Open Optimization **Course Notes**

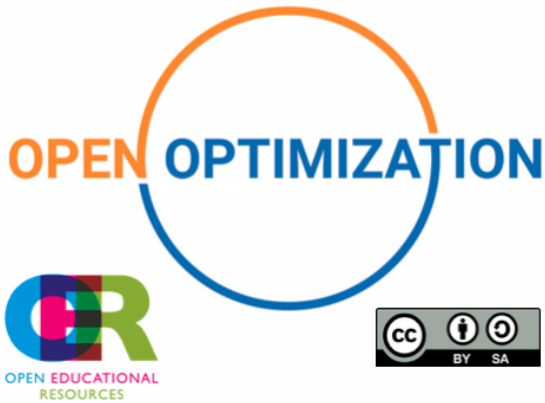
A Brief Introduction to Simulation

(Work in Progress)

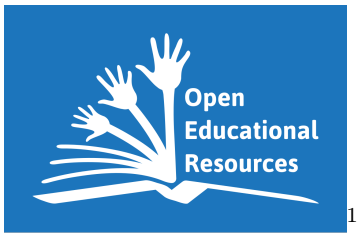
Timothy Yusun
Department of Mathematical and Computational Sciences
University of Toronto Mississauga

Tamon Stephen
Department of Mathematics
Simon Fraser University

Version 0.0 (January 2020)



This work is done in alignment with the mission of UNESCO Open Educational Resources `https://en.unesco.org/themes/building-knowledge-societies/oer`



[1]

The source code of this book is available from `https://github.com/open-optimization/`

[1] `https://en.wikipedia.org/wiki/Open_educational_resources#/media/File:Global_Open_Educational_Resources_Logo.svg`

Preface

These course notes were originally written to support the course Math 208W (Introduction to Operations Research) at Simon Fraser University (British Columbia, Canada). They are designed to for students with minimal mathematical background to get a hands-on taste of Operations Research, a branch of mathematics which treats large-scale decision problems.

The focus of the course is on exploring such problems, which are too big for analysis via pen-and-paper methods: indeed, simply collecting and managing the data are a big part of the challenge. The first part of the course studies deterministic models, based on linear programming and extensions, following Baker's *Optimization Modeling with Spreadsheets* [1]. This book suits the purpose quite well, as it assumes little in terms of prerequisites and provides many worked examples and exercises, including data files on a supporting Website. The examples and exercises are written in `Microsoft Excel`, which is ubiquitous in the user community, and a skill that Operations Researchers hoping to make an impact in practice should acquire.

For the second part of the course, we move on to non-deterministic models, that incorporate an element of *simulation*. These notes are designed to support this part of the course. The notes continue in the style of Baker, with light prerequisites (Introductory Calculus) and extensive, `Excel`-based examples and exercises, supported by data files. The aim is not a rigorous presentation, but rather an exposure to the basic ideas of simulation and an invitation to experiment.

Chapter 1 begins the discussion with an introduction to random numbers and how to generate them. Chapter 2 then shows how a simple random source can be used to generate different types of distributions that may be useful in simulations. Chapter 3 proceeds to use these distributions to build some very basic simulations involving coin-flipping and random walks, while Chapter 4 builds on this to examine simple queues and discrete-event simulations.

There are several texts that introduce simulation. We mention some which are good sources for further reading, though they often require more extensive mathematical prerequisites. Hillier and Liebermann [6] provide an extensive introduction to Operations Research, including a chapter on simulation. Simulation textbooks include Bertsimas and Freund [4], Banks et al. [2], Guiasu [5], Law [9], Leemis and Park [10] and Parlar [12].

This work was started with support from a Teaching and Learning Development Grant (TLDG) through the Institute for the Study of Teaching and Learning in the Disciplines (ISTLD) at Simon Fraser University. We thank the ISTLD staff who helped with the project, including Cheryl Amundsen and Laura D'Amico for encouragement and discussion, and the other participants in our workshop for feedback.

We thank Ismael Martinez and the students who took Math 208W *Introduction to Operations Research* at SFU for 2017-2019 for their comments and suggestions.

These notes are a work in progress, and comments, corrections and suggestions are encouraged. The authors' contact details are available on their Webpages.[2]

[2] `https://www.utm.utoronto.ca/math-cs-stats/faculty-staff/yusun-dr-timothy` and `http://people.math.sfu.ca/~tamon/`.

Contents

1 Randomness and Random Numbers

> The generation of random numbers is too important to be left to chance.
>
> Robert R. Coveyou
> Oak Ridge National Laboratory, 1969.

Randomness is unpredictability, or a lack of any discernible pattern.

Consider the experiment of flipping a coin – there are two possible outcomes, heads or tails. This is a *random experiment*, since we cannot predict the outcome. Moreover a coin flip is not reproducible as an individual event; however it is still possible to understand the relative frequencies of the possible outcomes by repeating this experiment multiple times. Flipping a fair coin 1000 times, for instance, will give roughly 500 heads and 500 tails.

With a view towards modelling these random experiments, first we discuss random numbers and how to generate them.

Question: Would you characterize the sequence of digits of π as *random*?

$$\pi = 3.14159265358979323846\cdots$$

This sequence looks random, in the sense that these digits do not seem to follow any prescribed pattern. On the other hand, π is a fixed number, and any digit of this sequence can be calculated given sufficient resources. We call such a sequence *pseudorandom*. This means that the process that outputs digits of π in sequence *appears random*, statistically speaking, while being completely deterministic.

In fact, the following process can be used as a [bad] algorithm for generating random numbers from the set $\{0, 1, \ldots, 9\}$:

Algorithm 1: A simple random number generator

initialize i a non-negative integer;
function rand():
    $i \leftarrow i + 1$;
    return ith digit of π ;

One nice property of this is algorithm is that as long as we know the initial index i, we can repeat exactly the sequence of digits it outputs. This is called the *seed* of the generator.

This is especially useful for running computer simulations, as sometimes we would like to reproduce results of simulating a random process. Note that we would not actually use Algorithm 1 in a simulation, due to a large part in the computational effort required to compute digits of π. Moreover, patterns have been found in its digits[1]. Another simple example is Algorithm 2.

Algorithm 2: The Linear Congruential Method

initialize integers a, m, c appropriately selected, a seed x_0, $i = 0$;
function rand():
    $i \leftarrow i + 1$;
    $x_i = (ax_{i-1} + c) \pmod{m}$;
    return x_i ;

The mod here is the *modulo operation* – it computes the remainder when $(ax_{i-1} + c)$ is divided by m. This means each x_i will be in the set $\{0, 1, 2, \ldots, m-1\}$. As in Algorithm 1, Algorithm 2 is very easy to replicate, in this case by simply knowing the parameters a, m, c, and the seed x_0.

There are certain conditions that these parameters have to satisfy in order to make this algorithm viable. One desirable property of the produced sequence is a long period, that is, that the algorithm goes through as many values from the set $\{0, 1, 2, \ldots, m-1\}$ as possible. For instance, if $c = 0$ and m is prime, the algorithm generates the longest possible period with length $m - 1$ for many values of a.[2] This algorithm can be modified to output real numbers in $(0, 1)$, by dividing the outputs by m.

Example 1.1 (The Linear Congruential Method)**.** Using $a = 6$, $m = 11$, $c = 0$, and $x_0 = 1$, we get the sequence $1, 6, 3, 7, 9, 10, 5, 8, 4, 2, \ldots$ which goes through all values in $\{1, 2, \ldots, 10\}$ before repeating.

If we use the same parameters but with $m = 10$ instead, we get $1, 6, 6, 6, \ldots$, reminding us that we need to pay attention to m, a, and c.

This example illustrates the operations performed. Observe that as in Algorithm 1, the output of this algorithm is completely deterministic. It also becomes more predictable the more numbers you've generated, since you do not repeat x_i's before the period ends (and then after that, the same sequence is produced repeatedly).

In practice however, m is chosen to be very large, and if the other parameters are chosen correctly, the sequence produced will look no different than a random sequence (in the sense that it passes known statistical tests that check for this property).

There are more sophisticated algorithms for random number generation, based on similar ideas to Algorithm 2. One popular one is called the *Mersenne Twister*, and which is

[1] See `http://mathworld.wolfram.com/PiDigits.html`

[2] This holds if a is chosen so that the smallest integer k such that $a^k - 1$ is divisible by m is $k = m - 1$. Such an a is called a *primitive root modulo* m – see [9].

used by most computational software, including MATLAB, Excel, C++, and others. For running simulations on a computer, the primary concern in this course is knowing how to use a computer to generate random numbers (although it is helpful understanding how it is done in the background).

1.1 Generating Random Numbers in Excel

In Microsoft Excel, the `=RAND()` command outputs a random real number[3] in the interval $[0, 1)$. Whenever the worksheet is re-calculated (by editing a cell, for example), a new real number is returned in this interval.

This function can be in turn used to generate a random real number in any interval $[a, b)$, using the formula `= a + (b - a)*RAND()`. Intuitively, this transformation computes the number in the interval $[a, b)$ that is `RAND()` fraction of the way from a to b.

	A	B
1	0.344538338	= RAND()
2	11	a
3	17	b
4	13.06723003	= A4 + (A5 - A4)*A1

Figure 1.1: Computing random real numbers in Excel

Note that this process is *not* reproducible; once the worksheet is refreshed, each `RAND()` call is reset, and the old values cannot be retrieved. There are two ways of fixing the values generated – one is to copy and paste these in a separate worksheet; another way is to press F9 after entering the formula instead of Enter. This makes it so that recalculations are not done when the worksheet is refreshed, and the formula in the cell is overwritten by the computed values.

The type of randomness we discuss in this section is called *uniform*, since numbers from the set under consideration are all *equally likely* to be generated. We also call this a *uniform distribution* of numbers.

One way of visualizing the random numbers we are generating is by using *histograms*, which are graphs with rectangles or bars representing the frequency of each range of values. In Figure 1.2 we show the result of generating 100 real numbers using `RAND()`, then grouping results into 'buckets' $[0, 0.1), [0.1, 0.2), \ldots, [0.9, 1)$.

Note that there are 17 numbers in $[0.1, 0.2)$ while only 5 in $[0.8, 0.9)$. This is not unusual – what we would expect the average to be when repeating an experiment multiple times does not necessarily reveal itself in only a few trials. Observe that the histogram in Figure 1.3, where we generate a thousand real numbers using `RAND()`, looks more 'evened out.'

[3] In computing, real numbers are represented as *floating point numbers* – see `https://en.wikipedia.org/wiki/Floating_point`.

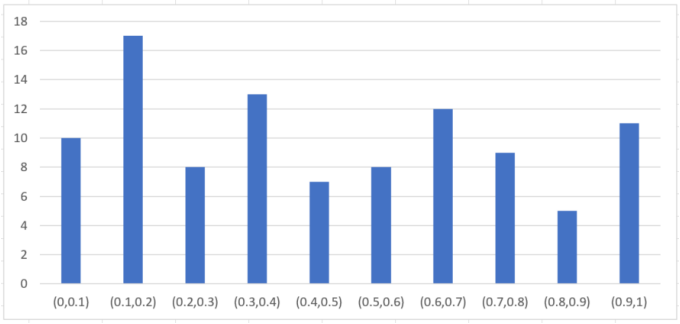


Figure 1.2: A histogram of one hundred `RAND()` outputs in Excel

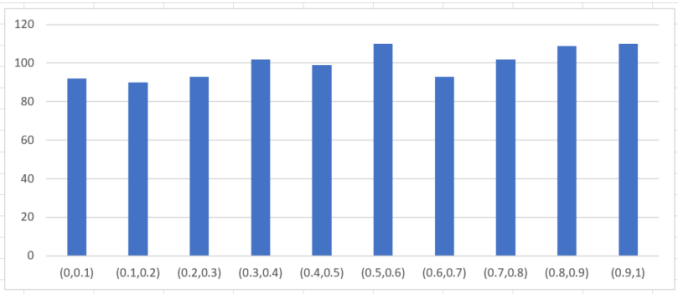


Figure 1.3: A histogram of one thousand `RAND()` outputs in Excel

In the next section, we will introduce other types of distributions, where some outcomes are more likely than others. From this point we will use the term *random number* to describe a random real number from $[0, 1)$, and the term *random variable* to describe an output that may be generated from any general probability distribution.

EXERCISES

1.1 **Explore.** Write a sequence of H's and T's to describe what you think will happen if you flip a coin twenty times. How about fifty? Then actually perform the experiments, and compare the two results. What observations can you make?

1.2 An urn contains ten balls - three blue, two green, and five red.

(a) If you pick a ball at random, what are the chances that the ball is blue? red?

(b) If you pick two balls in succession (without putting back the first ball), what are the chances that the two balls are green?

1.3 What are the chances that you roll a die twice and you never see the number 1? How about three times?

1.4 When we take $c = 0$ in Algorithm 2, it is known as the *Multiplicative Congruence Method*, also called the *Lehmer random number generator* after the mathematician D. H. Lehmer. A classical choice for its parameters is $m = 2^{31} - 1 = 2147483647$

and $a = 7^5 = 16807$, and this generator has the special name `MINSTD`. Using Excel, generate the first ten terms of `MINSTD` when the initial seed is your SFU student ID number. (Divide each term by m so that each output lies in $[0, 1)$.)

1.5 Write a formula in Excel to generate a number uniformly from the following sets:

(a) The interval $[0, 100)$.

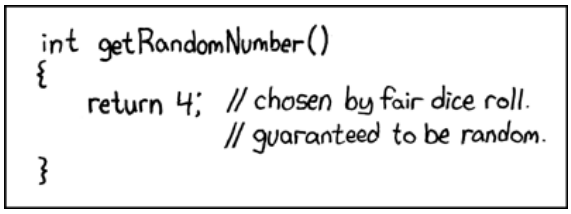
(b) The interval $(-20, -15]$.

(c) The set of integers in the interval $[-20, -15]$.

1.6 Write a formula in Excel that simulates the process of picking a ball at random from the urn in Exercise 1.2

1.7 **Explore.** Replicate the process used in Figure 1.2 and Figure 1.3 but using different bucket sizes - 0.2 and 0.05. That is, generate one hundred, then one thousand random numbers from $[0, 1)$, and then aggregate the results using (a) 5 buckets of equal length; (b) 20 buckets of equal length. What do you think the effect of changing the bucket size will be? Compare with the results of your experiment.

1.8 Explain the following `xkcd` comic:



Source: `https://xkcd.com/221/`

2 Probability Distributions

> One sees... that the theory of probabilities is basically just common sense reduced to calculus.
>
> Pierre-Simon Laplace
> *Philosophical Essay on Probabilities*, 1814

2.1 The Discrete Uniform Distribution, PDFs and CDFs

In this section we consider probability distributions and how to sample them on a computer.[1] The **probability** of an event is a measure of how likely that event will occur, it can be understood as the ratio of the number of ways it can occur to the entire space of possible outcomes. **Probability distributions** contain information about the probabilities of different possible outcomes in an experiment. In the next example we define two important probability functions.

