can imagine that if a susceptible person had two or more sick neighbors, we might expect the odds of catching the disease to increase. We won't worry about that right now, but it's the kind of thing you can do to try to improve your model.

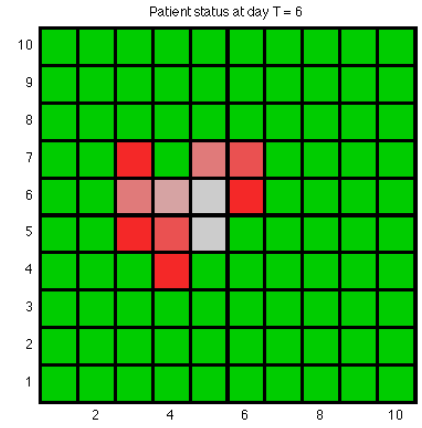

Patient status at day T = 1
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10


Patient status at day T = 2
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10


Patient status at day T = 3
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10

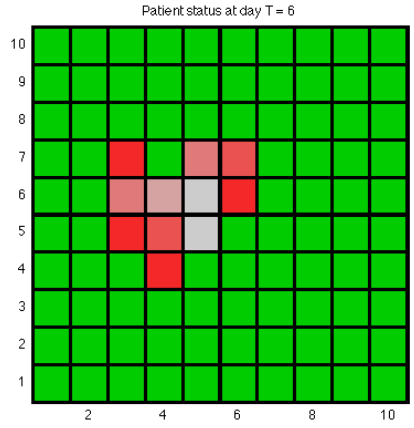

Patient status at day T = 4
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10

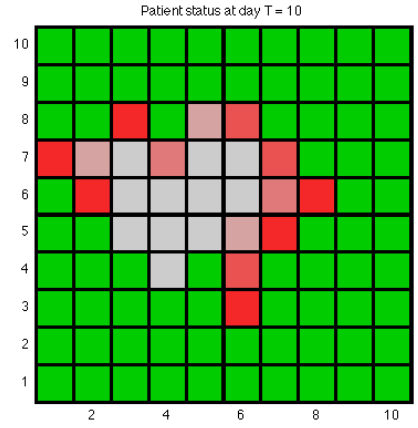

Patient status at day T = 5
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10

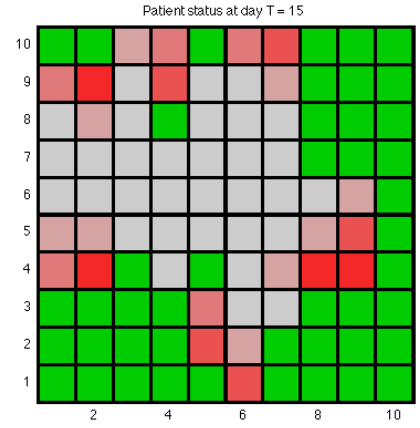

Patient status at day T = 6
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10

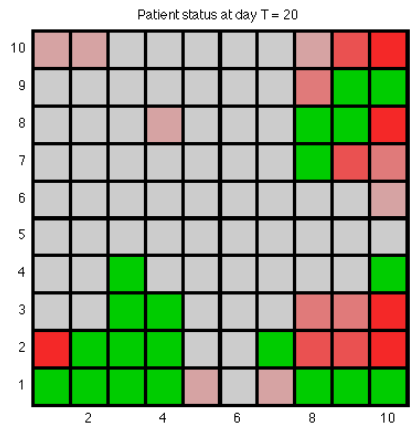
Here are the first 6 days of the disease.

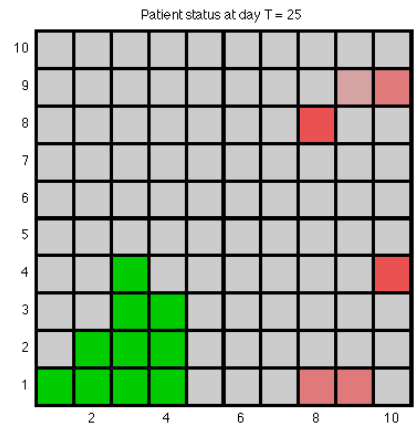
You can see how it spreads a bit like a fire, and the gray area that shows up in the center is the “recovered” patients, something like the burned out area in the center of a fire.

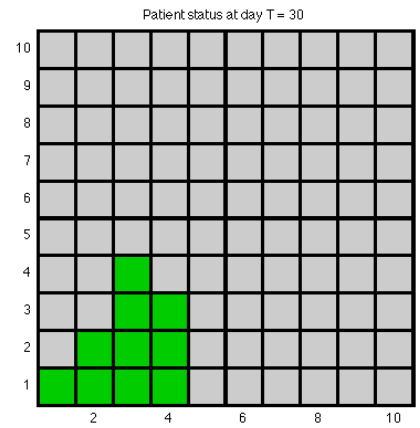

Patient status at day T = 6
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10


Patient status at day T = 10
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10


Patient status at day T = 15
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10