Example 2.1. Consider the experiment of rolling a fair six-sided die with the numbers 1 to 6 on its faces. Each face has an equal chance of coming up when the die is rolled – so the probability that we get the number 1 (e.g.) is $\frac{1}{6}$. Using X to denote the random variable representing the outcome of this experiment, we can state this as $P(X = 1) = \frac{1}{6}$. Similarly, $P(X = 2) = \frac{1}{6} = P(X = 3) = \cdots$.

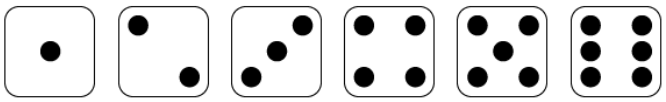


Figure 2.1: Each outcome has equal probability $= \frac{1}{6}$.

The probability distribution for X assigns a probability of $\frac{1}{6}$ to each of the six possible outcomes – this is the *uniform distribution* from Section 1 applied to the integers $\{1, 2, \ldots, 6\}$. We can capture this information concisely using a **probability density function**, or a **pdf**. This is a function $f(k)$ that takes as input the outcome k, and outputs the probability that $X = k$. That is, the pdf for the random variable X is

$$f(k) = P(X = k) = \begin{cases} \frac{1}{6}, & k = 1, 2, \ldots, 6 \\ 0, & \text{otherwise.} \end{cases}$$

[1] For further reading on probability distributions, see [8].

Recall from Section 1 that we call this a uniform distribution, since each outcome is equally likely. In particular, X is said to have a **discrete uniform distribution**, since there are only a finite number of possible outcomes.

We also define the **cumulative distribution function** of a random variable, also called its **cdf**. As the name suggests, this function $g(k)$ takes as input an outcome k, and outputs the probability that $X \leq k$; that is, $g(k) = P(X \leq k)$. For this example, the cdf of X is

$$g(k) = P(X \leq k) = \frac{\lfloor k \rfloor}{6}, \ k = 1, 2, \ldots, 6. \quad \square$$

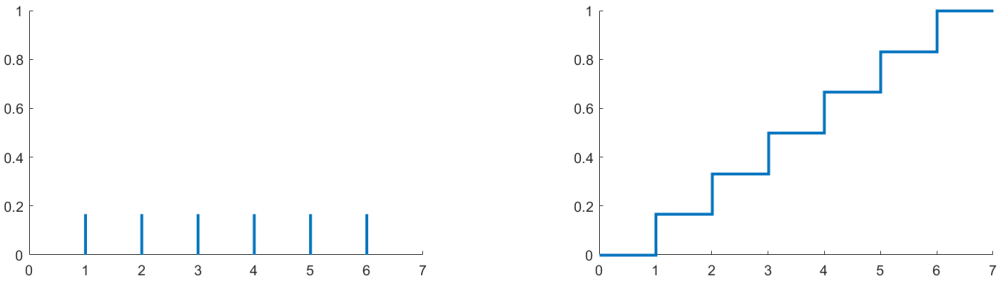


Figure 2.2: The pdf (L) and cdf (R) of X

In general, given a finite set of outcomes $S = \{a, a+1, \ldots, b\}$, with $n = |S| = b - a + 1$, the discrete uniform distribution over this set has pdf

$$f(k) = \begin{cases} \frac{1}{n}, & k \in S \\ 0, & \text{otherwise.} \end{cases}$$

while its cdf is

$$g(k) = \begin{cases} 0, & k < a \\ \lfloor \frac{k-a+1}{b-a+1} \rfloor, & a \leq k \leq b \\ 1, & k > b \end{cases}$$

If a random variable X has such a distribution, we denote it by $X \sim U\{a, b\}$.

To sample from the discrete uniform distribution $U\{a, b\}$ in Excel, we generate a random number using `RAND()`, then we divide the interval $[0, 1)$ into an subintervals of equal length. Recall Figure 1.3, which illustrated a thousand `RAND()` outputs in Excel, aggregated into intervals of width 0.1. This is equivalent to sampling from $U\{1, 10\}$ a thousand times, mapping intervals $\left[\frac{i-1}{10}, \frac{i}{10}\right)$ to the integer i. In Figure 2.3 we show the results of generating 10, 100, 1000, and 10000 random integers from $\{1, 2, \ldots, 10\}$.

Excel has a built in function which does this transformation for you; `RANDBETWEEN(a,b)` generates a uniform random integer between integers a and b (inclusive).

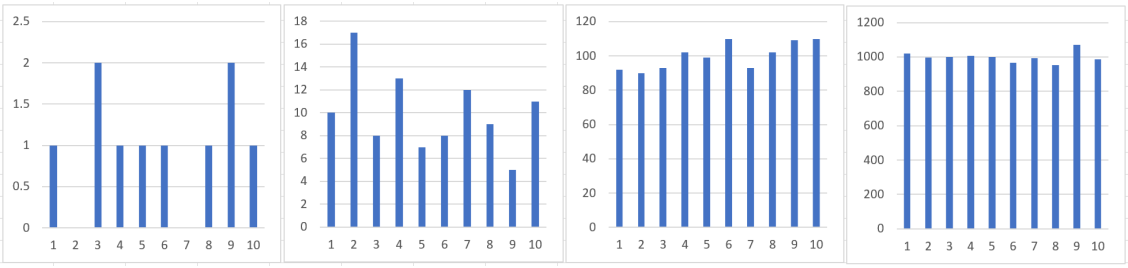


Figure 2.3: Generating 10, 100, 1000, and 10000 values from $U\{1, 10\}$

2.2 The Bernoulli and Binomial Distributions

One of the most basic distributions is the **Bernoulli distribution**, where there are only two possibilities: *success* or *failure*, with a parameter p describing probability of success, and a corresponding probability of $q = 1-p$ for failure. Because there are only two possible outcomes, this is a discrete distribution. Note that for $p = 0.5$ this is the discrete uniform distribution $U\{0, 1\}$, and simulates a fair coin flip; for other values of p it simulates an unfair coin flip. We denote a Bernoulli-distributed random variable with parameter p as $B(p)$.

The pdf of $B(p)$ is

$$f(k) = \begin{cases} 1-p, & k = 0 \\ p, & k = 1 \\ 0, & \text{otherwise} \end{cases}$$

while its cdf is

$$g(k) = \begin{cases} 0, & k < 0 \\ 1-p, & 0 \leq k < 1 \\ 1, & k \geq 1 \end{cases}$$

Figure 2.4: The pdf (L) and cdf (R) of the Bernoulli distribution with $p = 0.6$

To simulate this in Excel we use `RAND()` as well. If u was the result of `RAND()`, then we output

$$\begin{cases} \text{success}, & \text{if } 0 \leq u < p \\ \text{failure}, & \text{if } p \leq u < 1 \end{cases}.$$

If we perform a Bernoulli trial multiple times, we get another distribution – the **binomial distribution**, which counts the number of successes in n independent Bernoulli trials. Hence a binomially-distributed variable has two parameters - p the probability of success for each trial, and n the number of trials. We denote this by $B(n, p)$ (so the Bernoulli distribution can also be written as $B(1, p)$).[2]

Example 2.2. Consider the experiment of flipping two fair coins and counting the number of heads that come up. This is a binomial random variable, with parameters $n = 2$ and $p = 0.5$. There are three possible outcomes - zero, one, or two heads. We can compute the probability of each outcome happening:

Ways to get zero heads: TT.

Ways to get one heads: HT or TH.

Ways to get two heads: HH.

Thus, we get the probabilities $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$ for each possible outcome. □

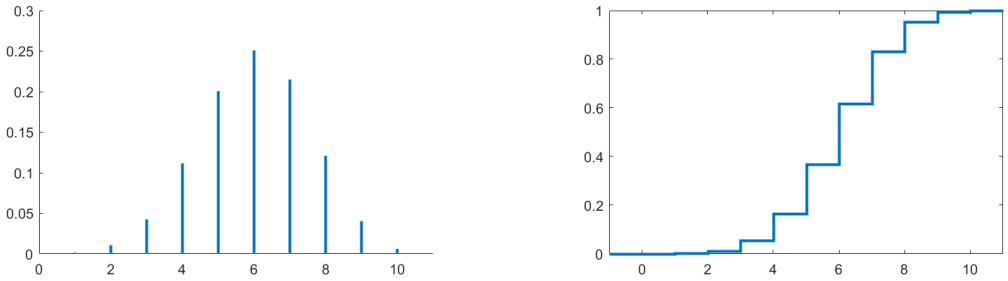


Figure 2.5: The pdf (L) and cdf (R) of the binomial distribution with $n = 10$ and $p = 0.6$

In Excel, multiple Bernoulli trials in separate cells can be used to simulate a binomial random variable by counting the number of successes. Note in Figure 2.6 the use of the `IF` function – it will output either 1 or 0 depending on whether the first condition `A3 < $C$1` is true or false.

The exact probabilities of each outcome can be calculated analytically, and in Excel using the built-in `BINOM.DIST(`x`,`n`,`p`,cumulative)`. This outputs the value of the pdf of $B(n, p)$ at x when the argument `cumulative` is set to False. If `cumulative` is set to True, this function instead evaluates the cdf of $B(n, p)$ at x, or the probability of getting at most x successes.

[2] So if $X_1, X_2, \ldots, X_n \sim B(1, p)$ independently, then $X_1 + X_2 + \cdots + X_n \sim B(n, p)$.

	A	B	C	D
1		p =	0.6	
2	RAND()	Trial #	Outcome	
3	0.9866582	1	0	=IF(A3<$C$1,1,0)
4	0.1899253	2	1	
5	0.1539927	3	1	
6	0.8555625	4	0	
7	0.1575868	5	1	
8	0.8712667	6	0	
9	0.9693323	7	0	
10	0.5708276	8	1	
11	0.2722038	9	1	
12	0.1362309	10	1	
13			6	# successes

Figure 2.6: Simulating multiple Bernoulli trials in Excel

The function `BINOM.INV(`n`,`p`,prob)` computes the inverse of `BINOM.DIST`, and can be used to generate samples from the binomial distribution. It outputs, given the parameters n, p, and a given probability `prob`, the smallest integer for which the cumulative binomial distribution is at least `prob`. Hence we can generate binomially-distributed variates by using `RAND()` as an input. Figure 2.7 illustrates this process while Figure 2.8 shows the inverse of the cdf for $B(10, 0.6)$.

	A	B	C	D
1	0.123565	4	=BINOM.INV(10,0.6,A1)	
2	0.427576	6		
3	0.05764	4		
4	0.898984	8		
5	0.481114	6		
6	0.769007	7		
7	0.594473	6		
8	0.755489	7		
9	0.891132	8		
10	0.588629	6		

Figure 2.7: 10 outcomes from $B(10, 0.6)$

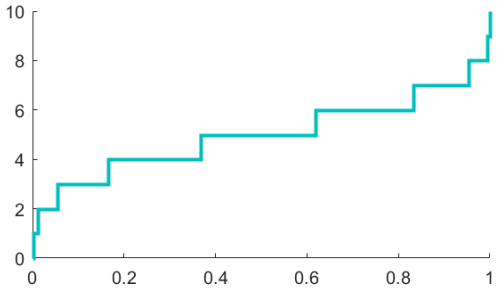


Figure 2.8: The inverse of the cdf for $B(10, 0.6)$

In Figure 2.9 we show the result of using Excel to generate 10, 100 and 1000 outcomes from $B(10, 0.6)$. Observe that outcomes are clustered around 6 and that the curve has a distinct 'bell' shape. We discuss this phenomenon in Section 2.4.

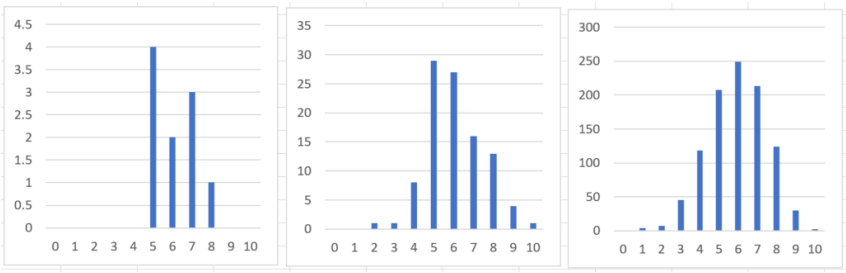


Figure 2.9: Generating 10, 100, and 1000 outcomes from $B(10, 0.6)$ in Excel

2.3 The Continuous Uniform Distribution, PDFs and CDFs

We discussed in Section 2.1 the discrete uniform distribution $U\{a, b\}$, where each outcome in the set $\{a, a+1, a+2, \ldots, b\}$ is equally likely.

There is also a continuous version of this distribution, where the possible set of outcomes is an interval of real numbers. Let us denote by $U(a, b)$ the continuous uniform distribution on the interval $[a, b]$. Sampling a value u from the standard continuous uniform distribution $U(0, 1)$ is exactly what the `RAND()` function in Excel is designed to approximate.[3] We can use u to simulate the random variable $U(a, b)$, using the transformation $a + (b - a)u$.

Question: Do continuous random variables have probability density functions?

The answer to this question is *yes*, but we will have to define them differently. Since continuous random variables have infinitely many possible outcomes, we do not want to talk about probabilities of individual outcomes, but instead, the probability that an outcome will lie *in a given interval*.

To illustrate, suppose we know that $X \sim U(0, 2)$. Then, intuitively, the probability that $X < 0.5$ should be 0.25, which is the length of the interval $[0, 0.5)$ divided by the length of the interval $[0, 2)$.

So, for a continuous random variable X, its **probability density function** or **pdf** is a function $f(x)$, that satisfies the property that

$$P(a \leq X \leq b) = \text{the area under the graph of } f(x) \text{ in between } x = a \text{ and } x = b. \ [4]$$

For example, for $U(a, b)$, its pdf is $f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{otherwise} \end{cases}$.

The **cdf** $g(x)$ of a continuous random variable is defined in the same way:

$$g(x) = P(X \leq x).$$

[3] Note that since the sample space is an infinite set, the probability of generating any specific value from $U(0, 1)$ is 0.

[4] If you have taken integral calculus you may recognize this as the integral of $f(x)$ from a to b: $P(a \leq x \leq b) = \int_a^b f(x)\, dx$.

The cdf of $U(a,b)$ is

$$g(x) = \begin{cases} \frac{x-a}{b-a}, & a \le x \le b \\ 0, & \text{otherwise} \end{cases}.$$

For now, do not worry about dealing with these functions themselves; what is more important is that we know how to simulate these random variables using Excel.

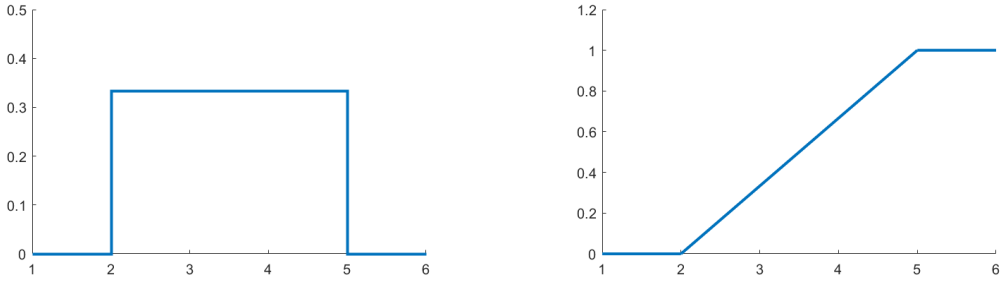


Figure 2.10: The pdf (L) and cdf (R) of the continuous uniform distribution $U(2,5)$

2.3.1 Application: Area and Volume Computations

One surprising application of the continuous uniform distribution is the approximation of quantities that are not random at all: areas, volumes, and their higher-dimensional analogues. For instance, suppose we want to calculate the area of the quarter-circle in Figure 2.11.

We know from grade-school geometry that this area is equal to $\frac{\pi}{4}$. We can also approximate this area as follows. We generate points at random inside the unit square, count the number of points inside the given region, and divide this by the total number of points generated. Since the unit square has area 1, the result will be an approximation of the area of the given region.

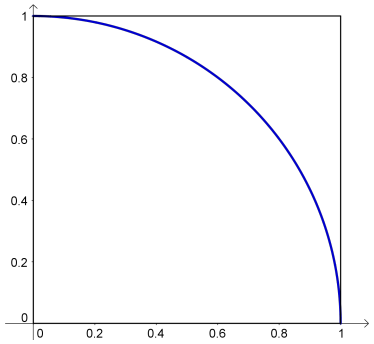


Figure 2.11

This is easy to do. Since the area under consideration is contained within the unit square $[0,1]\times[0,1]$, we can uniformly pick a point inside the square at random by using `RAND()` for each coordinate. It is also simple to check whether a point is inside this circle: test if its distance from the origin is less than 1. The following figures show the results of generating 20, 200, and 2000 points in this manner.

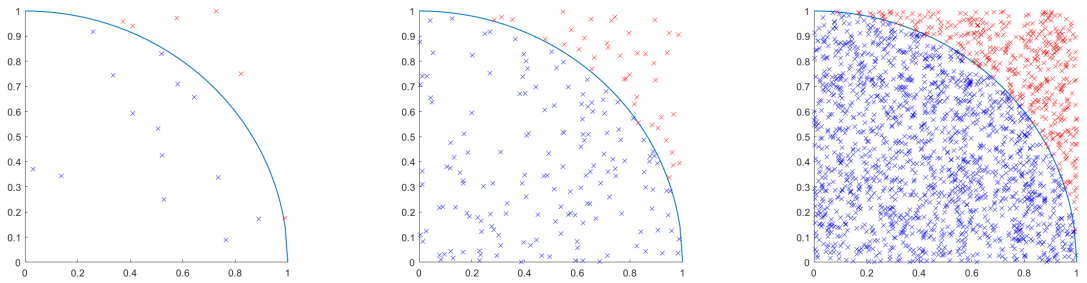


Figure 2.12: 20, 200, and 2000 randomly-generated points.

	A	B	C	D	E
1	x	y	inside?		
2	0.137781	0.343744	1	=IF(A2^2+B2^2<1,1,0)	
3	0.334951	0.744073	1		
4	0.4102	0.591695	1	counts	
5	0.259717	0.916392	1	14	=COUNTIF(C:C,1)
6	0.521272	0.425347	1	20	=COUNT(C:C)
7	0.735075	0.337929	1	approx.	
8	0.990122	0.176337	0	0.7	=D2/D3
9	0.822536	0.750946	0		
10	0.529121	0.250384	1		
11	0.506297	0.5327	1		
12	0.64321	0.65708	1		
13	0.577985	0.972232	0		
14	0.765571	0.088098	1		
15	0.727064	0.998083	0		
16	0.373298	0.957587	0		
17	0.580231	0.709169	1		
18	0.410599	0.941622	0		
19	0.890149	0.171932	1		
20	0.519994	0.829297	1		
21	0.031182	0.371519	1		

Figure 2.13: Area approximation of the quarter circle using 20 random points.

A big advantage of this method is that it can be easily implemented when we have a way to test if a point in the set. This is the case, for instance, if the set is the area under a curve.

Another approach to an area calculation like this involves breaking up the space $[0,1]\times[0,1]$ around the region into smaller squares, and colouring the squares whose centre is in the region under consideration – see Figure 2.15 for an example. Summing the areas of the coloured squares gives an estimate of the area; taking the limit of this as the squares get smaller is effectively equivalent to the integration technique from calculus.

But there are situations when one would prefer a random process (dropping points randomly and calculating the proportion) over a deterministic one (computing the integral, or estimating it from a count of grid points). For instance, in two dimensions using an evenly-spaced grid makes sense, while in twenty-dimensional space it may not. In particular, in high dimensions, grids have too many points: in dimension 20 the unit cube

	A	B	C	D	E
1	x	y	inside?		
2	0.5770338	0.9249764	0	=IF(A2^2+B2^2<1,1,0)	
3	0.049012	0.6368925	1		
4	0.2842498	0.1286711	1	counts	
5	0.8254521	0.0652154	1	159	=COUNTIF(C:C,1)
6	0.9063688	0.5349853	0	200	=COUNT(C:C)
7	0.8934579	0.423937	1	approx.	
8	0.4162457	0.5120693	1	0.795	=D2/D3

	A	B	C	D	E
1	x	y	inside?		
2	0.173794	0.99394	0	=IF(A2^2+B2^2<1,1,0)	
3	0.805487	0.857562	0		
4	0.206793	0.629817	1	counts	
5	0.043221	0.737561	1	1573	=COUNTIF(C:C,1)
6	0.755929	0.06447	1	2000	=COUNT(C:C)
7	0.479959	0.924161	0	approx.	
8	0.825032	0.881108	0	0.7865	=D2/D3

Figure 2.14: Area approximations using 200 and 2000 random points.

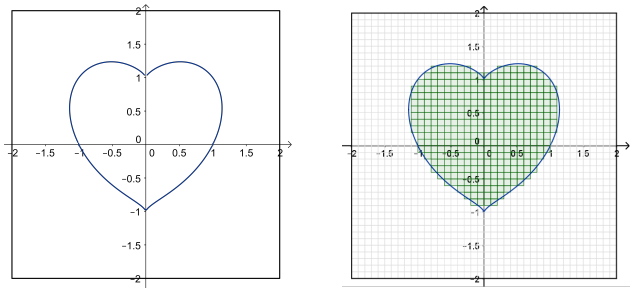


Figure 2.15: Approximating the area of a general region.

has over a million vertices and a $3 \times 3 \times \cdots \times 3$ grid has more than three trillion points while missing huge spaces inside the cube. It is very difficult to find a deterministic way of producing well-spaced points in high dimension. The method of using random points to estimate areas is called *Monte Carlo integration*.

2.4 The Normal Distribution, more PDFs and CDFs

The **normal distribution** is often called the **Gaussian distribution** or the **bell curve**. The two parameters associated with this distribution are the *mean* μ and the *standard deviation* σ. Random variables drawn from this distribution will tend to be gathered around the mean, while the variance measures how 'spread out' values are (a lower variance means values are more likely to stay close to the mean). This distribution is denoted as $N(\mu, \sigma)$, and is one of the most important statistical distributions, because it is commonly seen in nature. For instance, in error analysis, measurement errors are modelled by the normal distribution (e.g. when calibrating instruments, etc).

In Figure 2.16 we show the probability density function of the standard normal distribution $N(0, 1)$. The density function for $N(\mu, \sigma)$ is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

In Excel, the cumulative probability $P(Z < k)$ for $Z \sim N(\mu, \sigma)$ is calculated by the function `NORM.DIST(k,`μ`,`σ`,cumulative)` function when the parameter `cumulative` is set

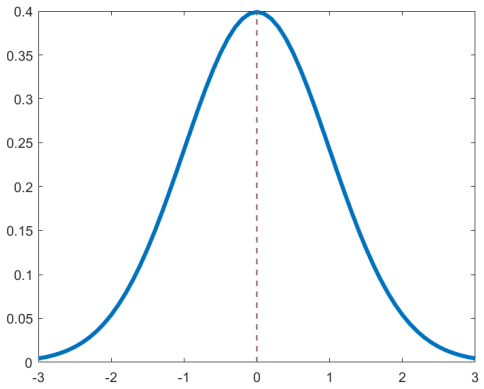


Figure 2.16: The probability density function of the normal distribution $N(0, 1)$

to True (if it is set to False, it just outputs the value of the probability density function at that k). The function `NORMINV(prob,`μ`,`σ`)` is the inverse of this; it calculates the point k for which $P(Z < k) =$ `prob`.

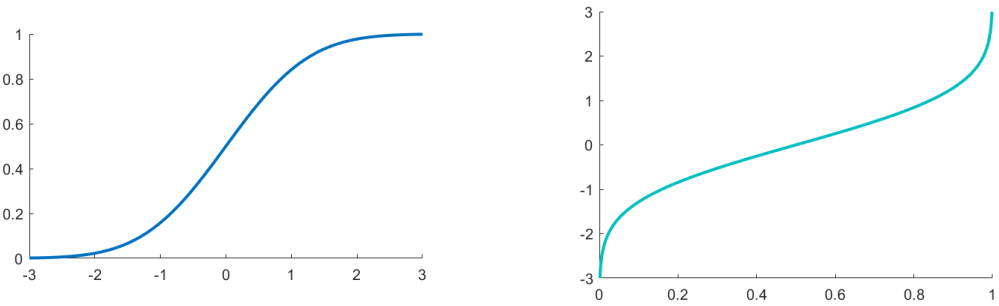


Figure 2.17: The cdf (L) and its inverse (R) for the standard normal distribution $N(0, 1)$

Example 2.3. Suppose that we sample from a normal distribution with mean $\mu = 50$ and standard deviation $\sigma = 10$. What is the probability that the random sample is (a) smaller than 40? (b) in between 60 and 70?

Solution Using `NORM.DIST` in Excel, we can compute the required probabilities. For (a), we get `NORM.DIST(40,50,10,TRUE)` $= 0.158655$. This is also equal to the area under the curve of $N(50, 10)$ from $x = 0$ to 40.

(b) This is calculated by the formula

`NORM.DIST(70,50,10,TRUE)-NORM.DIST(60,50,10,TRUE)` $= 0.135905$,

and this is the area under the curve of $N(50, 10)$ from $x = 60$ to $x = 70$. □

To generate values from a normal distribution in Excel, one could just call the `NORMINV()` function with a random probability for its first argument. For instance, the formula

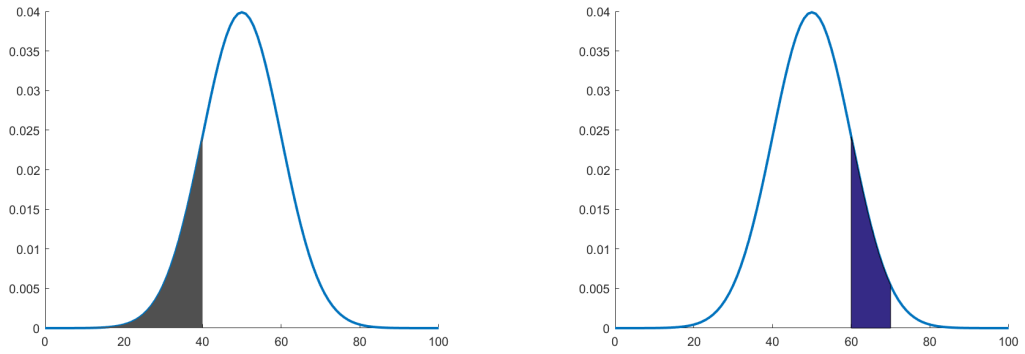


Figure 2.18: The probabilities in Example 2.3 as areas under the normal curve

`NORMINV(RAND(),50,10)` will generate values from $N(50, 10)$, as is illustrated in Figure 2.19.

55.740915	=NORMINV(RAND(),50,10)
42.088421	
35.479225	
52.055506	
52.577764	
49.880345	
54.756096	
42.464377	
46.337251	
38.258119	

Figure 2.19: Generating values from $N(50, 10)$

Example 2.4. The heights of women in a certain country are normally-distributed with mean $\mu = 65$in and standard deviation $\sigma = 5$in. What is the probability that a randomly-selected person from this country is taller than 6ft?

Solution: The probability is `1 - NORM.DIST(72,65,5,TRUE)` $= .008076$, or 0.81%. □

2.5 The Exponential Distribution

The **exponential distribution** is another widely-used distribution in applications, since it models interarrival times between two events – for example, the time it takes between customers at the bank, or the amount of time in between calls at a customer service centre. It also models the lifetimes of electrical equipment. This distribution is denoted as $\text{Exp}(\lambda)$, with the single parameter $\lambda > 0$ representing the average rate of arrivals/services per unit time. The probability density function for $\text{Exp}(\lambda)$ is the following:

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & \text{elsewhere} \end{cases}$$

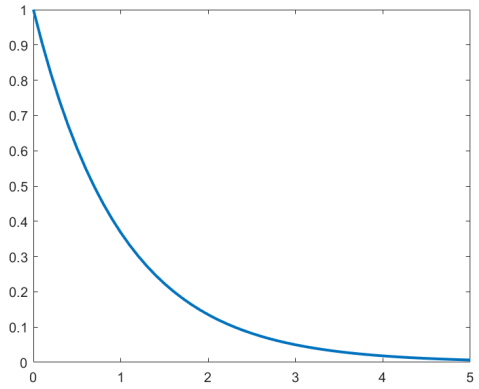


Figure 2.20: The probability density function of the exponential distribution Exp(1)

As this is a continuous distribution, we will use cumulative probabilities $P(X < k)$ to describe events. If $X \sim \text{Exp}(\lambda)$, then its cdf is

$$P(X < k) = 1 - e^{-\lambda k}, k \geq 0.$$

Example 2.5. Suppose that 10 customers arrive at a restaurant per hour on average. Find the probability that when the restaurant opens, the first customer arrives in the first 6 minutes.

Solution: The interarrival times of customers is an exponentially-distributed random variable with $\lambda = 10$, or $X \sim \text{Exp}(10)$. Hence $P(X < 0.1) = 1 - e^{-10(0.1)} = 0.632121$, or 63.21%. □

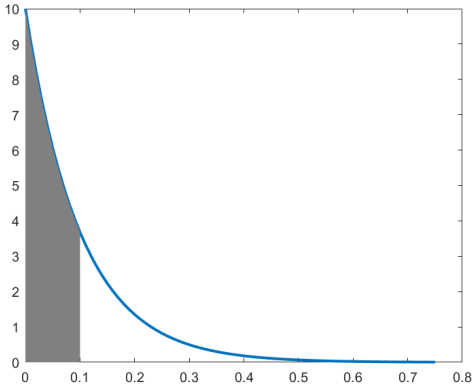


Figure 2.21: The probability density function for Exp(10) and the probability $P(X < 0.1)$.