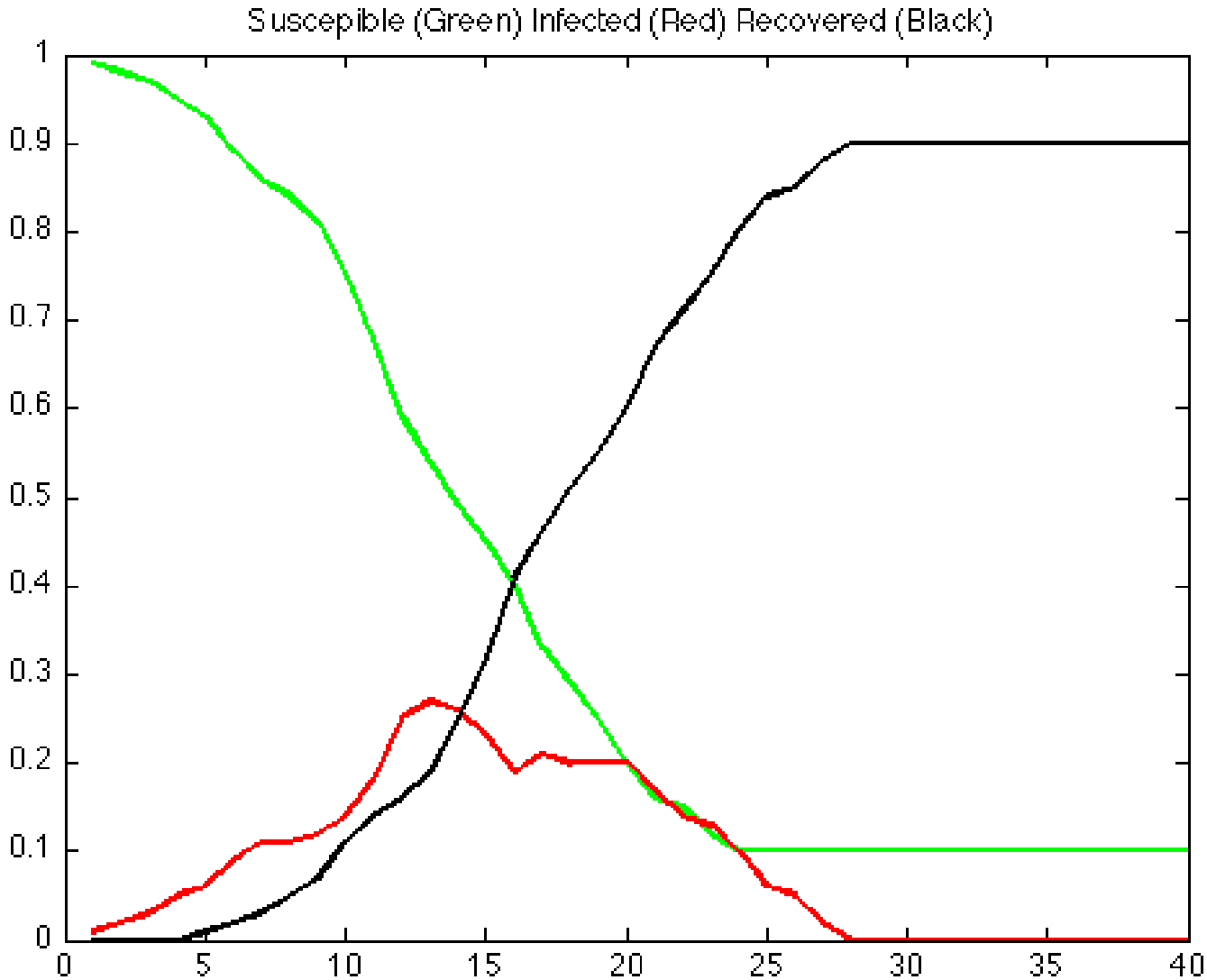

Patient status at day T = 20
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10


Patient status at day T = 25
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10


Patient status at day T = 30
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10

Now we see days 6, 10, 15, 20, 25, and 30.

The disease seems to be progressing very well, but in the lower left corner, it dies out before catching everyone.


Suscepible (Green) Infected (Red) Recovered (Black)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40

We can make day-to-day plots of the number S, I, and R patients.

If everyone got sick at some point, then at the end of the plot, if the disease has finished spreading, we'd have no susceptibles, no infecteds, but only recovereds.

In this example, we managed to spare a few patients from the disease.

Using the computer model, we could investigate whether the disease rate would be affected by such strategies as spreading the beds further apart, or changing the shape of the room.


$900
$1000
$300
$500
BANKRUPT
ONE
MILLION
BANKRUPT
$550
$400
$
BerryFactory.com

Chance is involved in many games, social situations, business propositions, and physical occurrences.

If a random event is repeated often enough, we may be able to count the number of times each possible outcome occurs. Then we may estimate the probability of each outcome, even though we can't predict exactly what will happen on the next try.

Using probabilities, we can simulate situations involving random events.

This can allow us to understand the typical behavior to be expected.

A computer is ideal for such studies, since these simulations should be done many times for best results, and may involve a lot of record keeping.