To calculate the cumulative probability $P(X < k)$ in Excel, the function is `EXPON.DIST(k,lambda,TRUE)`, with the last parameter again indicating either the pdf

(False) or the cdf (True). However there is no built-in function for its inverse as in the normal distribution, since it has a simple algebraic expression: if $P(X < k) = y = 1 - e^{-\lambda k}$, then $k = -\frac{\ln(1-y)}{\lambda}$. Hence we can randomly generate exponentially-distributed variables by using $y =$ `RAND()`, then computing k from this expression.

	A	B	C
1	=RAND()	=-LN(1-A2)/($C$3)	
2	0.4905838	0.0674490	lambda
3	0.8041859	0.1630590	10
4	0.7738098	0.1486379	
5	0.4858436	0.0665228	
6	0.8758250	0.2086064	
7	0.6795648	0.1138075	
8	0.5094054	0.0712137	
9	0.1323678	0.0141987	
10	0.4339096	0.0569001	
11	0.9650668	0.3354318	

Figure 2.22: Ten values generated from Exp(10) using `RAND()` and the inverse cumulative distribution function

One important property of the exponential function is that it is *memoryless*, in that it models the time in between two events *regardless of how many times that event has happened*. In Example 2.5, this means that after the first customer has arrived, the time it takes for the second customer to arrive is also modelled by Exp(10). Similarly for the third, fourth, etc. This is a common assumption when modelling queueing processes.

EXERCISES

2.1 A pair of dice is rolled; let X be the sum of the two numbers that appear. Tabulate the pdf for X.

2.2 Plot the cdf of the random variable X of Exercise 2.1

2.3 **Explore**. We can simulate drawing a card at random from a standard deck of cards using $U\{1, 52\}$, by assigning cards to numbers (1-13 are the A-K of clubs, 14-26 are diamonds, 27-39 hearts, and 40-52 spades).

(a) In Excel, generate 20 numbers from $U\{1, 52\}$ and list the cards they correspond to. Do you see any repeats? How about when you generate 40 numbers?

(b) Guess the average number of draws from a deck of cards until *the first repeat occurs*, assuming the draws are done with replacement. Run a few experiments to test your theory.

(c) Guess the average number of draws from a deck of cards until *the first card is repeated*, assuming the draws are done with replacement. Again run some experiments on Excel to test your theory.

2.4 If $X \sim B(n, p)$, the probability that $X = k$ is known to be

$$P(X = k) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}.$$

Use Excel to generate one hundred random variables from $B(20, 0.7)$. Is your experiment consistent with this formula?

2.5 A company that manufactures alarm clocks has determined that for each batch of 100 clocks produced, there are 5 that are defective. Suppose a clock is randomly selected from a batch of 100, and let X be the random variable that is equal to 1 if this clock is defective, and 0 otherwise.

a) Determine the probability distribution of X (with the associated parameter/s).

b) A clock is randomly selected from each of ten batches. What is the probability that none of the selected clocks are defective?

2.6 Suppose that the lifetime of a car's battery is exponentially distributed with parameter $\lambda = 1/1000$ hours. Use Excel to estimate the probability that (a) the battery fails before 500 hours of use; (b) the battery lasts longer than 750 hours but fails before 1200 hours.

2.7 Two runners A and B run a race, and their running times are normally distributed with $N(60, 8)$ and $N(65, 5)$ (respectively).

(a) Use Excel to estimate the probability that runner A wins.

(b) If runner C also joins the race, with running time distributed as $N(58, 7)$, estimate the individual probabilities that each runner wins.

2.8 Using simulation, estimate the number of rolls needed in Exercise 2.1 until all possible values of X (from 2 to 12) are seen.

2.9 The heart-shaped region in Figure 2.15 is defined by the relation

$$\{(x, y) : (x^2 + y^2 - 1)^3 - x^2 y^3 \leq 0\}.$$

Approximate its area using 1000 randomly-generated points in Excel.

2.10 Choose three points uniformly at random on the unit circle $x^2 + y^2 = 1$. The probability that the triangle formed by these points contains the centre is $\frac{1}{4}$. Simulate this in Excel and verify this statement.

Hint: To generate a point on a circle at random, generate an angle in $[0, 2\pi)$.

Hint #2: You can fix one of the points to be at $(1, 0)$

3 Basic Simulations in Excel

> I am rarely happier than when spending an entire day programming my computer to perform automatically a task that would otherwise take me a good ten seconds to do by hand.
>
> Douglas Adams & Mark Carwardine
> *Last Chance to See*, 1990

Simulation is defined in [2] as

> *the imitation of the operation of a real-world process or system over time.*

There are many types of simulation, varying in the methods used and the type of systems being analyzed. For instance, a simulation can be *deterministic*, or *stochastic*, depending on whether random variables are involved or not. The first step in a simulation is usually to formulate the problem and the corresponding model – here, assumptions will be made about the system being studied. This might include deciding which variables to ignore, or assuming how an input is distributed probabilistically.

In the previous sections, we discussed how to generate random variables in Excel, using a few of the most common probability distributions. Before working on examples of simulations, we first discuss **data tables** in Excel.

3.1 Data Tables in Excel

In Excel, data tables are used for scenario analysis. They show the output of a calculation for different values of an input (similar to performing a sensitivity analysis). We demonstrate this using an example.

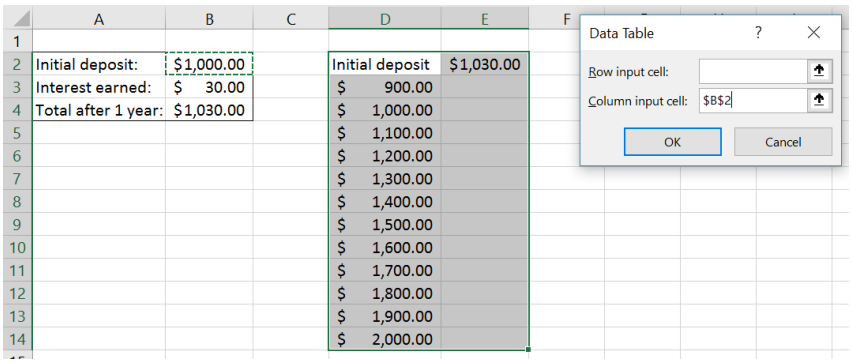
Example 3.1. Suppose that you open a savings account at your bank on January 1 of this year, that earns you a fixed rate of 3% per year. If you deposit $1,000 on January 1, at the end of the year you will have $1,030. What happens if we change the initial amount deposited?

Consider the following Excel spreadsheet: in cell `B2` we enter the initial amount deposited; `B3` has the interest earned (with formula `= 0.03 * B2`); `B4` has the total amount `= B2 + B3`.

To begin constructing the data table, in cells `D3:D14` input the different deposit amounts you would like to check (from \$900 to \$2,000). In cell `E2` enter the formula `=B4`.

	A	B	C	D	E
1					
2	Initial deposit:	$1,000.00		Initial deposit	$1,030.00
3	Interest earned:	$ 30.00		$ 900.00	
4	Total after 1 year:	$1,030.00		$ 1,000.00	
5				$ 1,100.00	
6				$ 1,200.00	
7				$ 1,300.00	
8				$ 1,400.00	
9				$ 1,500.00	
10				$ 1,600.00	
11				$ 1,700.00	
12				$ 1,800.00	
13				$ 1,900.00	
14				$ 2,000.00	

Now highlight cells `D2:E14`, and click 'Data Table' (Data → What-If Analysis → Data Tables). For 'Column input cell' enter the cell corresponding to the initial deposit, `B2`, then click OK.



The cells beside the input column will be populated with the one-year totals for each corresponding initial deposit.

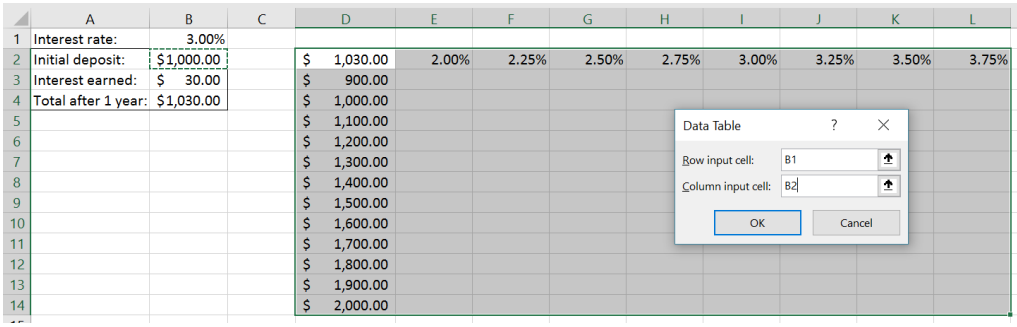
	A	B	C	D	E
1					
2	Initial deposit:	$1,000.00		Initial deposit	$1,030.00
3	Interest earned:	$ 30.00		$ 900.00	$ 927.00
4	Total after 1 year:	$1,030.00		$ 1,000.00	$1,030.00
5				$ 1,100.00	$1,133.00
6				$ 1,200.00	$1,236.00
7				$ 1,300.00	$1,339.00
8				$ 1,400.00	$1,442.00
9				$ 1,500.00	$1,545.00
10				$ 1,600.00	$1,648.00
11				$ 1,700.00	$1,751.00
12				$ 1,800.00	$1,854.00
13				$ 1,900.00	$1,957.00
14				$ 2,000.00	$2,060.00

Data tables do the work of entering information and recalculating the worksheet based on the changes. Example 3.1 demonstrated a *one-way data table*, because one input variable was allowed to vary. We can also construct *two-way data tables*.

Example 3.2. Continuing the previous example, suppose that you wanted to compute your year-end savings account balance given different interest rates for the account. To do this using a two-way data table, see the following spreadsheet:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Interest rate:	3.00%										
2	Initial deposit:	$1,000.00		$ 1,030.00	2.00%	2.25%	2.50%	2.75%	3.00%	3.25%	3.50%	3.75%
3	Interest earned:	$ 30.00		$ 900.00								
4	Total after 1 year:	$1,030.00		$ 1,000.00								
5				$ 1,100.00								
6				$ 1,200.00								
7				$ 1,300.00								
8				$ 1,400.00								
9				$ 1,500.00								
10				$ 1,600.00								
11				$ 1,700.00								
12				$ 1,800.00								
13				$ 1,900.00								
14				$ 2,000.00								

Enter the formula `=B4` in cell `D2`, and also change cell `B3` to the formula `=B1*B2`, to account for a varying interest rate. In cells `E2:L2` enter the different interest rates under consideration. Now highlight cells `D2:L14`, and click Data Table. For 'Column input cell' enter `B2`, and for 'Row input cell' enter `B1`.



Click OK. The table should now be populated with the different year-end totals for the given combinations of initial deposit and interest rate.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Interest rate:	3.00%										
2	Initial deposit:	$1,000.00		$ 1,030.00	2.00%	2.25%	2.50%	2.75%	3.00%	3.25%	3.50%	3.75%
3	Interest earned:	$ 30.00		$ 900.00	$ 918.00	$ 920.25	$ 922.50	$ 924.75	$ 927.00	$ 929.25	$ 931.50	$ 933.75
4	Total after 1 year:	$1,030.00		$ 1,000.00	$1,020.00	$1,022.50	$1,025.00	$1,027.50	$1,030.00	$1,032.50	$1,035.00	$1,037.50
5				$ 1,100.00	$1,122.00	$1,124.75	$1,127.50	$1,130.25	$1,133.00	$1,135.75	$1,138.50	$1,141.25
6				$ 1,200.00	$1,224.00	$1,227.00	$1,230.00	$1,233.00	$1,236.00	$1,239.00	$1,242.00	$1,245.00
7				$ 1,300.00	$1,326.00	$1,329.25	$1,332.50	$1,335.75	$1,339.00	$1,342.25	$1,345.50	$1,348.75
8				$ 1,400.00	$1,428.00	$1,431.50	$1,435.00	$1,438.50	$1,442.00	$1,445.50	$1,449.00	$1,452.50
9				$ 1,500.00	$1,530.00	$1,533.75	$1,537.50	$1,541.25	$1,545.00	$1,548.75	$1,552.50	$1,556.25
10				$ 1,600.00	$1,632.00	$1,636.00	$1,640.00	$1,644.00	$1,648.00	$1,652.00	$1,656.00	$1,660.00
11				$ 1,700.00	$1,734.00	$1,738.25	$1,742.50	$1,746.75	$1,751.00	$1,755.25	$1,759.50	$1,763.75
12				$ 1,800.00	$1,836.00	$1,840.50	$1,845.00	$1,849.50	$1,854.00	$1,858.50	$1,863.00	$1,867.50
13				$ 1,900.00	$1,938.00	$1,942.75	$1,947.50	$1,952.25	$1,957.00	$1,961.75	$1,966.50	$1,971.25
14				$ 2,000.00	$2,040.00	$2,045.00	$2,050.00	$2,055.00	$2,060.00	$2,065.00	$2,070.00	$2,075.00

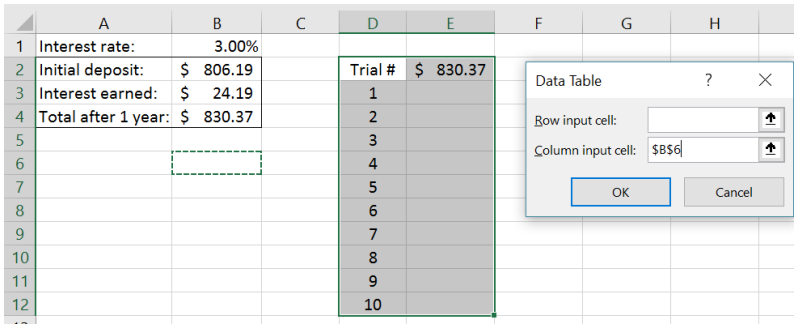
Rather than looking at an evenly-spaced set of input values, we can also construct tables based on random inputs to help us develop intuition for what the transformed random output should look like.

Note that Examples 3.1 and 3.2 are *deterministic*: each entry has a value that is computed directly from its inputs, and repeating the calculation will produce an identical result. The following example, on the other hand, is *stochastic*: it incorporates some randomness.

Example 3.3. Suppose that your initial deposit amount is a normally-distributed random variable with mean \$1,000 and standard deviation \$100. In cell `B2`, following the discussion in Section 2.4 we enter the formula `=NORM.INV(RAND(),1000,100)`. To initialize the data table cell `E2` still refers to `B4`, while the first column can just have trial numbers (say we want to run this experiment 10 times).

	A	B	C	D	E
1	Interest rate:	3.00%			
2	Initial deposit:	$ 806.19		Trial #	$ 830.37
3	Interest earned:	$ 24.19		1	
4	Total after 1 year:	$ 830.37		2	
5				3	
6				4	
7				5	
8				6	
9				7	
10				8	
11				9	
12				10	

Highlight cells `D2:E12`, then click Data Table. Now for 'Column input cell' choose any blank cell. The value in `B2`, and hence the output `B4`, will change each time the worksheet is recalculated so we do not need to specify a changing input cell.



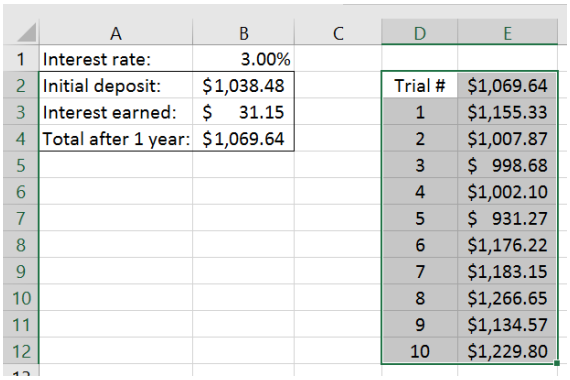
Click OK. Now the table is populated by the different year-end values obtained given the assumption on the initial deposit.

Now we begin our foray into simulation by looking at some simple problems.

3.2 A Beginning Example – Coin-Flipping

The following problem recently appeared on the blog *The Riddler*[1]:

[1] A weekly puzzle blog on the site `http://www.fivethirtyeight.com`.



	A	B	C	D	E
1	Interest rate:	3.00%			
2	Initial deposit:	$1,038.48		Trial #	$1,069.64
3	Interest earned:	$ 31.15		1	$1,155.33
4	Total after 1 year:	$1,069.64		2	$1,007.87
5				3	$ 998.68
6				4	$1,002.10
7				5	$ 931.27
8				6	$1,176.22
9				7	$1,183.15
10				8	$1,266.65
11				9	$1,134.57
12				10	$1,229.80

You and I find ourselves indoors one rainy afternoon, with nothing but some loose change in the couch cushions to entertain us. We decide that we'll take turns flipping a coin, and that the winner will be whoever flips 10 heads first. The winner gets to keep all the change in the couch! Predictably, an enormous argument erupts: We both want to be the one to go first.

What is the first flipper's advantage? In other words, what percentage of the time does the first flipper win this game?[2]

This is a question about probability, and if you have taken some courses in probability and statistics, then you will have seen exercises like this. The exact probability that the first flipper wins the game can be computed analytically, but this is not easy to do. However, the process is very simple to simulate.

To simulate the coin flips, we use a Bernoulli random variable with a $p = 0.50$ probability of heads. Using 100 flips each, we keep track of the number of heads for each player, and compare the number of flips needed to get to 10. Note that we have to cap the number of flips at a certain point, even though in theory there is a nonzero probability that neither player gets at least 10 heads in the first 100 flips.[3]

The formulas we use are: `RANDBETWEEN(0,1)` to generate the coin flips, and `SUM($B$2:B2)` and `SUM($D$2:D2)` for the running totals (autofilled).

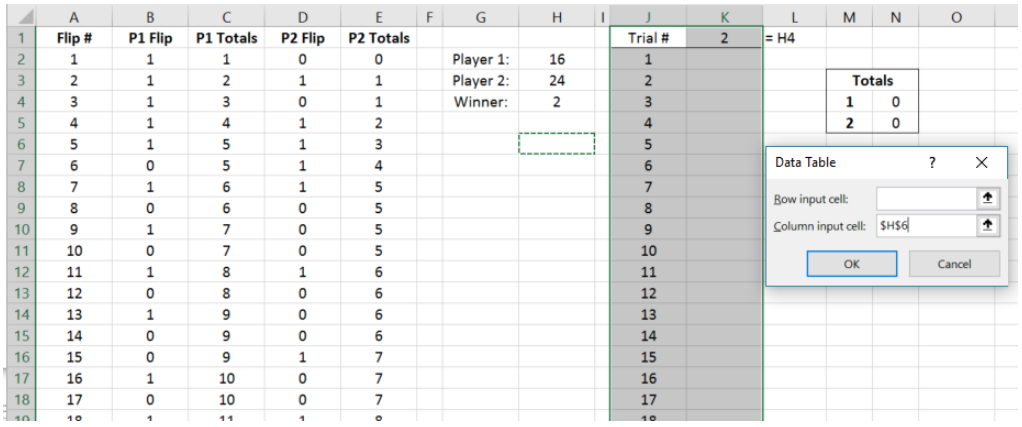
To output the number of flips to get to 10, we use the `MATCH(lookup_value, lookup_array, [match_type])` function in Excel, which finds the first occurrence of `lookup_value` in the array `lookup_array`, with parameter `match_type` equal to 1, 0, or -1 depending on whether you want `MATCH` to find the first value less than or equal to, equal to, or greater than or equal to the lookup value, respectively.

In the spreadsheet shown, cell `H2` has the formula `MATCH(10,C2:C101,0)`, and cell `H3` has formula `MATCH(10,E2:E101,0)`. In this run Player 1 gets 10 heads on the 17th flip, while Player 2 gets 10 heads on the 20th. Lastly, in `H4` we enter the formula `IF(H2<=H3,1,2)`.

[2] `https://fivethirtyeight.com/features/can-you-time-the-stoplight-just-right/`, January 20, 2017

[3] Compute this probability using the inverse binomial function in Excel!

	A	B	C	D	E	F	G	H
1	**Flip #**	**P1 Flip**	**P1 Totals**	**P2 Flip**	**P2 Totals**			
2	1	1	1	0	0		Player 1:	17
3	2	1	2	0	0		Player 2:	20
4	3	0	2	1	1		Winner:	2
5	4	0	2	0	1			
6	5	1	3	0	1			
7	6	0	3	1	2			
8	7	0	3	1	3			
9	8	0	3	0	3			
10	9	0	3	1	4			
11	10	1	4	0	4			
12	11	1	5	0	4			
13	12	1	6	1	5			
14	13	1	7	1	6			
15	14	1	8	0	6			
16	15	1	9	0	6			
17	16	0	9	1	7			
18	17	1	10	0	7			
19	18	0	10	1	8			
20	19	1	11	1	9			
21	20	1	12	1	10			



To run this multiple times, we use a data table with 500 rows. We set a blank cell as the column input cell, as explained in Example 3.3.

J	K	L	M	N
Trial #	1	= H4		
1	1			
2	2		**Totals**	
3	2		**1**	279
4	1		**2**	221
5	2			
6	2			
7	1			
8	2			
9	1			
10	1			
11	1			

Our results give 279 wins for Player 1, and 221 wins for Player 2. Hence our simulation approximates the probability of Player 1 winning as 53.8%. The actual probability is given by the expression

$$\sum_{n=10}^{\infty} \left[\frac{1}{2^n} \frac{(n-1)!}{9!(n-10)!} \left(1 - \sum_{i=10}^{n-1} \frac{1}{2^i} \frac{(i-1)!}{9!(i-10)!} \right) \right] \approx 53.2909\%.$$

This is in some sense a "lucky" result, if you repeat the calculation you will see that it is usually not so accurate.

3.3 Generating Permutations and Sampling without Replacement

Here is another puzzle from *The Riddler*:

> On snowy afternoons, you like to play a solitaire 'game' with a standard, randomly shuffled deck of 52 cards. You start dealing cards face up, one at a time, into a pile. As you deal each card, you also speak aloud, in order, the 13 card faces in a standard deck: ace, two, three, etc. (When you get to king, you start over at ace.) You keep doing this until the rank of the card you deal matches the rank you speak aloud, in which case you lose. You win if you reach the end of the deck without any matches.
>
> What is the probability you win?[4]

One can compute solution exactly using combinatorics, but the calculation is difficult and subtle enough that the authors of the blog presented an oversimplified incorrect solution the following week, despite the fact that several people presumably sent the correct answer

[4] `https://fivethirtyeight.com/features/can-you-deal-with-these-card-game-puzzles/`, December 30, 2016.

(while many others sent incorrect ones).[5] Simulation can be a good sanity check on an analytic solution. In this case, we would have to compute the solution quite accurately to spot the error.

To simulate this game, we need a way to generate a random arrangement of a 52-card deck. Let us assign cards to numbers using the following scheme:

	A	2	3	···	J	Q	K
♠	1	2	3	···	11	12	13
♦	14	15	16	···	24	25	26
♥	27	28	29	···	37	38	39
♣	40	41	42	···	50	51	52
(mod 13)	1	2	3	···	11	12	0

Note that we only have to consider these numbers modulo 13, since the suit is ignored. Now we need to generate a random permutation of the numbers 1 to 52, corresponding to the shuffled deck of cards.

Refer to the spreadsheet below: first enter in cells `C2:C53` the numbers 1 to 52 – this will correspond to the cards being announced. Then, in the first column generate 52 random numbers using `RAND()`. We then generate a permutation based on the relative positions of each cell in the array `A2:A53`. To do this we enter in cell `B2` the formula

```
=INDEX($C$2:$C$53,MATCH(LARGE($A$2:$A$53,ROW()-ROW($A$2)+1),$A$2:$A$53,0)).
```

This instructs Excel to look for the kth largest entry in the array of random numbers, then output its relative position in that array. Auto-filling the rest of the cells in `B2:B53` yields the random permutation. In Figure 3.1, for example, the first few cells in column B mean that the largest entry in column A is in the 6th position (`A7`), the second largest entry is in the 1st position (`A2`), and the third largest entry is in the 35th position.

	A	B	C
1	RAND()	deck	called
2	0.967922	6	1
3	0.448037	1	2
4	0.0145	35	3
5	0.655991	52	4
6	0.643384	48	5
7	0.987506	40	6
8	0.308553	38	7
9	0.791775	8	8
10	0.079951	45	9

Figure 3.1: Generating a random permutation in Excel.

Observe that this method simulates sampling from a finite set without replacement, although generating a permutation is not practical for large sets. Drawing only k cards

[5] Original solution: `https://fivethirtyeight.com/features/dont-throw-out-that-calendar/`.
Correction: `https://fivethirtyeight.com/features/how-long-will-it-take-to-blow-out-the-birthday-candles/#correction`.
Solution method: `http://math.stackexchange.com/questions/1891958/derangements-of-a-deck-of-cards-where-ranks-are-equal`, giving a probability of 1.6233%.

out of the deck without replacement corresponds to choosing the first k entries in the permutation.

To check whether the drawn card matches the rank being called out, we just do a modulo check:

```
=IF(MOD(B2,13)=MOD(C2,13),1,0)
```

will output 1 if the two ranks match, and 0 otherwise. Summing up column C will result in the total number of times that the ranks match, so you win if this sum is zero. Using a data table, we repeat this experiment 1000 times.

	A	B	C	D	E	F	G	H	I	J	K
1	RAND()	deck	called	check mod	sum		Trial #	3		Count	1000
2	0.967922	6	1	0	3		1	5		0	17
3	0.448037	1	2	0			2	5		1	57
4	0.0145	35	3	0			3	5		2	130
5	0.655991	52	4	0			4	4		3	214
6	0.643384	48	5	0			5	2		4	203
7	0.987506	40	6	0			6	3		5	164
8	0.308553	38	7	0			7	5		6	109
9	0.791775	8	8	1			8	4		7	63
10	0.079951	45	9	0			9	1		8	27
11	0.013268	27	10	0			10	3		9	10
12	0.204743	46	11	0			11	5		10	4
13	0.49463	36	12	0			12	7		11	1
14	0.50034	4	13	0			13	4		12	1
15	0.303023	5	14	0			14	3		13	0
16	0.397258	21	15	0			15	4		14	0
17	0.115443	22	16	0			16	6		15	0

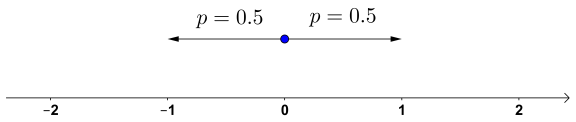
The `COUNTIF` function in Excel counts how many instances of 0, 1, and so on, appear in the data table; as the spreadsheet shows we won 17 times out of 1000 trials (so we estimate the probability to be 1.7%). More trials will lead to greater accuracy in estimating the true probability, which as noted is around 1.62%.

3.4 Random Walks

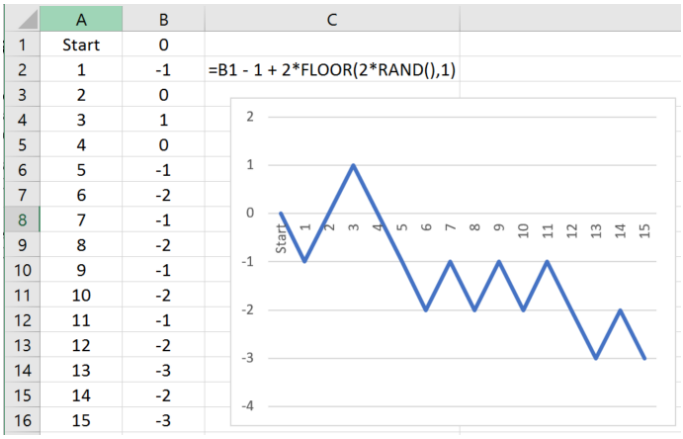
A **random walk** is a path where each step depends on a probability distribution. Random walks have many applications, from physics (Brownian motion) to finance (movement of stocks on the market).

3.4.1 Random Walks on a Line

A *1-dimensional random walk* is a path on the real number line that starts at 0, and with probability p moves one unit to the right, and with probability $1-p$ moves one unit to the left. Here we consider the simplest case, where $p = 0.5$. We can think of each step as being determined by a fair coin flip.
This is easy to simulate in Excel, as shown in Figure 3.2. Verify for yourself that the formula shown outputs `B1`-1 or `B1`$+1$ with equal probability. Using the Chart function



produces a visual of the random walk produced.



	A	B	C
1	Start	0	
2	1	-1	=B1 - 1 + 2*FLOOR(2*RAND(),1)
3	2	0	
4	3	1	
5	4	0	
6	5	-1	
7	6	-2	
8	7	-1	
9	8	-2	
10	9	-1	
11	10	-2	
12	11	-1	
13	12	-2	
14	13	-3	
15	14	-2	
16	15	-3	

Figure 3.2: Generating a random walk in Excel

A basic problem about random walks is whether they return to their starting point. Let us generate random walks (using data tables) and then count what fraction of those generated do return to 0. The next two figures show the results of generating a thousand random walks of length 100 and 1000, respectively. Note the usage of the `MATCH` function to look for the first occurrence of zero in the walk.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Start	0		**first return**		Trial #	20		=D2			
2	1	-1		20		1	2			**928** /1000 walks return to the origin		
3	2	-2		=MATCH(0,B2:B101,0)		2	26		=COUNT(G2:G1001)			
4	3	-1				3	2					
5	4	-2				4	4			**8.627** average steps to return		
6	5	-3				5	2		=AVERAGEIF(G2:G1001,">=0")			
7	6	-2				6	2					
8	7	-3				7	16					
9	8	-4				8	48					
10	9	-5				9	#N/A					
11	10	-4				10	6					

Figure 3.3: 1000 random walks of length 100.

A `#N/A` entry denotes that the walk did not return to 0, while the `COUNT` function counts the number of cells in the data table that have numbers. Observe that only 92.2% of the 100-step walks returned to 0, while 97.4% of the 1000-step walks did. This suggests that all the random walks will eventually return to zero, but some take a long time to do so.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Start	0		**first return**		Trial #	2	=D2				
2	1	1		2		1	4			**971** /1000 walks return to the origin		
3	2	0		=MATCH(0,B2:B1001,0)		2	2		=COUNT(G2:G1001)			
4	3	1				3	4					
5	4	2				4	58		**25.096** average steps to return			
6	5	3				5	10		=AVERAGEIF(G2:G1001,">=0")			
7	6	2				6	16					
8	7	3				7	4					
9	8	4				8	2					
10	9	3				9	4					
11	10	2				10	48					

Figure 3.4: 1000 random walks of length 1000.

We can also keep track of the average number of steps it takes for a walk to return to zero. We calculate this using `AVERAGEIF(G2:G1001,“>=0”)`, which takes the average of the cells in the data table while ignoring the cells with `#N/A`. Increasing the length of the walk thus increases the number of walks that do return to zero, but the average number of steps continues to increase as we increase the number of steps and capture the walks that take a long time to return to zero. In fact, it is possible to prove (with considerable effort) that while walks will eventually return to zero, the expected time to return is infinite. This is because the average time is dominated by a few long walks, a fact that we can appreciate experimentally.

Using Python we generated 100,000 walks for each of the following lengths, and obtained these results. (The code to implement this can be found after the exercises.)

length of walks	% returning	average time to return
10	73.513%	3.27
100	92.049%	8.57
1000	97.471%	25.86
10000	99.183%	76.30
100000	99.744%	238.20
1000000	99.932%	786.75
10000000	99.976%	2949.73

Observe that even for walks with ten million steps there is a small fraction that do not return to zero.

Using the same simulation we can also investigate the expected distance from the origin after a fixed number of steps. Try this yourself for walks of 100 steps and then 1000 steps, and try to understand the pattern.

3.4.2 Random Walks on a Grid

Suppose now that we start at the origin $(0, 0)$ in 2-dimensional space, and consider the lattice of integer points $\{(x, y) : x, y \text{ integers}\}$. Assume that there is an equal probability (of 0.25 each) of moving to an adjacent integer point at each step. We would now like to estimate similar quantities as in the 1-dimensional case: what is the probability of returning to the origin? What is the expected distance from the origin for a walk of fixed length? We remark that the expected time to return to zero must be ∞ as it is that for each coordinate separately. Moreover there are multiple notions of distance we can consider – the Euclidean distance ($\sqrt{x^2 + y^2}$) and the Manhattan distance ($|x| + |y|$) are examples.

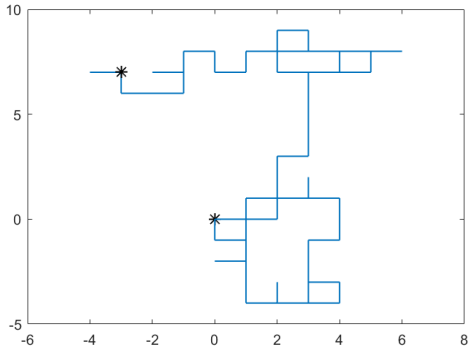


Figure 3.5: 100 steps of a random walk on a grid, visualized.

To simulate this in Excel, we now need to keep track of two columns, one each for the x- and y-coordinate. Refer to Figure 3.6 for the formulas used (note that `F2` has a formula that changes based on the length of this walk, which is input in `F5`).

	A	B	C	D	E	F	G
1		**Start**	0	0	**distance from (0,0)**	**first return**	
2	0.483351	1	1	0	1.000	8	=MATCH(0,INDIRECT($F$8),0)
3	0.522945	2	1	-1	1.414		
4	0.856885	3	1	0	1.000	**max length of walk**	
5	0.450542	4	2	0	2.000	100	
6	0.782893	5	2	1	2.236		
7	0.132647	6	1	1	1.414	**array**	
8	0.552644	7	1	0	1.000	$E$2:$E$101	=ADDRESS(2,5)&":"&ADDRESS(1+$F$5,5)
9	0.169862	8	0	0	0.000		
10	0.691686	9	0	-1	1.000		
11	0.264265	10	1	-1	1.414	Formula in	
12	0.212846	11	0	-1	1.000	A2:	= RAND()
13	0.442498	12	1	-1	1.414	C2:	= C1 + IF(A2<0.25,-1,IF(A2<0.5,1,0))
14	0.71708	13	1	-2	2.236	D2:	= D1 + IF(A2>0.75,1,IF(A2>0.5,-1,0))
15	0.188812	14	0	-2	2.000	E2:	= SQRT(C2^2 + D2^2)
16	0.250576	15	1	-2	2.236		

Figure 3.6: 100 steps of a random walk on a grid.

Using a data table, we simulate a thousand random walks of length 100 and 1000. Observe that the percentages are much smaller than what we saw in the one-dimensional case: 58.1% for walks of length 100, and 69.9% for walks of length 1000. Even increasing the length to 5000 only increases this observed frequency to 71.8% (try it yourself and see what percentages you get!).

	A	B	C	D	E	F	G	H	I	J	K
1		**Start**	0	0	**distance from (0,0)**	**first return**		Data Table	4		**# returns**
2	0.1298	1	-1	0	1.000	4	=MATCH(0,INDIRECT($F$8),0)	1	#N/A		581
3	0.78605	2	-1	1	1.414			2	46		
4	0.28578	3	0	1	1.000	**max length of walk**		3	#N/A		**average**
5	0.74163	4	0	0	0.000	100		4	#N/A		13.38726
6	0.78743	5	0	1	1.000			5	#N/A		
7	0.86206	6	0	2	2.000	**array**		6	#N/A		
8	0.1534	7	-1	2	2.236	$E$2:$E$101	=ADDRESS(2,5)&":"&ADDRESS(1+$F$5,5)	7	#N/A		
9	0.03065	8	-2	2	2.828			8	2		
10	0.33632	9	-1	2	2.236			9	#N/A		
11	0.56188	10	-1	1	1.414	Formula in		10	2		
12	0.83926	11	-1	2	2.236	A2:	= RAND()	11	#N/A		
13	0.65666	12	-1	1	1.414	C2:	= C1 + IF(A2<0.25,-1,IF(A2<0.5,1,0))	12	#N/A		
14	0.00657	13	-2	1	2.236	D2:	= D1 + IF(A2>0.75,1,IF(A2>0.5,-1,0))	13	#N/A		
15	0.0183	14	-3	1	3.162	E2:	= SQRT(C2^2 + D2^2)	14	2		

Figure 3.7: 1000 random walks of length 100.

Based on this, it is hard to tell if the probability of returning to the origin might be smaller than 1 – however, it has been shown analytically that as the length of the walk approaches

	A	B	C	D	E	F	G	H	I	J	K
1		**Start**	0	0	**distance from (0,0)**	**first return**		**Data Table**	#N/A		**# returns**
2	0.29829	1	1	0	1.000	#N/A	=MATCH(0,INDIRECT($F$8),0)	1	#N/A		699
3	0.49999	2	2	0	2.000			2	2		
4	0.69422	3	2	-1	2.236	**max length of walk**		3	#N/A		**average**
5	0.26629	4	3	-1	3.162	1000		4	10		70.34907
6	0.90236	5	3	0	3.000			5	2		
7	0.05479	6	2	0	2.000	**array**		6	2		
8	0.29837	7	3	0	3.000	$E$2:$E$1001	=ADDRESS(2,5)&":"&ADDRESS(1+$F$5,5)	7	2		
9	0.82858	8	3	1	3.162			8	40		
10	0.85737	9	3	2	3.606			9	4		
11	0.70956	10	3	1	3.162	Formula in		10	14		
12	0.22666	11	2	1	2.236	A2:	= RAND()	11	2		
13	0.27656	12	3	1	3.162	C2:	= C1 + IF(A2<0.25,-1,IF(A2<0.5,1,0))	12	4		
14	0.93347	13	3	2	3.606	D2:	= D1 + IF(A2>0.75,1,IF(A2>0.5,-1,0))	13	8		
15	0.1271	14	2	2	2.828	E2:	= SQRT(C2^2 + D2^2)	14	2		

Figure 3.8: 1000 random walks of length 1000.

infinity, this probability approaches 1.[6] This illustrates one difficulty of performing simulations – if the sample size is too small, the results might not necessarily reflect the true behaviour accurately. In this example, even working with walks of length 5000, we still found a return rate of smaller than 75%. On the other hand, it is hard to run a large number of experiments on Excel, as we don't have a good way to truncate walks that have already returned to zero. Other programming languages are better for this purpose, however the trade-off between accuracy and running time remains.

EXERCISES

3.1 Consider the following experiment: roll three 6-sided dice at the same time, and let X be the random variable that is equal to the number of different outcomes seen. For example, a roll of $\{3, 5, 5\}$ gives $X = 2$. Estimate the probability distribution for X using an Excel simulation. Then, by enumerating all possibilities completely, determine the exact probabilities.

3.2 Another puzzle from *The Riddler*:

> You and I stumble across a 100-sided die in our local game shop. We know we need to have this die – there is no question about it – but we're not quite sure what to do with it. So we devise a simple game: We keep rolling our new purchase until one roll shows a number smaller than the one before. Suppose I give you a dollar every time you roll. How much money do you expect to win?[7]

Estimate this quantity using a simulation.

[6] See the article on *Pólya's Random Walk Constants* at `http://mathworld.wolfram.com/PolyasRandomWalkConstants.html`.

[7] `https://fivethirtyeight.com/features/how-long-will-it-take-to-blow-out-the-birthday-candles/`, January 13, 2017.

3.3 The probability that two positive integers picked randomly are coprime (that is, they share no divisors other than 1) is $\frac{6}{\pi^2}$. Provide estimates for π by taking 100, 1000, and 10000 pairs of integers from the ranges $\{1, 2, \ldots, 100\}$, $\{1, 2, \ldots, 1000\}$, and $\{1, 2, \ldots, 10000\}$.[8]

3.4 For a random walk in one dimension, estimate the expected distance from the origin after 200, 500, and 1000 steps by generating 1000 walks for each case.

3.5 **Biased random walks**. Consider a one-dimensional random walk where you move one unit to the left with probability p and one unit to the right with probability $1 - p$. Simulate this for $p = 0.6$. Do you expect all walks to return to the origin in this case? How about if $p = 0.8$?

3.6 **Biased random walks**. Consider a one-dimensional random walk where at the origin, you move one unit to the left or right with equal probability, but everywhere else, you move *towards the origin* with probability p, and *away from the origin* with probability $1 - p$. Simulate this for $p = 0.4$ and $p = 0.6$, and explain your results. Note that the formula to generate the next step is more involved, since you have to deal with three different cases.

3.7 **Random walks in three dimensions**. Consider a random walk in 3D, where you start at the origin $(0, 0, 0)$, and with equal probability move 1 or -1 units in one of the coordinate directions. Simulate this in Excel for a maximum length of 100 steps, and generate 100 walks using a data table. How many return to the origin? Try to generate longer walks (as far as your computer allows). Can you make any conclusions?

Python code for generating random walks

```
from __future__ import division
import random as rd

def randwalk(N,p=0.5,marker=0):
# Function for generating a random walk
# Input: N, p (prob of -1 step), marker (starting point)
#      *defaults are p = 0.5 and marker = 0
# Output: number of steps to first return to origin
#         (or 0 if it does not return)
    current = marker # Initialize
    count = 0
    while count < N:
        # Generate random numbers, and move left or right
# corresponding to probabilities
        r = rd.random()
        if r < p:
            current -= 1
        else:
```

[8] This video by mathematician-comedian Matt Parker on his *Youtube* channel *standupmaths* demonstrates this process: `https://www.youtube.com/watch?v=RZBhSi_PwHU` .

```
            current += 1
        count += 1
        if current == 0:
            return count
    return 0


# Now aggregate up to M walks, and take return rate in %
# and the average # steps to return

# Initialize parameters
N = 100
p = 0.5
marker = 0
M = 100000 # Number of walks to aggregate

total_returns = 0
total_steps = 0
counter = 0

while counter < M:
    steps = randwalk(N,p,marker)
    if steps > 0:
        total_returns += 1
        total_steps += steps
    counter += 1

average_steps = total_steps/total_returns
return_rate = 100*total_returns/M

print "%d random walks of length %d were generated." % (M, N)
print "%7.4f%% of all walks returned to 0." % (return_rate)
print "On average it took %10.2f steps to return." % (average_steps)
```

4 An Introduction to Queueing Theory

3. There is no point in waiting at the end of the line.
4. If you don't wait at the end of the line, you'll never get to the front.

Charles Wright
Wright's Axioms of Queueing Theory

4.1 A Framework for Simulation

The goal of simulation is the analysis of a real-world process by imitating its behaviour using computer software. In the previous chapters we saw examples of how we could use randomness to perform calculations that are deterministic in nature (e.g. the area calculations in Section 2.3.1). Such experiments are referred to as **Monte Carlo Methods**.

Moreover, we also performed simulations of processes for which randomness was an inherent characteristic (rolling a die, etc., in Chapter 3). Arguably this scenario is more interesting than the former, since one cannot *solve* these processes analytically as one can theoretically do for area and volume calculations; outcomes of these experiments cannot be predicted (although one can derive quantities like the long-term behaviour of random processes). And herein lies the main advantage of simulation – it is a low-risk method for obtaining knowledge about how a random system operates (or *will operate*).

In particular, what if questions are addressed easily by simulations. *What if we change one of the starting assumptions? What if we remove a certain part of the system?* These are not easy to analyze in general, but simulations may reveal what the outcomes are.

In [2] some commonly-used terms in simulations are defined:

- *system*: A group of objects that interact with each other or are interdependent, joined together to work toward a common goal.
- *system environment*: Where the system is 'located,' i.e. refers to the space outside the system but from where changes to the system can be caused.
- *entity*: An object in the system.
- *attribute*: A property of an entity.
- *activity*: A time period of given length.

- *state*: The collection of variables necessary to describe the system at any given time.
- *event*: An instantaneous occurrence that might change the state of the system.
- *model*: A representation of the system.

There are a few more classifications that are used to refer to systems and their simulation.

- A system is *discrete* if changes can only happen at a discrete set of time points, and *continuous* if variables can change continuously.
- A simulation model is *static* if it represents a system at a fixed point in time, and *dynamic* if it represents a system as it changes over time.
- A model is called *stochastic* if it contains random variables, and *deterministic* otherwise.

Note that we can also refer to models as discrete or continuous, and that the type of model used may not necessarily match the type of system being observed (so a discrete model can be used to study a continuous system, and vice versa). The term **discrete-event simulation** refers to the case when the types of models studied are discrete, dynamic, and stochastic.

Typically, simulation-based studies will follow the same framework. First, a problem is identified and formulated. Then, objectives are set – what do we want to know? Is simulation the appropriate tool? Then a model is formulated; this may be a very involved process, and there is no single approach that will work for all problems. In general, it is a good idea to start with a simple model, and then increase the complexity as the study moves forward.

Once the model is constructed, input data is collected, and the model programmed into a computer. Then *model verification* and *model validation* are done. This means that the model is first checked, to make sure the data representations used are appropriate, and the logic is sound (verification). Then, the model is run on a few test cases where the real-world outcome is known, and the outputs are compared (validation).

After these steps, the actual experiment begins. Parameters such as the maximum number of iterations, maximum time of simulation, and number of replications of each run are fixed, and input data is selected. Then the model is run, and an analysis is carried-out on the system being studied.

4.2 Simulating Simple Queues

Queueing is something everyone experiences in daily life – we wait in lines at the supermarket, at the bank, when taking transit, and so on. Routers also queue information to send to your wireless devices. Modelling the behaviour of queues can be very involved and complicated, especially for more complex systems. When studying a system with queues, we are interested in various properties of such as:

(1) Average waiting times,

(2) Average total times in the system (including the processing time),

(3) Average size of queues,

(4) Probability of spending time in a given queue,

(5) Proportion of idle time of the server.

To understand a queueing system, we should model arrival times for *customers*, their routes through the system, and *service times* once a customer is being processed. For instance, at a supermarket we might want to analyze its queueing system for checking out. This would involve gathering data empirically to determine what probability distribution governs the time between two customer arrivals at the queue (which may vary depending on time of day) and studying how long it takes to serve a customer. This analysis could then be used to predict the effects of adding counters or varying the checkout procedures, e.g. through introducing express counters or self-checkout counters.

The following assumptions are typical in simple queueing models:

(a) Interarrival times of the customers are drawn independently from some probability distribution A.

(b) Service times of successive customers are drawn independently from some probability distribution S.

(c) When a customer arrives and the server is idle, the customer is immediately served. Otherwise, if the server is busy, the customer joins the end of the queue.

Key characteristics that differentiate one queueing system from another are the probability distributions for arrivals and service time, and the number of servers. For now we limit our discussion to single-server queues.

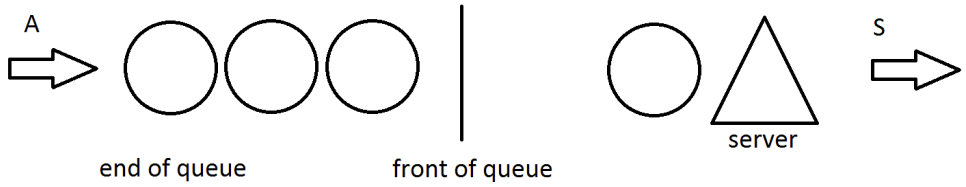


Figure 4.1: A single-server queue.

Example 4.1. In a single-server queue, interarrival times follow the discrete uniform distribution $U\{1, 6\}$. Customers are served in 3 minutes with probability 0.25, 4 minutes with probability 0.5, and 7 minutes with probability 0.25. Simulate this process until 10 customers are served, and find the average waiting time of a customer.

First, we have to generate interarrival times. Suppose that we used Excel to do this, and obtained the following: 4, 3, 5, 5, 5, 1, 4, 2, 5, 3 (using the `=RANDBETWEEN()` function).

Next, we generate service times, and obtain: 4, 7, 4, 3, 3, 3, 4, 7, 4, 4 (using the `IF()` function with `RAND()`).

Assuming we start at time $t = 0$, the first customer arrives at time $t = 4$. Since the server is idle when he arrives, he is served immediately.

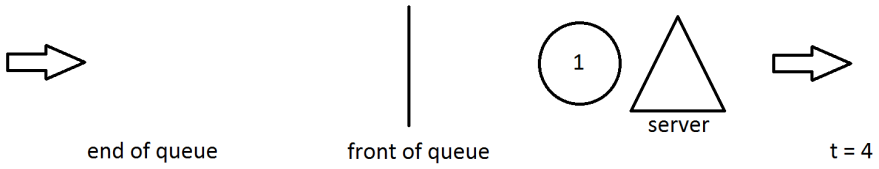


Figure 4.2: State of the system at $t = 4$.

The service time for the first customer is 4 minutes. Before he finishes being served, however, the second customer arrives at time $t = 4 + 3 = 7$.

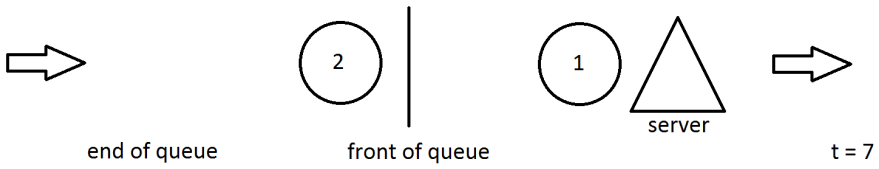


Figure 4.3: State of the system at $t = 7$.

The first customer exits the system at $t = 8$, at which point the second customer is served, with service time 7 minutes. The third customer arrives at time $t = 12$, and has to wait in the queue for 3 minutes until the second customer exits.

The rest of the system behaviour can be derived similarly. Putting everything in a table, we obtain the following values. The waiting time of a customer is the time between arrival

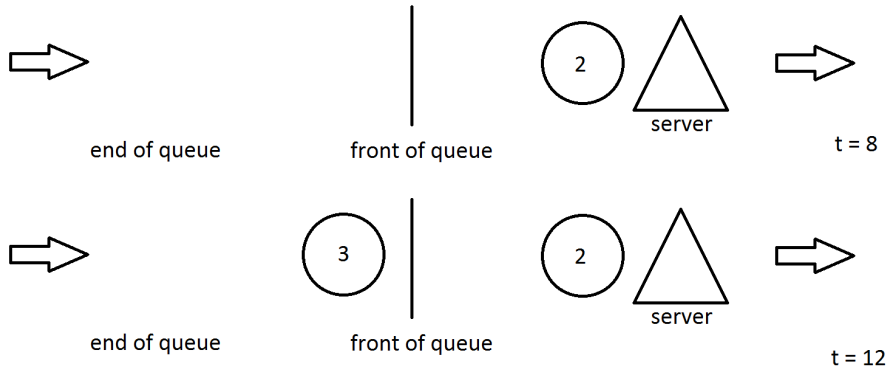


Figure 4.4: States of the system at $t = 8$ and $t = 12$.

time and when service begins, while the total time in the system is the sum of the waiting time and the service time.

Customer	Interarrival times	Time of arrival	Service time	Service start	Service end	Waiting time	Total time in system
1	4	4	4	4	8	0	4
2	3	7	7	8	15	1	8
3	5	12	4	15	19	3	7
4	5	17	3	19	22	2	5
5	5	22	3	22	25	0	3
6	1	23	3	25	28	2	5
7	4	27	4	28	32	1	5
8	2	29	7	32	39	3	10
9	5	34	4	39	43	5	9
10	3	37	4	43	47	6	10

The average waiting time of a customer in this system is therefore 2.3 minutes, while the average time a customer spends in the system is 6.6 minutes. Certainly, the table can (and should) be calculated using Excel. The long-run averages can be computed analytically – however in this example, in the long run the queue will grow as the average service time is longer than the average interarrival time, meaning the long-run wait times diverge to infinity. However this behaviour may not be apparent when simulating only a limited number of customers in Excel. □

In general, queues are represented using the notation A/S/n, where A represents the arrival time distribution, S the service time distribution, and n the number of servers.

When interarrival times and service times are both exponentially distributed, we call the queue an M/M/n queue (where 'M' stands for *memoryless* or *Markov*). This is one of the simplest types of queue to analyze, and provides a convenient model for many applications. Conventionally, the parameters for these distributions are called λ and μ (so interarrival times are Exp(λ) and service times are Exp(μ)).

The queue in Example 4.1 can be referred to as a G/G/1 queue, where G stands for general probability distribution – that is, it is not one of the common ones encountered. [1]

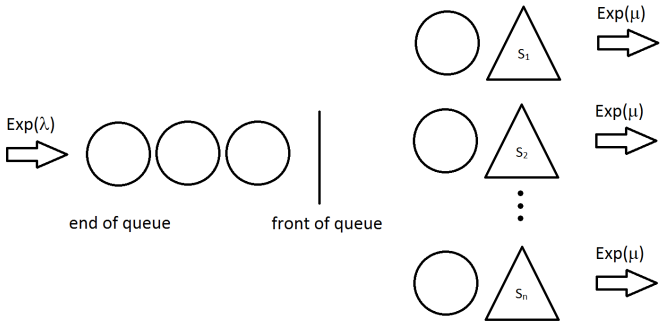


Figure 4.5: An M/M/n queue.

Example 4.2. Suppose that a queue follows the M/M/1 model with interarrival times distributed as Exp(3) and service times distributed as Exp(5), where a unit of time is 1 hour. This means that on average, 3 customers arrive in an hour, and the server can serve 5 customers in an hour, i.e. $\lambda = 3$ and $\mu = 5$. Simulate this queue for 20 customers and compute the average waiting time and the average total time in the system for a customer.

To simulate this in Excel, we need to generate two sequences of values from Exp(λ) and Exp(μ), and use these values to calculate waiting times and other statistics about the system. Recall that we use the formula $-\frac{\ln(1-\texttt{RAND()})}{\lambda}$ to sample from Exp(λ). The spreadsheet in Figure 4.6 shows an Excel simulation for 20 customers given $\lambda = 3$/hour and $\mu = 5$/hour. Observe that the time a customer starts to be served depends on both the arrival time and the service end time of the previous customer, hence in cell `D6` we have the formula `=MAX(C6,E5)`. Also note that we multiply the randomly-generated times by 60 to express them in minutes. The waiting time being 0 means that when the customer arrives the server is immediately available.

In order to calculate the statistics we are interested in, we take the average of the entries for waiting times and total time in system. The spreadsheet allows for varying the parameters λ and μ, so that the queueing system can be studied effectively for different cases. For instance, if customers arrive faster than they can be served ($\lambda > \mu$), the system intuitively would not be able to keep up, and the total time a customer spends in the system grows exponentially as more arrive.

To produce a visual representation for this simulation, we input the arrival time, waiting time, and service time for each customer in three columns, and use the *stacked bar* chart in Excel. Changing the fill of the first series to 'no fill' will produce the following, where the red bars denote waiting times, and green denotes service times. [2] □

[1] See the wiki page `https://en.wikipedia.org/wiki/Kendall's_notation`.
[2] This follows the method described in [7].

	A	B	C	D	E	F	G	H	I	J	K	L
1	**lambda**	**mu**	**M/M/1**									
2	3	5										
3					Averages:	9.79	23.68					
4	interarrival time (minutes)	service time (minutes)	System time	Service start	Service end	Waiting time	Total time in system					
5	2.42	10.24	2.42	2.42	12.66	0.00	10.24			*Formulas*		
6	8.25	24.61	10.67	12.66	37.27	1.99	26.60		*A5*	=ROUND(-60*LN(1-RAND())/$A$2,2)		
7	0.76	5.62	11.43	37.27	42.89	25.84	31.46		*B5*	=ROUND(-60*LN(1-RAND())/$B$2,2)		
8	23.65	2.09	35.08	42.89	44.98	7.81	9.90		*C5*	=A5	*D5*	=A5
9	22.68	4.89	57.76	57.76	62.65	0.00	4.89		*C6*	=C5+A6	*D6*	=MAX(C6,E5)
10	10.91	15.62	68.67	68.67	84.29	0.00	15.62		*E5*	=B5+D5		
11	18.50	11.26	87.17	87.17	98.43	0.00	11.26		*F5*	=D5-C5	*G5*	=E5-C5
12	0.76	3.42	87.93	98.43	101.85	10.50	13.92		*F3*	=AVERAGE(F5:F24)		
13	10.80	10.53	98.73	101.85	112.38	3.12	13.65		*G3*	=AVERAGE(G5:G24)		
14	49.12	20.45	147.85	147.85	168.30	0.00	20.45					

Figure 4.6: Simulating an M/M/1 queue in Excel.

Arrival time	Waiting time	Service time
2.42	0.00	10.24
10.67	1.99	24.61
11.43	25.84	5.62
35.08	7.81	2.09
57.76	0.00	4.89
68.67	0.00	15.62
87.17	0.00	11.26
87.93	10.50	3.42
98.73	3.12	10.53
147.85	0.00	20.45
156.54	11.76	5.15
156.66	16.79	16.49
214.22	0.00	24.08
217.8	20.50	40.18
240.21	38.27	0.80
268.09	11.19	38.59
291.89	25.98	2.41
306.51	13.77	8.53
340.49	0.00	9.63
341.91	8.21	23.36

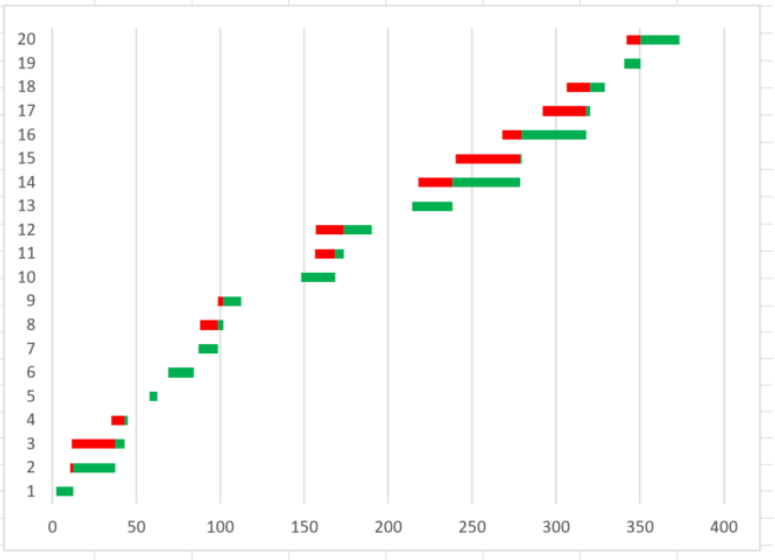


Figure 4.7: Visualizing an M/M/1 queue in Excel.

The M/M/1 queue is the simplest model to analyze; given interarrival time and service time distributions of Exp(λ) and Exp(μ) respectively, the following quantities can be derived for the case $\lambda < \mu$:

- Utilization rate is ρ, which is defined to be $\rho = \lambda/\mu$
- Average waiting time is $\frac{\rho}{\mu(1-\rho)}$
- Average total time in system is $\frac{1}{\mu(1-\rho)}$
- Average number of customers in the system at any given time is $\frac{\rho}{1-\rho}$

If $\lambda \leq \mu$, the queue grows indefinitely since the server cannot keep up with the number of arriving customers.

Similar quantities can be derived for the more general M/M/n case. These models are more difficult to simulate in Excel – for instance, when simulating an M/M/2 queue, for each arrival there are now three cases: server #1 is available, server #2 is available (but not #1), or neither is available.

In Figure 4.8 we show a spreadsheet simulation for the M/M/2 queue, where interarrival times are Exp(λ) and service times are Exp(μ) for both servers, with $\lambda = 5$/hour and $\mu = 4$/hour. Note that cells in columns `E` and `H` keep track of the earliest time each server will be available for that customer. A graph similar to the M/M/1 case can also be generated here, by taking the same three series: arrival time, waiting time, and service time.

	A	B	C	D	E	F	G	H	I	J	K
1		**lambda**	**mu**								
2		5	4								
3					**server 1**			**server 2**			
4		interarrival	arrival time	server #	can start	start	end	can start	start	end	service time
5		20.09	20.09	1	20.09	20.09	60.90	20.09			40.81
6		13.31	40.18	2	60.90			40.18	40.18	44.87	4.69
7		14.20	53.49	2	60.90			53.49	53.49	59.3	5.81
8		0.29	67.69	1	67.69	67.69	85.24	67.69			17.55
9		13.13	67.98	2	85.24			67.98	67.98	102	34.02
10		8.48	81.11	1	85.24	85.24	96.54	102.00			11.30
11		27.87	89.59	1	96.54	96.54	134.08	102.00			37.54
12		29.87	117.46	2	134.08			117.46	117.46	140.16	22.70
13		13.06	147.33	1	147.33	147.33	185.12	147.33			37.79

Figure 4.8: Simulating an M/M/2 queue in Excel – times are in minutes.

Formulas			
B5	=ROUND(-60*LN(1-RAND())/$B$2,2)		
K5	=ROUND(-60*LN(1-RAND())/$C$2,2)		
C5	=B5	E5	=C5
		F5	=C5
C6	=C5+B6	G5	=C5+K5
D5	=IF(E5<=H5,1,2)		
E6	=MAX(C6,G$5)		
F6	=IF($D6=1,$E6,"")		
G6	=IF($D6=1,$E6+$K6,"")		
H6	=MAX(C6,J$5)		
I6	=IF($D6=2,$H6,"")		
J6	=IF($D6=2,$H6+$K6,"")		
E7	=MAX(C7,G$5:G6)		
H7	=MAX(C7,J$5:J6)		

Figure 4.9: Excel formulas for simulating an M/M/2 queue.

arrival	waiting	service
20.09	0.00	40.81
40.18	0.00	4.69
53.49	0.00	5.81
67.69	0.00	17.55
67.98	0.00	34.02
81.11	4.13	11.30
89.59	6.95	37.54
117.46	0.00	22.70
147.33	0.00	37.79
160.39	0.00	51.23
174.03	11.09	7.17
190.39	1.90	30.94
199.71	11.91	49.62
207.18	16.05	21.26
220.63	23.86	19.68
224.13	37.11	14.51
242.61	21.56	3.22
246.94	20.45	5.15
269.09	3.45	12.91
274.80	0.95	10.34

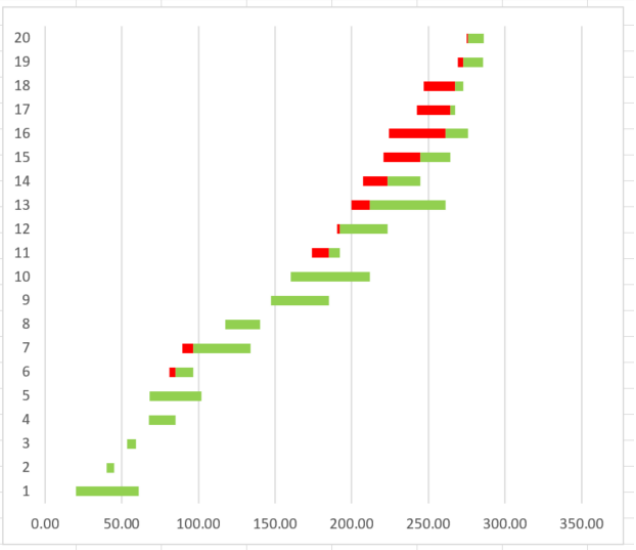


Figure 4.10: Visualizing an M/M/2 queue in Excel.

4.3 Simulating Real-time Games

Here we consider a more detailed simulation of a real-time system: modeling a hockey game. In hockey, teams of skaters score goals by shooting a puck into their opponent's net. It is played between two teams of six: this includes the goaltender and five other players who are trying to score. The game is composed of three 20-minute periods, the team with more goals at the end wins. Ties are broken in varying ways depending on the type of game played (professional/amateur, regular/playoff).

Image by Calle Eklund
Wikimedia Commons

Figure 4.11: A hockey game.

There is a substantial body of research into modeling goal scoring in hockey. One tactical problem that has been carefully studied is the question of when the goalie should be pulled – that is, when a team is down by one goal (or more) in the late game, their coach must decide when the right time is to replace the goalie with another attacking skater. Modeling the game allows analyses to be performed in different scenarios. In [13] and [3], for example, a hockey game is modeled as a random (Markov) process where interarrival times of goals follow either an exponential or a generalized exponential distribution. We will not go into as complex a model as these ones; instead we follow [11] and try to simulate a more rudimentary model of the game.

When the teams are of equal quality, we assume that while both teams have all six players on the ice (the goalie and five attackers), they are of equal ability, and each team scores goals independently at a rate of λ per minute. This is the simplest framework that we can assume for modeling a hockey game; let us try to model it in Excel.

To simulate one period of a hockey game, for each team we generate a sequence of exponentially-distributed random variables, and stop once the sum of the generated values exceeds 20 minutes. For instance, using $\lambda = 0.1$ goals/min and the inverse function technique from Section 2.5, we obtain the following interarrival times: 5.88, 12.13, 13.22, 2.43, ... This means that goals were scored by that team at 5.88 and 18.01 minutes, for a total of two goals in the first period (since the sum of the first three values already exceeds

20).

We repeat the same process for the other team, and for the second and third periods (resetting when each period starts). Here's what it looks like in Excel:

	A	B	C	D	E	F	G
1		**lambda**	**0.1**	**A**	**B**		
2		Exponential R.V.'s		Goals		Score at end of period	
3	*First Period*	5.88	32.91	5.88	~~32.91~~	2	0
4		12.13	23.50	18.01	~~56.42~~		
5		13.22	7.42	~~31.24~~	~~63.83~~		
6		2.43	6.71	~~33.67~~	~~70.55~~		
7		0.96	1.76	~~34.63~~	~~72.31~~		
8		2.20	14.63	~~36.82~~	~~86.93~~		
9	*Second Period*	13.11	9.21	33.11	29.21	5	3
10		3.43	5.45	36.55	34.66		
11		2.37	0.90	38.92	35.56		
12		3.81	5.32	~~42.73~~	~~40.89~~		
13		1.67	13.53	~~44.39~~	~~54.42~~		
14		12.17	7.94	~~56.57~~	~~62.36~~		
15	*Third Period*	0.12	3.03	40.12	43.03	**9**	**7**
16		4.12	0.48	44.24	43.51	**FINAL**	
17		3.11	8.90	47.35	52.41		
18		9.03	6.52	56.39	58.94		
19		18.77	3.45	~~75.15~~	~~62.39~~		
20		3.62	1.71	~~78.78~~	~~64.10~~		

Figure 4.12: Simulating a hockey game.

The corresponding formulas are as follows:

Formulas		
B3:C20	*=-LN(1-RAND())/$C$1*	
D3:D8	*=SUM($B$3:B3)*	*(drag down)*
E3:E8	*=SUM($C$3:C3)*	
D9:D14	*=SUM(B$9:B9)+20*	
E9:E14	*=SUM(C$9:C9)+20*	
D15:D20	*=SUM(B$15:B15)+40*	
E15:E20	*=SUM(C$15:C15)+40*	
F3	*=COUNTIF(D3:D8,"<20")*	
F9	*=F3+COUNTIF(D9:D14,"<40")*	
F15	*=F9+COUNTIF(D15:D20,"<60")*	
G3	*=COUNTIF(E3:E8,"<20")*	
G9	*=G3+COUNTIF(E9:E14,"<40")*	
G15	*=G9+COUNTIF(E15:E20,"<60")*	

Figure 4.13: Formulas for the Excel spreadsheet in Figure 4.12.

Observe that to get times in the second and third periods (columns `C` and `D`), we added 20 and 40 (respectively) to the same formula we used in the first period. Furthermore, these cells have been formatted such that values that are above the possible period times are struck-through. Finally, the `COUNTIF` function outputs the actual number of goals at the end of each period.

Indeed, the scoring rate λ for the two teams can be different, and it is an easy change in the spreadsheet to adopt this. Nevertheless, it is an entirely different problem to estimate

λ for actual teams – for this one needs to go into data fitting and parameter estimation, which are beyond the focus of this Chapter. What we are interested in is running the simulation *given* the λ values.

A few more observations:

- The Excel spreadsheet we used only generated six exponential random variables in each period, for each team. This might not be enough if λ is increased; if $\lambda = 0.4$ goals/min, for example, this translates into a rate of 8 goals per 20-minute period. While not necessarily realistic, the model should generate enough random variates such that the sum of the interarrival times is larger than 20. This is one limitation of Excel, in that we need to specify in advance how many we are generating.
- Penalties and power plays are not considered in the above simulation; a complete model would take into account different 'states': 5 vs. 5, 5 vs. 4, etc, and have different values of λ for each team, for each of these cases.
- A team can have different values of λ depending on the *individual players* on the ice. This will be extremely difficult to model, though.
- Simulating goalie-pulling is a bit more involved; generally, a team will only pull their goalie when it is late in the third period, and they are only one or two points behind. One also needs to run simulations for different scenarios: pulling the goalie with a minute left, with 2 minutes left, etc.

	A	B	C	D	E	F	G
1		A	B				
2		**0.05**	**0.1**	**A**	**B**		
3		Exponential R.V.'s		Goals		Score at end of period	
4	First Period	2.85	2.36	2.85	2.36	2	3
5		8.30	4.90	11.15	7.26		
6		30.48	12.11	~~41.62~~	19.37		
7		3.44	1.76	~~45.07~~	~~21.13~~		
8		25.10	15.65	~~70.17~~	~~36.78~~		
9		10.71	0.14	~~80.88~~	~~36.92~~		
10	Second Period	37.38	0.64	~~57.38~~	20.64	2	5
11		27.28	15.16	~~84.66~~	35.80		
12		16.27	6.04	~~100.93~~	~~41.84~~		
13		36.29	16.09	~~137.22~~	~~57.93~~		
14		35.29	2.41	~~172.51~~	~~60.33~~		
15		5.67	7.13	~~178.18~~	~~67.46~~		
16	Third Period	1.00	14.75	41.00	54.75	3	7
17		26.29	2.73	~~67.29~~	57.49	**FINAL**	
18		3.92	2.71	~~71.21~~	~~60.19~~		
19		13.86	8.54	~~85.07~~	~~68.73~~		
20		7.09	10.82	~~92.16~~	~~79.55~~		
21		3.96	14.72	~~96.12~~	~~94.27~~		

Figure 4.14: Simulating a hockey game where the two teams have different scoring rates.

EXERCISES

4.1 Simulate an M/M/1 queue in excel with $\lambda = 3$/hour and $\mu = 6$/hour until 20 customers are served. Compute the average waiting time and average time in the system for customers. Then compute the proportion of time that the server is idle.

4.2 Simulate an M/M/1 queue in excel with $\lambda = 6$/hour and $\mu = 3$/hour until 20 customers are served. Compute the average waiting time and average time in the system for customers. Then compute the proportion of time that the server is idle.

Use the technique in Example 4.2 to produce a graph of the simulation you produced. In theory, since customers are arriving faster than they can be served, the length of the queue will blow up as time goes to infinity. Does your graph confirm this?

4.3 Simulate an M/M/2 queue in Excel with $\lambda = 8$/hour and $\mu = 5$/hour until 20 customers are served. Compute the average waiting time for customers, then compute the proportion of time that the server is idle.

4.4 Compare the M/M/2 queue in Exercise 4.3 with an M/M/1 queue where $\lambda = 8$/hour and $\mu = 10$/hour. Do you think the expected waiting times will be similar? Simulate this queue and compare.

How about <u>two</u> M/M/1 queues with $\lambda = 4$/hour and $\mu = 5$/hour – can this be compared to the single M/M/2 queue in Exercise 4.3? Explain what assumptions might have to be made.

4.5 In the M/D/1 model, the 'D' stands for degenerate, meaning service times are constant. This is sometimes a reasonable assumption to make; it comes up in systems where the service requires the same tasks for each customer. Let the service time be μ. Create a spreadsheet in Excel to model this queue for varying values of λ and μ.

4.6 Simulate an M/G/1 queue where the service times are normally distributed $\sim N(\mu, \sigma)$. For fixed λ and μ, what effect do you think changing σ has on the average waiting time of a customer? Use Excel to study the behaviour of the queue. (Hint: What happens as $\sigma \to 0$?)

4.7 **Critique**. Is it reasonable to assume that interarrival times between goals in a hockey game are exponentially-distributed? Why or why not?

4.8 Simulate a number of hockey games between two teams where $\lambda = 0.07$ for both. Mathematically, excluding ties, each team should win over the other half of the time. Does your experiment confirm this?

4.9 **Overtime**. Adapt the spreadsheet in Figure 4.14 to include an overtime scenario when the score is tied after regulation play. Is it reasonable to assume that a 3-on-3 manpower situation will have the same λ values as 5-on-5?

4.10 **Pulling the goalie**. Simulate a few hockey games where the goalie is pulled by the team that is trailing with <u>one minute</u> left in regulation play. Make assumptions on the λ parameters for the two teams, and justify your choices.

Bibliography

[1] Kenneth R. Baker, *Optimization modeling with spreadsheets*, 3rd ed., Wiley, 2015.

[2] Jerry Banks, John S. Carson, II, Barry L. Nelson, and David M. Nicol, *Discrete-event system simulation*, Pearson Education, Inc., 2010.

[3] David Beaudoin and Tim Swartz, *Strategies for pulling the goalie in hockey*, The American Statistician **64** (2010), no. 3, 197–204.

[4] Dimitris Bertsimas and Robert M. Freund, *Data, models, and decisions : the fundamentals of management science*, Dynamic Ideas, 2004.

[5] Silviu Guiasu, *Probabilistic models in operations research*, Nova Science Publishers, Inc., New York, 2009.

[6] Frederick S. Hillier and Gerald J. Lieberman, *Introduction to operations research*, ninth ed., McGraw-Hill, 2010.

[7] Armann Ingolfsson and Thomas A. Grossman, Jr., *Graphical spreadsheet simulation of queues*, INFORMS Transactions on Education **2** (2002), no. 2, 27–39.

[8] Richard J. Larsen and Morris L. Marx, *An introduction to mathematical statistics and its applications*, Pearson Education, Inc., 2004.

[9] Averill M. Law, *Simulation modeling and analysis*, McGraw-Hill, 2015.

[10] Lawrence M. Leemis and Stephen K. Park, *Discrete-event simulation: A first course*, Pearson Education, Inc., 2006.

[11] Donald G. Morrison and Rita D. Wheat, *Misapplications reviews: Pulling the goalie revisited*, Interfaces **16** (1986), no. 6, 28–34.

[12] Mahmut Parlar, *Interactive operations research with Maple*, Birkhäuser Boston, Inc., Boston, MA, 2000, Methods and models.

[13] Andrew C. Thomas, *Inter-arrival times of goals in ice hockey*, Journal of Quantitative Analysis in Sports **3** (2007), no. 3.